Highly Efficient Deep Intelligence via Multi-Parent Evolutionary Synthesis of Deep Neural Networks

by

Audrey Chung

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2020

Examining Committee

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner	Dr. James H. Elder Professor, Electrical Engineering & Computer Science York University, Toronto, ON
Supervisor	Dr. Alexander Wong Associate Professor, Systems Design Engineering University of Waterloo, Waterloo, ON
Supervisor	Dr. Paul Fieguth Professor, Systems Design Engineering University of Waterloo, Waterloo, ON
Internal Member	Dr. John Zelek Associate Professor, Systems Design Engineering University of Waterloo, Waterloo, ON
Internal Member	Dr. Bryan Tripp Associate Professor, Systems Design Engineering University of Waterloo, Waterloo, ON
Internal-External Member	Dr. Mark Crowley Assistant Professor, Electrical & Computer Engineering University of Waterloo, Waterloo, ON

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

The following six papers are used in this thesis. I was a co-author with major contributions on the design, development, evaluation, and writing of the papers' material.

A. G. Chung, M. J. Shafiee, P. Fieguth, and A. Wong, "The Mating Rituals of Deep Neural Networks: Learning Compact Feature Representations through Sexual Evolutionary Synthesis," *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1220-1227, 2017.
This paper is incorporated in Chapter 5 of this thesis.

A. G. Chung, P. Fieguth, and A. Wong, "Polyploidism in Deep Neural Networks: m-Parent Evolutionary Synthesis of Deep Neural Networks in Varying Population Sizes," *Journal of Computational Vision and Imaging Systems*, vol. 3, no. 1, 2017.
This paper is incorporated in Chapter 5 of this thesis.

A. G. Chung, P. Fieguth, and A. Wong, "Mitigating architectural mismatch during the evolutionary synthesis of deep neural networks," *Workshop on Meta-Learning (MetaLearn 2018), Conference on Neural Information Processing Systems (NeurIPS)*, 2018.
This paper is incorporated in Chapter 6 of this thesis.

A. G. Chung, P. Fieguth, and A. Wong, "Assessing Architectural Similarity in Populations of Deep Neural Networks," *Women in Computer Vision Workshop (WiCV 2019), Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
This paper is incorporated in Chapter 6 of this thesis.

A. G. Chung, P. Fieguth, and A. Wong, "Nature vs. Nurture: The Role of Environmental Resources in Evolutionary Deep Intelligence," *2018 15th Conference on Computer and Robot Vision (CRV)*, pp. 368-374, 2018.
This paper is incorporated in Chapter 7 of this thesis.

A. G. Chung, P. Fieguth, and A. Wong, "Nature vs. Nurture II: Environmental Resources in Evolutionary Deep Intelligence Revisited," submitted to *14th Women in Machine Learning Workshop (WiML 2019), Conference on Neural Information Processing Systems (NeurIPS)*, 2019.
This paper is incorporated in Chapter 7 of this thesis.

Abstract

Machine learning methods, and particularly deep neural networks, are a rapidly growing field and are currently being employed in domains such as science, business, and government. However, the significant success of neural networks has largely been due to the increasingly large model sizes and enormous amounts of required training data. As a result, powerful neural networks are accompanied by growing storage and memory requirements, making these powerful models infeasible for practical scenarios that use small embedded devices without access to cloud computing.

As such, methods for significantly reducing the memory and computational requirements of high-performing deep neural networks via sparsification and/or compression have been developed. More recently, the concept of *evolutionary deep intelligence* was proposed, and takes inspiration from nature and allows highly-efficient deep neural networks to organically synthesize over successive generations. However, current work in evolutionary deep intelligence has been limited to the use of asexual evolutionary synthesis where a newly synthesized offspring network is solely dependent on a single parent network from the preceding generation.

In this thesis, we introduce a general framework for synthesizing efficient neural network architectures via multi-parent evolutionary synthesis. Generalized from the asexual evolutionary synthesis approach, the framework allows for a newly synthesized network to be dependent on a subset of all previously synthesized networks. By imposing constraints on this general framework, the cases of asexual evolutionary synthesis, 2-parent sexual evolutionary synthesis, and m-parent evolutionary synthesis can all be realized.

We explore the computational construct used to mimic heredity, and generalize it beyond the asexual evolutionary synthesis used in current evolutionary deep intelligence works. The efficacy of incorporating multiple parent networks during evolutionary synthesis was examined first in the context of 2-parent sexual evolutionary synthesis, then generalized to m-parent evolutionary synthesis in the context of varying generational population sizes. Both experiments show that the use of multiple parent networks during evolutionary synthesis allows for increased network diversity as well as steeper trends in increasing network efficiency over generations.

We also introduce the concept of gene tagging within the evolutionary deep intelligence framework as a means to enforce a like-with-like mating policy during the multi-parent evolutionary synthesis process, and evaluate the effect of architectural alignment during multi-parent evolutionary synthesis. We present an experiment exploring the quantification of network architectural similarity in populations of networks.

In addition, we investigate the the computational construct used to mimic natural selection. The impact of various environmental resource models used to mimic the constraint of available computational and storage resources on network synthesis over successive generations is explored, and results clearly demonstrate the trade-off between computation time and optimal model performance.

The results of m-parent evolutionary synthesis are promising, and indicate the potential benefits of incorporating multiple parent networks during evolutionary synthesis for highly-efficient evolutionary deep intelligence. Future work includes studying the effects of inheriting weight values (as opposed to random initialization) on total training time and further investigation of potential structural similarity metrics, with the goal of developing a deeper understanding of the underlying effects of network architecture on performance.

Acknowledgements

Learning is a weightless treasure you can always carry easily.

– *Chinese Proverb*

First and foremost, I would like to thank my supervisors Prof. Alexander Wong and Prof. Paul Fieguth, whose support and eagerness to explore new ideas has long been (and will always be) a source of inspiration. Thank you for being wonderful mentors over the past few years, and for guiding me as I grow as a researcher.

I would also like to thank Prof. John Zelek, Prof. Bryan Tripp, and Prof. Mark Crowley for being a part of my PhD examining committee, and Prof. James Elder for agreeing to be my external examiner. Carving out the time from your exceedingly busy schedules to read and revise my thesis is greatly appreciated. In addition, thank you to the Natural Sciences and Engineering Research Council (NSERC) of Canada for providing me with funding to not only conduct the research in this thesis, but to also enrich my PhD graduate studies with a research internship abroad at the University of Tokyo.

A huge thank you to the wonderful mosaic of people that make up the Vision and Image Processing Research Group. Having been a part of this group long enough to see us grow from a dozen students to over 50, I will forever be grateful for the enthusiasm and sense of community we foster every day – but I can sadly no longer afford to make enough cupcakes for all of us.

I would like to thank my family and friends for their constant support and blind faith in my abilities. Thanks for listening to me spew information about topics you were unfamiliar with, but hopefully are more familiar with now. To Katrina and Joanne – this marks the 10 year anniversary of when we met (and very determinedly became friends) back in first year, and I will forever be grateful to UWaterloo's roommate matching algorithm. To Rob and Kaylen – thank you for all the love and support you have given over the last few years; the days spent writing the comprehensive proposal and final thesis were rough, and I doubt I would have emerged relatively unscathed without your help.

Lastly but most importantly, Brendan, you have been a steady and reliable part of my life, and I appreciate your calm and unflappable nature. Worry not – I am confident I can express enough exhilaration and devastation for the both of us. Thanks for all your support during this adventure, and I look forward to the next one with you.

To Brendan

... And Charlie, I suppose.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The field of machine learning (ML) and artificial intelligence (AI) is rapidly growing in modern society, and these techniques are currently being used in a variety of domains, including science, business, and government. In the last few years, deep neural networks (DNNs) [1–4] have taken over the field of AI due to their demonstrated ability to accurately model abstract problem spaces and significantly improve the performance over other machine learning methods. Deep neural networks, especially deep convolutional neural networks (CNNs), have exploded in popularity and have regularly been applied to various challenging areas of research, such as image recognition [4–7] and speech recognition [2, 8–10]. However, this boost in performance of deep neural networks is largely attributed to increasing model sizes and complexity. In addition to requiring enormous amounts of data to properly train, these increasingly large models have resulted in growing storage and memory requirements [11].

1.1 Problem Definition

The astronomical computational requirements and memory requirements make high performance deep neural networks infeasible for on-the-edge devices without access to cloud computing. For many practical situations such as self-driving cars and smartphone applications, the available computing resources are limited to low-power, embedded graphics processing units (GPUs) and central processing units (CPUs). For example, the NVIDIA Jetson Nano has 4GB of memory [12] while the Raspberry Pi 4 has 1GB, 2GB, or 4GB of memory available [13]. In addition, much of the power consumption of these edge devices is attributed to simply loading the neural network (as opposed to the computations

themselves) [14]. As shown in [15], reducing a network model size by one order of magnitude (343Mb to 16.7Mb) can be the difference between insufficient memory and model tractability. With such limited computational power and storage, smaller and more compact versions of deep neural networks that can run on these stand-alone systems are highly desirable. As such, research into highly efficient deep neural networks has been conducted, and methods have been developed (as detailed in Section 2.3) for significantly reducing the memory and computational requirements with minimal drop in performance via the sparsification and/or compression of high-performing deep neural networks.

Rather than attempting to compress existing deep neural networks into smaller and more compact models directly, Shafiee *et al.* [16] recently proposed a radically different approach to efficient neural networks: "Can deep neural networks evolve naturally over successive generations into highly efficient deep neural networks?" Taking inspiration from nature, Shafiee *et al.* introduced the concept of *evolutionary deep intelligence*, where highly efficient deep neural networks are organically synthesized over successive generations. While the genetic encoding scheme used to mimic heredity during the evolutionary synthesis of new networks has been further explored via synaptic cluster-driven genetic encoding [17, 18], previous work in evolutionary deep intelligence [16–18] has been limited to the use of asexual evolutionary synthesis where a new synthesized offspring network is solely dependent on a single parent network from the preceding generation.

Asexual evolutionary synthesis, while effective at synthesizing increasingly efficient networks, results in limited network diversity and only explores a limited range of possible offspring networks as the structure of newly synthesized networks is highly constrained by its parent network. Relative to asexual reproduction, sexual two-parent reproduction allows for rapid adaptation to changing environments and has the potential to accelerate evolution by several orders of magnitude due to the increased diversity in the population [19].

Generalizing on the idea of sexual evolutionary synthesis, multi-parent evolutionary synthesis for highly efficient deep neural networks is proposed to increase offspring network diversity and architectural efficiency over generations. To the best of our knowledge, no research has been conducted in multi-parent evolutionary synthesis of deep neural networks to statistically generate efficient neural network architectures from populations of neural networks via network mating and environmental resource constraints. The focus of other existing research was to produce efficient neural networks via the direct sparsification of computationally expensive networks (as described in Chapter 2.3) or, by applying principles from genetic algorithms, stochastically and iteratively converge on an optimally efficient network in the search space of possible networks (as described in Chapter 2.5) rather than the guided optimization of network architectures via evolutionary synthesis.

1.2 Challenges and Objectives

Although highly desirable, creating efficient deep neural network architectures is complicated, and research to enable the use of deep learning on embedded systems has only begun to emerge in the last few years [11, 20, 21]. The abstract nature of DNNs is ill-suited for human intuition (making the manual creation of efficient architectures difficult) [22] and awkward to parameterize for higher-level learning-based methods [23, 24]. Falling into the last category, the guided optimization of neural network architectures via evolutionary deep intelligence presents a unique set of challenges:

- *Generalization.* The first challenge relates to the generalization of the evolutionary deep intelligence method. The seminal works present a constrained realization of the method, and a general formulation must allow for both flexibility in network dependencies and synthesis as well as the recreation of past realizations via user-imposed constraints.

- *Parameter optimization.* The evolutionary deep intelligence framework is relatively new and, as a result, largely unexplored beyond the parameters previously used. A primary challenge is therefore to study how these various parameters and biologically-inspired computational constructs can best be leveraged to allow for the synthesis of efficient network architectures.

- *Interpretability.* DNNs are regularly referred to as a "black box" technology [25–29], and it is notoriously difficult to interpret their predictions. Even more difficult to understand, then, are the networks themselves. How do we compare the synthesized neural network architectures, and how can we begin to understand the advantages and disadvantages of various architectures?

Evolutionary deep intelligence shows great promise towards guiding the creation of highly efficient architectures; however, the method is relatively early in its development, and further exploration beyond its initial realization can lead to both a deeper understanding of the proposed computational constructs and neural network architectures themselves. To this end, the following objectives are addressed throughout this thesis:

- The primary objective is to propose a general formulation of the evolutionary deep intelligence method, and to study the use of multiple parents during the evolutionary synthesis of efficient neural network architectures.

- The second objective is to investigate the computational constructs used to mimic biological evolution. Specifically, heredity and natural selection should be considered, as random mutation is somewhat analogous to neural network training.
- The third objective is to explore how we can begin to compare different neural network architectures, and the implications of architecture similarity (or dissimilarity) on performance. In the context of this thesis, this will be limited to comparing the synthesized network architectures.

1.3 Contributions

The main contributions of this thesis are as follows:

- *The general formulation and evaluation of m-parent evolutionary synthesis relative to asexual evolutionary synthesis (Chapter 3,Chapter 4,Chapter 5).* Generalizing the original evolutionary deep intelligence to m-parent evolutionary synthesis, we investigate the effects of the number of parent networks used to synthesize a new network and network population (i.e., the number of networks in each generation) on network performance accuracy and architectural efficiency over generations. The incorporation of multiple networks during the synthesis process allows for increased network architecture diversity and exploration of the architecture search space.
- *A gene tagging system to mitigate architectural mismatch during evolutionary synthesis (Chapter 6).* Gene tagging is explored within the context of m-parent sexual evolutionary synthesis and evaluated over a range of environmental resource models. The gene tagging system allows for the proper alignment of architectural structures
- *Assessment of the role of simulated external environmental resources (Chapter 7).* Used to mimic the abundance or scarcity of resources in biological evolution, the environmental resource model is thoroughly explored using a range of possible models during the evolutionary synthesis process. This study shows that there is an optimal range of environmental resource models to facilitate a more gradual trade-off between performance accuracy and model size. that originated from the same location in the ancestor network. The similarity of the resulting network architectures shows less architectural variability than networks synthesized without gene tagging as quantified by relatively higher overlap percentages of architectural clusters, indicating that enforcing a like-with-like mating policy via gene tagging potentially restricts the exploration of the search space of possible network architectures.

1.4 Thesis Structure

Chapter 2 introduces deep neural networks (with an emphasis on convolutional neural networks), and presents a literature review of current neural network sparsity methods and research on evolutionary computing. Chapter 3 proposes a general problem formulation for the synthesis of a new network architecture, and presents realizations of the framework for the asexual evolutionary synthesis and sexual evolutionary synthesis cases via assumptions on network dependency during synthesis. Chapter 4 presents an initial experiment comparing asexual and sexual (i.e., two-parent) evolutionary synthesis, and Chapter 5 extends the formulation and preliminary results to the general m-parent evolutionary synthesis case in varying population sizes per generation. Chapter 6 introduces a gene tagging system to mitigate architectural mismatch during synthesis and enforces a like-with-like mating policy, and Chapter 7 details a study on the role of simulated external environmental resources during the evolutionary synthesis process. Chapter 8 presents an overview of the achieved performance of m-parent evolutionary synthesis with a comparison to state-of-the-art methods. Lastly, Chapter 9 concludes the thesis, providing a summary of thesis contributions and proposing potential research directions for future work.

Chapter 2

Background

2.1 Neural Networks

At the core of nearly all modern deep neural networks (NNs) is the feedforward deep network model [30]. Feedforward models are named for the directionality of information flow in the model (unlike recurrent neural networks which incorporate feedback connections), and the goal of the feedforward model is to approximate some function f^* using a parametric approximation f. In the case of classification, feedforward NNs find a mapping $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ and learn the values for parameters $\boldsymbol{\theta}$ that best approximate a function that maps an input $\mathbf{x}$ into category y, i.e., $y = f^*(\mathbf{x})$. The optimal values for $\boldsymbol{\theta}$ are found via a training algorithm and, depending on the size and complexity of the model, can require millions of training samples to reach convergence. In practice, stochastic gradient descent (SGD) is one of the most commonly used optimization methods during training [3]. Thus, feedforward networks can be seen as nonlinear function approximators that are trained via gradient descent to minimize the approximation error. In general, this is done by minimizing the expected loss on the training set:

$$\frac{1}{m} \sum_{i=1}^{m} L(f(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \mathbf{y}^{(i)}) \tag{2.1}$$

where L is some loss function and m is the number of training examples.

Figure 2.1 shows a visual representation of a general fully connected feedforward NN. Inspired by the human brain, NNs consist of neurons (coloured circles) with connections between them (black lines). The input layer takes in the data samples, and the output layer

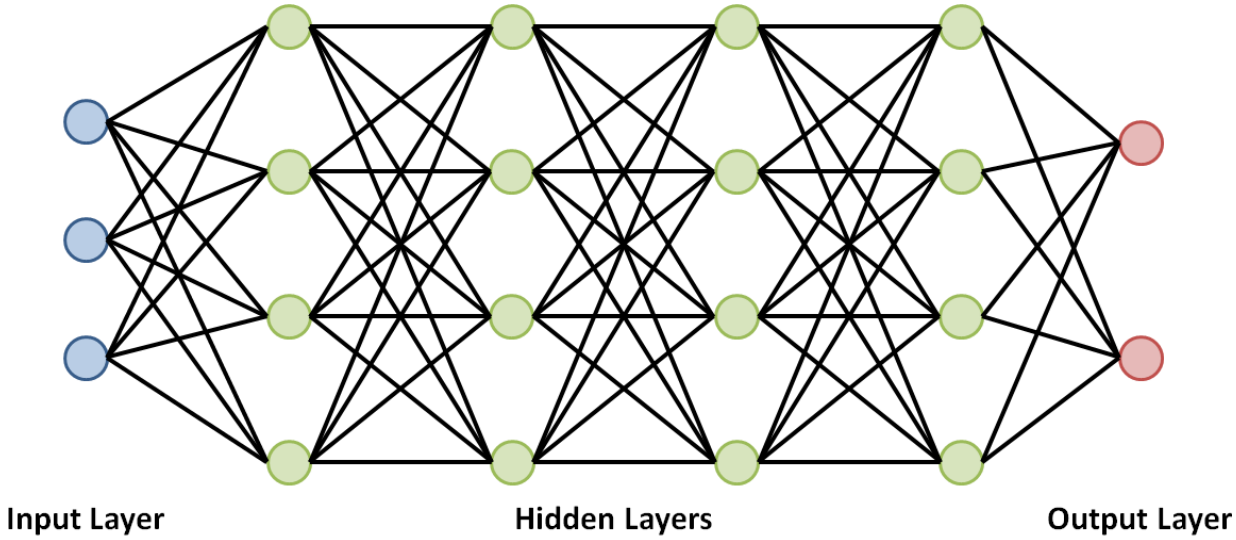


Figure 2.1: Visual representation of a general fully connected feedforward neural network, with an input layer (blue), four hidden layers (green), and an output layer (red).

presents the model's corresponding prediction. While the training data directly specify the behaviour of the output layer, the desired output of each individual layer is left unspecified; as a result, these layers are referred to as hidden layers. The design of the hidden layers has been the focus of a lot of research, and most neurons in general can be described as some affine transformation of a vector of inputs $\mathbf{x}$, i.e., $\mathbf{z} = \mathbf{W}^T\mathbf{x} + \mathbf{b}$, followed by an element-wise nonlinear activation function $g(\mathbf{z})$. The choice of $g(\mathbf{z})$ impacts how easily these neurons can be optimized; as such, rectified linear units ($g(\mathbf{z}) = max\{0, \mathbf{z}\}$) are the most commonly used based on the principle that models with closer to linear behaviour are easier to optimize [30].

2.2 Convolutional Neural Networks

Deep convolutional neural networks (CNNs) [3] are a specialized kind of neural network for processing data with grid-like structure, and essentially refers to any deep neural network that uses convolution in place of the classic matrix multiplication in at least one of its layers. The main advantage of CNNs is their ability to drastically reduce the number of parameters that must be learned via parameter sharing and locality while improving computational efficiency on parallel computing hardware. Based on a spatial stationarity assumption,

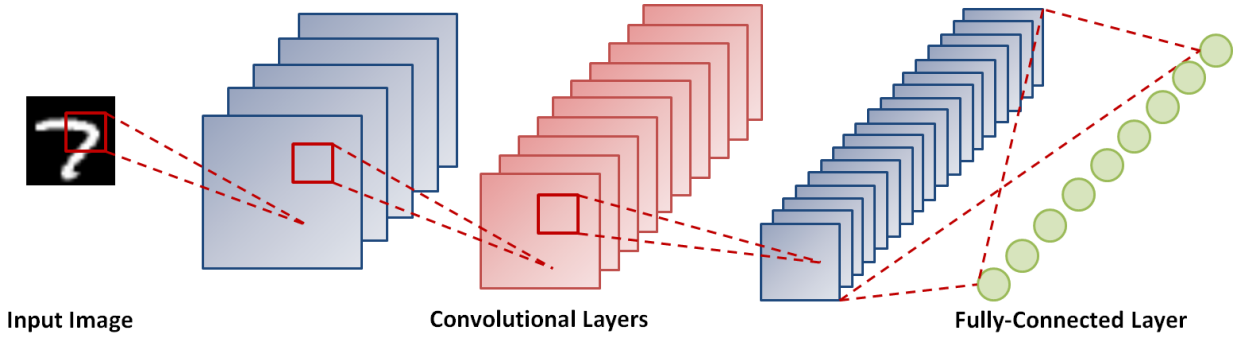


Figure 2.2: Visual representation of a convolutional neural network. The convolutional layers (represented using stacks of kernels) generally act as feature extractors, and are often followed by at least one fully-connected layer (green).

CNNs leverage the idea of local receptive fields and only the elements in the convolution kernels that are applied across the data need to be learned. Relative to feedforward NNs of similar size, CNNs have comparably fewer network parameters and connections to train with minimal drop in the theoretical best performance. As a result, CNNs have been used in a wide variety of computer vision and image-based applications [3], and have been extensively applied to the field of image classification in particular [5, 7, 31–33].

Figure 2.2 shows a visual representation of a convolutional neural network. CNNs have at least one convolutional layer (shown as stacks of kernels) and are often followed by at least one fully-connected later (coloured circles). A convolutional layer typically consists of a convolution stage where the kernels are applied to the layer's input, a nonlinear activation function (similar to a feedforward NN), and a pooling stage to help make the representation invariant to small translations of the input data [30]. Unlike traditional NN layers that use matrix multiplication by a matrix of separate parameters describing the interaction between each unit, CNN typically have sparse interactions which are accomplished by making the convolution kernels smaller than the input.

One of the earliest CNNs that was proposed is the LeNet architecture [34]. Comparatively small relative to recently developed CNNs, LeNet was used for digit recognition and only has three convolutional layers. Due to its shallow architecture, LeNet fails to perform well on more challenging datasets (such as object recognition); however, as there are relatively few parameters to learn, LeNet is still used in applications with limited training samples. Since LeNet, however, a number of notable CNNs have been proposed.

Credited with re-popularizing NNs in 2012, Krizhevsky *et al.* [5] trained AlexNet, one of the largest convolutional neural networks at the time, on the subsets of ImageNet database

used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) for ILSVRC-2010 [35] and ILSVRC-2012 [36] competitions and achieved the best results reported on those databases at the time. Consisting of five convolutional layers and three fully-connected layers, Krizhevsky *et al.* also employed the "dropout" regularization method to prevent overfitting. Krizhevsky *et al.* clearly demonstrated the efficacy of neural networks in solving visual perception problems, and a large number of architectural design advances have been introduced over the past several years to improve the original architecture.

In 2014, Simonyan and Zisserman [31] explored the effects of increasing the depth of Krizhevsky *et al.*'s architecture by fixing other parameters of the architecture, using very small (3×3) convolutional receptive fields, and steadily increasing the depth by adding more convolutional layers. The final VGG architectures (up to 19 weight layers) not only achieved state-of-the-art accuracy on ILSVRC classification and localization tasks, but also generalized well to a wide range of tasks and datasets.

Szegedy *et al.* [7] introduced the "Inception module" (inspired by Lin *et al.*'s Network-in-Network approach [37]) and proposed the Inception architecture to consider how an optimal local sparse structure of a CNN can be approximated by dense components. The Inception architecture is a network consisting of stacked Inception modules, and allows for increased network depth and width while maintaining a computational budget. The architecture used in the ILSVRC-2014 submission was named GoogLeNet, and consisted of 22 layers through repeated Inception layers.

In 2016, He *et al.* proposed ResNet [32] via a deep residual learning framework to mitigate the training requirements of substantially deeper networks than the ones that had been previously used. He *et al.* addressed the degradation problem that exists with deeper networks (i.e., the accuracy of deeper networks saturate during convergence, and then rapidly degrade) by allowing layers to fit a residual mapping $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$ rather than directly fit a desired underlying mapping $\mathcal{H}(\mathbf{x})$. Using the ImageNet dataset, He *et al.* were able to train ResNets of up to 152 layers with lower complexity than VGG networks.

Most recently, Szegedy *et al.* [33] proposed the combination of ResNets [32] and a revised version of their Inception architecture [7] by replacing the filter concatenation in the Inception architecture with residual connections. Szegedy *et al.* noted that the incorporation of residual connections allowed for significantly improved training speed in the Inception architecture, and that the residual Inception networks marginally outperformed similarly expensive Inception networks without residual connections. However, all the recent advances in performance has largely been by virtue of increased model size and, as such, do little for the use of deep neural networks in practical situations with limited computing resources.

2.3 Network Sparsity and Compression Methods

One of the first approaches for adapting the size of a neural network was optimal brain damage [38]. The method removed unimportant parameters (as determined using the second derivative of the objective function as a saliency approximation of each parameter) from the network before retraining the network to a reasonable solution. This procedure can be repeated to iteratively improve network generalizability, increase the speed of learning, and reduce the number of training samples required.

Another early approach was proposed by Suzuki *et al.* [39]. A relatively simple neural network pruning algorithm, Suzuki *et al.* removed units from the neural network to assess the influence of removing each unit on the error. The unit resulting in the least increase in error was removed, and the neural network was retrained to recover the damage from removing the unit. This process was repeated until the network error increased to the predetermined allowable error.

Gong *et al.* [20] proposed a network compression framework where vector quantization was leveraged to shrink the storage requirements of deep neural networks trained for computer vision tasks. Gong *et al.* noted that vector quantization has clear advantages over existing matrix factorization methods, and found a good balance between model size and accuracy could be achieved via the application of k-means clustering to the weights or product quantization.

Chen *et al.* [40] introduced the HashedNet architecture for the compression of neural networks via the hashing trick, which uses a low-cost hash function to randomly group network weights into hash buckets with a single shared parameter value. The parameters are then re-tuned given the weight sharing architecture. Exploiting the redundancy in network layers, the HashedNet architecture leverages the idea of weight-sharing and allows for considerable savings in terms of memory and storage.

Han *et al.* [11] reduced the storage and computational requirements of neural networks with no drop in accuracy by training a network to learn which weights are important, pruning the unimportant connections, and retraining the network to fine tune the remaining weights. Han *et al.* [21] extended their previous method and introduced deep compression to address the limitations of computational power and memory that comes with embedded systems via a three stage pipeline: pruning, trained quantization, and Huffman coding. Deep compression first prunes the network by removing small-weight connections and retraining the network. The remaining network connections are then quantized (8-bit bins for convolutional layers, 5-bit bins for fully-connected layers) and k-means clustering is used to identify the shared weights for each layer. The network is then retrained to fine-tune

the pruned and quantized centroids before the weights and clustering indices are encoded via Huffman coding. The method reduced the storage requirements of a neural network by 35x to 49x with no loss in accuracy.

Other methods for reducing the computational requirements of neural networks include low rank approximations [41–43]. Sainath *et al.* [41] proposed a low-rank matrix factorization of the final weight layer of neural networks, reducing training time with no significant loss in accuracy. Jaderberg *et al.* [42] used low-rank expansions to speed up the computation of deep neural networks (specifically the convolutional layers of convolutional neural networks) by exploiting cross-channel or filter redundancy to construct a low rank basis of filters. Similarly, Ioannou *et al.* [43] created computationally efficient networks using low rank representations of convolutional filters by learning a set of small basis filters that are then combined into more complex filters.

Sparsity learning [44–46] is another strategy used to sparsify deep neural networks. Feng and Darrell [44] demonstrated a novel method for learning components of the structure of a neural network by incorporating the Indian Buffet Process prior; especially effective when there is limited labelled training data, this method captures complex data distributions in an unsupervised generative manner. Liu *et al.* [45] showed how to reduce the number of parameters in neural networks via sparse decomposition by exploiting both intra-channel and inter-channel redundancy. Lastly, Wen *et al.* [46] recently proposed a Structured Sparsity Learning (SSL) method to regularize the structures within deep neural networks (e.g., filters shapes, channels, layer depth).

Hinton *et al.* [47] proposed the use of "distillation" to transfer the knowledge of an ensemble of models or a large highly regularized model to a small deployable model. Class probabilities from the softmax layer of the large model (or ensemble of models) are relaxed to produce a soft set of targets for a transfer dataset, and the small model is trained using the soft targets. Hinton *et al.* showed that the resulting small model performs significantly better than a model of the same size learned directly from the same training data.

Sun *et al.* [48] introduced a method for sparsifying convolutional neural networks for face recognition. Layer-by-layer, the method iteratively sparsifies and retrains the network using the weights learned in previous iterations as initialization. Sun *et al.* noted that training the sparse networks from scratch without the weight initializations of the denser network failed to produce good solutions for face recognition.

Guo *et al.* [49] presented dynamic network surgery, a novel network compression method that reduces network complexity by making on-the-fly connection pruning. Most interestingly, Guo *et al.* incorporate connection splicing into the process to avoid incorrect pruning and perform continual network maintenance. This combination of pruning and splicing al-

lows for model compression via pruning, and connection recovery if pruned connections are found to be important via splicing.

Scardapane *et al.* [50] proposed a group sparse regularization method for neural networks that simultaneously optimized the weights of a neural network, the number of neurons per layer, and the subset of active input features. Extending the group Lasso penalty, each group is defined as the set of outgoing weights from a unit and a group-level sparsity is imposed, resulting in highly compact networks.

Inspired by Han *et al.* [21], Ullrich *et al.* [51] showed that similar compression rates can be achieved using a version of "soft weight-sharing", achieving both the quantization and pruning done in [21] in one re-training procedure. Ullrich *et al.* optimize the minimum description length (MDL) complexity lower bound, and present results indicating that this method works well in practice when applied to different models.

Molchanov *et al.* [52] explored the Variational Dropout technique and extend it to the case of unbounded dropout rates. By training Sparse Bayesian DNNs using Sparse Variational Dropout, the authors found that this method led to extremely sparse solutions in both convolutional and fully-connected layers. Molchanov *et al.* also noted that this approach can be combined with other model compression techniques such as quantization and Huffman coding [21, 51].

