2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

A Two-Stage Hybrid GA-Cellular Encoding Approach to Neural Architecture Search

Trevor Londt, Xiaoying Gao, Peter Andreae, Yi Mei
Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science
Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand
trevor.londt@ecs.vuw.ac.nz, xiaoying.gao@ecs.vuw.ac.nz, peter.andreae@ecs.vuw.ac.nz, yi.mei@ecs.vuw.ac.nz

***Abstract*—Neural Architecture Search (NAS) aims to automate the creation of Artificial Neural Networks, including Convolutional Neural Networks (CNN), lessening the reliance on labour-intensive manual design by human experts. A CNN architecture can be decomposed into a micro- and macro-architecture, each influenced by distinct design and optimisation strategies to contribute to the overall construction and performance of the CNN. Cellular Encoding (CE), an evolutionary computation technique, has been successfully used to represent diverse network topologies of varying complexities. Recently, CE has been applied to evolve CNN architectures, showing promising results. However, current CE-based NAS approaches focus on evolving either the micro- or macro-architectures without considering the evolution of both in the same algorithm. Evolving the micro- and macro-architecture together can increase the performance of evolved CNN architectures. This research introduces a novel two-stage hybrid approach, combining Genetic Algorithms (GA) and CE to evolve both the micro- and macro-architectures to synthesise CNNs for classification tasks. Candidate macro-architectures are evolved using a CE approach, while a GA approach is used to explore the micro-architecture search space. The proposed algorithm is evaluated across four commonly used datasets and compared against six NAS peer competitors and five state-of-the-art manually designed CNN architectures. The results validate the approach's high competitiveness, outperforming several peer competitors on image and text classification tasks.**

***Index Terms*—Neural Architecture Search, Convolutional Neural Networks, Cellular Encoding.**

I. Introduction

Designing state-of-the-art CNNs for classification tasks is non-trivial, demanding expert proficiency and substantial developmental time. Moreover, as the complexity of CNN architectures has increased, the time required for training has been significantly extended. Researchers have concentrated on developing innovative Neural Architecture Search (NAS) algorithms to counter these challenges. These automated solutions aim to design CNNs without human intervention. Evolutionary Neural Architecture Search (ENAS) is a promising approach [1]. Current ENAS approaches primarily revolve around the evolution of micro-architectures consisting of network layers such as convolutional, pooling, and activation layers and then constructing the macro-architecture as an arrangement of sequentially stacked micro-architectures. The rationale behind this strategy lies in the reduced training time associated with micro-architectures as opposed to training the entire macro-architecture as a whole [2]. This evolved micro-architecture is used as a building block that forms the backbone of the CNN macro-architecture. While this approach has evolved performant CNNs, especially within image classification tasks, it inherently limits the exploration of multi-path CNN macro-architectures. Multi-path CNN architectures [3] facilitate the reuse of features across network layers and provide diverse processing pathways similar to network ensembles, thereby improving classification performance. Furthermore, there are limited research works that evolve multi-path macro-architectures, and those that do typically use a predefined micro-architecture. While this approach has succeeded [4], particularly for text classification tasks [5], the opportunity to evolve micro-architectures tailored to the dataset being modelled is missed. In this work, we propose a new two-stage hybrid ENAS algorithm to mitigate these limitations and automatically evolve both micro- and macro-architectures to synthesise CNN architectures of varying sizes and complexity for both image and text classification tasks. The evolution of the micro- and macro-architecture are separated into two distinct evolutionary processes: a Genetic Algorithm (GA) approach to evolve micro-architectures and a Cellular Encoding-based approach to evolve macro-architectures. Experiment results show that the algorithm is performant in both the image and text domains. For image classification tasks, the proposed algorithm attains test classification accuracies of 95.83% and 95.66% on Fashion-MNIST and CIFAR-10, respectively. On text classification tasks, the proposed algorithm achieved test classification accuracies of 92.44% and 61.96% on AG's News and Yelp Reviews Full, respectively. The results are highly competitive compared to the current state-of-the ENAS algorithms and manually designed CNN architectures.