Louizos *et al.* [53] proposed that the most principled and effective approach to model compression and computational efficiency with via a Bayesian method, where large parts of the network can be pruned using sparsity inducing priors. Louizos *et al.* use hierarchical priors to prune nodes rather than individual weights, and posterier uncertainties to determine the optimal encoding precision for the weights.

Recently, He *et al.* [54] introduced AutoML for Model Compression (AMC). Foregoing the hand-crafted features associated with conventional model compression methods, He *et al.* leverage reinforcement learning to efficiently sample the design space for improved model compression quality. The authors show that state-of-the-art model compression rates are achieved in a fully automated manner without any human efforts.

2.4 Genetic Algorithms

Genetic algorithms [55–58] are the most prominent example of evolutionary computation methods, and have largely been inspired by studying evolution in nature. First popularized by Holland [59], genetic algorithms evolve a population of candidate solutions (often

referred to as "chromosomes" or "phenotypes") to solve an optimization problem. The candidate solutions are assessed via a fitness function, and a new population of candidate solutions is created via selection and genetic operators (crossover and mutation) within the solution search space. A simple genetic algorithm typically follows these steps [55]:

1. A randomly generated population of candidate solutions are initialized.
2. The fitness of each candidate solution is calculated.
3. A new generation of candidate solutions is created via selection and genetic operators.
4. Repeat from step 2.

After several generations, one or more highly fit candidate solutions will often be present in the population; however, the fitness of candidate solutions is defined relative to the fitness function rather than the underlying optimization problem, making the design of an appropriately representative fitness function crucial. In addition, selection can be implemented in a variety of ways, such as uniformly replicating the best 50% of the current population or replicating candidate solutions proportional to their fitness [60].

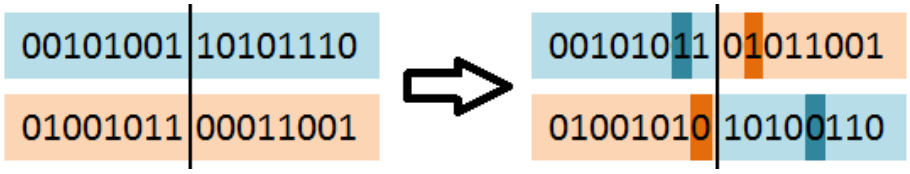


Figure 2.3: Visualization of crossover (split at black line) and random mutation (indicated by darker orange and blue) to create two new candidate solutions from two candidate solutions in the current population. The two new candidate solutions are created by taking the first part of one solution (e.g., blue) and the second part of the other solution (e.g., orange) prior to random mutation.

New candidate solutions are created via genetic operators, the most fundamental being crossover and random mutation. Figure 2.3 shows a basic crossover between two candidate solutions where a crossover point is selected at random, and two new candidate solutions are created by taking the first part of one solution and the second part of the other solution. Random mutation modifies a random fraction of the candidate solutions to prevent a uniform population incapable of further evolution [59].

2.5 Evolutionary Computing for Neural Networks

Previous studies [61–65] have been conducted that leverage the idea of using evolutionary techniques to generate and train neural networks. In 1994, Angeline *et al.* [61] introduced an evolutionary program GNARL (GeNeralized Acquisition of Recurrent Links) that non-monotonically constructs recurrent networks. The GNARL algorithm simultaneously acquires the network topology structure and weight values, minimizing network architectural restrictions and avoiding structural hill climbing.

Stanley *et al.* [62] proposed NeuroEvolution of Augmenting Topologies (NEAT) to investigate the advantage of evolving neural network topologies along with weights. NEAT resulted in faster learning and increased network efficiency by employing a principled crossover method of varying topologies via gene mutation tracking through historical markings, protecting structural innovation through speciation, and incrementally growing networks from minimal structure. The contributions of each component was verified using a series of ablations, and show that historical markings, protecting innovation through speciation, and incremental growth from minimal structure work together to evolve solutions of minimal complexity.

Stanley *et al.* [63] extended the previous work to real-time NeuroEvolution of Augmenting Topologies (rtNEAT) for evolving increasingly complex neural networks real-time during a Neuroevolving Robotics Operatives (NERO) game. NERO was created as a game where learning is indispensable in which the player trains a team of virtual robots controlled by rtNEAT for military combat against other players' teams. The rtNEAT method only produces one new offspring at a time and allows agents to change and evolve increasingly sophisticated behaviours in real time by continually replacing poorly performing agents with new agents while preserving speciation dynamics.

Gauci and Stanley [64] further extended this research to HyperNEAT, a Hypercube-based NeuroEvolution of Augmenting Topologies method. HyperNEAT evolves a generative encoding called connective Compositional Pattern Producing Networks (connective CPPNs) to discover geometric regularities in the application domain. HyperNEAT uses NEAT to evolve CPPNs that represent spatial patterns, with the aim of encoding connectivity patterns independent of the number of inputs and outputs to evolve large-scale neural networks. Unlike directly-encoded methods such as Perceptron NEAT (also proposed in [64]), HyperNEAT can scale solutions to higher resolutions without further evolution as it discovers a general connectivity concept that naturally generalizes to locations outside the training set.

Most recently, Tirumala *et al.* [65] proposed a prospect for evolving network architec-

tures that can reduce training time and provide a warm start to the deep learning process by using optimized deep neural networks as a starting point rather than a randomly initialized architecture. Using two populations of networks where one population evolves using competitive co-evolution and the other population evolves using cooperative co-evolution, the approach uses two different types of co-evolutionary processes by implementing a migration process between the two network populations after a few generations, and encourages combinations with networks from different populations to create more diversity.

2.6 Evolutionary Deep Intelligence

While previous studies have been conducted that leverage the idea of using evolutionary techniques to generate and train neural networks, Shafiee *et al.* [16] proposed an entirely novel concept: *Can deep neural networks naturally evolve to be highly efficient?* Introducing the notion of *evolutionary deep intelligence*, Shafiee *et al.* took inspiration from biological evolution and developed an approach to produce highly efficient and compact deep neural networks by allowing these networks to synthesize new networks with increasingly compact representations with minimal drop in performance over successive generations based on high-performing ancestor neural networks. The evolutionary deep intelligence approach is mimics biological evolutionary mechanisms via three computational constructs: i) heredity, ii) natural selection, and iii) random mutation.

Shafiee *et al.* incorporate the idea of heredity by encoding architectural traits of deep neural networks into synaptic probability models or "DNA" sequences, which are then passed on through successive generations. In addition to the synaptic probability models, external environmental factor models are also considered to mimic the environmental conditions that drive natural selection. Offspring networks are stochastically synthesized based on these "DNA" sequences and computational environmental factor models prior to being trained, thus mimicking heredity, natural selection, and random mutation.

There are key differences between existing evolutionary computing methods and the evolutionary deep intelligence method proposed by Shafiee *et al.* [16]. Past works have primarily focused on improving a network's training and accuracy, while evolutionary deep intelligence shifts the focus to organically synthesizing networks with high architectural efficiency with minimal to no drop in performance accuracy. In addition, these previous studies use classical evolutionary computation approaches such as genetic algorithms and evolutionary programming, while Shafiee *et al.* introduced a novel probabilistic framework that models genetic encoding and environmental conditions via probability distributions.

Shafiee *et al.* has also proposed a modification of the original evolutionary deep intelligence approach via synaptic cluster-driven genetic encoding [17, 18]. Further investigating the genetic encoding scheme used to mimic heredity, Shafiee *et al.* proposed the incorporation of synaptic clustering into the genetic encoding scheme, and introduced a multi-factor synapse probability model. Modelling the synaptic probability as a product of the probability of synthesis of a particular cluster of synapses and the probability of synthesis of a particular synapse within the cluster, this new genetic encoding scheme demonstrated state-of-the-art performance while producing significantly more efficient network architectures and compact feature representations specifically tailored for GPU-accelerated applications.

Shafiee *et al.* explored the imposition of synaptic precision restrictions and its impact on the evolutionary synthesis of DNNs [66], where half-precision (16-bit) synaptic precision was enforced during the synthesis of networks at each generation that were then trained at full precision (32-bit). The resulting networks shows both a reduction in the number of synapses as well as reduced precision requirements while maintaining performance accuracy. Shafiee *et al.* has also introduced the notion of trans-generational genetic transmission of environmental stresses [67], where the imposition of environmental stresses on a network during training produces genetic coding favourable to more efficient offspring networks.

Most recently, Shafiee *et al.* introduced StressedNets [68], a stress-induced evolutionary synthesis framework where stress signals are imposed during training to guide the evolutionary synthesis process towards more efficient architectures over successive generations. The results presented show spectacular increases in efficiency, synthesizing networks capable of state-of-the-art performance accuracy with orders of magnitude reduction in model size. While the StressedNets framework clearly outperforms the seminal evolutionary deep intelligence works [16–18], these improvements are compatible with the general framework proposed in this thesis and can be combined accordingly.

Previous works in evolutionary deep intelligence [16–18] and their subsequent extensions [66–68], however, formulate the evolutionary synthesis process based on asexual reproduction; that is, offspring neural networks are solely dependent on a single parent network. In this thesis, we present a generalized framework for synthesizing new network architectures that can be dependent on any subset of previously synthesized networks, and evaluate the efficacy of multi-parent evolutionary synthesis.

2.7 Summary

This chapter presented the requisite neural network, convolutional neural network, and genetic algorithms background for the remainder of the thesis, along with the relevant

background literature in neural network sparsity and compression methods, evolutionary computing for neural networks, and existing works in evolutionary deep intelligence. Motivated by the need for increasingly powerful deep neural networks that can run on the edge, Chapter 3 extends the seminal evolutionary deep intelligence framework to general architecture synthesis.

Chapter 3

Evolutionary Synthesis Formulation

From Chapter 2, we see that neural networks are fundamentally comprised of neurons and connections. In this chapter, we present the general architecture synthesis formulation, and abstract away from the low-level design of neural network architecture to consider how a new neural network architecture might be synthesized given existing network architectures.

3.1 General Architecture Synthesis

Let the network architecture $\mathcal{H}$ be formulated as $\mathcal{H}(N, S)$, where N denotes the set of possible neurons and S denotes the set of possible synapses in the network. In general, the set of possible neurons and the set of possible synapses are unbounded, representing all (i.e., infinite) possible network architectures. Each neuron $j \in N$, the existence of which is indicated by a binary indicator n_j, is connected to neuron $k \in N$, as indicated by n_k, via a synapse $(j, k) \in S$ such that the synaptic connectivity indicated is by $s_{j,k}$ and has an associated $w_{j,k} \in \mathbb{R}$ to denote the connection's strength. As such, n_j and n_k binarily indicate the existence of neurons j and k, respectively, where $n_j = 0$ indicates that neuron j does not exist in the network architecture while $n_j = 1$ indicates that neuron j does exist. Similarly, $s_{j,k}$ is a binary indicator that denotes the existence of synapse (j, k) (the synaptic connection between neuron j and neuron k) where $s_{j,k} = 0$ indicates that neurons j and k are not connected in the network architecture while $s_{j,k} = 1$ indicates that neurons j and k are connected. The associated $w_{j,k}$, then, represents the strength of the existing synaptic connections (i.e., $s_{j,k} = 1$), and the set of weights for the architecture is defined as $W = \{w_{j,k}\}$.

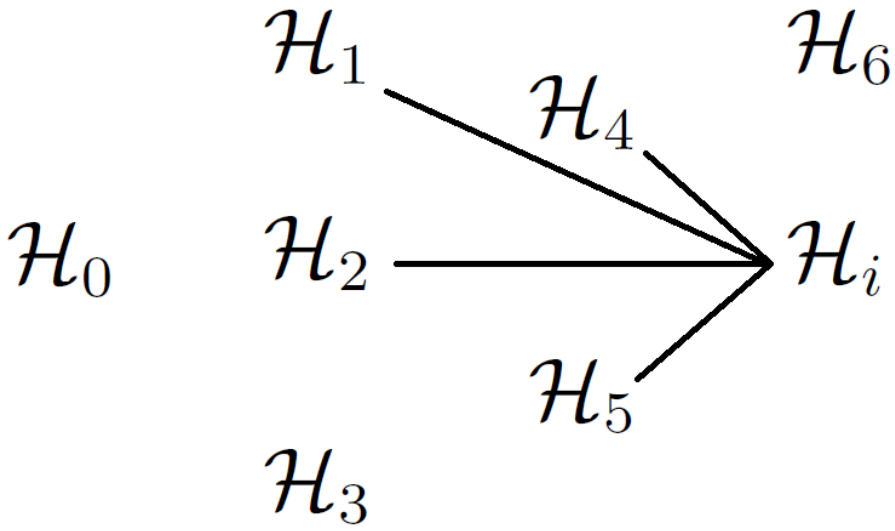


Figure 3.1: Visualization of dependency for the synthesis of a new network. In this realization, $\mathcal{H}_i$ is the 7^{th} network $\mathcal{H}_7$, and is dependent on a subset of all previously synthesized networks $\mathcal{H}_{G_7}$ (in this case, $\mathcal{H}_{G_7} = \mathcal{H}_1, \mathcal{H}_2, \mathcal{H}_4, \mathcal{H}_5$).

The probability of synthesis for a new network architecture, then, is encoded as

$$P(\mathcal{H}_{new}|\mathcal{H}_{old}, \mathcal{R}) \tag{3.1}$$

where the new architecture $\mathcal{H}_{new}$ is dependent on an existing architecture $\mathcal{H}_{old}$ and subject to an environmental factor model $\mathcal{R}$, the computational construct used the mimic natural selection via the constraint of simulated available environmental resources (further described in Section 3.6).

In the case of evolutionary synthesis, a newly synthesized network $\mathcal{H}_i$ is formulated to be dependent on a subset of all previously synthesized networks $\mathcal{H}_{G_i}$, and is encoded as

$$P(\mathcal{H}_i|\mathcal{H}_{G_i}, \mathcal{R}_i) \tag{3.2}$$

where G_i is the set of network indices corresponding to previous networks on which $\mathcal{H}_i$ is dependent, and $\mathcal{R}_i$ denotes the environmental factor model applied to the i^{th} network. Figure 3.1 visualizes a sample realization of $\mathcal{H}_i$'s dependency on a subset of previously synthesized networks. Note that in the general case, the number of networks in subset $\mathcal{H}_{G_i}$ is only constrained by the number of already synthesized networks.

From previous evolutionary synthesis work [16, 17], the synthesis probability of the i^{th} network $\mathcal{H}_i = \mathcal{H}(N_i, S_i)$ can be modelled as the product of the conditional probability of

S_i and the conditional probability of N_i:

$$P(\mathcal{H}_i|\mathcal{H}_{G_i}, \mathcal{R}_i) \simeq P(S_i|S_{G_i}, W_{G_i}, \mathcal{R}_i^S) \cdot P(N_i|N_{G_i}, \mathcal{R}_i^N). \tag{3.3}$$

Here, the synthesis probability of $\mathcal{H}_i$ is approximated by the product of the synthesis probability of its set of synapses S_i given the corresponding set of dependent synapses S_{G_i} and weights W_{G_i} and synapse environmental factor model $\mathcal{R}_i^S$, and the synthesis probability of its set of neurons N_i given the set of existing neurons N_{G_i} and neuron environmental factor model $\mathcal{R}_i^N$. Thus, as a network architecture $\mathcal{H}$ is formulated as comprising a set of neurons N and a set of synapses S, i.e., $\mathcal{H}(N, S)$, we similarly formulate the synthesis probability of a network architecture $\mathcal{H}$ as comprising the synthesis probability of the set of neurons $P(N_i|N_{G_i}, \mathcal{R}_i^N)$ and the synthesis probability of the set of synapses $P(S_i|S_{G_i}, W_{G_i}, \mathcal{R}_i^S)$. As is the case in previous evolutionary synthesis work [16, 17], we model this formulation as a product of the synthesis probabilities of its set of neurons N_i and its set of synapses S_i.

Thus far [16, 17], the nodal structure of any newly synthesized network has been such that $n_j = 1$ for all $j \in N_i$ (i.e., every neuron exists in the network architecture) and only the effects of altering the synaptic structure has been examined. However, note that when all synapses connecting neuron j to other neurons (as indicated by $s_{j,*}$) have an associated weight of zero (i.e., $w_{j,*} = 0$), neuron j is effectively removed from that network architecture realization, i.e., $n_j = 1$, but neuron j exists in isolation since all $w_{j,*} = 0$. For the scope of this thesis, we will implement a similar $n_j = 1$ nodal structure for synthesized network architectures while allowing for changes in the synaptic structure; as such, the synthesis probability can be reformulated as

$$P(\mathcal{H}_i|\mathcal{H}_{G_i}, \mathcal{R}_i) \simeq P(S_i|S_{G_i}, W_{G_i}, \mathcal{R}_i) \tag{3.4}$$

where the synthesis probability of a network architecture $\mathcal{H}_i$ is approximated by the synthesis probability of its set of synapses S_i alone. By taking inspiration from nature, this general formulation for network synthesis can be simplified using the concepts of asexual and sexual reproduction.

In this thesis, the synthesis of a new network architecture $\mathcal{H}_i$ is effectively done via a two-step process:

1. the synaptic structure S_i of $\mathcal{H}_i$ is inherited from $\mathcal{H}_{G_i}$ via S_{G_i}, and
2. $\mathcal{H}_i$ is subject to simulated environmental resources via $\mathcal{R}_i$ and W_{G_i}.

The inheritance of the synaptic structure is first described in Section 3.3 for asexual evolutionary synthesis, and is extended to sexual evolutionary synthesis in Section 3.4. The use of synaptic strengths and environmental resources during evolutionary synthesis is described in Sections 3.5 and 3.6, respectively.

3.2 Practical Considerations

Before we begin synthesizing new neural networks, however, some practical considerations must first be addressed regarding the search space of potential network architectures. As noted at the beginning of Section 3.1, the theoretical formulation for a network architecture $\mathcal{H}(N, S)$ allows for the set of possible neurons N and the set of possible synapses S to be unbounded, effectively representing all possible network architectures. This results in an unbounded search space for potential network architectures, making the practical application of this formulation intractable. In the scope of this thesis, then, the set of possible neurons N and the set of possible synapses S are both bounded by the set of neurons and the set of synapses present in the ancestor network architecture.

The motivation of this thesis is to synthesize compact and efficient network architectures to reduce the memory and computational requirements of high-performance neural networks; to this end, we constrain the set of neurons N_i and the set of synapses S_i for newly synthesized architecture $\mathcal{H}_i$ to the network architecture of the original high-performance neural network. As such, we can denote the original ancestor network architecture as $\mathcal{H}$ (formulated as $\mathcal{H}(N, S)$), and any newly synthesized network $\mathcal{H}_i$ can be formulated as $\mathcal{H}_i(N_i, S_i)$ where $N_i \in N$ and $S_i \in S$ so that the set of neurons in the ancestor architecture N and the set of synapses in the ancestor architecture S serve as the practical constraints for the set of possible neurons and the set of possible synapses, respectively, during the evolutionary synthesis process.

3.3 Architecture via Asexual Evolutionary Synthesis

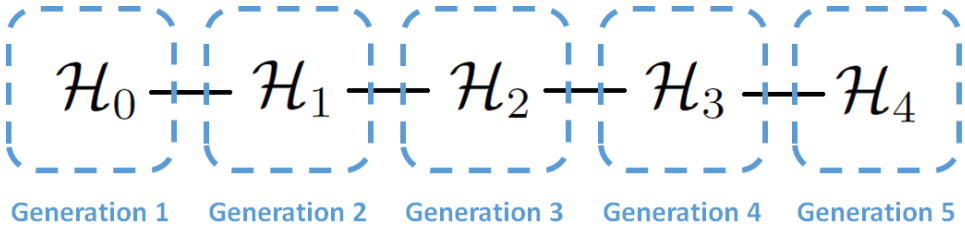


Figure 3.2: Visualization of dependency for the asexual evolutionary synthesis of a new network $\mathcal{H}_g$. In the asexual case shown here, $\mathcal{H}_g$ is solely dependent on the immediately preceding network $\mathcal{H}_{g-1}$. Relative to Figure 3.1, only a single network is synthesized at each generation.

In the case of asexual evolutionary synthesis, the synthesis probability of network $\mathcal{H}_i$ is encoded as $P(\mathcal{H}_i|\mathcal{H}_{i-1}, \mathcal{R}_{g(i)})$, where each newly synthesized network is its own generation and is only dependent on the immediately preceding network (as shown in Figure 3.2). In this case, i indicates the network number as well as the generation number, and i can be dropped in favour of using g to indicate both the generation number and the associated network number.

Similar to the asexual evolutionary synthesis formulation from Shafiee *et al.* [16], the synthesis probability can be rewritten as $P(\mathcal{H}_g|\mathcal{H}_{g-1}, \mathcal{R}_g)$ at generation g, and comprises an environmental factor model $\mathcal{R}_g$ to imitate natural selection via a changing environment for successive generations of networks to adapt to and the synaptic probability $P(S_g|S_{g-1}, W_{g-1})$ to emulate heredity through the generations of networks; that is, the synthesis probability can be formulated as follows:

$$P(\mathcal{H}_g|\mathcal{H}_{g-1}, \mathcal{R}_g) \simeq P(S_g|S_{g-1}, W_{g-1}, \mathcal{R}_g) \tag{3.5}$$

More recently, Shafiee *et al.* [17] introduced a synaptic cluster-driven genetic encoding scheme via a multi-factor synaptic probability model. Synaptic clustering was incorporated to improve memory and storage requirements by promoting the formation of strong synaptic clusters, as well as tailoring the neural network architecture for devices such as embedded GPUs by defining the synaptic cluster to allow for parallel operations. Similar to [17], we define synaptic clusters in this thesis to be a filter (i.e., a collection of convolutional kernels within a convolutional layer), and denote a single synaptic cluster as C within the set of all synaptic clusters $\mathcal{C}$. To incorporate the multi-factor synaptic probability model and different quantitative environmental factor models at the synapse and cluster levels [17], the synthesis probability is modelled as

$$P(\mathcal{H}_g|\mathcal{H}_{g-1}, \mathcal{R}_g) = \prod_{C \in \mathcal{C}} \left[P(s_{g,C}|s_{g-1,C}, w_{g-1,C}, \mathcal{R}_g^c) \cdot \prod_{j \in C} P(s_{g,j}|s_{g-1,j}, w_{g-1,j}, \mathcal{R}_g^s) \right] \tag{3.6}$$

where $P(s_{g,C}|s_{g-1,C}, w_{g-1,C}, \mathcal{R}_g^c)$ represents the probability of synthesis for a given cluster of synapses $s_{g,C}$; that is, $P(s_{g,C}|s_{g-1,C}, w_{g-1,C}, \mathcal{R}_g^c)$ denotes the likelihood that a synaptic cluster $s_{g,C}$ will exist in the network architecture in generation g given $s_{g-1,C}$, the cluster's synaptic indicator in generation $g-1$; $w_{g-1,C}$, the cluster's synaptic strength in generation $g-1$; and the cluster-level environmental factor model $\mathcal{R}_g^c$. Comparably, $P(s_{g,j}|s_{g-1,j}, w_{g-1,j}, \mathcal{R}_g^s)$ represents the likelihood of the existence of synapses $j, *$ within

the synaptic cluster C in generation g given $s_{g-1,j}$, the indicators for synapses $j,*$ in generation $g-1$; $w_{g-1,j}$, the synaptic strength in the previous generation $g-1$ of synapses $j,*$; and the synapse-level environmental factor model $\mathcal{R}_g^s$. Here, $\mathcal{C}$ is the set of possible synaptic clusters for network architecture $\mathcal{H}_\}$. Note that this multi-factor synthesis probability formulation allows for both a cluster-level environmental factor model $\mathcal{R}_g^c$ and a synapse-level environmental factor model $\mathcal{R}_g^s$; this allows for a more flexible computational construct for constraining the simulated availability of environmental resources (as further discussed in Section 3.6). This multi-factor probability model encourages both the persistence of strong synaptic clusters and the persistence of strong synaptic connectivity over successive generations.

In the context of this thesis, a given neuron j with synapses $j,* \in C$ that are zero weight synapses (i.e., $w_{j,*} = 0$ for all associated synapses $j,*$) is still recoverable with the possibility of non-zero weight synapses in future network architecture realizations provided there is at least one non-zero weight synapse in C, i.e., $j,* \in C$ can still recover non-zero weights $w_{j,*}$ provided C contains at least one non-zero weight synapse. When all synapses in a given synaptic cluster C are zero weight synapses, C is removed from the set of possible synaptic clusters $\mathcal{C}$. A neuron j, then, can no longer recover non-zero weight synapses in subsequently synthesized network architectures when all synaptic clusters containing its synapses $j,*$ have been removed from the set of possible synaptic clusters $\mathcal{C}$.

3.4 Architecture via Sexual Evolutionary Synthesis

Evolutionarily speaking, sexual reproduction is thought to have developed in living organisms due to the fact that it favours the survival of groups rather than individuals by allowing for accelerated adaptation to changing environments via the combination of mutations occurring in distinct individuals in a single descendant [69, 70]. Relative to asexual reproduction, sexual reproduction has the potential to accelerate evolution by several orders of magnitude [19], with its effects most prominent in a large population with a high frequency of beneficial mutations. This motivates the idea that the use of sexual reproduction in evolutionary synthesis can accelerate the generation-by-generation development of highly efficient neural networks.

With sexual evolutionary synthesis, a newly synthesized network $\mathcal{H}_i$ is dependent on m previously synthesized networks, i.e., m-parent sexual evolutionary synthesis. In the simplest two-parent case ($m = 2$), $\mathcal{H}_i$ is dependent on $\mathcal{H}_{G_i}$ where $\mathcal{H}_{G_i}$ comprises two networks from the immediately preceding generation (Figure 3.3). However, it is possible to increase

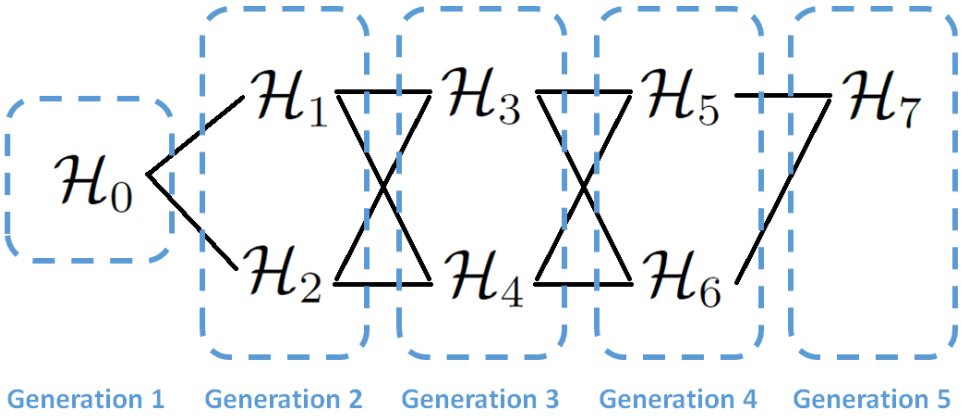


Figure 3.3: Visualization of dependency for the m-parent sexual evolutionary synthesis of a new network $\mathcal{H}_i$. In the two-parent case ($m = 2$), $\mathcal{H}_i$ is dependent on $\mathcal{H}_{G_i}$ where $\mathcal{H}_{G_i}$ is comprised of two networks from the immediately preceding generation. Here, we show the simplest case where it is only necessary to synthesize m networks per generation.

m beyond two (as will be explored in Chapter 5), with multiple network architectures being synthesized at each generation. It is similarly possible to extend $\mathcal{H}_{G_i}$ further back to other generations to have $\mathcal{H}_i$ dependent on more generationally distant ancestor networks (as is shown in Figure 3.1). In this thesis, however, $\mathcal{H}_{G_i}$ is restricted to the immediately preceding generation.

In contrast to the asexual evolutionary synthesis case, which was formulated earlier in this chapter in (3.6) as

$$P(\mathcal{H}_g|\mathcal{H}_{g-1}, \mathcal{R}_g) = \prod_{C \in \mathcal{C}} \left[P(s_{g,C}|s_{g-1,C}, w_{g-1,C}, \mathcal{R}_g^C) \cdot \prod_{j \in C} P(s_{g,j}|s_{g-1,j}, w_{g-1,j}, \mathcal{R}_g^s) \right]$$

where the probability of synthesis was dependent on a cluster's or neuron's synaptic strength of the parent network and the corresponding environmental factor models, we now extend the synthesis probability formulation to the sexual evolutionary synthesis case.

In the case of sexual evolutionary synthesis, more than one network architecture is synthesized at each generation. As such, we introduce a second subscript i to $\mathcal{H}$ to indicate the i^{th} network synthesized at any generation g. For sexual evolutionary synthesis, then,

the synthesis probability for a new network $\mathcal{H}_{g,i}$ can be encoded as

$$P(\mathcal{H}_{g,i}|\mathcal{H}_{G_i},\mathcal{R}_g) = \prod_{C\in\mathcal{C}} \Big[P(s_{g,C}|\mathcal{M}_C^S(s_{G_i,C}),\mathcal{M}_C^W(w_{G_i,C}),\mathcal{R}_g^c)\cdot \prod_{j\in C} P(s_{g,j}|\mathcal{M}_s^S(s_{G_i,j}),\mathcal{M}_s^W(w_{G_i,j}),\mathcal{R}_g^s)\Big]. \tag{3.7}$$

In the asexual evolutionary synthesis case, the probability of synthesis for a given cluster of synapses is dependent on S_{g-1} and W_{g-1}, and the likelihood of existence of a synapse is dependent on $s_{g-1,j}$ and $w_{g-1,j}$. In the sexual evolutionary synthesis case, however, the probability of synthesis for a given cluster of synapses is now dependent on a multiple sets of synapses $s_{G_i,C}$ subject to some synaptic structure cluster-level mating function $\mathcal{M}_C^S(.)$ and multiple sets of weights $w_{G_i,C}$ subject to some synaptic strength cluster-level mating function $\mathcal{M}_C^W(.)$. Similarly, the likelihood of existence of a synapse $(j,*)$ (indicated by $s_{j,*}$) is dependent on multiple possible synapses $s_{G_i,j}$ subject to some synaptic structure synapse-level mating function $\mathcal{M}_s^S(.)$ and multiple possible weights $w_{G_i,j}$ subject to some synaptic strength synapse-level mating function $\mathcal{M}_s^W(.)$. The selection of the mating functions $\mathcal{M}_C(.)$ and $\mathcal{M}_s(.)$ are of particular interest in the sexual evolutionary synthesis process, as they directly impact the viability and diversity of the synthesized networks. Simple mating functions used in the scope of the first part of this thesis are introduced in Chapter 4, and more sophisticated mating functions are introduced in Chapter 5.

3.5 Realization of Genetic Encoding

In this thesis, we use the genetic encoding scheme introduced by Shafiee *et al.* [17]. In both asexual and sexual evolutionary synthesis, the probability of synthesis for a given synaptic cluster $s_{g,C}$ is encoded as:

$$P(s_{g,C}=1|W_{g-1,C}) = \frac{1}{Z}\Big[\exp\Big(\Sigma_{j\in C}\lfloor w_{g-1,j}\rfloor\Big)-1\Big] \tag{3.8}$$
$$P(s_{g,C}=1|w_{g-1,C}) + P(s_{g,C}=0|w_{g-1,C}) = 1$$

where $\lfloor . \rfloor$ encodes a synaptic weight truncation and Z is the normalization factor required to construct a probability distribution, i.e., $P(s_{g,C}=1|w_{g-1,C}) \in [0,1]$. The truncation of synaptic weights lessens the impact of weak synapses in a synaptic cluster.

Similarly, the probability of synthesis for synapses $(j,*)$ associated with neuron j, as

indicated by $s_{g,j}$, within a synaptic cluster C is encoded as:

$$P(s_{g,j} = 1|w_{g-1,j}) = \frac{1}{z}\Big[\exp\Big(w_{g-1,j}z\Big) - 1\Big] \tag{3.9}$$
$$P(s_{g,j} = 1|w_{g-1,j}) + P(s_{g,j} = 0|w_{g-1,j}) = 1$$

where z is a layer-wise normalization factor. This genetic encoding scheme allows for the simultaneous consideration of both inter-synapse relationships and individual synapse strength, and is employed throughout this thesis.

3.6 Environmental Resources During Synthesis

In this section, we present the formulation behind the environmental factor models previously introduced in (3.6). These factor models are the computational construct used the mimic natural selection via the constraint of simulated available environmental resources. In nature, a main environmental factor in encouraging high efficiency during evolution is via the restriction of available resources; one such example is proposed in a study by Moran *et al.* [71] about the eyeless Mexican cavefish. Moran *et al.* hypothesized that the eyeless Mexican cavefish lost its vision system over successive generations due to the high energy cost of its visual system and the low food availability in its subterranean habitat. As such, the loss of its vision system substantially lowered the amount of energy expended on neural tissue, allowing for increased survivability in the food-limited habitat. Here, we aim to encourage highly efficient network architectures during the evolutionary synthesis process by simulating available (or lack thereof) environmental resources using computational constructs, i.e., the environmental factor models.

In evolutionary synthesis, (3.6) models these simulated environmental resources via environmental factor models $\mathcal{R}_g$, using a cluster-level environmental factor model $\mathcal{R}_g^C$ and a synapse-level environmental factor model $\mathcal{R}_g^s$. We use the synaptic cluster existence indicator $s_{g,C}$ and the synaptic existence indicator $s_{g,j}$ from Section 3.1 to denote whether a synaptic cluster or a synapse will be synthesized, respectively.

In this thesis, $\mathcal{R}_g^C$ and $\mathcal{R}_g^s$ are constrained to be between 0 and 1, where a lower environmental factor model is more permissive. As shown in (3.10) and (3.11), an environmental factor model of 0 guarantees that the given cluster or synapse will be synthesized, while an environmental factor model of 1 means that the probability of synthesis of the cluster or synapse is equivalent to the strength of the synaptic cluster or the strength of the synapse, respectively. As $w_{g-1,C}$ and $w_{g-1,j}$ correspond to clusters of or single network

weights, respectively, learned during the neural network training process [1–4], we leverage the synaptic weight truncation and normalization factor from the previous section to ensure that the weight magnitudes used in (3.10) and (3.11) are between 0 and 1. Here, we present the formulation behind the environmental factor models for the asexual evolutionary synthesis case before extending the formulation to multi-parent evolutionary synthesis in Chapter 4 and Chapter 5.

At the cluster level, then, the existence of all clusters $C \in \mathcal{C}$ is determined as:

$$P(s_{g,C} = 1|\mathcal{R}_g^C, s_{g-1,C}, w_{g-1,C}) = \begin{cases} 0 & s_{g-1,C} = 0 \\ 1 - (\mathcal{R}_g^C \cdot (1 - |w_{g-1,C}|)) & otherwise \end{cases} \tag{3.10}$$
$$P(s_{g,C} = 1|\mathcal{R}_g^C, s_{g-1,C}, w_{g-1,C}) + P(s_{g,C} = 0|\mathcal{R}_g^C, s_{g-1,C}, w_{g-1,C}) = 1$$

where $s_{g,C}$ incorporates both the strength of the synapses in a cluster ($w_{g-1,C}$) of the parent network and the cluster-level environmental resources available ($\mathcal{R}_g^C$), subject to the existence of the synaptic cluster as determined by $s_{g-1,C}$.

Similarly at the synapse level, the existence of all synapses is determined as:

$$P(s_{g,j} = 1|\mathcal{R}_g^s, s_{g-1,j}, w_{g-1,j}) = \begin{cases} 0 & s_{g-1,j} = 0 \\ 1 - (\mathcal{R}_g^s \cdot (1 - |w_{g-1,j}|)) & otherwise \end{cases} \tag{3.11}$$
$$P(s_{g,j} = 1|\mathcal{R}_g^s, s_{g-1,j}, w_{g-1,j}) + P(s_{g,j} = 0|\mathcal{R}_g^s, s_{g-1,j}, w_{g-1,j}) = 1$$

where $s_{g,j}$ incorporates both the strength of each synapse ($w_{g-1,j}$) and the synapse-level environmental resources available ($\mathcal{R}_g^s$), subject to the existence of the synapse as determined by $s_{g-1,j}$.

We formulate the environmental resource models separately at the cluster and synapse levels to allow for increased flexibility when applying this computational construct. While existing evolutionary synthesis methods [16, 17] have considered the cluster-level environmental factor model and synapse-level environmental factor model together, having separate environmental factor models allows for future work to explore more complex schemes for constraining environmental resources, e.g., specifically constraining the synapse-level environmental resources to encourage network filter sparsity, or constraining the cluster-level environmental resources more to decrease the number of filters in the network architecture. In the context of this thesis, we similarly assume the cluster-level and synapse-level environmental factor models to be equal [16, 17], and enforce that $\mathcal{R}_g^C = \mathcal{R}_g^s$. As such, the cluster-level and synapse-level environmental factor models, $\mathcal{R}_g^C$ and $\mathcal{R}_g^s$, were varied simultaneously such that $\mathcal{R}_g^C = \mathcal{R}_g^s$ in every experiment. The impact of environmental factor models is explored in Chapter 7.

3.7 Learning Offspring Networks

After the synthesis of a new network architecture $\mathcal{H}_i$ via the inheritance of its synaptic structure and being subject to simulated environmental resources, each newly synthesized architecture $\mathcal{H}_i$ is fully trained using the following training parameters:

- Epochs: 50
- Learning rates:
  - 0.05 (20 epochs)
  - 0.005 (20 epochs)
  - 0.0005 (10 epochs)
- Weight decay: 0.0001
- Batch size: 100

Network $\mathcal{H}_i$ is trained using the same dataset as the original ancestor network architecture. For Chapters 4, 5, 6, and 7, the first generation ancestor networks are initialized using the LeNet-5 architecture [34]. In this thesis, we constrain the training data made available to the offspring network to the data used to train the parent networks. While it is possible (and generally of interest) to investigate introducing new datasets to offspring networks, this thesis focuses on reducing the computational and memory requirements of existing high-performance neural networks rather than investigating the effects of new data on network generalization and robustness.

3.8 Assessing the Architecture

The performance of a given network architecture $\mathcal{H}$ can be most directly assessed via a dataset. In this case, the probability of error $\mathcal{E}$ given a synthesized network architecture and a dataset $\mathcal{D}$ can be evaluated as $P(\mathcal{E}|\mathcal{H}, \mathcal{D})$. Thus, the error $\mathcal{E}$ (or, more commonly, $1 - \mathcal{E}$ for the performance accuracy) of a synthesized network architecture is a key metric used to assess the efficacy of a neural network for a given dataset.

Conversely, the efficiency of the network can be assessed by evaluating the size of the network architecture. This can be done by computing the synaptic network sparsity via

the proportion of $s_{g,C} = 1$ for all $s_{j,k} \in S$ and examining the number of filters in each convolutional layer (though in Chapter 8, we use the more indirect $\frac{w \neq 0}{|W|}\%$ for the sake of comparing against state-of-the-art neural network sparsity methods), and we also propose a more direct measure of network size for this thesis by leveraging network storage size (see Chapter 4 and Chapter 5). Rather than implicitly assessing the amount of computational memory required for a neural network, we explicitly measure each network architecture size in terms of number of bytes on a computer hard drive.

In this thesis, we assess both the performance accuracy and the storage size of each network architecture, and monitor the trends over successive generations. While decreasing storage size indicates a more compact and efficient network architecture, it often corresponds to a drop of performance accuracy as the network loses the complexity required to model the problem space. As such, we also evaluate the performance accuracy of the synthesized network architectures as a function of network storage size to directly assess the trade-off between network size and accuracy.

Having extended the asexual evolutionary synthesis previously used in past evolutionary deep intelligence works [16, 17] to sexual evolutionary synthesis in this chapter, we now assess the efficacy of the proposed framework in the context of 2-parent evolutionary synthesis (Chapter 4) and m-parent evolutionary synthesis (Chapter 5).

Chapter 4

Asexual vs. Sexual Evolutionary Synthesis

In this chapter and the next, we explore the computational construct for network heredity in the evolutionary synthesis framework. As explained in Chapter 2, previous works in evolutionary deep intelligence [16–18] had constrained this construct to asexual evolutionary synthesis where a single offspring network is synthesized at each generation using the one parent network in the previous generation. Inspired by biological motivations for 2-parent sexual reproduction [19], we propose a realization of the sexual evolutionary synthesis process in Chapter 3 as an adaptation of the network synthesis process towards more compact offspring networks. In this chapter, we explore the use of a second parent network during the synthesis process and compare the performance to the single parent asexual evolutionary synthesis case.

We first extend the computational construct for network heredity to 2-parent evolutionary synthesis to begin exploring the effects of sexual evolutionary synthesis when synthesizing offspring network architectures. Thus, we augment the evolutionary deep intelligence scheme to incorporate a second parent network during the process of synthesizing new offspring networks as shown in Figure 4.1. At each generation, two parent networks from the preceding generation are combined via mating policies (as described below) to synthesize new offspring networks containing information from both parent networks to drive network diversity and adaptability by mimicking sexual reproduction.

Recall from (3.6) that for an asexually synthesized network, the synaptic cluster synthesis probability, i.e., $P(s_{g,C}|s_{g-1,C}, w_{g-1,C}, \mathcal{R}_g^C$, and the synapse synthesis probability, i.e., $P(s_{g,j}|s_{g-1,j}, w_{g-1,j}, \mathcal{R}_g^s)$, are conditional on the synaptic architecture and synaptic

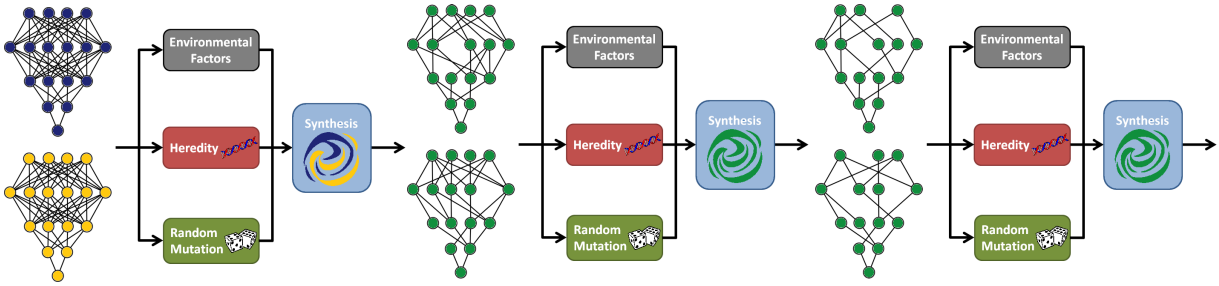


Figure 4.1: The proposed 2-parent evolutionary synthesis process over successive generations as an extension of asexual cluster-driven genetic encoding. The effects of sexual evolutionary synthesis are explored via the incorporation of a second parent network during the synthesis of offspring networks. At each generation, two parent networks from the preceding generation are combined via a mating function to synthesize new offspring networks.

strength of a single parent network in the previous generation and the environmental factor models. To explore the effects of sexual evolutionary synthesis in evolutionary deep intelligence, we reformulate the synthesis probability to combine the cluster and synapse probabilities of two parent networks, e.g., $\mathcal{H}_A$ and $\mathcal{H}_B$ as indicated with subscripts A and B, respectively, during the synthesis of an offspring network via some cluster-level mating functions $\mathcal{M}_C(\cdot)$ and some synapse-level mating functions $\mathcal{M}_s(\cdot)$. Specifically, we rewrite the synthesis probability $P(\mathcal{H}_{g,i}|\mathcal{H}_{G_i}, \mathcal{R}_g)$ in (3.7) as

$$P(\mathcal{H}_{g,i}|\mathcal{H}_A, \mathcal{H}_B, \mathcal{R}_g) = \prod_{C \in \mathcal{C}} \Big[P(s_{g,C}|\mathcal{M}_C^S(s_{A,C}, s_{B,C}), \mathcal{M}_C^W(w_{A,C}, w_{B,C}), \mathcal{R}_g^C) \cdot \prod_{j \in C} P(s_{g,j}|\mathcal{M}_s^S(s_{A,j}, s_{B,j}), \mathcal{M}_s^W(w_{A,j}, w_{B,j}), \mathcal{R}_g^s) \Big] \tag{4.1}$$

where $\mathcal{M}_C^S(.)$ and $\mathcal{M}_C^W$ are the cluster-level mating functions for the synaptic structure and synaptic weights, respectively, and $\mathcal{M}_s^S(.)$ and $\mathcal{M}_s^W$ are the synapse-level mating functions for the synaptic structure and synaptic weights, respectively.

4.1 2-Parent Mating of Deep Neural Networks

In the context of this thesis, we restrict the the parent networks, $\mathcal{H}_A$ and $\mathcal{H}_B$, to the immediately preceding generation; that is, for an offspring $\mathcal{H}_{g,i}$ at generation g, the parent

networks $\mathcal{H}_A$ and $\mathcal{H}_B$ are from generation $g-1$. To inherit the synaptic structure $S_{g,i}$ of $\mathcal{H}_{g,i}$, we propose a simple intersection-based synaptic architecture mating policy, where only synaptic clusters and synapses present in both parent networks $\mathcal{H}_A$ and $\mathcal{H}_B$ are synthesized in $\mathcal{H}_{g,i}$. An intersection-based synaptic architecture mating policy encourages a sparser synaptic architecture in $\mathcal{H}_{g,i}$, as it is unlikely the synaptic architecture of one parent network perfectly overlaps with the other parent network; this results in fewer synaptic clusters and synapses being synthesized in the offspring network. As such, we propose the cluster-level mating function $\mathcal{M}_C^S(.)$ and synapse-level mating function $\mathcal{M}_s^S(.)$ to be as follows:

$$\mathcal{M}_C^S(s_{A,C}, s_{B,C}) = s_{A,C} \cdot s_{B,C} \tag{4.2}$$

$$\mathcal{M}_s^S(s_{A,j}, s_{B,j}) = s_{A,j} \cdot s_{B,j} \tag{4.3}$$

where $s_{A,C}$ and $s_{B,C}$ represent the cluster's synaptic architecture for parent networks $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively. Similarly, $s_{A,j}$ and $s_{B,j}$ represent the existence of a synapse j within cluster C for parent networks $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively.

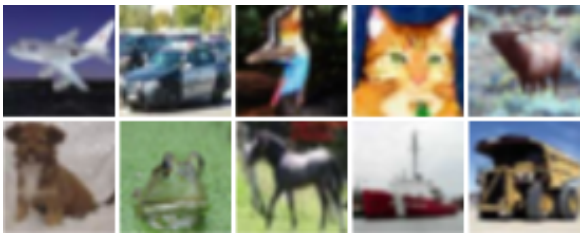
Given the inherited synaptic structure $S_{g,i}$ of $\mathcal{H}_{g,i}$, $\mathcal{H}_{g,i}$ is then subject to simulated environmental resources via $\mathcal{R}_g$ and the synaptic strengths of parent networks $\mathcal{H}_A$ and $\mathcal{H}_B$. Extending (3.10) and (3.11) to the 2-parent evolutionary synthesis case, we propose that the simulated environmental resources can be applied as follows at the cluster level and at the synapse level:

$$P(s_{g,C} = 1 | \mathcal{R}_g^C, s_{G_i,C}, w_{G_i,C}) = \begin{cases} 0 & s_{G_i,C} = 0 \\ 1 - (\mathcal{R}_g^C \cdot (1 - |\mathcal{M}_C^W(w_{G_i,C})|)) & otherwise \end{cases} \tag{4.4}$$

$$P(s_{g,j} = 1 | \mathcal{R}_g^s, s_{G_i,j}, w_{G_i,j}) = \begin{cases} 0 & s_{G_i,j} = 0 \\ 1 - (\mathcal{R}_g^s \cdot (1 - |\mathcal{M}_s^W(w_{G_i,j})|)) & otherwise \end{cases} \tag{4.5}$$

where $\mathcal{M}_C^W(.)$ is the cluster-level synaptic strength mating function and $\mathcal{M}_s^W(.)$ is the synapse-level synaptic strength mating function. From previous evolutionary deep intelligence methods [16, 17], we know that a newly synthesized network also inherits the corresponding synaptic weights from its parent network as an initialization. Thus in the 2-parent case, we propose the synaptic strength mating policies for the purposes of applying

(a) Sample images from the MNIST hand-written digits dataset [72]



(b) Sample images from the CIFAR-10 object classification dataset [73].

Figure 4.2: Sample images from the MNIST hand-written digits dataset and the CIFAR-10 object classification dataset.

the simulated environmental resources $\mathcal{R}_g$ to the newly synthesized network $\mathcal{H}_{g,i}$ to be

$$\mathcal{M}_C^W(w_{A,C}, w_{B,C}) = \frac{w_{A,C} + w_{B,C}}{2} \tag{4.6}$$

$$\mathcal{M}_s^W(w_{A,j}, w_{B,j}) = \frac{w_{A,j} + w_{B,j}}{2} \tag{4.7}$$

where $w_{A,C}$ and $w_{B,C}$ represent the cluster's synaptic strength for parent networks $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively. Similarly, $w_{A,j}$ and $w_{B,j}$ represent the synaptic strength of a synapse j within cluster c for parent networks $\mathcal{H}_A$ and $\mathcal{H}_B$, respectively.

4.2 Experimental Setup

The asexual and sexual evolutionary synthesis of deep neural networks were performed over multiple generations, and the effects of sexual evolutionary synthesis relative to asexual evolutionary synthesis were explored using the full MNIST [72] hand-written digits and full CIFAR-10 [73] object classification datasets with the first generation ancestor networks trained using the LeNet-5 architecture [34]. Figure 4.2 shows sample images from the MNIST and CIFAR-10 datasets.

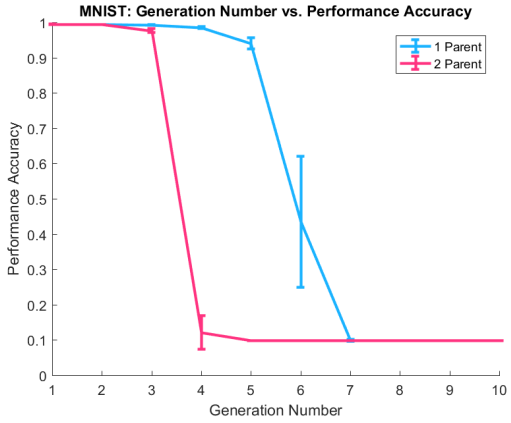
Similar to Shafiee *et al.*'s work [17], we designed the environmental factor models $\mathcal{R}_g^C$ and $\mathcal{R}_g^s$ to be 70%, thus decreasing the total number of clusters and synapses in the newly synthesized network relative to its parent networks in the previous generation; this allows increasingly more compact feature representations and for the synthesized deep

neural networks to become progressively more efficient in the successive generations while minimizing any loss in accuracy. As in Shafiee *et al.*'s work [17], each filter (i.e., collection of convolution kernels) was considered as a synaptic cluster in the multi-factor synapse probability model.

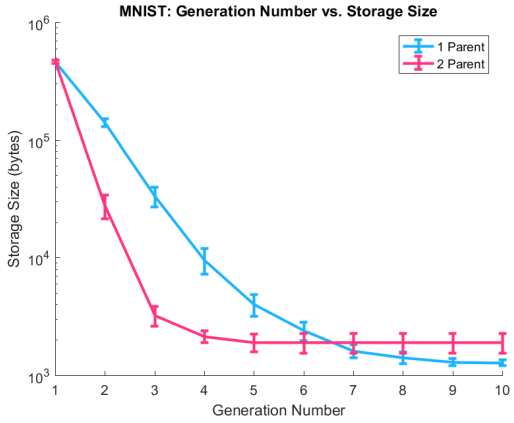
At each generation, the neural network with the highest validation accuracy was selected as representative of that generation, and was assessed for network performance and efficiency. In particular, we use validation accuracy as a metric for network performance accuracy and leverage network storage size (in bytes) as a direct measure of network efficiency. Each experiment was performed 5 times, and error bars showing the standard deviation of the synthesized networks at each generation are plotted at each data point. The difference in validation accuracy and network storage size trends between asexual evolutionary synthesis and sexual evolutionary synthesis is assessed for statistical significance via a multivariate analysis of variance (MANOVA) test [74], the generalization of the analysis of variance (ANOVA) method [75]. There are, however, notable limitations of the MANOVA method, as assessing all the synthesized networks include networks that are fundamentally expected to not be significantly different; these networks occur in the earliest generations when the synthesized networks are still similar to the common ancestor network, and in the last generations when the synthesized networks are no longer complex enough to model the dataset (i.e., 10% performance accuracy representative of random guessing). As a result, networks synthesized using asexual evolutionary synthesis and sexual evolutionary synthesis are also assessed using a paired t-test [76], where the theoretical storage size of the networks are compared at a selected performance threshold via linear interpolation between the nearest data points.

4.3 Results

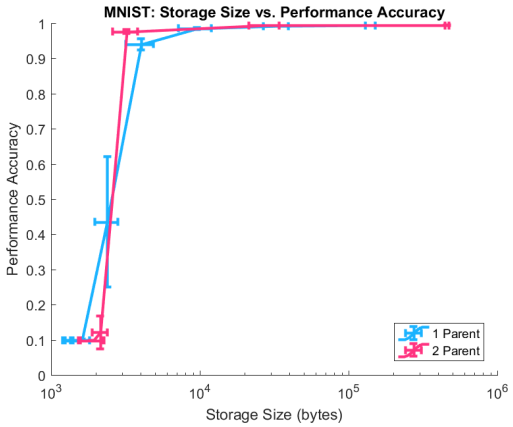
In this chapter, we explore the use of 2-parent sexual evolutionary synthesis during the synthesis of offspring networks, and compare these networks to the network architectures synthesized via asexual evolutionary synthesis. At each generation, the network validation accuracy and the network storage size (in bytes) were evaluated. Figure 4.3 and Figure 4.4 show the validation accuracy and network storage size of networks synthesized using asexual and sexual evolutionary synthesis as a function of generation number, and evaluated on the MNIST and CIFAR-10 datasets, respectively. Note that in both the asexual and sexual evolutionary synthesis cases, there is a trade-off between performance accuracy and architectural efficiency, i.e., validation accuracy decreases as the network storage size decreases; this trade-off is also shown in Figure 4.3 and Figure 4.4.



(a) MNIST validation accuracy.



(b) MNIST network storage size.



(c) MNIST accuracy vs. storage size.

Figure 4.3: MNIST average validation accuracy and average neural network storage size vs. generation number (top row) for synthesized networks for networks synthesized using asexual (blue) and sexual (pink) evolutionary synthesis, with error bars showing the standard deviation over 5 runs. Note that while sexual evolutionary synthesis results in networks that more rapidly decrease in accuracy and storage, these networks also show potential for a better the accuracy vs. storage size trade-off (bottom row), as shown by the pink data points close to the top left corner of the scatter plot.

Figure 4.3 shows the MNIST performance accuracy and network efficiency for networks synthesized using asexual (blue) and sexual (pink) evolutionary synthesis. Figure 4.3 (a) shows the validation accuracy and Figure 4.3 (b) shows the storage size in bytes for networks synthesized using asexual and sexual evolutionary synthesis. For this experiment, the original fully-trained ancestor network (generation 1) had a performance accuracy of approximately 99.5% with 143,136 synapses and 7,200 kernels (corresponding to a 1-channel input LeNet architecture [34]). As expected, both asexual and sexual evolutionary synthesis produced networks that decrease in performance accuracy and storage size over successive generations; however, note that sexual evolutionary synthesis produced a network with a $1-2\%$ drop in validation accuracy and a decrease in storage size of approximately two orders of magnitude at generation 3 while asexual evolutionary synthesis produced a similar network with a higher drop in validation accuracy ($5-6\%$) and slightly smaller decrease in storage size at generation 5.

Figure 4.3 (c) shows the trade-off between performance accuracy and network storage size for networks synthesized using asexual and sexual evolutionary synthesis and evaluated using the MNIST dataset. The best networks are those closest to the top-left corner, indicating a high performance accuracy for a smaller network storage size. While networks synthesized using asexual (blue) and sexual (pink) evolutionary synthesis generally show similar trade-offs in performance accuracy and network efficiency, note the two data points near the top left corner of Figure 4.3 (c); interestingly, these two data points correspond to the generation 3 network produced via sexual evolutionary synthesis (pink) and the generation 5 network produced via asexual evolutionary synthesis (blue). Note that the network produced via sexual evolutionary synthesis has both a higher performance accuracy and a smaller storage size, indicating the potential of sexual evolutionary synthesis to produce more efficient and compact networks.

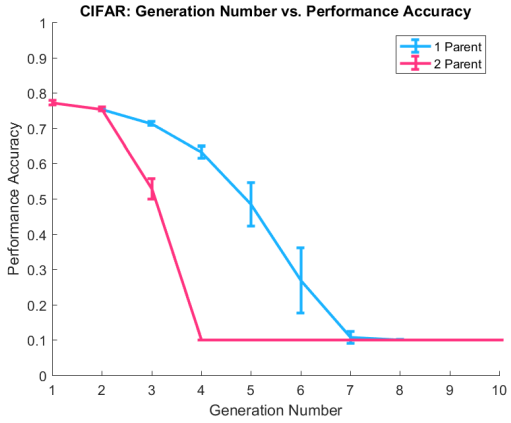
To assess the statistical significance of 2-parent sexual evolutionary synthesis relative asexual evolutionary synthesis across multiple runs, we use the multivariate generalization of the analysis of variance (MANOVA) method to assess the performance accuracy and network storage size as dependent variables. For this experiment with MNIST, the MANOVA statistical significance analysis resulted in a p-value of 0.0405, indicating that there is statistical significance (based on a $p < 0.05$ significance criterion) between networks synthesized using asexual evolutionary synthesis and networks synthesized using 2-parent sexual evolutionary synthesis. We also assess statistical significance via a paired t-test by comparing the linearly interpolated network storage sizes at a performance threshold of 95% accuracy; this threshold was selected for this experiment as representative of the performance accuracy of the best networks (as per Figure 4.3 (c)). The paired t-test resulted in a p-value of 0.0448, similarly indicating that there is statistical significance (based on

a $p < 0.05$ significance criterion) between networks synthesized using asexual evolutionary synthesis and networks synthesized using 2-parent sexual evolutionary synthesis.

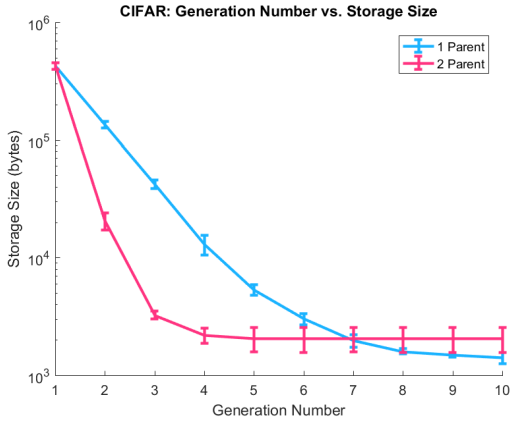
Figure 4.4 shows the CIFAR-10 performance accuracy and network efficiency for networks synthesized using asexual (blue) and sexual (pink) evolutionary synthesis. Figure 4.4 (a) shows the validation accuracy and Figure 4.4 (b) shows the storage size in bytes for networks synthesized using asexual and sexual evolutionary synthesis. For this experiment, the original fully-trained ancestor network (generation 1) had a performance accuracy of approximately 77.5% with 144,736 synapses and 7,264 kernels (corresponding to a 3-channel input LeNet architecture [34]). Similar to the MNIST dataset, both asexual and sexual evolutionary synthesis produced networks that decrease in performance accuracy and storage size over successive generations. Sexual evolutionary synthesis produced a network with an approximately 2% drop in validation accuracy and a decrease in storage size of approximately one order of magnitude at generation 2; in comparison, asexual evolutionary synthesis produced a network with a similar drop in validation accuracy and notably smaller decrease in storage size at generation 2, and a network with a higher drop in validation accuracy (approximately 6%) and slightly smaller decrease in storage size at generation 3.

Figure 4.4 (c) shows the trade-off between performance accuracy and network storage size for networks synthesized using asexual and sexual evolutionary synthesis and evaluated using the CIFAR-10 dataset. The best networks are those closest to the top-left corner, indicating a high performance accuracy for a smaller network storage size. While networks synthesized using asexual (blue) and sexual (pink) evolutionary synthesis show similar trade-offs between performance accuracy and network efficiency, the two pink data points near the top left corner of Figure 4.4 (c) (corresponding to the networks produced via sexual evolutionary synthesis at generation 2 and 3) have both higher performance accuracy and smaller storage size relative to the five comparable blue data points (corresponding to the networks produced via asexual evolutionary synthesis at generation 2 through 6). As with the MNIST dataset, this further indicates the potential of sexual evolutionary synthesis to produce more efficient and compact networks.

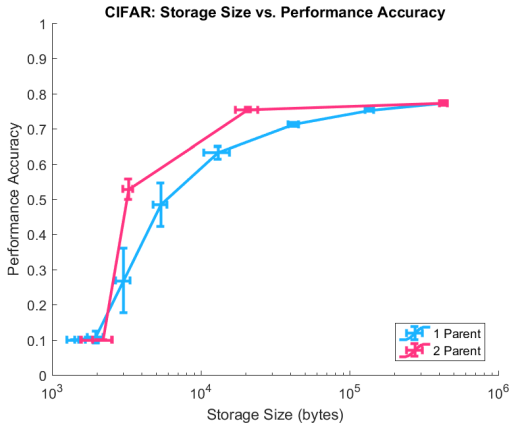
We similarly assess the statistical significance of 2-parent sexual evolutionary synthesis relative asexual evolutionary synthesis across multiple runs for the CIFAR-10 dataset using the multivariate generalization of the analysis of variance method (MANOVA) with performance accuracy and network storage size as dependent variables. The MANOVA statistical significance analysis resulted in a p-value of 0.0556, which could be considered as not statistically significant given a statistical significance criterion of $p < 0.05$. As with the MNIST dataset, we also assess the statistical significance using the paired t-test by comparing the linearly interpolated network storage sizes at a performance threshold of



(a) CIFAR validation accuracy.