II. Background and Related Work

A. Evolutionary NAS

Early Evolutionary Neural Architecture Search (ENAS) involved evolving the topology, weights and biases of artificial neural networks (ANN). With the advent of CNNs, containing millions of trainable weights, back-propagation (BP) was found to be a superior approach to training CNNs. However, with modern innovations and the ever-increasing complexity of CNN architectures, the need to automatically design the complex architectures and hyperparameters of CNNs has resulted in renewed interest in ENAS. The modern approach to ENAS is to use Evolutionary Computation (EC) techniques to evolve a candidate CNN architecture and to use BP to train

978-0-7381-4408-5/23/$31.00 ©2023 IEEE

the architecture [4]. This combined approach has resulted in impressive performance gains and the introduction of novel CNN architectures for a range of problem domains and tasks. A CNN architecture can conceptually be decomposed into a macro- and micro-architecture, each representing a distinct search space. It is a common approach for ENAS algorithms to focus on the micro-architecture search space when involved in the image domain and the macro-architecture space when dealing with the text domain [4]. ENAS algorithms in the image domain typically evolve micro-architectures stacked one after the other to form the macro-architecture [2]. In contrast, ENAS algorithms for text domains use a predefined micro-architecture to be placed as nodes (cells) in the topology of the evolved macro-architectures [5]. Limited research works consider the evolution of the micro- and macro-architecture in the same algorithm, which could offer novel CNN architectures with increased performance. Therefore this work considers the evolution of both the micro- and macro-architecture.

Genetic Algorithms (GA) have been successfully used in numerous ENAS algorithms. GAs are generally used for direct encoding of CNN architectures, which implies that the size of a chromosome is directly related to the size of an architecture. This property means that a CNNs architecture is restricted to the size of the chromosome. Modern CNN architectures, particularly macro-architectures, are large due to their increasing depth over recent years. Therefore this work uses a GA to evolve micro-architectures instead, which are significantly smaller in size.

B. Cellular Encoding

Cellular Encoding (CE) [6] draws inspiration from how biological cells divide to create complex organisms. It constitutes an indirect encoding approach for generating artificial neural networks. CE has proven its efficacy in evolving Convolutional Neural Network (CNN) architectures [5], [7]. In CE, genotypes take the form of tree structures and are encoded as a sequence of program instructions. These instructions are applied to an initial neural network, denoted as the *ancestor network*. The ancestor network comprises a single ancestor neuron, referred to as a *cell*. With each execution of an instruction in the genotype, the ancestor network undergoes transformations, leading to complex topologies as cells are added, modified, or removed. This method permits the neural network topology to expand organically until a suitably sized network containing enough capacity to model the given problem is produced. The set of instructions within cellular encoding is comprehensive, with the core instructions pertaining to cell duplication. These encompass Sequential (SEQ), Parallel (PAR), and Recursion (REC) duplication operations. Sequential duplication involves a cell undergoing division into two cells connected in series. Parallel duplication results in a cell dividing into two cells connected in parallel, sharing their inputs and outputs. Recursion involves reprocessing the genotype from the root node up to a point in the genotype where the recursion operation was executed. Recursion operations are only performed once. Terminal operations include the END operation, which signifies a leaf node. Additional instructions relate to manipulating weights and bias values within the network. The term *mother cell* means a cell currently undergoing a CE operation. In contrast, a *child cell* refers to the resultant duplicate cell emerging from SEQ or PAR operations performed on a mother cell. Our previous research [5] on CE uses Genetic Programming (GP) [8] to facilitate evolutionary processes. GP genotypes are variable-length programs represented by tree structures. The variable length property of individuals in GP algorithms is an attractive property that allows the representation of various network topologies sizes ranging from small to very large. Therefore, this work uses GP to facilitate CE to represent macro-architectures, allowing both shallow and very deep CNN macro-architectures to be explored.