(b) CIFAR network storage size.



(c) CIFAR accuracy vs. storage size.

Figure 4.4: CIFAR-10 average validation accuracy and average neural network storage size vs. generation number (top row) for synthesized networks for networks synthesized using asexual (blue) and sexual (pink) evolutionary synthesis, with error bars showing the standard deviation over 5 runs. Note that while sexual evolutionary synthesis results in networks that more rapidly decrease in accuracy and storage, these networks also show potential for a better the accuracy vs. storage size trade-off (bottom row), as shown by the pink data points close to the top left corner of the scatter plot.

70% accuracy; this threshold was similarly selected as representative of the performance accuracy of the best synthesized networks (as per Figure 4.4 (c)). Unlike the MANOVA analysis, the paired t-test resulted in a p-value of 7.4454×10^{-5} and indicates that there is statistical significance (based on a $p < 0.05$ significance criterion) between networks synthesized using asexual evolutionary synthesis and networks synthesized using 2-parent sexual evolutionary synthesis at the crucial trade-off point between performance accuracy and storage size.

Notice that in both the experiments with MNIST and CIFAR-10 datasets, fewer generations were required to reach similar levels of network performance (e.g., similar drops in validation accuracy) using sexual evolutionary synthesis relative to asexual evolutionary synthesis. This is likely due to the intersection-based synaptic architecture mating policy proposed in (4.2) and (4.3). With two parent network architectures, the synaptic architecture for a newly synthesized network only contains synapses that are present in both parent architectures; this results in a synaptic architecture that is sparser in nature than if the synthesized network only inherited from a single parent network architecture. Lastly, it is worth noting that the MNIST dataset allows for larger increases in architectural efficiency relative to the CIFAR-10 dataset (i.e., a larger decrease in network storage size for a comparable drop in validation accuracy); this is likely due to the simplicity of the MNIST dataset (1-channel images of handwritten digits) relative to the CIFAR-10 dataset (3-channel natural images of objects).

4.4 Summary

An extension of asexual evolutionary synthesis, the use of 2-parent sexual evolutionary synthesis during the synthesis of offspring networks for evolutionary deep intelligence was explored in this chapter. The efficacy of 2-parent evolutionary synthesis was examined by comparing networks synthesized via asexual and sexual evolutionary synthesis using both the MNIST and CIFAR-10 datasets. The 2-parent sexual evolutionary synthesis experiment resulted in the synthesis of more efficient deep neural networks at earlier generations relative to asexual evolutionary synthesis. Showing clear trends in increasing architecture efficiency over successive generations for both asexual and sexual evolutionary synthesis, 2-parent sexual evolutionary synthesis results in a networks with better accuracy to storage size trade-offs relative to the asexual case for both the MNIST (p-value of 0.0405 for MANOVA analysis, p-value of 0.0448 for paired t-test at 95% performance accuracy threshold) and CIFAR-10 (p-value of 0.0556 for MANOVA analysis, p-value of 7.4454×10^{-5} for paired t-test at 70% performance accuracy threshold) datasets. In par-

ticular, the MNIST dataset showed more obvious gains in network efficiency compared to the CIFAR-10 dataset, likely due to the relative simplicity of the MNIST dataset. The results of both datasets indicate that the use of 2-parent evolutionary synthesis allows for the synthesis of increasingly efficient network architectures at relatively earlier generations, accelerating the rate at which successive offspring networks increase in efficiency.

Chapter 5

m-Parent Evolutionary Synthesis

In this chapter, we further explore the computational construct for network heredity in the evolutionary synthesis framework and generalize the 2-parent evolutionary synthesis from Chapter 4 to m-parent evolutionary synthesis in varying population sizes. Having considered the computational construct for network heredity in the 2-parent evolutionary synthesis case in Chapter 4, we now generalize this construct to inherit from any m parent networks during offspring synthesis. In this chapter, we investigate the effects of m-parent evolutionary synthesis in varying population sizes. Two main concepts are explored here: i) the effect of varying the number of parent networks used when synthesizing a new network, and ii) the effect of varying population size where there are multiple potential parent network candidates. The evolutionary deep intelligence scheme from Chapter 4 is generalized to use m parents during the evolutionary synthesis process (as shown in Figure 5.1). At each generation, m parent networks from the preceding generation are combined via a mating function to synthesize new offspring networks containing information from all parent networks to drive network diversity and adaptability.

We use the generalized synthesis probability $P(\mathcal{H}_{g,i}|\mathcal{H}_{G_i}, \mathcal{R}_g)$ for m-parent evolutionary synthesis. This m-parent synthesis probability was originally detailed in (4.1), where the cluster and synapse probabilities of m parent networks $\mathcal{H}_{g,i}$ are combined during the synthesis of an offspring network via some cluster-level mating function $\mathcal{M}_C(\cdot)$ and some synapse-level mating function $\mathcal{M}_s(\cdot)$:

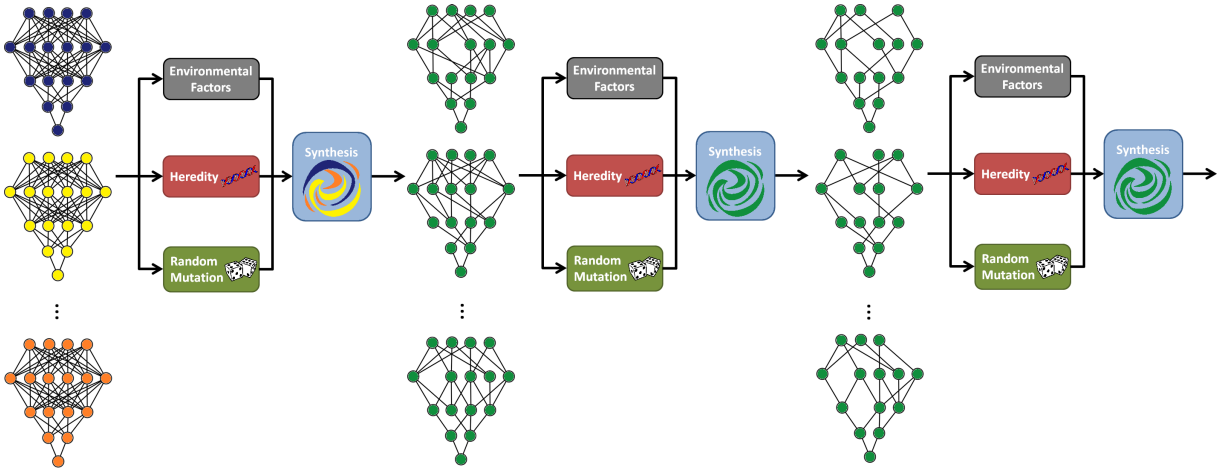


Figure 5.1: A generalization of Figure 4.1, the proposed m-parent evolutionary synthesis process over successive generations. The effects of m-parent evolutionary synthesis are explored via the combination of m parent networks during the synthesis of offspring networks. At each generation, m parent networks from the preceding generation are combined via a mating function to synthesize new offspring networks.

$$P(\mathcal{H}_{g,i}|\mathcal{H}_{G_i}, \mathcal{R}_g) = \prod_{C \in \mathcal{C}} \Big[P(s_{g,C}|\mathcal{M}_C^S(s_{G_i,C}), \mathcal{M}_C^W(w_{G_i,C}), \mathcal{R}_g^C) \cdot \prod_{j \in C} P(s_{g,j}|\mathcal{M}_s^S(s_{G_i,j}), \mathcal{M}_s^W(w_{G_i,j}), \mathcal{R}_g^s) \Big] \tag{5.1}$$

where $\mathcal{M}_C^S(.)$ and $\mathcal{M}_C^W$ are the cluster-level mating functions for the synaptic structure and synaptic weights, respectively, and $\mathcal{M}_s^S(.)$ and $\mathcal{M}_s^W$ are the synapse-level mating functions for the synaptic structure and synaptic weights, respectively. In this chapter, we extend the idea of 2-parent sexual evolutionary synthesis from Chapter 4 to incorporate m parent networks during the network synthesis process.

5.1 m-Parent Mating of Deep Neural Networks in Varying Population Sizes

As in Chapter 4, we limit $\mathcal{H}_{G_i}$ to networks in the immediately preceding generation, i.e., for a newly synthesized network $\mathcal{H}_{g,i}$ at generation g, the m parent networks in $\mathcal{H}_{G_i}$ are

from generation $g-1$. In this chapter, we first generalize the proposed mating functions (4.2) and (4.3) by extending the idea to m parent networks during evolutionary synthesis, and rewrite the synaptic architecture cluster-level and synapse-level mating functions to be as follows:

$$\mathcal{M}_C^S(s_{G_i,C}) = \prod_{k=1}^{m} s_{k,C} \tag{5.2}$$

$$\mathcal{M}_s^S(s_{G_i,j}) = \prod_{k=1}^{m} s_{k,j} \tag{5.3}$$

where $s_{k,C}$ represents the cluster's synaptic architecture for the k^{th} parent network $\mathcal{H}_k \in \mathcal{H}_{G_i}$. Similarly, $s_{k,j}$ represents the existence of a synapse j within cluster C for the k^{th} parent network $\mathcal{H}_k \in \mathcal{H}_{G_i}$.

As demonstrated in Chapter 4, an intersection-based synaptic architecture mating policy results in the synthesis of more efficient deep neural network architectures at earlier generations relative to asexual evolutionary synthesis. Note that Figure 4.3 and Figure 4.4 show that 2-parent evolutionary synthesis only allows for two to three generations of viable (i.e., 2% drop in performance accuracy) network architectures, indicating that a simple intersection-based synaptic architecture mating policy is unlikely to produce any viable network architectures at all with additional parent networks.

In this chapter, we relax this intersection-based synaptic architecture mating policy via the introduction of a cluster-level percent of population parameter χ to control the proportion of parent network architectures that must contain any given cluster during evolutionary synthesis. This introduces a parameter-based lower bound on the subset of parent networks with existing architectural clusters. In Chapter 4, we employed an intersection-based mating policy where a cluster was synthesized only if the cluster existed within all m parent networks. To allow for more flexibility within the m-parent evolutionary synthesis process, we propose χ so that only synaptic clusters that exist within the specified proportion of parent networks are synthesized, and the intersection-based policy previously used in Chapter 4 can be achieved using $\chi = 1$.

As such, we rewrite the synaptic architecture cluster-level mating function (5.2) as

$$\mathcal{M}_C^S(s_{G_i,C}) = \begin{cases} 1 & \frac{1}{m}\sum_{k=1}^{m} s_{k,C} \geq \chi \\ 0 & otherwise \end{cases} \tag{5.4}$$

where χ is the proportion of parent network architectures that must contain any given synaptic cluster during evolutionary synthesis for the newly synthesized network to inherit the cluster. The synapse-level mating function shown in (5.3) remains the same.

Given the inherited synaptic structure $S_{g,i}$ of $\mathcal{H}_{g,i}$, $\mathcal{H}_{g,i}$ is then subject to simulated environmental resources via $\mathcal{R}_g$ and the synaptic strengths of parent networks $\mathcal{H}_{G_i}$. Using (4.4) and (4.5) from Chapter 4, the simulated environmental resources are applied at the cluster-level and at the synapse-level:

$$P(s_{g,C} = 1|\mathcal{R}_g^C, s_{G_i,C}, w_{G_i,C}) = \begin{cases} 0 & s_{G_i,C} = 0 \\ 1 - (\mathcal{R}_g^C \cdot (1 - |\mathcal{M}_C^W(w_{G_i,C})|)) & otherwise \end{cases}$$

$$P(s_{g,j} = 1|\mathcal{R}_g^s, s_{G_i,j}, w_{G_i,j}) = \begin{cases} 0 & s_{G_i,j} = 0 \\ 1 - (\mathcal{R}_g^s \cdot (1 - |\mathcal{M}_s^W(w_{G_i,j})|)) & otherwise \end{cases}$$

where $\mathcal{M}_C^W(.)$ is the cluster-level synaptic strength mating function and $\mathcal{M}_s^W(.)$ is the synapse-level synaptic strength mating function.

In contrast to Chapter 4, we now modify the synaptic strength mating policies to incorporate the effects of the cluster-level percent of population parameter χ when applying the simulated environmental resources $\mathcal{R}_g$ to the newly synthesized network $\mathcal{H}_{g,i}$:

$$\mathcal{M}_C^W(w_{G_i,C}) = \frac{1}{m} \sum_{k=1}^{m} s_{k,C} \cdot w_{k,C} \tag{5.5}$$

$$\mathcal{M}_s^W(w_{G_i,j}) = \frac{1}{m} \sum_{k=1}^{m} s_{k,j} \cdot w_{k,j} \tag{5.6}$$

where $w_{G_i,C}$ represents the cluster's synaptic strength for parent networks $\mathcal{H}_{G_i}$, and $w_{G_i,j}$ represents the synaptic strength of a synapse j within cluster C for parent networks $\mathcal{H}_{G_i}$.

5.2 Experimental Setup

The m-parent evolutionary synthesis of deep neural networks was performed over several generations for $m = 1$, 2, 3, and 5, and the effects of m-parent evolutionary synthesis in varying population sizes of three, five, and eight synthesized networks per generation were explored using a 10% subset of the MNIST [72] hand-written digits dataset (a sample of

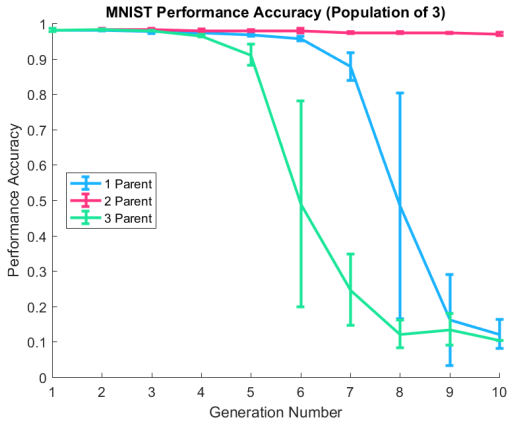
which is shown in Figure 4.2 in Chapter 4) with the first generation ancestor networks trained using the LeNet-5 architecture [34]. In this study, m parents were randomly selected (without replacement) from the population of networks in the immediately preceding generation, and we select the cluster-level percent of population parameter χ to be 50% for this experiment. Similar to Chapter 4, we assess the statistical significance of our results using a multivariate analysis of variance (MANOVA) test, as well as a paired t-test where the theoretical storage size of the networks are compared at a selected performance threshold via linear interpolation between the nearest data points.

As in Chapter 4, we designed the environmental factor models $\mathcal{R}_g^C$ and $\mathcal{R}_g^s$ to enforce that an offspring deep neural network is limited to a fraction of the total number of synapses available in the previous generation, allowing for the synthesized deep neural networks to become progressively more efficient in the successive generations while minimizing any loss in accuracy. In Chapter 4, there were only a few generations of viable network architectures before performance accuracy dropped too low; as a result, we leverage both a more relaxed synaptic architecture mating policy (as detailed in the previous section) as well as a less aggressive environmental factor model. In this chapter, we select an environmental factor model of 50% (relative to the 70% used in Chapter 4); the environmental factor models are more thoroughly explored in Chapter 7. As in Shafiee *et al.*'s work [17] and Chapter 4, each filter (i.e., collection of kernels) was considered as a synaptic cluster in the multi-factor synapse probability model, and both the synaptic efficiency and cluster efficiency were assessed along with testing accuracy.

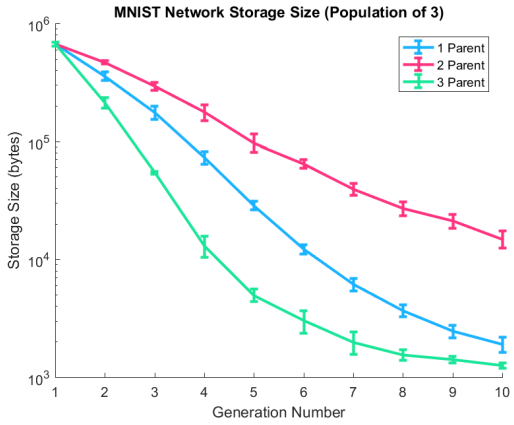
5.3 Results

In this chapter, the effects of m-parent evolutionary synthesis in varying population sizes were investigated using m = 1, 2, 3, and 5, and population sizes of three, five, and eight synthesized networks per generation. At each generation, the network validation accuracy and the network storage size (in bytes) were evaluated. As was the case in Chapter 4, there is generally a trade-off between performance accuracy and network storage size, i.e., accuracy decreases as the network storage size decreases. For all experiments, 10% of the MNIST dataset was used for training and the original fully-trained ancestor network (generation 1) had a testing accuracy of 98% with 143,136 synapses and 7,200 kernels (corresponding to a 1-channel input LeNet architecture [34]).

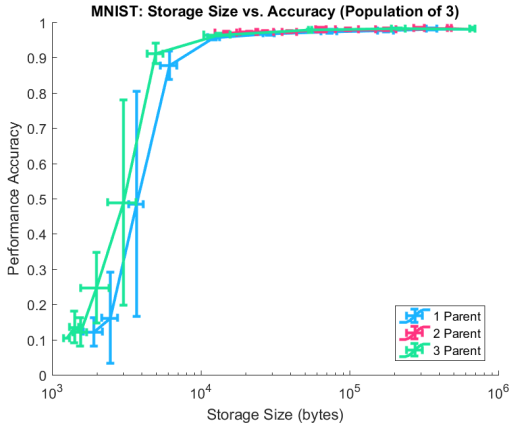
Figure 5.2, Figure 5.3, and Figure 5.4 show the MNIST validation accuracy and network storage size for networks synthesized via evolutionary synthesis using one parent (blue), two parents (pink), three parents (green), and five parents (black, not shown in Figure 5.2) as a



(a) MNIST validation accuracy.

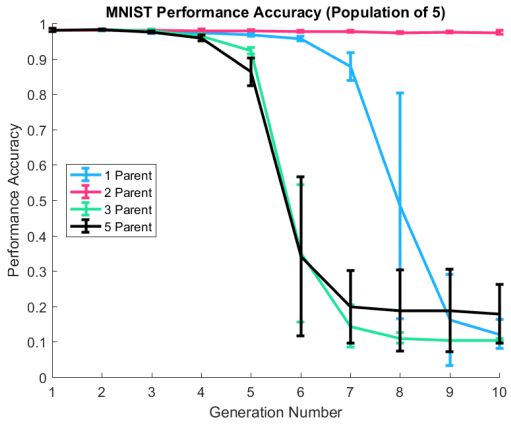


(b) MNIST network storage size.

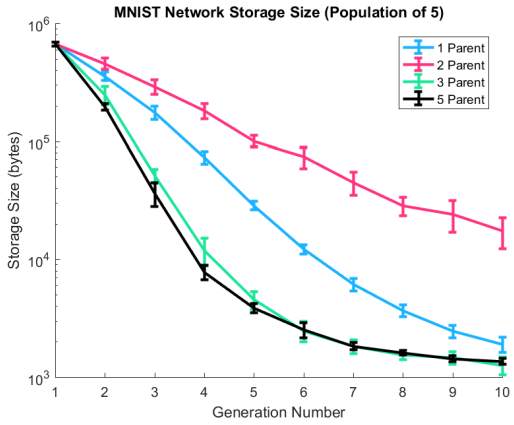


(c) MNIST accuracy vs. storage size.

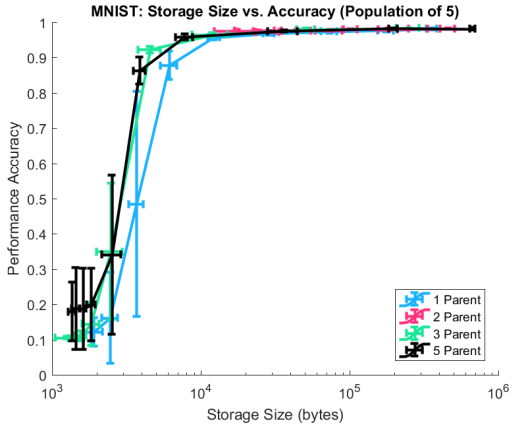
Figure 5.2: MNIST average validation accuracy and average neural network storage size vs. generation number (top row) for networks synthesized using one (blue), two (pink), and three (green) parents, with error bars showing the standard deviation over 5 runs. These plots show the effects of m-parent evolutionary synthesis for a population of three synthesized networks per generation. Like the five network population case (Figure 5.3) and the eight network population case (Figure 5.4), increasing the number of parent networks results in improved accuracy vs. storage size trade-off (bottom row). Notice that the 2-parent (pink) case appears to drop in both accuracy and storage size more slowly than the other cases.



(a) MNIST validation accuracy.

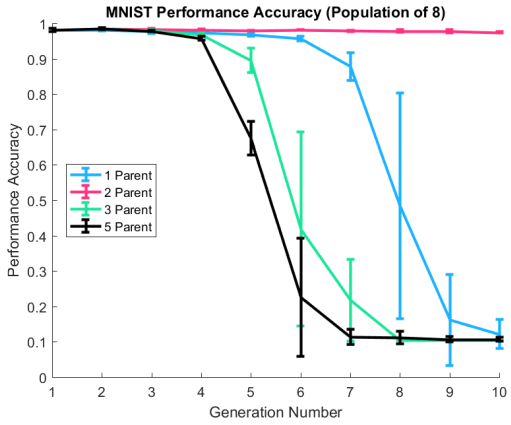


(b) MNIST network storage size.

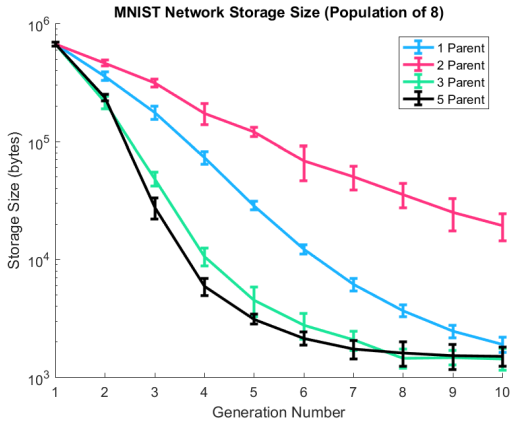


(c) MNIST accuracy vs. storage size.

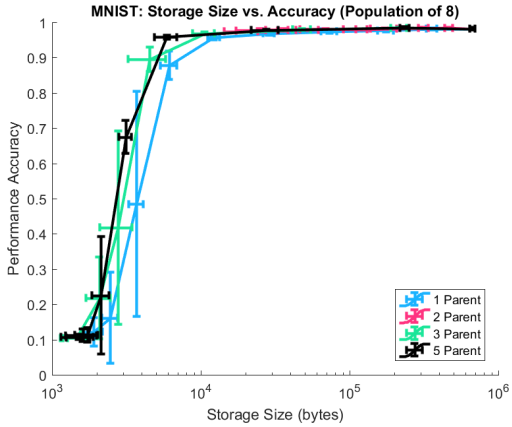
Figure 5.3: MNIST average validation accuracy and average neural network storage size vs. generation number (top row) for networks synthesized using one (blue), two (pink), three (green), and five (black) parents, with error bars showing the standard deviation over 5 runs. These plots show the effects of m-parent evolutionary synthesis for a population of five synthesized networks per generation. Networks synthesized using three (green) and five (black) parents show the best accuracy vs. storage size trade-off (bottom row), with data points closest to the top left corner of the scatter plot. As in Figure 5.2, the 2-parent (pink) case appears to drop in both accuracy and storage size more slowly than the other cases.



(a) MNIST validation accuracy.



(b) MNIST network storage size.



(c) MNIST accuracy vs. storage size.

Figure 5.4: MNIST average validation accuracy and average neural network storage size vs. generation number (top row) for networks synthesized using one (blue), two (pink), three (green), and five (black) parents, with error bars showing the standard deviation over 5 runs. These plots show the effects of m-parent evolutionary synthesis for a population of eight synthesized networks per generation. Similar to the five network population case (shown in Figure 5.3), networks synthesized using three (green) and five (black) parents show the best accuracy vs. storage size trade-off (bottom row). As in Figure 5.2 and Figure 5.3, the 2-parent (pink) case appears to drop in both accuracy and storage size more slowly than the other cases.

function of generation number, and evaluated on the 10% subset of the MNIST dataset for populations of three, five, and eight networks per generation, respectively. Figure 5.2 (a), Figure 5.3 (a), and Figure 5.4 (a) show the validation accuracy for networks synthesized using m-parent evolutionary synthesis in a population of three, five, and eight networks per generation. Similarly, Figure 5.2 (b), Figure 5.3 (b), and Figure 5.4 (b) show the neural network storage size for networks synthesized using m-parent evolutionary synthesis in a population of three, five, and eight networks per generation. Note that Figure 5.2 only shows networks synthesized using one, two, and three parent networks (but not five) due to having only a population size of three networks at each generation, and therefore having insufficient networks at each generation for 5-parent evolutionary synthesis.

In all three figures, 3-parent and 5-parent evolutionary synthesis produced networks with a $1 - 3\%$ drop in validation accuracy and a decrease in model storage size of approximately two orders of magnitude at generation 4. Conversely, 1-parent evolutionary synthesis produced networks of comparable performance accuracy with a decrease in storage size of approximately one and a half orders of magnitude at generation 6. This indicates that increasing the number of parents during evolutionary synthesis can allow for the synthesis of more efficient network architectures with minimal loss in testing accuracy. While there is generally a trend of faster decrease in both performance accuracy and network storage size as the number of parent networks increase (i.e., validation accuracy and storage size decrease at an earlier generation number), the 2-parent evolutionary synthesis case appears to be an anomaly in that the validation accuracy has minimal to no decrease over 10 generations while the network storage size also decreases more gradually. This will be investigated in Section 5.4.

Figure 5.2 (c), Figure 5.3 (c), and Figure 5.4 (c) shows the trade-off between performance accuracy and network storage size for networks produced via evolutionary synthesis using one parent (blue), two parents (pink), three parents (green), and five parents (black, not shown in Figure 5.2). As in Chapter 4, the best networks are those closest to the top-left corner, indicating a high performance accuracy for a smaller network storage size. Networks synthesized via 3-parent (green) and 5-parent (black) evolutionary synthesis appear to have better trade-offs between accuracy and storage size relative to 1-parent evolutionary synthesis (blue). In addition, the anomaly with the 2-parent evolutionary synthesis case can be seen in the pink data points along the top of the plot. Note that the trend of these data points do not appear to have progressed sufficiently to where the network architectures are at an optimal trade-off between accuracy and storage size (i.e., close to the top left corner), implying that the network mating in the 2-parent evolutionary synthesis case is not reducing the number of synaptic clusters and/or synapses at a rate comparable to 1-parent, 3-parent, or 5-parent evolutionary synthesis over the 10 generations.

Table 5.1: Statistical significance assessment for networks synthesized using one, two, three, and five parent networks in population sizes of three, five, and eight networks per generation over 5 runs and trained using 10% of the MNIST dataset. We use the MANOVA statistical significance analysis method, and present the p-values resulting from both overall and one-to-one comparisons (statistical significance criterion of $p < 0.05$).

	Network Population Size (Per Generation)		
	Three (p-value)	**Five (p-value)**	**Eight (p-value)**
1-Parent & 2-Parent	7.947×10^{-5}	7.706×10^{-5}	6.754×10^{-5}
1-Parent & 3-Parent	0.1235	0.0624	0.0849
1-Parent & 5-Parent	N/A	0.0953	0.0148
3-Parent & 5-Parent	N/A	0.9122	0.7773
Overall	0.0000	0.0000	0.0000
Overall (excluding 2-parent)	0.1235	0.1478	0.0626

As in Chapter 4, we assess the statistical significance of the networks synthesized using 1-parent, 2-parent, 3-parent, and 5-parent evolutionary synthesis in varying population sizes via the multivariate analysis of variance (MANOVA) method [74]. Table 5.1 presents the p-values resulting from both overall (i.e., m-parent evolutionary synthesis for a given population size) and one-to-one comparisons (e.g., comparing 1-parent asexual evolutionary synthesis and 3-parent sexual evolutionary synthesis in a network population size of 5 networks per generation). For the overall MANOVA analysis (second last row), the p-value for all three population sizes is 0, indicating that that there is clear evidence that the group (i.e., various m-parent evolutionary synthesis methods) means are not at the same point in space and, thus, are significantly different.

We also directly compare the networks synthesized using 2-parent, 3-parent, and 5-parent (where available) sexual evolutionary synthesis to the networks synthesized via 1-parent asexual evolutionary synthesis via the statistical significance MANOVA method, as shown in the first three rows of Table 5.1. While a MANOVA analysis of 3-parent and 5-parent evolutionary synthesis compared with 1-parent evolutionary synthesis results in relatively low p-values (e.g., < 0.15), we see that only a MANOVA analysis of 1-parent and 5-parent evolutionary synthesis in a population of eight networks is statistically significant with a p-value of 0.0148 (given a statistical significance criterion of $p < 0.05$). As expected given the trends show in Figure 5.2, Figure 5.3, and Figure 5.4, the networks synthesized using 3-parent and 5-parent evolutionary synthesis are not significantly different, with p-values of 0.9122 and 0.7773 in populations of five and eight networks, respectively.

We also see from Table 5.1 that 2-parent evolutionary synthesis is statistically significant relative to the asexual case (first row), with p-values less than 0.0001 for populations of three, five, and eight networks per generation. This is consistent with the anomalous 2-parent evolutionary synthesis trends shown in Figure 5.2, Figure 5.3, and Figure 5.4, and is likely the cause for the p-values of 0 shown in the second last row of the table. As such, we re-evaluate the overall statistical significance excluding the 2-parent evolutionary synthesis networks (bottom row), with resulting p-values of 0.1235, 0.1478, and 0.0626 for population sizes of three, five, and eight networks, respectively; note that excluding 2-parent evolutionary synthesis in a population size of three is the same as comparing 1-parent and 3-parent directly, resulting in the same p-value. While the p-values for 3-parent and 5-parent evolutionary synthesis do not indicate statistical significance relative to the 1-parent evolutionary synthesis case, the corresponding p-values are consistently low and suggest that m-parent evolutionary synthesis can potentially synthesize good (i.e., efficient) network architectures.

Lastly, we applied the MANOVA method to networks synthesized using 3-parent evolutionary synthesis in varying population sizes (three, five, and eight networks per generation), and produced a p-value of 0.9962. Similarly, we applied the MANOVA method to networks synthesized using 5-parent evolutionary synthesis in varying population sizes (five and eight networks per generation), and produced a p-value of 0.6531. Both these p-values indicate that the networks synthesized in different population sizes are not significantly different (using a statistical significance criterion of $p < 0.05$). As such, we will only synthesize the required number of network architectures at each generation for m-parent evolutionary synthesis in the subsequent experiments in this thesis, i.e., we will synthesis m networks per generation.

As in Chapter 4, we also assess the statistical significance via paired t-test by comparing the linearly interpolated network storage sizes at a performance threshold of 90% accuracy. This threshold was selected for this chapter as representative of the performance accuracy of the best networks (as per Figure 5.2 (c), Figure 5.3 (c), and Figure 5.4 (c)); however, note that networks synthesized using 2-parent evolutionary synthesis were omitted as none of the synthesized networks had a performance accuracy below the 90% threshold (due to the anomalous trends). As such, paired t-tests were performed between networks synthesized using 3-parent and 5-parent (where available) sexual evolutionary synthesis and networks synthesized using 1-parent asexual evolutionary synthesis (shown in Table 5.2).

From Table 5.2, we see that the paired t-test of 3-parent and 5-parent evolutionary synthesis compared with 1-parent evolutionary synthesis results in relatively low p-values (e.g., approximately 0.15 or less). In particular, the paired t-test between 1-parent and 3-parent in a population size of five networks per generation (p-value of 0.0325) and the paired

Table 5.2: Statistical significance assessment via paired t-test for networks synthesized using one, three, and five parent networks (two parent networks omitted due to anomalous trends) in population sizes of three, five, and eight networks per generation over 5 runs and trained using 10% of the MNIST dataset. We compare interpolated network storage sizes at a performance threshold of 90% accuracy, and present the p-values resulting from one-to-one comparisons (statistical significance criterion of $p < 0.05$).

	Network Population Size (Per Generation)		
	Three (p-value)	**Five (p-value)**	**Eight (p-value)**
1-Parent & 3-Parent	0.1508	0.0325	0.0848
1-Parent & 5-Parent	N/A	0.0639	0.0215

t-test between 1-parent and 5-parent in a population size of eight networks per generation (p-value of 0.0215) showing statistical significance using a statistical significance criterion of $p < 0.05$. The paired t-tests, then, show statistical significance results similar to the MANOVA statistical significance analysis in Table 5.1.

5.4 2-Parent and 4-Parent Evolutionary Synthesis

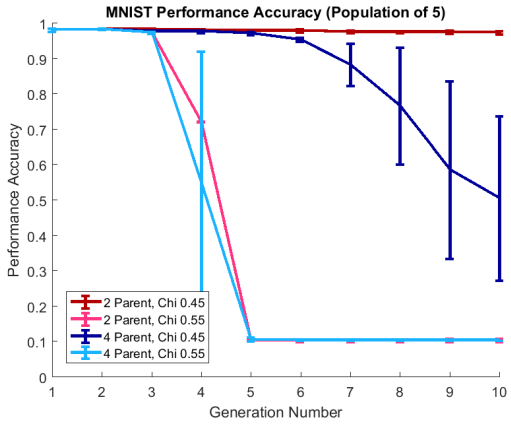
In the previous section, we see that there is anomalous behaviour in the 2-parent evolutionary synthesis case (as shown in Figure 5.2, Figure 5.3, and Figure 5.4). In this section, we now investigate these trends more thoroughly and speculate as to the cause of the comparatively slow decrease in performance accuracy and network storage size relative to the 1-parent, 3-parent, and 5-parent evolutionary synthesis trends.

In the previous section, we had selected the cluster-level percent of population parameter χ to be 50% for the experiment. However, this value of χ is ill-suited for 2-parent evolutionary synthesis as a χ of 50% aligns exactly with having one of the two parent networks containing any given cluster, making the χ threshold ambiguous. The algorithmic implementation of this specifies that a cluster present in ***less*** than χ parent networks would be removed. As a result, a χ of 50% results in a synaptic cluster being synthesized in an offspring network if present in either of the two parent networks, and essentially produces an offspring network containing the union of the synaptic clusters present in the two parent networks. To investigate this potential ambiguity, we investigate 2-parent evolutionary synthesis using cluster-level percent of population parameters of $\chi = 45\%$ and $\chi = 55\%$. In addition, we also examine the previously unexplored 4-parent evolutionary synthesis

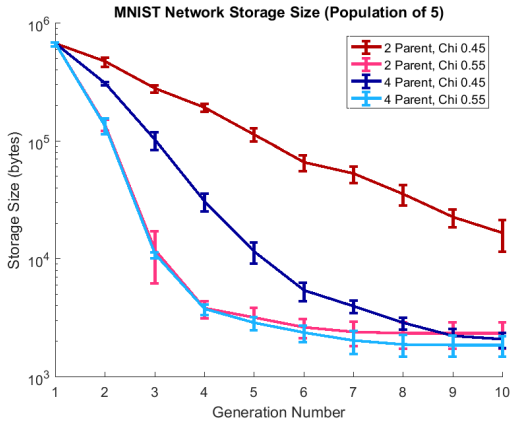
(as 2-parent evolutionary synthesis was the only even number of parent networks in the previous section that would suffer from this ambiguity with $\chi = 50\%$) using $\chi = 45\%$ and $\chi = 55\%$ for comparison. To allow for direct comparison between 2-parent and 4-parent evolutionary synthesis, we only synthesize population sizes of five and eight networks per generation using 10% of the MNIST dataset for training, and the original fully-trained ancestor network (generation 1) had a testing accuracy of 98% with 143,136 synapses and 7,200 kernels (corresponding to a 1-channel input LeNet architecture [34]).

Figure 5.5 and Figure 5.6 show the MNIST validation accuracy and network storage size for networks synthesized using two parents and $\chi = 45\%$ (dark red), two parents and $\chi = 55\%$ (pink), four parents and $\chi = 45\%$ (dark blue), and four parents and $\chi = 55\%$ (light blue) as a function of generation number for populations of five and eight networks per generation, respectively. Specifically, Figure 5.5 (a) and Figure 5.6 (a) show the validation accuracy as a function of generation number while Figure 5.5 (b) and Figure 5.6 (b) show the network storage size as a function of generation number. Notice that for 2-parent evolutionary synthesis, the accuracy and storage size trends for $\chi = 45\%$ (dark red) closely resemble those shown in Figure 5.2, Figure 5.3, and Figure 5.4; this is expected, as (algorithmically speaking) a χ of 45% is functionally equivalent to a χ of 50% and essentially results in a union-based mating policy as explained earlier in this section. Similarly, notice that the accuracy and storage size trends for 2-parent evolutionary synthesis using $\chi = 55\%$ (pink) a similar to those shown in Figure 4.3. While the steep drop in validation accuracy and storage size is perhaps delayed by a generation (i.e., accuracy drops to approximately 10% at generation 5 rather than at generation 4 as in Chapter 4), the trends themselves are comparable as a χ of 55% requires that both parent networks contain any given synaptic cluster and results in an intersection-based mating policy, and the delay in performance accuracy drop is likely due to the environmental factor model being 50% in this chapter (as compared to the 70% in Chapter 4).

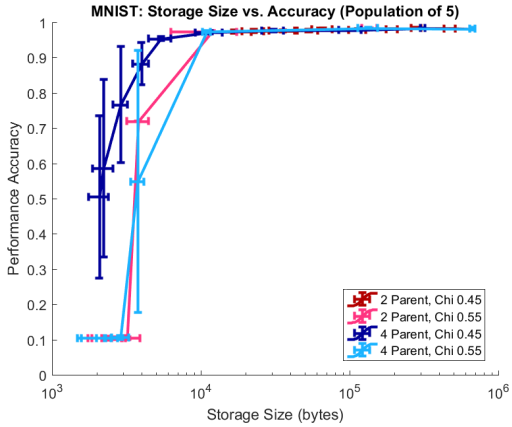
More interesting, then, are the accuracy and storage size trends with 4-parent evolutionary synthesis. In the 4-parent case, a χ of 45% (dark blue) requires that at least two of the four parent networks contain any given cluster while a χ of 55% (light blue) requires that at least three of the four parent networks contain any given cluster. As expected, a χ of 45% results in a slower decrease in validation accuracy and network storage size over the 10 generations relative to a χ of 55%; however, the decrease in accuracy and storage size is notably faster than the 2-parent case for a χ of 45% where only one parent network must contain any given cluster. Conversely, a χ of 55% results in similar accuracy and storage size trends for both 2-parent and 4-parent evolutionary synthesis. This indicates that requiring three of the four parent networks to contain any given cluster in 4-parent evolutionary synthesis is comparable to an intersection-based mating policy in 2-parent



(a) MNIST validation accuracy.

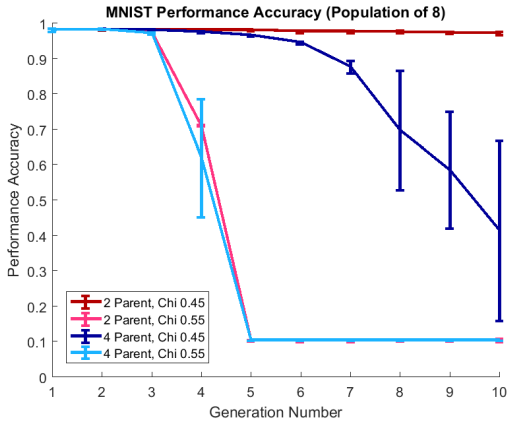


(b) MNIST network storage size.

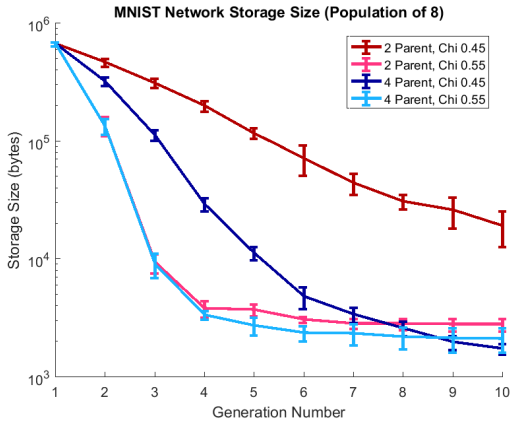


(c) MNIST accuracy vs. storage size.

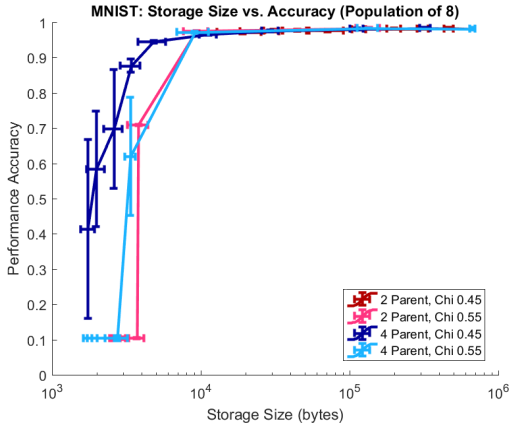
Figure 5.5: MNIST average validation accuracy and average neural network storage size vs. generation number (top row) for networks synthesized using two ($\chi = 0.45$ dark red, $\chi = 0.55$ pink) and four ($\chi = 0.45$ dark blue, $\chi = 0.55$ light blue) parents, with error bars showing the standard deviation over 5 runs. These plots show the effects of 2-parent and 4-parent evolutionary synthesis for a population of five synthesized networks per generation given a percent of population χ of 45% and 55%. The bottom row shows the accuracy vs. storage size trade-off, with data points closest to the top left corner of the scatter plot indicating the best networks. Notice that a χ of 55% results in a much faster drop in accuracy relative to a χ of 45%.



(a) MNIST validation accuracy.



(b) MNIST network storage size.



(c) MNIST accuracy vs. storage size.

Figure 5.6: MNIST average validation accuracy and average neural network storage size vs. generation number (top row) for networks synthesized using two ($\chi = 0.45$ dark red, $\chi = 0.55$ pink) and four ($\chi = 0.45$ dark blue, $\chi = 0.55$ light blue) parents, with error bars showing the standard deviation over 5 runs. These plots show the effects of 2-parent and 4-parent evolutionary synthesis for a population of eight synthesized networks per generation given a percent of population χ of 45% and 55%. The bottom row shows the accuracy vs. storage size trade-off, with data points closest to the top left corner of the scatter plot indicating the best networks. Similar to Figure 5.5, a χ of 55% results in a much faster drop in accuracy relative to a χ of 45%.

Table 5.3: Statistical significance assessment for networks synthesized two and four parent networks in population sizes of five and eight networks per generation over 5 runs and trained using 10% of the MNIST dataset; a percent of population χ of 45% and 55$ were used for both 2-parent and 4-parent evolutionary synthesis. We use the MANOVA statistical significance analysis method, and present the p-values resulting from both overall and one-to-one comparisons (statistical significance criterion of $p < 0.05$).

	Network Population Size (Per Generation)	
	Five (p-value)	**Eight (p-value)**
2-Parent $\chi = 45\%$ & $\chi = 55\%$	1.451×10^{-14}	1.372×10^{-14}
4-Parent $\chi = 45\%$ & $\chi = 55\%$	4.752×10^{-10}	3.151×10^{-9}
2-Parent & 4-Parent ($\chi = 45\%$)	2.614×10^{-4}	1.052×10^{-4}
2-Parent & 4-Parent ($\chi = 55\%$)	0.9723	0.9919
Overall	0.0000	0.0000

evolutionary synthesis, and implies that there is an optimal value of χ that can produce efficient network architectures in relatively few (e.g., less than 10) generations.

Figure 5.5 (c) and Figure 5.6 (c) show the trade-off between performance accuracy and network storage size for networks synthesized using two parents and $\chi = 45\%$ (dark red), two parents and $\chi = 55\%$ (pink), four parents and $\chi = 45\%$ (dark blue), and four parents and $\chi = 55\%$ (light blue). Notice that 2-parent evolutionary synthesis produced data points similar to those shown in Figure 5.2 (c), Figure 5.3 (c), and Figure 5.4 (c) for $\chi = 45\%$ and data points similar to those shown in Figure 4.3 for $\chi = 55\%$. While the trend for 4-parent evolutionary synthesis is similar to that of 2-parent evolutionary synthesis for $\chi = 55\%$, the data points corresponding to networks produced via 4-parent evolutionary synthesis for $\chi = 45\%$ (dark blue) appears to result in comparably better network architectures (as indicated by their closeness to the top-left corner of the plot), and further supports the idea that the incorporation of multiple parent networks during the evolutionary synthesis process can produce network architectures with improved trade-offs between accuracy and storage size.

Similar to the experiment in the previous section, we leverage the multivariate analysis of variance (MANOVA) method to test for statistical significance between the networks synthesized using 2-parent and 4-parent evolutionary synthesis in population sizes of five and eight networks with a percent of population χ of 45% and 55$. The resulting p-values for both overall (i.e., 2-parent and 4-parent evolutionary synthesis using a χ of 45% and 55$) statistical significance for a given population size and one-to-one comparisons are

shown in Table 5.3. The overall p-value (bottom row) for both populations of five and eight networks per generation is 0; this indicates that there is clear evidence that the means of the different groups are significantly different.

Table 5.3 also shows that networks synthesized using a χ of 45% (regardless of 2-parent or 4-parent evolutionary synthesis) are significantly different than networks synthesized using a χ of 55%, as indicated by the small p-values ($< 1.0 \times 10^{-8}$) and given a statistical significance criterion of $p < 0.05$. While networks synthesized using 2-parent evolutionary synthesis are significantly different than networks synthesized using 4-parent evolutionary synthesis for a χ of 45% with p-values of 2.614×10^{-4} (population of five networks) and 1.052×10^{-4} (population of eight networks), the opposite is true for a χ of 55%. A χ of 55% produced networks that were not significantly different for 2-parent and 4-parent evolutionary synthesis with p-values > 0.95. The p-values shown in Table 5.3 are congruent with the trends shown in Figure 5.5 and Figure 5.6. In particular, Table 5.3 shows that networks synthesized using 4-parent evolutionary synthesis and a χ of 45% achieve a significantly better trade-off between performance accuracy and network storage size relative to the other combinations of number of parent networks and χ evaluated in this experiment.

As per Section 5.3, we also assess the statistical significance of the synthesized networks using the paired t-test by comparing the linearly interpolated network storage sizes at a performance threshold of 90% accuracy. Note that similar to the previous section, the anomalous behaviour in 2-parent sexual evolutionary synthesis persists in this section; as such, networks synthesized via 2-parent evolutionary synthesis were omitted from this analysis, and the paired t-test was only performed on networks synthesized via 4-parent evolutionary synthesis. The paired t-test comparing networks synthesized via 4-parent evolutionary synthesis using a χ of 45% and using a χ of 55% resulted in a p-value of 0.0086 for a population of five networks per generation, and a p-value of 0.0100 for a population of eight networks per generation. In both cases, the paired t-test shows statistical significance using a significance criterion of $p < 0.05$, indicating that networks synthesized via 4-parent evolutionary synthesis with a χ of 45% are significantly different than those synthesized using a χ of 55%.

5.5 Summary

Building on the concept of sexual evolutionary synthesis from the previous chapter, the use of multi-parent sexual evolutionary synthesis during the synthesis of offspring networks for evolutionary deep intelligence was explored in this chapter. The efficacy of sexual evolutionary synthesis was examined via m-parent evolutionary synthesis in the context of

varying generational population sizes using 10% of the MNIST dataset. In contrast to the intersection-based mating policy in Chapter 4, we leverage a more flexible mating policy via the introduction of a cluster-level percent of population parameter χ, allowing for the synthesis of synaptic clusters present in a subset of the m parent networks.

Initially, we investigated $m = 1, 2, 3, 5$ number of parent networks during evolutionary synthesis using a χ of 50%, and demonstrate that the incorporation of multiple parent networks during evolutionary synthesis allows for a better trade-off between accuracy and storage size. However, upon discovering anomalous behaviour in the 2-parent evolutionary synthesis cases, we further investigated 2-parent and 4-parent evolutionary synthesis with cluster-level percent of population parameters of $\chi = 45\%$ and $\chi = 55\%$. While 2-parent evolutionary synthesis can only synthesize networks using a union-based mating policy (in the case of $\chi = 45\%$) or an intersection-based mating policy (in the case of $\chi = 55\%$), the difference in trends for 4-parent evolutionary synthesis using $\chi = 45\%$ and $\chi = 55\%$ implies that there is an optimal value of χ that can produce efficient network architectures in relatively few (e.g., less than 10) generations. We will further investigate the effect of the cluster-level percent of population parameters of χ in the next chapter.

Both experiments indicate that the use of multiple parent networks during evolutionary synthesis allows for the synthesis of networks increasingly efficient architectures, and (generally) the addition of more parent networks during the evolutionary synthesis process results in networks with improved trade-off between accuracy and storage size. While assessing the performance accuracy and network storage size as a function of generation number is appropriate in the context of the evolutionary synthesis formulation presented in Chapter 3, it is difficult to interpret such an abstract concept outside of evolutionary deep intelligence. As such, we propose evaluating the performance accuracy and network storage size as function of computation time as a more direct measure of network progression in the following chapters.

Chapter 6

Gene Tagging

In Chapters 4 and 5, we explored the computational construct used to mimic heredity via the incorporation of multiple parent networks during the evolutionary synthesis process using both intersection-based mating functions (Chapter 4) and more flexible proportional mating functions (Chapter 5). The use of these mating functions during the evolutionary synthesis process, however, essentially combines architectural structures sequentially regardless of their relative positions in each parent network. Without a system to track the original locations of the architectural synaptic clusters, offspring networks created during the evolutionary synthesis process are produced by combining parent networks independent of architectural alignment, resulting in a mismatch of network structures. The implications and mitigation of this architectural mismatch are explored in this chapter.

Figure 6.1 shows a simple visualization of architectural mismatch due to a lack of gene tagging where two offspring network architectures are synthesized using the same ancestor architecture. Because not all synapses are synthesized in the offspring networks, the two offspring networks show in Figure 6.1 appear to have comparable architectural structures, despite their bottom left nodes originating from different locations in the ancestor network (blue on the left, red on the right). This misalignment in structure, then, results in an architectural mismatch if the two offspring networks shown were to be used as parent networks for a new offspring network. We propose to mitigate this architectural mismatch by tracking the originating locations of each architectural cluster (represented in Figure 6.1 as single nodes) to prevent the combination of different architectural clusters due to misalignment in the parent network architectures.

In order to track architectural clusters over successive generations, we introduce the concept of gene tagging into the multi-parent evolutionary synthesis process. We evaluate

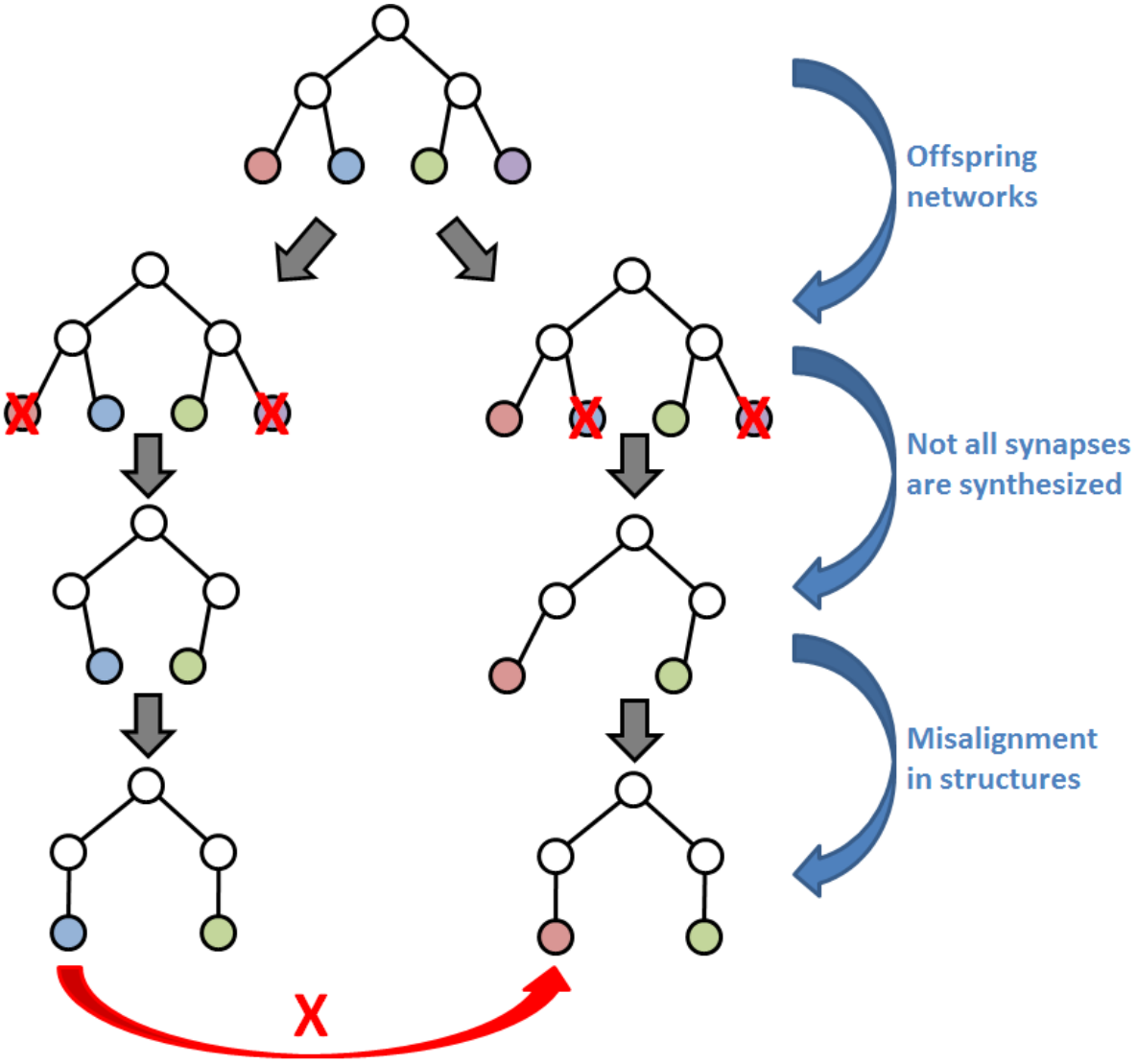


Figure 6.1: Visualization of architectural mismatch during m-parent (2-parent shown) evolutionary synthesis due to lack of gene tagging. While the bottom left node of the left offspring network (blue) appears in the same architectural location as the bottom left node of the right offspring network (red), they originate from different locations in the ancestor network (shown at the top).

the efficacy of gene tagging using the most complex multi-parent evolutionary synthesis case, i.e., 5-parent evolutionary synthesis, to maximize the occurrence of architectural

mismatch. First, we present a study into the effects of architectural alignment during evolutionary synthesis via the introduction of a gene tagging system. Gene tagging is explored within the context of m-parent sexual evolutionary synthesis and evaluated over a range of environmental resource models. The gene tagging system allows for the proper alignment of architectural structures that originated from the same location in the ancestor network, and enforces a like-with-like mating policy during evolutionary synthesis. Recall from Chapter 5 that we introduced χ in (5.4), the cluster-level percent of population parameter to allow for more flexibility during the evolutionary synthesis process. To better understand its effect (especially in the context of gene tagging), we evaluate a range of χ in the context of 3-parent evolutionary synthesis, and evaluate a limited range of χ in combination with varying environmental factor models for 5-parent evolutionary synthesis.

Results from the first two sections show that networks synthesized with gene tagging have a slight drop in trade-off between accuracy and storage size, leading to speculation whether gene tagging produces networks with less architectural variability. In the last section of this chapter, we attempt to answer the following question: "how can we assess the architectural similarity of DNNs in a meaningful and useful way?" We present an experiment exploring the quantification of network architectural similarity in populations of evolutionary synthesized neural networks via percentage overlap of architectural clusters. Architectural similarity is explored within the context of multi-parent sexual evolutionary synthesis (specifically, 5-parent evolutionary synthesis), and will hopefully allow for the development of improved similarity-based mating policies during the evolutionary synthesis of highly efficient networks.

6.1 Mitigating Architecture Mismatch

In prior chapters, offspring networks were created during the evolutionary synthesis process by combining parent networks via a simple mating function that operated independent of architectural alignment. As a result, architectural clusters were combined sequentially, regardless of relative positions in the parent networks, and resulted in a mismatch of architectural structures.

To encourage like-with-like mating during evolutionary synthesis, we introduce a gene tagging system to track the original locations of each cluster over successive generations. This allows us to enforce structural alignment during the mating process, i.e., only allowing architectural clusters originating from the same location in the ancestor network to mate. As such, we reformulate the synaptic architecture cluster-level mating function (5.4) from

the previous chapter as follows:

$$\mathcal{M}_C^S(\overline{s}_{G_i,C}) = \begin{cases} 1 & \frac{1}{m}\sum_{k=1}^{m} \overline{s}_{k,C} \geq \chi \\ 0 & otherwise \end{cases} \tag{6.1}$$

where $\overline{s}$ is the indicator for the gene tagged and structurally-aligned synaptic clusters, meaning that any given synaptic cluster in one network in G_i is only ever mated with the *same* synaptic cluster in another network in G_i. Each gene tagged architectural cluster holds information with its original location in the ancestor network, and this gene tag can be used during the evolutionary synthesis process to ensure that only clusters with the same original location (i.e., same gene tag) are combined during mating. Similar to Chapter 5, the synapse-level mating function shown in (5.3) remains the same. Given the inherited structurally-aligned synaptic structure $\overline{S}_{g,i}$ of $\mathcal{H}_{g,i}$, $\mathcal{H}_{g,i}$ is then subject to simulated environmental resources via $\mathcal{R}_g$ and the synaptic strengths of parent networks $\mathcal{H}_{G_i}$ as per (4.4) and (4.5). We also leverage the χ-based synaptic strength mating policies from Chapter 5 proposed in (5.5) and (5.6).

Figure 6.2 shows the performance accuracy and storage size for 5-parent evolutionary synthesis with gene tagging (top row) and no gene tagging (bottom row) given various cluster-level and synapse-level environmental factor models for a single run. The incorporation of gene tagging resulted in a more gradual decrease in both performance accuracy and storage size relative to computational time, while the omission of gene tagging produced network architectures with more rapidly decreasing performance accuracy and storage size. Note that the bottom plateau in performance accuracy is once again at 10% akin to random guessing, as the MNIST dataset consists of 10 classes of handwritten digits. While there is no inherent limit in network reduction when using the evolutionary deep intelligence approach, Figure 6.2 shows that there are natural plateaus in network storage size, particularly with the higher environmental factor models, i.e., environmental factor models of 80%, 85%, and 95%. In addition, note that these plateaus in network storage size tend to occur more when synthesizing networks using gene tagging.

Figure 6.3 shows performance accuracy as a function of storage size for 5-parent sexual evolutionary synthesis using various cluster-level and synapse-level environmental factor models, where the best synthesized networks are closest to the top left corner, i.e., high performance accuracy and low storage size. Networks synthesized using gene tagging (diamond points) appear to be minimally worse relative to networks synthesized without gene tagging (round points) in terms of maintaining performance accuracy while decreasing storage size. However, this raises an interesting question (addressed at the end this chapter): how can we assess the architectural similarity of DNNs in a meaningful and useful way?

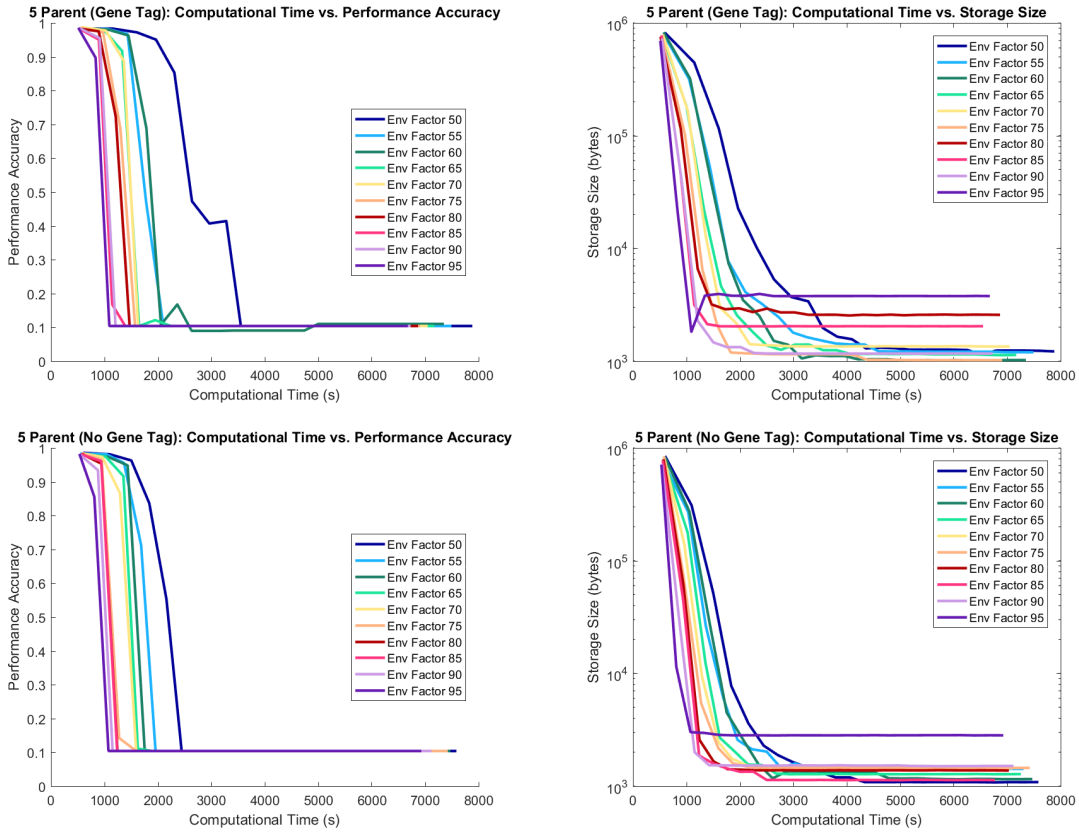


Figure 6.2: Performance accuracy (left) and storage size (right) for 5-parent evolutionary synthesis with gene tagging (top row) and no gene tagging (bottom row) using various environmental factor models. Plots best viewed in colour.

6.2 Percent of Population Parameter χ

To better understand the percent of population parameter χ introduced in Chapter 5, we further explore its impact within the context of 3-parent and 5-parent evolutionary synthesis. Percent of population parameter χ is used to control the proportion of parent network architectures that must contain any given cluster in order for that cluster to be synthesized during evolutionary synthesis.

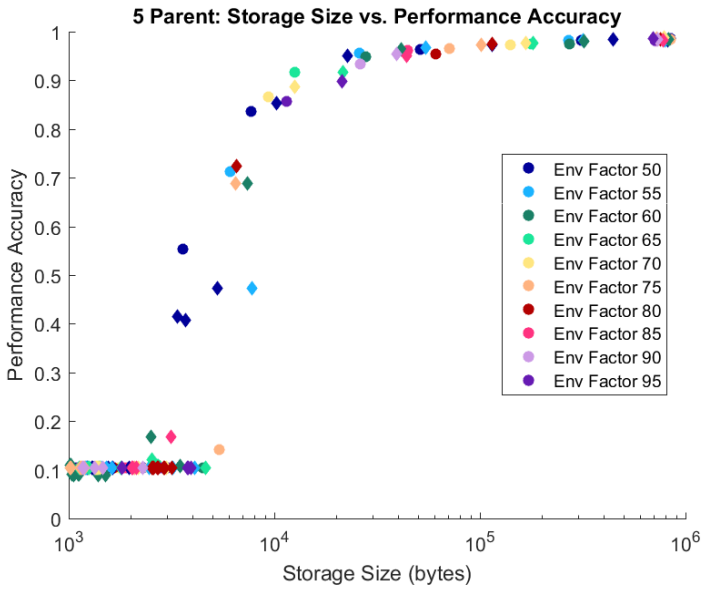


Figure 6.3: Performance accuracy as a function of storage size for 5-parent sexual evolutionary synthesis using various environmental factor models. Networks synthesized using gene tagging (diamond) show minimal to no noticeable difference relative to networks synthesized without gene tagging (round) in terms of maintaining performance accuracy while decreasing storage size. Plots best viewed in colour.

6.2.1 3-Parent Evolutionary Synthesis with Varying χ

As an initial assessment of percent of population parameter χ, we evaluate a range of values within the context of gene tagged 3-parent evolutionary synthesis for ten generations. Having used $\chi = 0.5$ (or 50%) in the previous section where we introduced the gene tagging system, we now evaluate the following values of χ:

$$\chi = \{0, 0.25, 0.5, 0.75, 1\}. \tag{6.2}$$

As in the previous section, we use an environmental factor model of 50% and, to validate its efficacy in a less complex case, we use 3-parent evolutionary synthesis.

Figure 6.4 shows the performance accuracy (left) and storage size (right) of the gene tagged 3-parent evolutionary synthesis as a function of computation time for a single run. Note that in the case of 3-parent evolutionary synthesis, $\chi = 0$ and $\chi = 0.25$ are functionally equivalent where a cluster needs to exist in any of the parent networks to be synthesized in the offspring network (as existing in a single parent network would result in

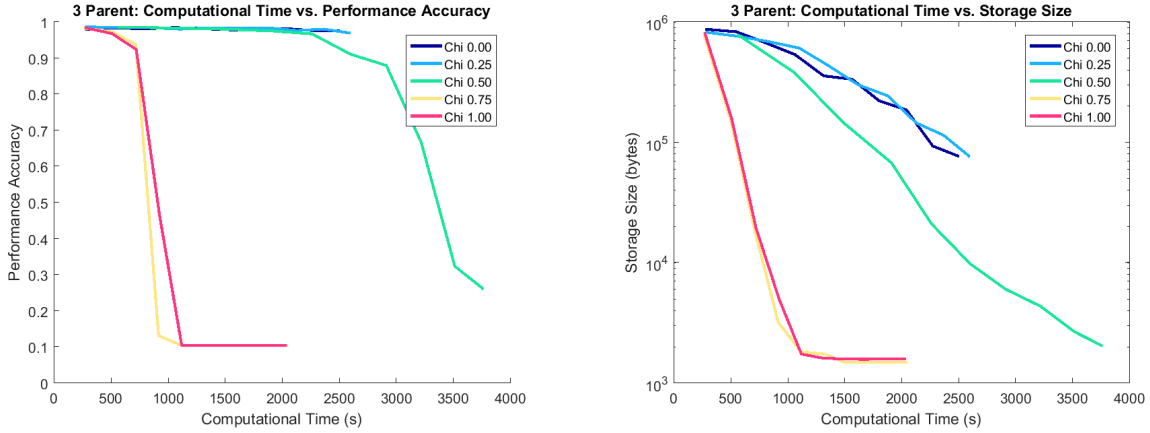


Figure 6.4: Performance accuracy (left) and storage size (right) as a function of computation time for 3-parent sexual evolutionary synthesis using various percent of population parameter χ. Plots best viewed in colour.

a percent of population of approximately 0.33 or 33%). Similarly, $\chi = 0.75$ and $\chi = 1$ are functionally equivalent where a cluster needs to exist in all three of the parent networks to be synthesized in the offspring network (as existing in two parent networks would result in a percent of population of approximately 0.67 or 67%). We see this reflected in Figure 6.4, as the trends are very similar between $\chi = 0$ and $\chi = 0.25$ as well as between $\chi = 0.75$ and $\chi = 1$ for both performance accuracy and storage size.

Figure 6.4 also confirms some speculations with respect to how the percent of population parameter χ affects the synthesis of increasingly efficient network architectures. As expected, enforcing that any given cluster must exist within all parent networks in order to be synthesized (as is the case for $\chi = 0.75, 1$ in this section) results in a rapid sparsification of the network structure; this can clearly be seen in the steep drops in both performance accuracy and storage size for $\chi = 0.75$ (yellow) and $\chi = 1$ (pink). Conversely, allowing a cluster to be synthesized if it exists in any parent network ($\chi = 0, 0.25$ in this section) results in a much slower sparsification process. Figure 6.4 shows that there is minimal drop in performance accuracy for $\chi = 0$ (dark blue) and $\chi = 0.25$ (light blue); this corresponds to a much more gradual decline in storage size, having achieved a reduction of only one order of magnitude by the end of the ten generations in this experiment.

Most interesting, then, are the performance accuracy and storage size plots for $\chi = 0.5$ (green). In the context of 3-parent evolutionary synthesis, $\chi = 0.5$ enforces that any given cluster must exist within at least two parent networks to be synthesized in an offspring

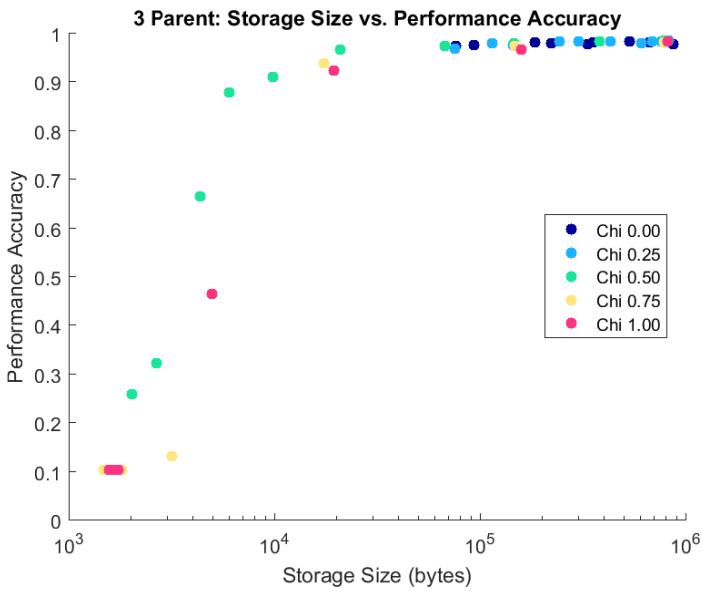


Figure 6.5: Performance accuracy as a function of storage size for the first ten generations of 3-parent sexual evolutionary synthesis using various percent of population parameter χ. Plots best viewed in colour.

network. Figure 6.4 implies that $\chi = 0.5$ achieves a good trade-off between maintaining performance accuracy and reducing model storage size, as the accuracy is maintained for the first few generations before dropping off after the storage size is reduced by two orders of magnitude.

Figure 6.5 shows the performance accuracy of 3-parent evolutionary synthesis as a function of storage size. By comparing the networks synthesized using various percent of population parameter χ, we see that the best networks (those closest to the top-left corner) with the highest accuracy to storage size ratio are synthesized using $\chi = 0.5$ (green). This is congruent with the implications of Figure 6.4. As such, this experiment implies that the optimal percent of population parameter χ is one that allows for the synthesis of clusters that exist in some, but not all, of the parent network architectures.

6.2.2 5-Parent Evolutionary Synthesis with Varying χ

From the previous section, we see that the optimal percent of population parameter χ tends to enforce that a cluster can be synthesized in the offspring network given that it exists within some non-zero subset of the parent networks. We now extend this experiment to the most complex multi-parent case in this thesis, i.e., 5-parent evolutionary synthesis, and evaluate the following subset of χ:

$$\chi = \{0.25, 0.5, 0.75\}. \tag{6.3}$$

These selected χ values will be used to investigate the effect of χ when a given cluster exists in two parent networks ($\chi = 0.25$), three parent networks ($\chi = 0.5$), or four parent networks ($\chi = 0.75$) during 5-parent evolutionary synthesis. To better evaluate the percent of population parameter χ within the context of the gene tagging system proposed earlier in this chapter, the above χ parameters are used to synthesize network architectures both with and without gene tagging using the same range of environmental factor models, i.e., 50% to 95% at 5% increments.

Figure 6.6 shows the performance accuracy (left) and storage size (right) for the first ten generations using percent of population parameter $\chi = 0.25$ with various environmental factor models (for a single run). As it is 5-parent evolutionary synthesis, a χ of 0.25 enforces that any given cluster must exist in at least two parent networks to be synthesized in an offspring network for both with gene tagging (top row) and without gene tagging (bottom row). This is the most lenient of existence criteria tested in this experiment, and we see the corresponding leniency in how the synthesized networks maintain performance accuracy over successive generations; this is most obvious in the low environmental factor models, such as 50% (dark blue), 55% (light blue), 60% (dark green), and 65% (light green). We likewise see a more gradual decrease in storage size over the generations, which is congruent with similar efficiency trends in this thesis.

Figure 6.7 shows the performance accuracy (left) and storage size (right) for the first ten generations using percent of population parameter $\chi = 0.5$ with various environmental factor models. This enforces that a cluster must exist within a minimum of three of the parent networks during 5-parent evolutionary synthesis both with gene tagging (top row) and without gene tagging (bottom row). This figure is essentially a truncated version of Figure 6.2 limited to ten generations for comparison purposes in this section. As expected, χ of 0.5 enforces a stricter existence criterion than a χ of 0.25, and we observe steeper and more sudden drops in performance accuracy as the model storage size decreases by more than two orders of magnitude.

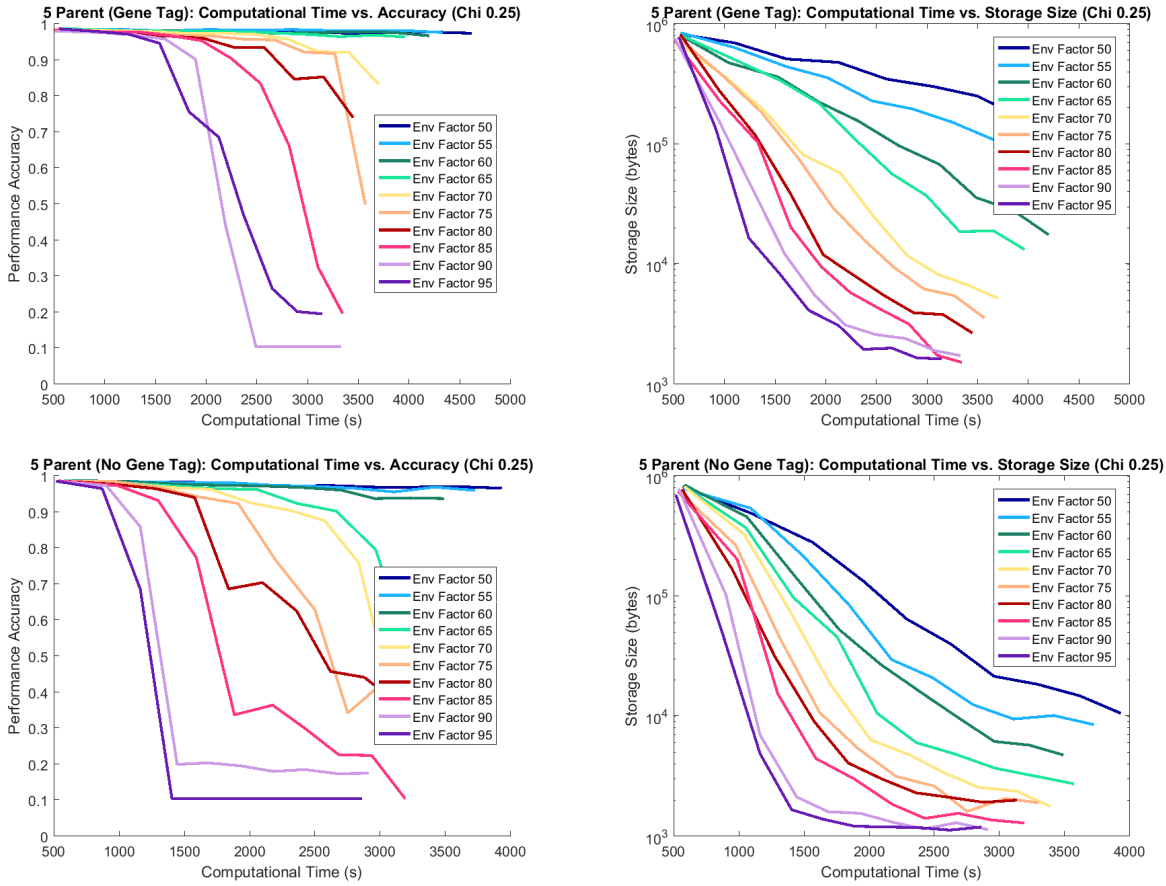


Figure 6.6: Performance accuracy (left) and storage size (right) for 5-parent evolutionary synthesis with gene tagging (top row) and no gene tagging (bottom row) using percent of population parameter $\chi = 0.25$ with various environmental factor models. Plots best viewed in colour.

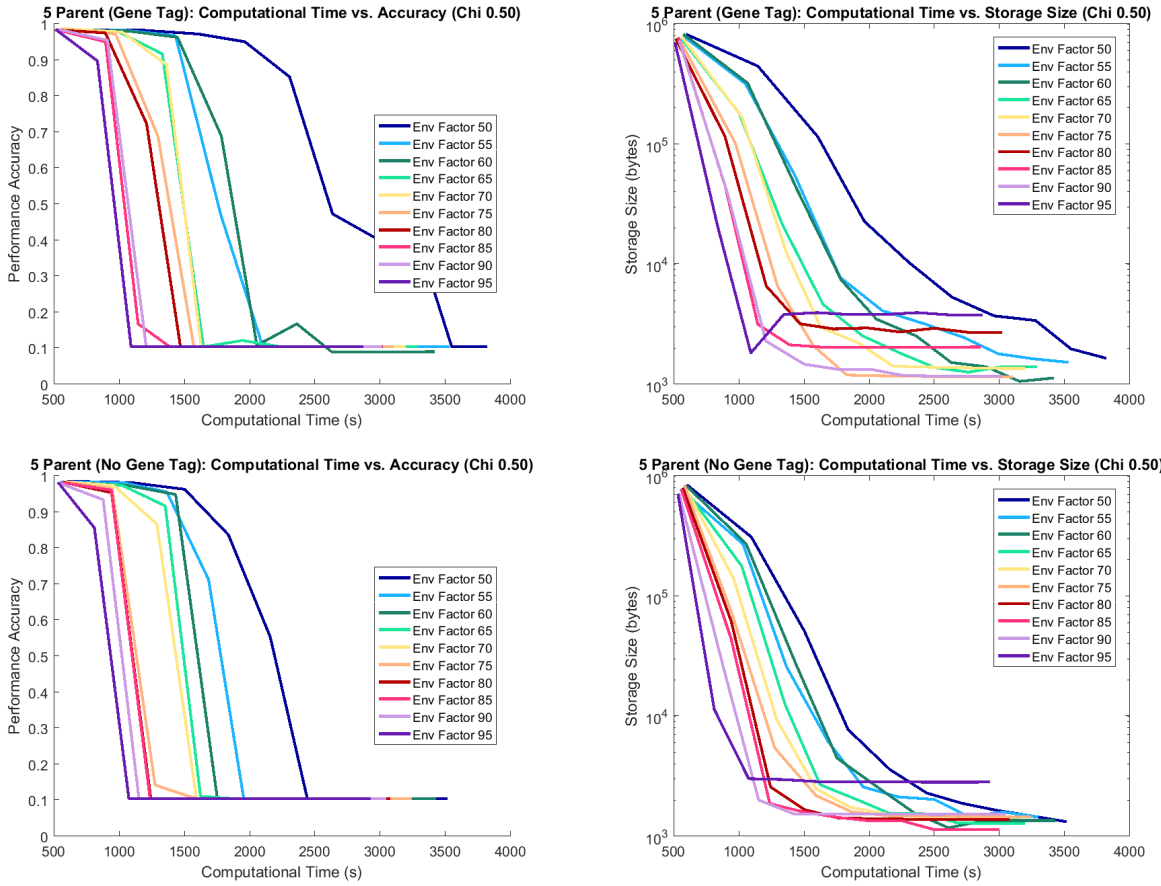


Figure 6.7: Performance accuracy (left) and storage size (right) for 5-parent evolutionary synthesis with gene tagging (top row) and no gene tagging (bottom row) using percent of population parameter $\chi = 0.5$ with various environmental factor models. Plots best viewed in colour.

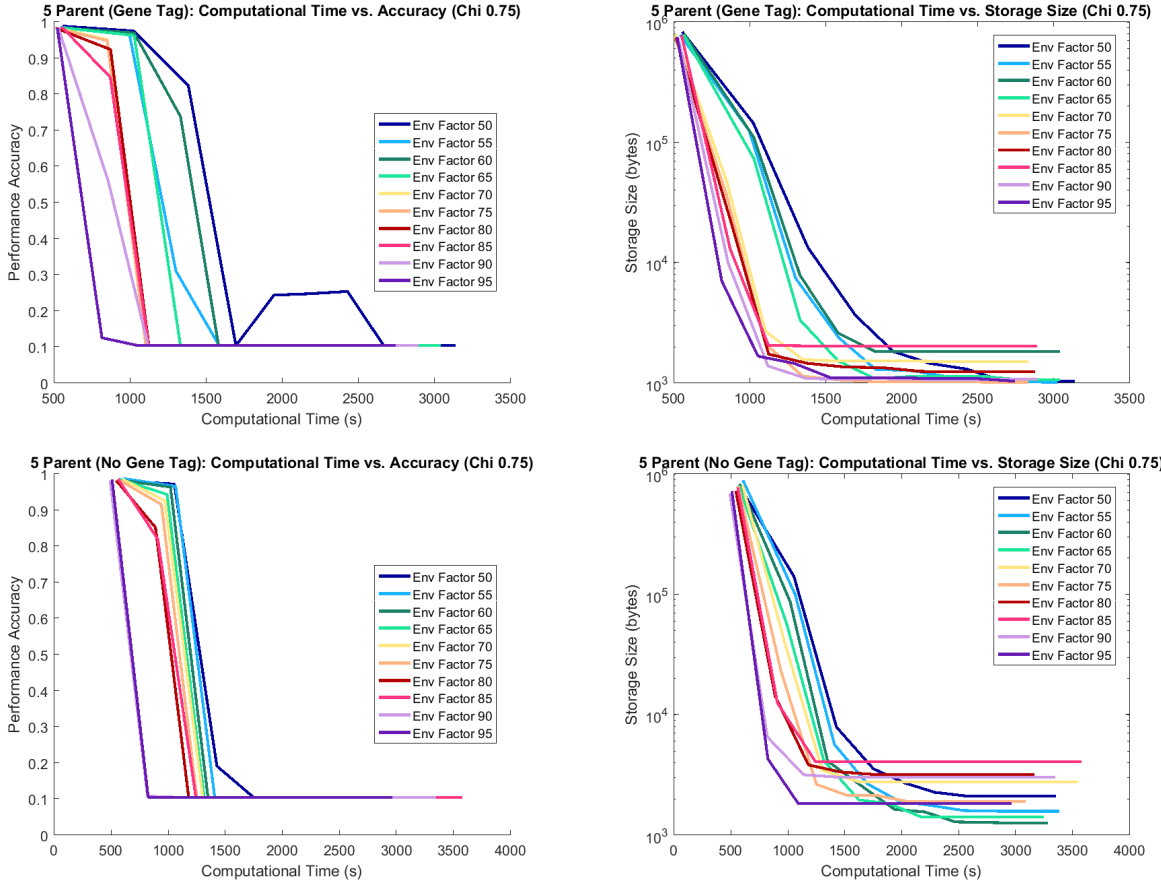


Figure 6.8: Performance accuracy (left) and storage size (right) for 5-parent evolutionary synthesis with gene tagging (top row) and no gene tagging (bottom row) using percent of population parameter $\chi = 0.75$ with various environmental factor models. Plots best viewed in colour.

Lastly, Figure 6.8 shows the performance accuracy (left) and storage size (right) as a function of computation time for the first ten generations using percent of population parameter $\chi = 0.75$ with various environmental factor models. A χ of 0.75 enforces that any given cluster must exist in at least four parent networks to be synthesized in an offspring network during the evolutionary synthesis process with gene tagging (top row) and without gene tagging (bottom row). This is the strictest of existence criteria evaluated in this section, and the effects are reminiscent of the trends seen in the 3-parent evolutionary synthesis plots using $\chi = 0.75, 1$ (see Figure 6.4) where the accuracy and

storage size rapidly decline within the first few generations. Relative to $\chi = 0.25$ and $\chi = 0.5$, $\chi = 0.75$ only produces viable offspring networks (i.e., network architectures with enough complexity to represent the problem space) during the first few generations. This is most obvious with the highest environmental factor models, such as 95% (dark purple), 90% (light purple), and 85% (pink), where the accuracy drops to about 10% within the first two generations.

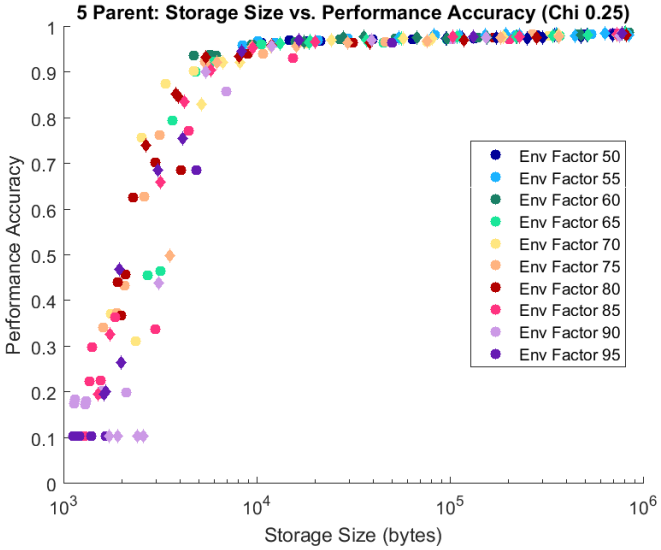


Figure 6.9: Performance accuracy as a function of storage size for the first ten generations of 5-parent sexual evolutionary synthesis using percent of population parameter $\chi = 0.25$ with gene tagging (diamond) and without (round). Plots best viewed in colour.

Figures 6.9, 6.10, and 6.11 show the performance accuracy of the networks synthesized using $\chi = 0.25$, $\chi = 0.5$, and $\chi = 0.75$, respectively, as a function of storage size. The networks are synthesized during the first ten generations of 5-parent evolutionary synthesis both with gene tagging (diamond) and without gene tagging (round). The plots confirm the earlier observations regarding percent of population parameter χ, where a more lenient existence criterion produces networks with maintained performance accuracy (as can be seen with the dense distribution of networks in the top right of Figure 6.9) and a stricter existence criterion mostly produces networks unable to model the problem space (as can be seen with the dense distribution of networks in the bottom left of Figure 6.11). This suggests that a percent of population parameter χ of 0.5 is a good choice for maintaining performance accuracy over successive generations while reducing the model storage size.

6.3 Assessing Architectural Similarity of Networks

Thus far in the chapter, experimental results have shown that networks synthesized with and without gene tagging have similar trends in performance accuracy and model storage size, with networks synthesized using gene tagging perhaps slightly worse at maintaining accuracy while decreasing storage size. This slight drop in trade-off between accuracy and storage size has led us to speculate whether gene tagging produces networks with less architectural variability, and raises an interesting question: "how can we assess the architectural similarity of DNNs in a meaningful and useful way?"

In this last section, we present an experiment exploring the quantification of network architectural similarity in populations of evolutionary synthesized neural networks via percentage overlap of architectural clusters. Architectural similarity is explored within the context of multi-parent sexual evolutionary synthesis, and will allow for the development of improved similarity-based mating policies during the evolutionary synthesis of highly efficient networks.

To investigate the quantification of architectural similarity in the context of multi-parent sexual evolutionary synthesis, the percentage overlap of architectural clusters be-

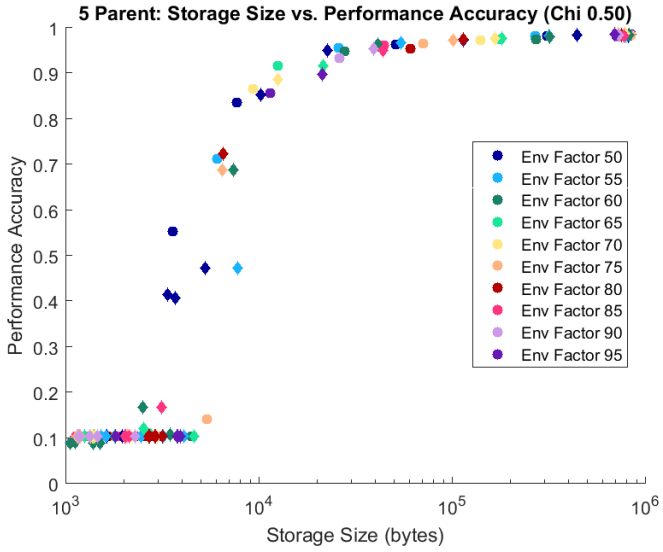


Figure 6.10: Performance accuracy as a function of storage size for the first ten generations of 5-parent sexual evolutionary synthesis using percent of population parameter $\chi = 0.5$ with gene tagging (diamond) and without (round). Plots best viewed in colour.

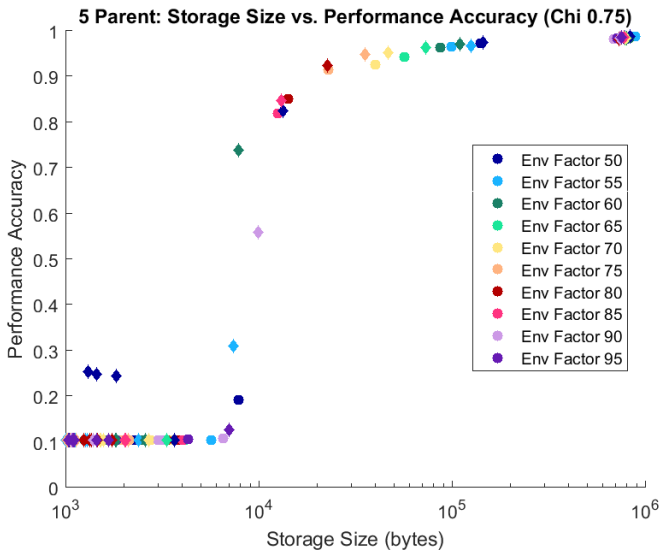


Figure 6.11: Performance accuracy as a function of storage size for the first ten generations of 5-parent sexual evolutionary synthesis using percent of population parameter $\chi = 0.75$ with gene tagging (diamond) and without (round). Plots best viewed in colour.

tween two networks is formulated as the proportion of intersecting clusters:

$$\%overlap_{AB} = \frac{|C_A \cap C_B|}{\frac{1}{2}(|C_A| + |C_B|)}, \tag{6.4}$$

where C_A and C_B are the sets of architectural clusters that exist in the compared networks.

Percentage overlap of architectural clusters is an intuitive representation of network architecture similarity made viable in the context of multi-parent evolutionary synthesis by leveraging the gene tagging system introduced in this chapter. As such, gene tagging (which allows for architectural alignment during evolutionary synthesis) can similarly be used to calculate percentage overlap of existing architectural clusters originating from the same location in the ancestor network. Percentage overlap is indicative of network population diversity within a generation, e.g., relatively low average percentage overlap would indicate a generation of synthesized networks with comparatively higher architectural variability.

Figure 6.12 shows the performance accuracy as a function of storage size for the populations of synthesized networks in the first seven generations of 5-parent sexual evolutionary synthesis, where the best synthesized networks are closest to the top left, i.e., high performance accuracy and low storage size. Networks synthesized using gene tagging (diamond

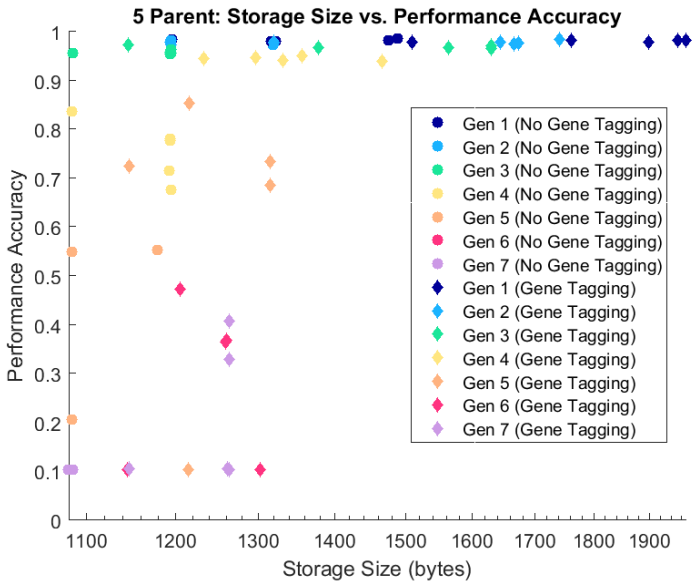


Figure 6.12: Performance accuracy as a function of storage size for the first seven generations of 5-parent sexual evolutionary synthesis for networks synthesized with gene tagging (diamond) and without gene tagging (round). Plots best viewed in colour.

plot points) show a slightly slower progression in maintaining performance accuracy while decreasing storage size relative to networks synthesized without gene tagging (round plot points), as can be seen by comparing generations 4 (yellow) and 5 (orange) in Figure 6.12.

Synthesizing networks with gene tagging and without gene tagging both produced architectures that increase in variability over successive generations; however, networks synthesized with gene tagging diversify more slowly than those without gene tagging (as shown in Tab. 6.1). Figure 6.12 and Tab. 6.1 also suggest that generations of networks approaching an optimal trade-off between performance accuracy and storage size tend to also have the highest architectural variability, e.g., in generations 3 and 4.

Lastly, it is worth noting that the increasing percentage overlap in generations 6 and 7 of networks synthesized without gene tagging is a result of sparse, low-variability architectures that can no longer represent the problem space (i.e., performance accuracy of 10% on the 10-class MNIST dataset, equivalent to random guessing). Similarly, the percentage overlap in generations 6 and 7 of networks synthesized with gene tagging increases as the performance accuracy begins to rapidly decrease. Conversely, there is unpredictability

Table 6.1: Average percentage overlap of architectural clusters in network models for the first seven generations of 5-parent sexual evolutionary synthesis. Note that the increasing percentage overlap in generations 6 and 7 of networks synthesized without gene tagging is a result of sparse, low-variability architectures that can no longer represent the problem space, while the unpredictability of percentage overlap in generations 6 and 7 of networks synthesized with gene tagging may be a result of some (but not all) networks having sparse, low-variability architectures.

Gen No.	**Gene Tagging**	**No Gene Tagging**
1	93.75%	93.71%
2	87.59%	78.11%
3	83.49%	68.84%
4	71.81%	66.64%
5	73.17%	68.44%
6	69.09%	82.74%
7	73.48%	91.05%

in percentage overlap in generations 6 and 7 of networks synthesized with gene tagging. Observing the distribution of architectures synthesized with gene tagging in generation 6 (pink diamond) and generation 7 (light purple diamond) shown in Figure 6.12, we speculate that this may be a result of some (but not all) networks in that generation having sparse,low-variability architectures.

In this section of the thesis, we presented a study in assessing architectural similarity between deep neural networks to improve the sexual evolutionary synthesis process. Results show that networks synthesized using gene tagging have less architectural variability than networks synthesized without gene tagging, as quantified by relatively higher overlap percentages of architectural clusters. This indicates that the use of gene tagging can potentially restrict the exploration of efficient network architectures in the search space.

6.4 Summary

In this chapter, we introduced a gene tagging system to explore the effect of architectural alignment and/or mismatch during multi-parent evolutionary synthesis. We evaluated gene tagging using 5-parent evolutionary synthesis over a range of environmental factor models, and found that networks synthesized using gene tagging appear to be minimally

worse at maintaining performance accuracy while minimizing storage size when compared to networks synthesized without gene tagging.

The introduction of gene tagging also introduced a percent of population parameter χ, which acts as an existence criterion where any given architectural cluster is only synthesized when it exists in a larger proportion of the parent networks than χ. The impact of χ was explored within the context of both 3-parent and 5-parent evolutionary synthesis. Results indicate that the optimal χ is one that allows for the synthesis of clusters that exist in some, but not all, of the parent networks. As such, we selected a percent of population parameter χ to be 0.5 as the optimal trade-off between maintaining performance accuracy over successive generations while reducing the model storage size.

Lastly, we explored how different neural network architectures can be compared. For the scope of this thesis, we limited the network architectures to those synthesized at each generation, and quantified the network architectural similarity in populations of synthesized networks via the percentage overlap of gene tagged architectural clusters. Results show that networks synthesized with gene tagging have a relatively higher overlap percentages of architectural clusters, implying that networks synthesized using gene tagging have less architectural variability than networks synthesized without gene tagging. This indicates that the use of gene tagging may be restricting the exploration of the efficient network architecture search space.

However, one key limitation of this architectural similarity assessment is that it is only applicable to network architectures that originate from the same ancestor network, as is the case in this thesis. Specifically, assessing the architectural similarity of networks is only enabled by the introduction of the gene tagging system, as it allows us to track which architectural clusters are being synthesized in successive generations (even when gene tagging is not actively being used during evolutionary synthesis). Thus, the architectural similarity assessment performed in this chapter is a preliminary step towards a more general method for comparing neural network architectures, but by no means solves the problem of comparing different network architectures trained from scratch. The question of how to assess network architectures that do not share a common ancestor network or inherit weight initialization from parent networks (i.e., we cannot ensure that we are comparing analogous parts of the networks) remains, and further investigation into both the quantification of architectural similarity and the associated implications on network variability will be discussed in future work (Chapter 9).

Chapter 7

Nature vs. Nurture: the Role of Environmental Resources

In the previous chapters, we explored the computational construct used to mimic heredity by studying the efficacy of incorporating multiple parent networks during the evolutionary synthesis process. In this chapter, we shift our focus to the computational construct used to simulate natural selection within the evolutionary deep intelligence framework and consider the environmental resource model.

We propose a study to determine the role of the simulated environmental resources during the evolutionary synthesis process and their effects on generations of synthesized neural networks. To do this, we leverage the evolutionary deep intelligence framework (as formulated in Chapter 3), using previously proposed cluster-driven genetic encoding [17] and m-parent evolutionary synthesis, and vary the availability of simulated external environmental resources.

7.1 The Role of Environmental Resources

Thus far, the effect of environmental factors (e.g., abundance or scarcity of resources) during the evolutionary synthesis process on generations of synthesized network architectures has largely been unexplored. Inspired by the nature versus nurture debate, we aim to better understand the role of external factors on the network synthesis process by varying the availability of simulated environmental resources during m-parent evolutionary synthesis.

In this chapter, we examine the computational construct used the mimic natural selection by studying the effects of the cluster-level environmental factor model $\mathcal{R}_g^c$ and the synapse-level environmental factor model $\mathcal{R}_g^s$ (as proposed in Section 3.6). From (3.10), we present a deterministic realization at the cluster level of the existence of all clusters $C \in \mathcal{C}$ as:

$$s_{g,C} = \begin{cases} 1 & \mathcal{R}_g^C \cdot (1 - |w_{G_i,C}|) \leq \gamma \\ 0 & \text{otherwise} \end{cases} \tag{7.1}$$

where the synthesis of a synaptic cluster $s_{g,C}$ incorporates both the strength of the synapses in a cluster ($w_{G_i,C}$) and the cluster-level environmental resources available ($\mathcal{R}_g^C$), and γ is a randomly generated number between 0 and 1.

Similarly from (3.11), we present the realization at the synapse level of the existence of all synapses as:

$$s_{g,j} = \begin{cases} 1 & \mathcal{R}_g^s \cdot (1 - |w_{G_i,j}|) \leq \gamma \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

where $s_{g,j}$ incorporates both the strength of each synapse ($w_{G_i,j}$) and the synapse-level environmental resources available ($\mathcal{R}_g^s$), and γ is a randomly generated number between 0 and 1. To better understand the role environmental resources play, we vary the cluster-level environmental factor model $\mathcal{R}_g^C$ and the synapse-level environmental factor model $\mathcal{R}_g^s$ at 5% increments. In the context of this thesis, we assume the cluster-level and synapse-level environmental factor models are equal, and enforce that $\mathcal{R}_g^C = \mathcal{R}_g^s$. As such, the cluster-level and synapse-level environmental factor models, $\mathcal{R}_g^C$ and $\mathcal{R}_g^s$, were varied simultaneously such that $\mathcal{R}_g^C = \mathcal{R}_g^s$ in every experiment, and can be represented using a single environmental factor model $\mathcal{R}_g$.

The m-parent evolutionary synthesis of deep neural networks was performed over multiple generations, and the effects of various environmental factor models were explored using 10% of the MNIST [72] hand-written digits dataset to increase the training speed of the synthesized network architectures as well as increase the inherently low intra-class variation within the MNIST dataset. The first generation ancestor networks trained using the LeNet-5 architecture [34]. Similar to Shafiee *et al.*'s work [17], each filter (i.e., collection of kernels) was considered as a synaptic cluster in the multi-factor synapse probability model, and we assessed the synthesized networks using performance accuracy on the MNIST dataset and storage size (representative of the architectural efficiency of a network) of the networks with respect to the computational time required.

7.2 Preliminary Range Showing Stochasticity

This section presents a preliminary study into a range of environmental factor models using the method proposed in previous evolutionary deep intelligence works [16–18]. Specifically, this section demonstrates the inherent stochasticity characteristic of these past works, and allows for a clear comparison against the methods used in this thesis (as presented in the following section) when evaluated using a range of environmental factor models. Of particular interest is the difference in the variability of trends for 5-parent evolutionary synthesis shown in this section (i.e., with the inherent stochasticity of past works [16–18]) in comparison to the method proposed in this thesis (see Figure 7.2).

Previous studies in evolutionary deep intelligence [16–18] employed environmental factor models of $\mathcal{R}_g^c = 70\%$ and $\mathcal{R}_g^s = 70\%$, i.e., the probability of synthesis for a given cluster and a given synapse in an offspring network during the evolutionary synthesis process was scaled by 70%. As such, we selected a preliminary range of environmental factor models to be from 50% to 95% at 5% increments, i.e.,:

$$\mathcal{R}_g^C, \mathcal{R}_g^s = \{50, 55, 60, 65, 70, 75, 80, 85, 90, 95\}\% \tag{7.3}$$

In this preliminary study, we evaluate these environmental factor models in the context of 1-parent evolutionary synthesis (as the most basic) and 5-parent evolutionary synthesis (as the most complex and variable).

The top row of Figure 7.1 shows the performance accuracy and storage size for 1-parent evolutionary synthesis using various environmental factor models, and shows a clear monotonic trend with respect to the environmental factors. As the environmental factor models decrease, the rate of decrease in storage size of synthesized networks slows accordingly and the rate of performance accuracy loss over generations is similarly slowed. As such, the top row of Figure 7.1 shows that networks synthesized with environmental factors of 50% (dark blue) has the most gradual decrease in performance accuracy and storage size, while networks synthesized with environmental factors of 90% (light purple) or 95% (dark purple) have the steepest decrease in performance accuracy and storage size over successive generations.

One aspect to note in Figure 7.1 is the bottom plateau in performance accuracy at 10% (most apparent in the 1-parent evolutionary synthesis plots); this is due to the MNIST dataset [72] itself which consists of 10 classes of individual handwritten digits (i.e., digits 0 to 9), and the 10% performance accuracy is akin to random guessing. There is similarly a bottom plateau in network storage size that is most easily seen in the trends for environmental factors of 90% (light purple) and 95% (dark purple). This plateau corresponds to

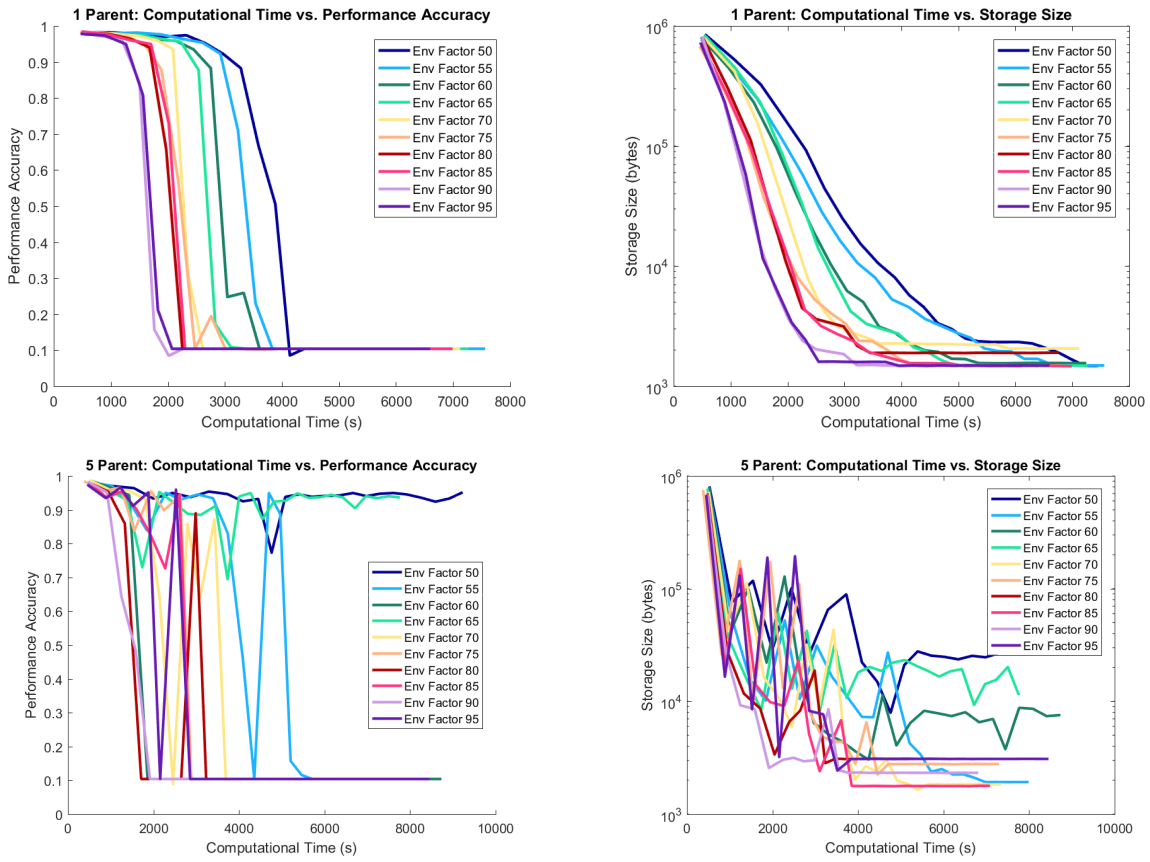


Figure 7.1: Performance accuracy (left) and storage size (right) with respect to computational time for 1-parent (top row) and 5-parent (bottom row) evolutionary synthesis using various environmental factor models. Networks synthesized via 1-parent evolutionary synthesis and subject to various environmental factors show a clear monotonic trend, while networks synthesized via 5-parent evolutionary synthesis are difficult to interpret due to the variability in the trends. Plots best viewed in colour.

networks where there is only a single synapse in each convolution layer and, thus, cannot be reduced any further. This network storage size reduction of approximately three orders of magnitude is a result of using the LeNet-5 [34] architecture as the first generation ancestor network, and there is no inherent limit in network reduction when using the evolutionary deep intelligence approach.

The bottom row of Figure 7.1 shows the performance accuracy and storage size for 5-parent evolutionary synthesis using various environmental factor models. There is noticeably more variability in both performance accuracy and storage size with the increased number of parent networks, making the trends difficult to interpret; this is likely a result of only one synthesized network being represented per generation in 5-parent evolutionary synthesis. As well, there is inherent stochasticity present from the implementations from the previous realizations of evolutionary deep intelligence [16–18] that was removed in this thesis, and the difference in variability of the trends can be seen in comparison with the next section.

7.3 Full Range of Environmental Resources

As stated in the previous section and at the end of Chapter 5, there is inherent stochasticity from previous evolutionary deep intelligence works [16–18] that resulted in increasing levels of noise in the efficiency trends as the number of parent networks used during evolutionary synthesis increased. This variability (as shown in Figure 7.1) makes it difficult to draw conclusions regarding the effects of the environmental resource model. As a result, the ad hoc stochasticity was removed in the context of this thesis to allow for cleaner and more easily interpretable trends in network efficiency over generations.

In this section, we revisit the environmental factor model by testing the full range of models from 5% to 95% at 5% increments, and re-evaluating its effect in more stable synthesis conditions. As the previous section found that the best network architectures were synthesized using the lowest tested environmental factor models, we propose the full range of environmental factor models evaluated to be:

$$\mathcal{R}_g^C, \mathcal{R}_g^s = \{05, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 95\}\%. \tag{7.4}$$

For this experiment, the full range of environmental factor models is evaluated only for the most complex case (i.e., 5-parent evolutionary synthesis) using the gene tagging structural alignment from Chapter 6, and the environmental factor models were tested using a percent of population parameter $\chi = 0.5$, i.e., only clusters that exist within at least half of the parent networks are synthesized.

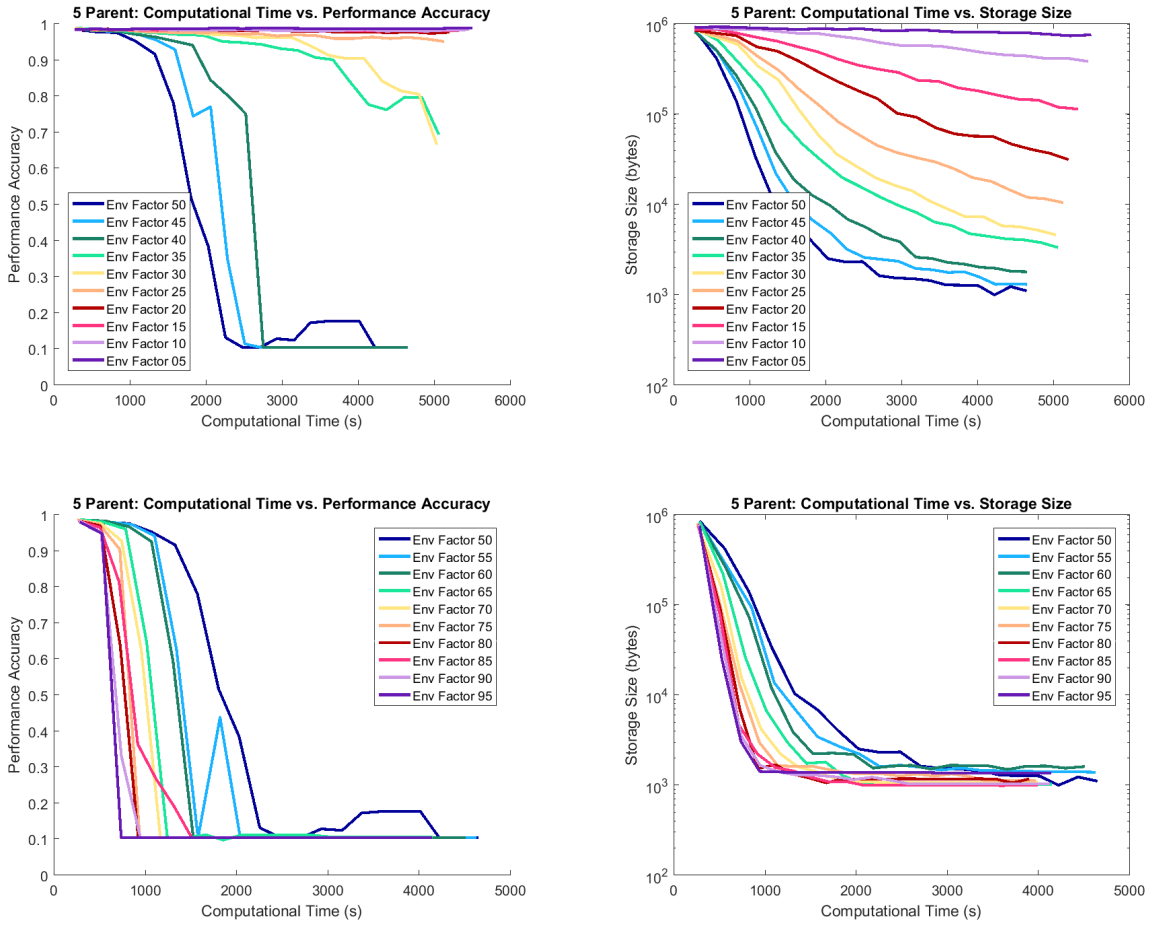


Figure 7.2: Performance accuracy (left) and storage size (right) with respect to computational time for 5-parent evolutionary synthesis using the full range of environmental factor models. The top row shows the accuracy and storage size for environmental factor models 5% to 50%, and the bottom row shows the plots for environmental factor models 50% to 95%. Environmental factor model 50% (dark blue) is shown on both sets of plots for comparison purposes. Note that the trends have much less variability than those shown in Figure 7.1. Plots best viewed in colour.

Figure 7.2 shows the performance accuracy (left) and storage size (right) with respect to computational time for 5-parent evolutionary synthesis using the full range of environmental factor models. The top row shows the accuracy and storage size for environmental factor models 5% to 50% and the bottom row shows the plots for environmental factor models 50% to 95%, with environmental factor model 50% (dark blue) shown on both sets of plots for comparison purposes. Note that the performance accuracy tends to reach a minimum of 10%, which is equivalent to random guessing given the 10-class MNIST dataset. Similarly, the storage size reaches a minimum of approximately 1 KB, which corresponds to a network architecture with only one convolutional kernal at each layer.

Figure 7.2 shows clear monotonic trends similar to those seen when evaluating the preliminary range of environmental factor models using 1-parent evolutionary synthesis (see top row of Figure 7.1), indicating that removing the inherent stochasticity from past evolutionary deep intelligence works [16–18] has correspondingly removed much of the variability associated with increasing the number of parent networks during synthesis seen in the previous section (as shown in the bottom row of Figure 7.1). This allows for an easier interpretation of the performance accuracy and storage size trends with respect to various environmental factor models.

While the shape of the plots for environmental factor models 50% to 95% are as expected given the previous section, the trends for environmental factor models 5% to 50% present more insight into how network performance changes over successive generations given more gradual decreases in simulated environmental resources. For environmental factor models 5% to 25%, we see that there is little to no decrease in performance accuracy over the generations (Figure 7.2, top left) while the storage size decreases (Figure 7.2, top right); however, we do see steeper decreases in storage size with increasing environmental factor models, indicating that decreasing the amount of available resources produces more efficient network architectures.

Most interesting are environmental resource models 30%, and 35%, as these parameters appear to allow for the synthesis of network architectures on the cusp of losing the computational complexity required to model the problem space. In particular, note that the networks synthesized at the latest generations have a performance accuracy of approximately 70% to 80% while the corresponding storage size has decreased by two orders of magnitude. This indicates that environmental resource models 30% and 35% can perhaps best capture the optimal tradeoff between performance accuracy and storage size.

To better evaluate this tradeoff between accuracy and storage size, we can plot the performance accuracy of the synthesized networks as a function of storage size. Figure 7.3 shows an overview of the network architectures synthesized using different environmental

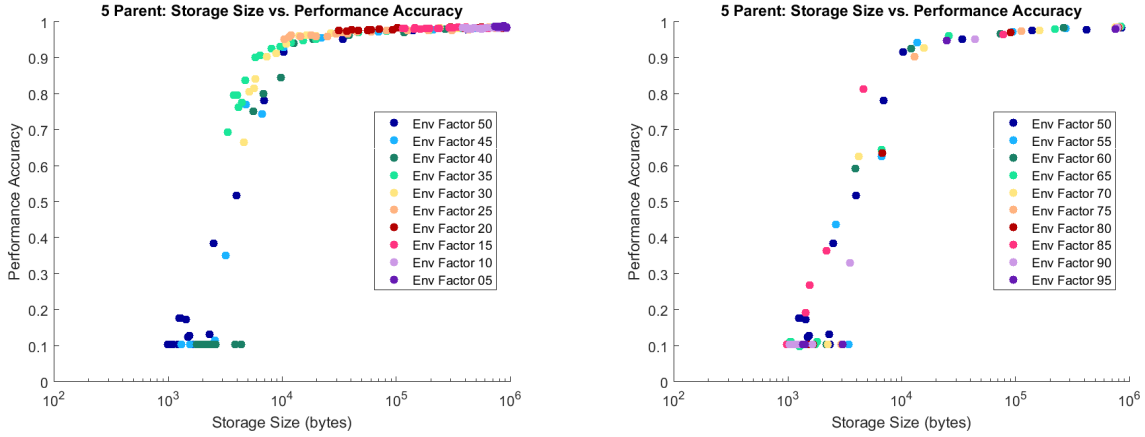


Figure 7.3: Performance accuracy as a function of storage size for 5-parent sexual evolutionary synthesis using the full range of environmental factor models. Split over two plots with environmental factor model 50% (dark blue) as overlap, environmental factor model 35% (light green, left) achieves the best accuracy and storage size trade-off. Plots best viewed in colour.

factor models split over two plots, with the best networks closest to the top-left corner, i.e., the highest accuracy for the lowest storage size. Environmental factor models 5% to 50% are shown on the left plot, and environmental factors models 50% to 95% are shown on the right; environmental factor model 50% is shown on both plots in the same colour (dark blue) to allow for easier comparison between the plots.

As speculated given Figure 7.2, environmental resource models 30% and 35% appear to synthesize the best network architectures. By comparing the synthesized networks on the two plots, the results shown in Figure 7.3 indicate that an environmental factor model of 35% (light green on the left plot) achieves the best performance accuracy to storage size trade-off, with networks closest to the top-left corner.

We can also observe that environmental factor models 30% and below maintain a higher performance accuracy, but lose accuracy more quickly with decreasing storage size. This results in a steep accuracy drop-off, and indicates that too low of an environmental factor model (i.e., too slow a decrease in simulated resources) fails to produce networks that can compensate for the loss in accuracy given decreasing storage sizes.

7.4 Summary

In this chapter, we investigated the computational construct used to mimic natural selection via the availability and/or scarcity of simulated environmental resources. The effect of these simulated resources was explored in two experiments. First, we evaluated a preliminary range of environmental factor models (based on the environmental factor models used in previous evolutionary deep intelligence works [16–18]) in the context of 1-parent and 5-parent evolutionary synthesis, and show that the inherent stochasticity from previous evolutionary deep intelligence works resulted in high variability trends 5-parent synthesis. The effect of these simulated resources was then explored using a full range of environmental factor models in more stable synthesis conditions after the inherent stochasticity from previous works was removed. The experiment evaluating the full range of environmental factor models leveraged the gene tagging structural alignment from in Chapter 6, and were evaluated using a percent of population parameter $\chi = 0.5$.

The results of the full range of environmental factor models were evaluated in terms of performance accuracy and storage size with respect to computational time, showing clear trends of increasingly gradual loss of both accuracy and storage size with decreasing environmental resource models. Most interestingly, the performance accuracy of the synthesized models were plotted as a function of storage size, and environmental factor model 35% was selected as the optimal environmental factor model with the corresponding synthesized networks achieving the best performance accuracy to storage size trade-off.

Chapter 8

Performance Comparison

In this chapter, we compare the networks synthesized via multi-parent evolutionary synthesis using two benchmark datasets. Based on the experiments from previous chapters in this thesis, we evaluate multi-parent evolutionary synthesis using the following:

- 5-parent evolutionary synthesis (Chapter 5)
- Networks synthesized with gene tagging (Chapter 6)
- Percent of population parameter χ as 0.5 (Chapter 6)
- Environmental factor model 35% (Chapter 7).

We evaluate the efficacy of multi-parent evolutionary synthesis using the Matlab version of the LeNet [34] architecture (32-32-64-64-10). This is the architecture used in Chapters 5, 7, and 6 due to its simplicity and ease of training, making it ideal for synthesizing multiple networks over successive generations. We also evaluate MNIST using the LeNet5-Caffe architecture (20-50-800-500), implemented in Matlab to the best of our abilities, and compare against other state-of-the-art network efficiency methods. In the previous chapters, we also trained using only a random 10% subset of the MNIST dataset to further minimize training times; however, in this chapter, we train using the full MNIST dataset [72] and CIFAR-10 dataset [73]. Experiments using the MNIST dataset and the CIFAR-10 dataset are described in sections 8.1 and 8.2, respectively.

Figure 8.1: Sample images from the MNIST hand-written digits dataset [72]. The MNIST dataset contains 10 different digits with grey scale images being 28×28 in size.

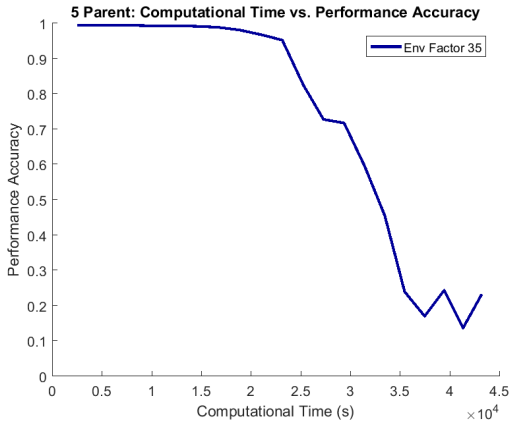
8.1 MNIST Dataset

In this section, we evaluate multi-parent evolutionary synthesis using the MNIST dataset [72] of 10 different hand-written digits using two different network architectures: a version of the LeNet architecture (as described in 8.1.1), and the LeNet5-Caffe architecture (8.1.2). Note that both architectures are relatively simple with only four layers, and can train to classify the MNIST dataset with over 99% accuracy.

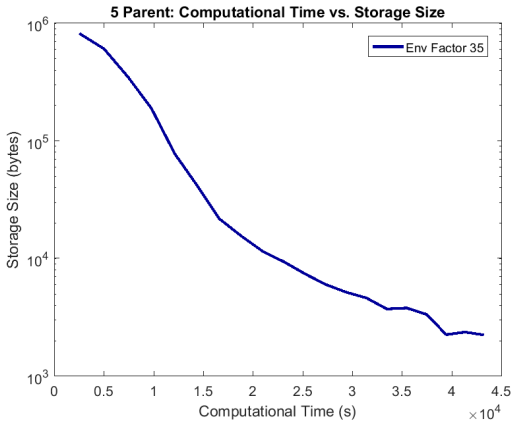
8.1.1 LeNet

First, we evaluate the efficacy of multi-parent evolutionary synthesis using the Matlab version of the LeNet [34] architecture (32-32-64-64-10). This is the architecture used in Chapters 5, 7, and 6 due to its simplicity and ease of training, making it ideal for synthesizing multiple networks over successive generations. In the previous chapters, we also trained using only a random 10% subset of the MNIST dataset to further minimize training times; however, in this section, we train using the full MNIST dataset to achieve benchmark performance accuracy, and evaluate the corresponding trends in performance accuracy and storage size.

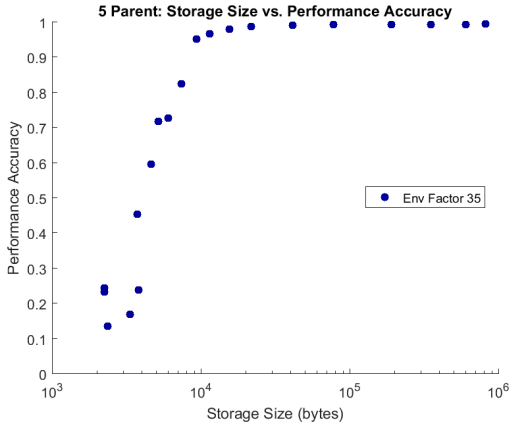
Figure 8.2 shows the performance accuracy (top left) and storage size (top right) as a function of computational time for networks synthesized via 5-parent evolutionary synthesis with an environmental factor of 35%. Both the performance accuracy and storage size trends are reminiscent of those seen in the previous chapters of this thesis. Figure 8.2 also presents the performance accuracy as a function of model storage size (bottom) for networks synthesized via 5-parent evolutionary synthesis with an environmental factor of 35%. Note that, as seen in previous chapters, we achieve a storage size reduction of approximately two orders of magnitude before the performance accuracy rapidly drops off. By counting the data points in Figure 8.2 (c), we can conclude that the accuracy steeply decreases after



(a) MNIST performance accuracy.



(b) MNIST network storage size.



(c) MNIST accuracy vs. storage size.

Figure 8.2: MNIST performance accuracy (top left) and storage size (top right) as a function of computational time for networks synthesized via 5-parent evolutionary synthesis with an environmental factor of 35% using the MNIST dataset [72]. The bottom row shows performance accuracy as a function of storage size of networks synthesized

Table 8.1: Experimental results for the LeNet architecture with the MNIST dataset. The original architecture for this version of the LeNet is 32-32-64-64, and synthesized networks via 5-parent evolutionary synthesis at different generations are shown for comparison.

Method	$\frac{w \neq 0}{\lvert W \rvert}\%$	**Error%**	Architecture
Gen 0 (Ancestor)	100.0	0.55	32-32-64-64
Gen 7 (Env 35%)	1.31	1.11	17-20-32-43
Gen 8 (Env 35%)	0.59	2.71	17-23-34-42
Gen 9 (Env 35%)	0.37	3.22	18-19-26-36
Gen 10 (Env 35%)	0.23	4.76	13-17-25-31

generation 10, indicating that the optimal network architectures are synthesized at, or just before, generation 10.

While the trends in Figure 8.2 are as expected, Table 8.1 presents a closer look into the synthesized network architectures. Table 8.1 shows the experimental results for the LeNet architecture with the MNIST dataset, and compares the networks synthesized via 5-parent evolutionary synthesis at generations 7, 8, 9, and 10 to the original fully-trained ancestor network (generation 0). We can clearly see the trade-off between performance accuracy and network efficiency; for example, the network at generation 7 is approximately 76.3× more efficient than the ancestor network with an error of 1.11%, while the network at generation 10 is approximately 434.8× more efficient than the ancestor network, but has an error of 4.76%.

8.1.2 LeNet5-Caffe

Here, we assess the efficacy of multi-parent evolutionary synthesis using the LeNet5-Caffe network architecture by performing a comparative analysis of the proposed method against eight state-of-the-art methods for achieving efficient DNNs [21,49,51–53,68]. The choice of leveraging the LeNet5-Caffe network architecture with MNIST as a means of benchmarking across different methods is motivated by this particular architecture-dataset configuration being the most commonly used across literature for cross-method comparisons.

A number of observations can be made from Table 8.2, which shows both the percentage of non-zero weights within the efficient DNNs produced by the eight different tested methods, as well as the corresponding test error. First of all, while the proposed multi-parent evolutionary synthesis method does not outperform the other state-of-the-art methods, it is

Table 8.2: Experimental results for the LeNet5-Caffe architecture with the MNIST dataset. Some of the results were derived from the original authors of [53] and [68].

Method	$\frac{w \neq 0}{\|W\|}\%$	**Error%**
DC [21]	8.0	0.7
DNS [49]	0.9	0.9
SWS [51]	0.5	1.0
Sparse VD [52]	0.7	1.0
BC-GNJ [53]	0.9	1.0
BC-GHS [53]	0.6	1.0
StressedNets#1 [68]	0.8	1.2
StressedNets#2 [68]	0.3	1.6
Asexual (Gen 9) [17]	3.1	1.2
Proposed (Gen 5)	1.9	1.3

regardless able to produce a very efficient neural network architecture with just 1.9% non-zero weights while still achieving a test error of 1.3%. Second, it is interesting to observe that, when compared to deep compression [21], the DNN produced using the proposed multi-parent evolutionary synthesis method is able to achieve significantly fewer non-zero weights (4× fewer than deep compression) with a trade-off of 0.6% increase in test error. This is particularly worth noting given that deep compression results in network architectures with only synapse sparsity while the proposed method results in architectures with both synaptic cluster sparsity and synapse sparsity. Third, it is interesting to observe that, when compared to the asexual evolutionary synthesis approach [17], the proposed m-parent sexual evolutionary synthesis method is able to achieve additional reductions in non-zero weights (1.6× fewer than the asexual approach) at a comparable test error, indicating the merit in leveraging multiple parents during the evolutionary synthesis process proposed in this thesis.

8.2 CIFAR-10 Dataset

In this section, we evaluate the efficacy of the multi-parent evolutionary synthesis method using the CIFAR-10 dataset [73], an object classification dataset containing 10 different classes, and we leverage a version of the LeNet architecture (as described at the beginning of this chapter) for this experiment. Note that while this architecture is insufficiently

Figure 8.3: Sample images from the CIFAR-10 object classification dataset [73]. The CIFAR-10 dataset contains 10 different classes with colour images being 32×32 in size.

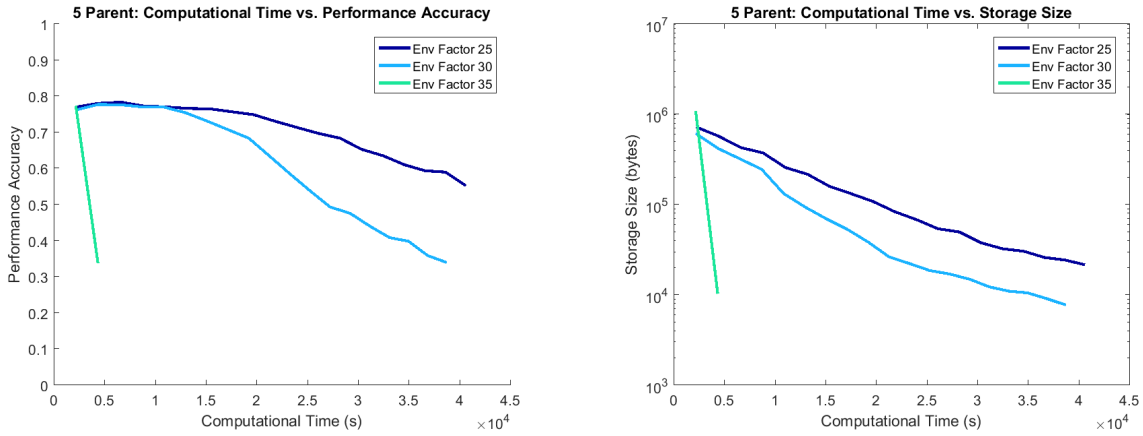


Figure 8.4: Performance accuracy (left) and storage size (right) as a function of computation time for networks synthesized via 5-parent evolutionary synthesis using the CIFAR-10 dataset [73]. Environmental factor models of 25% (dark blue), 30% (light blue), and 35% (green) were evaluated. Plots best viewed in colour.

complex to achieve state-of-the-art performance on the CIFAR-10 dataset, we can still observe trends in both the performance accuracy and model storage size over successive generations of multi-parent evolutionary synthesis.

Figure 8.4 shows the performance accuracy (left) and the model storage size (right) of synthesized networks as a function of computation time. The networks were synthesized via 5-parent evolutionary synthesis on the CIFAR-10 dataset using environmental factor models of 25% (dark blue), 30% (light blue), and 35% (green). The original environmental factor model of 35% (green) proved to be too aggressive given the more complex CIFAR-10 dataset, showing a rapid drop in performance accuracy and storage size in one generation. Thus, less aggressive environmental factor models (i.e., 25% and 30%) were also evaluated.

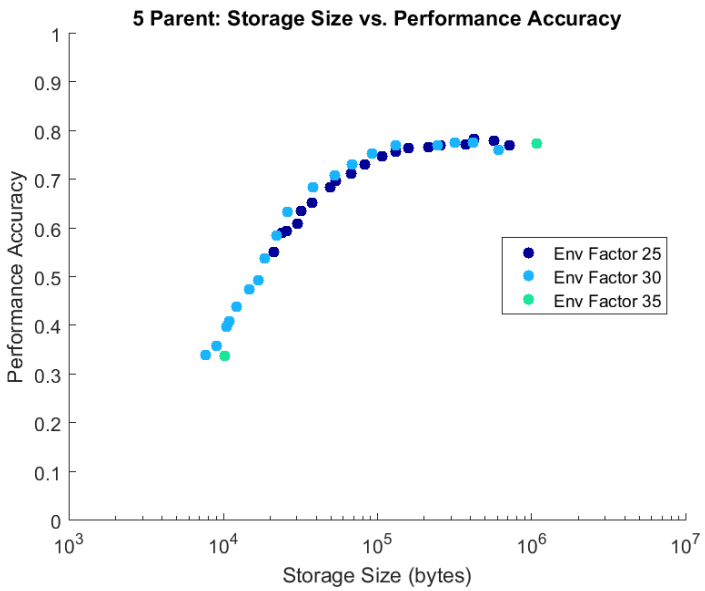


Figure 8.5: Performance accuracy as a function of storage size of networks synthesized via 5-parent evolutionary synthesis using the CIFAR-10 dataset [73]. Environmental factor models of 25% (dark blue), 30% (light blue), and 35% (green) were evaluated. Plots best viewed in colour.

Figure 8.5 shows the performance accuracy as a function of storage size for networks synthesized using 5-parent evolutionary synthesis on the CIFAR-10 dataset. As expected given Figure 8.4, an environmental factor model of 35% (green) is too aggressive given this dataset, and fails to produce any network architectures near the optimal trade-off between performance accuracy and storage size (represented by top-left corner of the plot). Environmental factor models of 25% (dark blue) and 30% (light blue), however, allow for the synthesis of networks that maintain accuracy while reducing storage size, with environmental factor model 30% synthesizing slightly better networks. In both cases, there is only a model storage size reduction of approximately one order of magnitude before the performance accuracy begins to drop off as well.

Table 8.3 shows the experimental results for the LeNet architecture with the CIFAR-10 dataset using the matconvnet version of the LeNet architecture. The synthesized networks via 5-parent evolutionary synthesis at generation 6 (environmental factor 30%) and generation 9 (environmental factor 25%) are shown for comparison. Note that we can see to effects of a lower environmental factor model, as the network synthesized using environmental factor 25% is at a later generation (generation 9) and is approximately 11.1× more

Table 8.3: Experimental results for the LeNet architecture with the CIFAR-10 dataset. The original architecture for this version of the LeNet is 32-32-64-64, and synthesized networks via 5-parent evolutionary synthesis at generation 6 (environmental factor 30%) and generation 9 (environmental factor 25%) are shown for comparison.

Method	$\frac{w \neq 0}{\lvert W \rvert}\%$	**Error%**	Architecture
Gen 0 (Ancestor)	100.0	22.65	32-32-64-64
Gen 9 (Env 25%)	9.0	24.29	31-23-59-49
Gen 6 (Env 30%)	13.5	23.01	29-30-55-54

efficient than the ancestor network, while the network synthesized using environmental factor 30% is at a relatively earlier generation (generation 6) and is approximately 7.4× more efficient. Similar to the LeNet5-Caffe experiment, this implies that a lower environmental factor model allows for more gradual decreases in accuracy and storage size, and can produce overall more efficient networks.

Chapter 9

Conclusion

In this thesis, we present a general formulation of the evolutionary deep intelligence framework and investigate the use of multiple parents during the evolutionary synthesis process, in addition to exploring the computational constructs used to mimic heredity and natural selection, for the purposes of highly efficient deep neural networks.

9.1 Summary of Contributions

9.1.1 m-Parent Evolutionary Synthesis

Chapter 3 presented a general framework for the evolutionary synthesis of new network architectures given existing neural networks. Abstracting away from the low-level design of neural networks, the framework formulates a network architecture as a set of possible neurons and a set of possible synapses, with associated weights to denote the strength of synaptic connectivity. Thus, we can leverage the computational construct used to mimic heredity by encoding a dependency on existing networks into the synthesis probability of a new network architecture. We can also incorporate the computational construct used to mimic natural selection by similarly encoding a dependency on an environmental factor model into the synthesis probability. By imposing constraints on this general framework, we showed that the cases of asexual evolutionary synthesis, 2-parent sexual evolutionary synthesis, and multi-parent (i.e., m-parent) evolutionary synthesis can all be realized.

The efficacy of multi-parent evolutionary synthesis was shown in Chapter 4 and Chapter 5 via two experiments. The first experiment (Chapter 4) compared asexual evolutionary

synthesis with 2-parent sexual evolutionary synthesis, and introduced a basic mating function for combining two parent networks. The second experiment (Chapter 5) extended this mating function to the general m-parent case, and assessed multi-parent evolutionary synthesis in the context of varying population sizes. Chapter 5 also introduced a more flexible mating policy relative to Chapter 4 via the use of a percent of population parameter χ. The statistical significance of the results was assessed in Chapter 4 and Chapter 5 via the multivariate analysis of variance (MANOVA) method, and both experiments indicated that incorporating multiple parent networks during the evolutionary synthesis process accelerated the rate at which successive offspring networks increased in efficiency.

9.1.2 Gene Tagging

Chapter 6 presented a gene tagging system to investigate the effect of architectural alignment and mismatch during evolutionary synthesis, and was evaluated using 5-parent evolutionary synthesis with a range of environmental factor models. The impact of χ was also assessed using both 3-parent and 5-parent evolutionary synthesis, and results indicated that the optimal percent of population χ is one that enforces that only architectural clusters that exist in some, but not all, parent networks are synthesized in an offspring network. While generally comparable, a slight drop in performance for networks synthesized with gene tagging relative to networks synthesized without gene tagging raised questions regarding whether gene tagging produced networks with less architectural variability and, more importantly, how the architectural similarity of neural networks can be assessed in a meaningful and useful way.

Thus, Chapter 6 also presented a brief study exploring the quantification of network architectural similarity via the percentage overlap of architectural clusters. Leveraging the gene tagging system introduced in this chapter, percentage overlap is indicative of network population diversity as low overlap would correspond to relatively high architectural variability among the networks being compared. Results show that networks synthesized with gene tagging tend to have higher percentage overlap, and imply that the use of gene tagging can potentially restrict the exploration of the search space of efficient network architectures during evolutionary synthesis.

9.1.3 Role of Environmental Resources

Chapter 7 presented an extensive analysis of the environmental resource model, the computational construct used to mimic natural selection via the simulated availability of com-

putational resources, and its effects during the evolutionary synthesis process. First, a preliminary range of environmental factor models based on previously used values in literature was evaluated in the context of asexual and 5-parent evolutionary synthesis to demonstrate the stochasticity inherent in previous evolutionary deep intelligence works. As the best networks tended to be synthesized using the lowest environmental factor models tested, a full range of environmental factor models were then revisited. Both experiments showed clear trends of increasingly gradual drops in performance accuracy and model storage size with decreasing environmental resource models, and an environmental factor model of 35% was selected as the optimal parameter to achieve an effective performance accuracy to storage size trade-off.

9.2 Limitations

The main limitation of this thesis is the up-front computational costs associated with synthesizing entire populations of neural networks over successive generations. As multi-parent evolutionary synthesis requires a minimum of m networks to be synthesized at each generation, increasing the number of parent networks used to synthesized offspring networks rapidly becomes too computationally costly to be feasible. In this thesis, we leverage a relatively simple network architecture (32-32-64-64-10) [34] for our exploratory experiments and set $m = 5$ to be the upper limit of parent networks used during evolutionary synthesis to keep the computational cost of repeated experiments tractable. Obviously, this also raises questions of scalability given the simplicity of these neural networks. While not presented in this thesis, recent evolutionary deep intelligence work [68] has applied the asexual evolutionary synthesis method to AlexNet [5] for CIFAR-10 image classification and YOLOv2 [77] for object detection, indicating that multi-parent evolutionary synthesis should be viable for smaller network architectures meant to be run on edge devices.

As seen in Chapter 8, networks synthesized via the multi-parent evolutionary synthesis process show notable increases in network efficiency for the matconvnet implementation of the LeNet architecture [34] when trained on both the MNIST and CIFAR-10 datasets. We also evaluate (to the best of our ability) the efficacy of multi-parent evolutionary synthesis using the LeNet5-Caffe network architecture, and compare against seven state-of-the-art methods for achieving efficient DNNs [21, 49, 51–53, 68]. Note that while [21, 49, 51–53, 68] used the Caffe implementation of this network directly, a Matlab version was implemented for comparison in this thesis as closely as possible, but fails to leverage some of the optimization intrinsic to the Caffe implementation. As a result, a version of the LeNet5-Caffe architecture is used for comparison in this section. While the proposed method performs

somewhat comparably to (but fails to outperform) state-of-the-art algorithms when applied to the LeNet5-Caffe network architecture, the performance of StressedNets [68] is promising, as this is a more recent realization of asexual evolutionary deep intelligence. The modifications that boost the performance of StressedNets are complementary to the contributions of this thesis, and it is likely that the combination of these modifications with multi-parent evolutionary synthesis can produce larger gains in architecture efficiency than the asexual StressedNets case.

The last limitation of multi-parent evolutionary synthesis process is that it is somewhat sensitive to how aggressive the environmental factor model is. While an environmental factor models of 35% was selected as optimal in Chapter 7 and used in section 8.1, it proved to be too aggressive in combination with the CIFAR-10 dataset. As a result, less aggressive environmental factor models (i.e., 25% and 30%) were also evaluated, and successfully synthesized networks with notable gains in network efficiency with a less than 2% drop in accuracy. However, this implies that an appropriate environmental factor model may need to be selected for each network and dataset combination.

Limitations in parameter sensitivity and up-front computational costs aside, the experiments in this thesis allow for a deeper understanding of the evolutionary synthesis framework via the exploration of the computation constructs used to mimic heredity and natural selection. In addition, some key insights into the implications of structural similarity between network architectures and its impact on navigating the search space of efficient network architectures are presented.

9.3 Future Work

One key area of future work is the further investigation into the quantification of architectural similarity between networks. The percentage overlap metric proposed in Chapter 6 is rudimentary, and more established quantities of information can be leveraged to better compare different network architectures both within and outside the scope of evolutionary deep intelligence. In particular, metrics inspired by information theory will be explored to simultaneously encourage minimizing overlapping performance strengths between networks (mutual information) and the better compaction of individual network architectures (self-information). With respect to this thesis, a better quantification of architectural similarity can hopefully lead to the development of a custom similarity metric for optimal architectural similarity during multi-parent evolutionary synthesis.

A potential application of the architectural similarity metric is the development of a

parent network selection scheme to better determine which networks should produce offspring networks at each generation. Through this, we can incorporate the idea of "survival of the fittest" for parent network selection given a population of networks. In addition to selecting which candidate parent networks will mate to produce offspring networks, we can leverage different selection criteria to explore how best to combine the candidate parent networks to optimize population performance and diversity. The development of such a set of criteria can allow for the selection of candidate networks purely on performance and/or efficiency and combining the "fittest" networks, as well as investigating less conventional combinations of candidates (e.g., "fittest" network with a relatively weak network) to study its effect on network population over generations.

Given that populations of neural networks are synthesized at each generation of the evolutionary synthesis process, a natural extension of this work would be to investigate the combined performance of multiple neural networks (e.g., an ensemble). In combination with the other future work, the impact of structural similarity and selection criteria should be systematically investigated by assessing the inference accuracy, computational complexity, and domain generalizability of both individual networks and ensembles of networks. While predicting a networks performance accuracy prior to training has proved to be one of the largest challenges in deep learning, studying the impact of architectural structure on network performance can potentially allow for the development of predictive performance metrics in the future.

Lastly, we present a preliminary experiment investigating the impact of architectural cluster inheritance on number of required training epochs both with and without gene tagging. To better understand how performance accuracy can be recovered through training, each network was synthesized at a generation where performance accuracy was just beginning to drop off. In the gene tagging case, the networks were synthesized at generation 13 via 5-parent evolutionary synthesis (environmental factor model 35%, $\chi = 0.5$); in the no gene tagging case, the networks were synthesized at generation 8 via 5-parent evolutionary synthesis (environmental factor model 35%, $\chi = 0.5$).

Figure 9.1 shows the validation accuracy as a function of training epochs for networks synthesized with gene tagging (dark and light blue) and without gene tagging (dark red and pink). In particular, Figure 9.1 compares the validation accuracy at each training epoch for networks synthesized using inherited starting weights from the corresponding architectural clusters in the parent networks (dark blue, dark red) against networks that inherit the architectural cluster structure, but begin training with randomly initialized weights (light blue, pink).

Note that there are clear advantages in initializing the synthesized architecture with

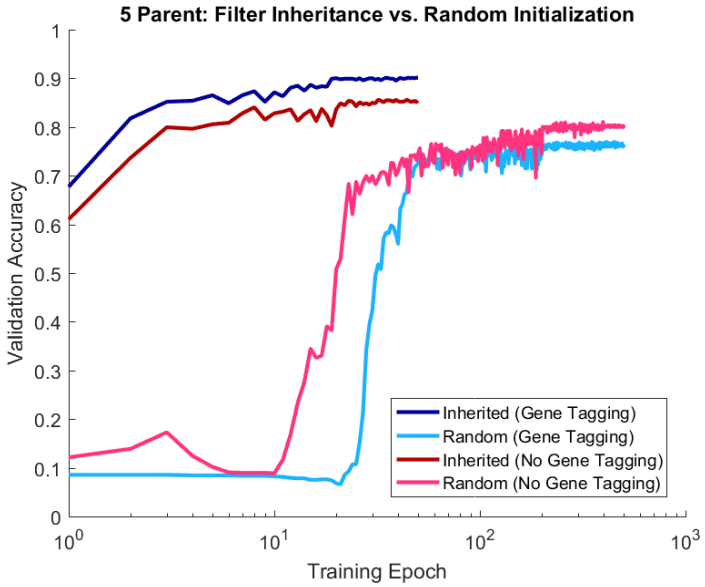


Figure 9.1: Validation accuracy as a function of training epochs for network architectures synthesized with gene tagging (dark and light blue) and without gene tagging (dark red and pink) with inherited starting weights (dark blue, dark red) and randomly initialized weights (light blue, pink).

the corresponding architectural cluster weights from the parent networks, as the networks initialized with inherited weights recover most of the validation accuracy within the first ten training epochs. Conversely, the networks that begin training with randomly initialized weights do not recover most of the validation accuracy until after approximately 50 training epochs. Interestingly enough, the difference in recovering validation accuracy over training epochs between networks initialized using inherited weights and networks with randomly initialized weights is more pronounced in the gene tagging case. As such, we speculate that there are potentially deeper implications to structural alignment during evolutionary synthesis that may result in network architectures that are more sensitive to training conditions, and merits further investigation.

While the proposed future work largely focuses on understanding the effects of structural similarity of network architectures, the aim of this research is to enable a deeper understanding of neural network architectures and the corresponding impact on model performance. This deeper understanding can lead to improved network transparency and interpretability. By unraveling the complexity behind understanding and improving deep neural network architectures, we can develop better and more systematic approaches to cre-

ating suitable deep neural networks for new and/or data-limited applications, and develop more guided methods for boosting inference performance using ensembles of networks.

References

[1] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[4] Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*, pages 1799–1807, 2014.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[8] Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*, pages 196–201. IEEE, 2011.

[9] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[10] Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8614–8618. IEEE, 2013.

[11] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in Neural Information Processing Systems*, pages 1135–1143, 2015.

[12] NVIDIA. Jetson Nano Developer Kit and Modules, 2020.

[13] Raspberry Pi Foundation. Raspberry Pi 4 Tech Specs, 2020.

[14] Mark Horowitz. 1.1 computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14. IEEE, 2014.

[15] Zhong Qiu Lin, Brendan Chwyl, and Alexander Wong. Edgesegnet: A compact network for semantic segmentation. *arXiv preprint arXiv:1905.04222*, 2019.

[16] Mohammad Javad Shafiee, Akshaya Mishra, and Alexander Wong. Deep learning with darwin: Evolutionary synthesis of deep neural networks. *arXiv preprint arXiv:1606.04393*, 2016.

[17] Mohammad Javad Shafiee and Alexander Wong. Evolutionary synthesis of deep neural networks via synaptic cluster-driven genetic encoding. In *Advances in Neural Information Processing Systems*, 2016.

[18] Mohammad Javad Shafiee, Elnaz Barshan, and Alexander Wong. Evolution in groups: A deeper look at synaptic cluster driven evolution of deep neural networks. *arXiv preprint arXiv:1704.02081*, 2017.

[19] James F Crow and Motoo Kimura. Evolution in sexual and asexual populations. *American Naturalist*, pages 439–450, 1965.

[20] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[21] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR, abs/1510.00149*, 2, 2015.

[22] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[23] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. *Automated Machine Learning: Methods, Systems, Challenges*. Springer, 2018. In press, available at http://automl.org/book.

[24] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[25] David Gunning. Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2, 2017.

[26] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. *arXiv preprint arXiv:1708.08296*, 2017.

[27] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[28] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.

[29] Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pages 80–89. IEEE, 2018.

[30] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[33] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284, 2017.

[34] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[35] Alex Berg, Jia Deng, and L Fei-Fei. Large scale visual recognition challenge 2010, 2010.

[36] Jia Deng, Alex Berg, Sanjeev Satheesh, Hao Su, Aditya Khosla, and Fei-Fei Li. Large scale visual recognition challenge. *www. image-net. org/challenges/LSVRC/2012*, 1, 2012.

[37] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[38] Yann LeCun, John S Denker, Sara A Solla, Richard E Howard, and Lawrence D Jackel. Optimal brain damage. In *NIPs*, volume 2, pages 598–605, 1989.

[39] Kenji Suzuki, Isao Horiba, and Noboru Sugie. A simple neural network pruning algorithm with application to filter synthesis. *Neural Processing Letters*, 13(1):43–53, 2001.

[40] Wenlin Chen, James T Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. *CoRR, abs/1504.04788*, 2015.

[41] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6655–6659. IEEE, 2013.

[42] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*, 2014.

[43] Yani Ioannou, Duncan Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*, 2015.

[44] Jiashi Feng and Trevor Darrell. Learning the structure of deep convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2749–2757, 2015.

[45] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[46] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances In Neural Information Processing Systems*, pages 2074–2082, 2016.

[47] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[48] Yi Sun, Xiaogang Wang, and Xiaoou Tang. Sparsifying neural network connections for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4856–4864, 2016.

[49] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. In *Advances In Neural Information Processing Systems*, pages 1379–1387, 2016.

[50] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

[51] Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. *arXiv preprint arXiv:1702.04008*, 2017.

[52] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2498–2507. JMLR. org, 2017.

[53] Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. In *Advances in Neural Information Processing Systems*, pages 3288–3298, 2017.

[54] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017.

[55] Melanie Mitchell and Stephanie Forrest. Genetic algorithms and artificial life. *Artificial life*, 1(3):267–289, 1994.

[56] Melanie Mitchell. Genetic algorithms: An overview. *Complexity*, 1(1):31–39, 1995.

[57] Mohammad Hasan Moradi and M Abedini. A combination of genetic algorithm and particle swarm optimization for optimal dg location and sizing in distribution systems. *International Journal of Electrical Power & Energy Systems*, 34(1):66–74, 2012.

[58] Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning. *IEEE Transactions on Industrial Informatics*, 9(1):132–141, 2013.

[59] John H Holland. Genetic algorithms. *Scientific american*, 267(1):66–73, 1992.

[60] David E Goldberg. Genetic algorithms in search, optimization, and machine learning, 1989. *Reading: Addison-Wesley*, 1989.

[61] Peter J Angeline, Gregory M Saunders, and Jordan B Pollack. An evolutionary algorithm that constructs recurrent neural networks. *IEEE transactions on Neural Networks*, 5(1):54–65, 1994.

[62] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[63] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. Real-time neuroevolution in the nero video game. *IEEE transactions on evolutionary computation*, 9(6):653–668, 2005.

[64] Jason Gauci and Kenneth Stanley. Generating large-scale neural networks through discovering geometric regularities. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 997–1004. ACM, 2007.

[65] Sreenivas Sremath Tirumala, Shahid Ali, and C Phani Ramesh. Evolving deep neural networks: A new prospect. In *Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD), 2016 12th International Conference on*, pages 69–74. IEEE, 2016.

[66] Mohammad Javad Shafiee, Francis Li, and Alexander Wong. Exploring the imposition of synaptic precision restrictions for evolutionary synthesis of deep neural networks. *arXiv preprint arXiv:1707.00095*, 2017.

[67] Mohammad J Shafiee, Elnaz Barshan, Francis Li, Brendan Chwyl, Michelle Karg, Christian Scharfenberger, and Alexander Wong. Learning efficient deep feature representations via transgenerational genetic transmission of environmental information during evolutionary synthesis of deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 979–986, 2017.

[68] Mohammad Javad Shafiee, Brendan Chwyl, Francis Li, Rongyan Chen, Michelle Karg, Christian Scharfenberger, and Alexander Wong. Stressednets: Efficient feature representations via stress-induced evolutionary synthesis of deep neural networks. *Neurocomputing*, 352:93–105, 2019.

[69] Ronald Aylmer Fisher. *The genetical theory of natural selection: a complete variorum edition*. Oxford University Press, 1930.

[70] Hermann Joseph Muller. Some genetic aspects of sex. *The American Naturalist*, 66(703):118–138, 1932.

[71] Damian Moran, Rowan Softley, and Eric J Warrant. The energetic cost of vision and the evolution of eyeless mexican cavefish. *Science advances*, 1(8):e1500363, 2015.

[72] Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

[73] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[74] Larry J Fish. Why multivariate methods are usually vital. *Measurement and Evaluation in Counseling and Development*, 21(3):130–137, 1988.

[75] Ellen R Girden. *ANOVA: Repeated measures*. Number 84. Sage, 1992.

[76] Henry Hsu and Peter A Lachenbruch. Paired t test. *Encyclopedia of Biostatistics*, 6, 2005.

[77] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[78] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.