C. Related Work

In recent years, CE-based NAS algorithms have successfully been applied to evolving CNN architectures. Broni-Bediako et al. [7] have proposed a new version of Cellular Encoding [7] representation. Their proposed representation is a symbolic linear generative encoding scheme that embeds local graph transformations in Cellular Encoding-based chromosomes. Evolved micro-architectures are stacked according to manually designed macro-architecture structures. This approach limits their algorithm to predefined macro-architecture configurations, missing out on exploring novel macro-architectures. Their work implements four CE operations: SEQ, CPI, CPO and END. SEQ is the sequential split operation. CPI is also a sequential split operation; however, the child and parent have the same inputs but different outputs. CPO is a sequential split operation, but the child and parent share the same outputs but different inputs. The final implemented operation is the END terminal. Another limitation of their approach is that they have not implemented the parallel split operation (PAR), where a child and parent share the same inputs *and* outputs. This limitation removes a significant and potentially important subset of the micro-architecture search space. Each chromosome has a fixed-length number of genes representing *grouped* CE operations *and* network layers. In essence, they have produced a hybrid direct-indirect encoding scheme where Cellular Encoding is used to evolve the CNN's micro-architecture topology, and the configuration parameters of convolutional layers are embedded in the chromosome. Results indicated the algorithm is highly performant, particularly on the CIFAR-10 dataset. Their algorithm has not been extended to other domains, such as the text domain. Our previous work [5] proposed a CE-based NAS algorithm, GP-Dense, to evolve character-level CNN architectures for text classification tasks. We define a network cell as containing a manually designed micro-architecture. Our algorithm evolves macro-architectures containing several network cells. The core operations of CE are implemented, namely, the SEQ, PAR and END operations. The SEQ operation duplicates a parent cell to produce a child cell, and the two are connected in series. The PAR operation performs similarly, except the child and parent cell are connected in parallel, sharing inputs and outputs. Results

indicated competitive performance across various text datasets against manually designed CNN architectures. A limitation of our algorithm is using manually designed micro-architectures, as it is unknown which are optimal for a particular text dataset. Furthermore, our algorithm has yet to be extended to the image domain.

III. Proposed Method

The performance of a micro-architecture is measured in conjunction with an appropriate macro-architecture, which together constitute the main parts of a CNN architecture. However, different macro-architectures may display different performance accuracies, even when using the same micro-architecture. This situation presents the problem of which macro-architecture to use to measure the performance of an evolved micro-architecture. To overcome this challenge, we propose a two-stage NAS algorithm, GACE-NAS, which first focuses on evolving micro-architectures using a set of predefined macro-architectures and then, in stage two, evolves a population of candidate macro-architectures using the best-evolved micro-architecture from stage one. A new GA encoding is proposed to represent candidate micro-architectures, and the CE representation, as proposed in our previous work, GP-Dense [5], is used to facilitate the representation and evolution of macro-architectures. However, the CE representation introduced by GP-Dense is modified to facilitate the evolution of CNNs for both image and text classification tasks.

The overall process of the proposed algorithm is presented in Fig. 1. The algorithm begins with generating a subset of the training set as the GP-Dense algorithm does. Three candidate macro-architectures are generated using the CE representation as used by GP-Dense. These macro-architectures are used to determine the fitness of the micro-architectures that will be evolved during this stage. Only three candidate macro-architectures are considered due to the high computational cost and limited hardware resources available. Next, a population of randomly generated micro-architectures is constructed. Each of the candidate micro-architectures is evaluated in all three macro-architectures. The micro-architecture fitness is calculated as the average of the validation classification accuracies of the three macro-architectures using the candidate micro-architecture. The evolutionary process to further evolve micro-architectures is conducted until a terminating criterion is achieved. The evaluation and calculation of the fitness of a candidate micro-architecture is the average fitness value attained from evaluating all three candidate macro-architectures. Note that if a CNN architecture does not fit into the GPU memory, the micro-architecture's fitness measure in the current macro-architecture is assigned a value of -1.0. This approach means that a micro-architecture's fitness is partially reliant on the ability of a CNN to fit into the GPU memory. This approach is acceptable because the GPU memory is part of the evolutionary environment in which the micro-architecture must be fit enough to survive. Once a terminating criterion has been met, the best-evolved micro-architecture is supplied to stage two of the proposed algorithm. A population of macro-architectures using the GP-Dense [5] representation is created, with each candidate macro-architecture using the best-evolved micro-architecture from stage one to synthesise a CNN architecture. The macro-architecture evolution process is conducted until a terminating criterion is met. The best-evolved macro-architecture with the best-evolved micro-architecture from stage one is then, along with the entire training dataset, used to synthesise and train a CNN architecture, after which the test set is used to conduct inference to attain the final test accuracy result.

A. Micro-architecture Representation

A new micro-architecture representation is proposed, and a visual representation is displayed in Fig. 2. Each genotype is represented as a chromosome consisting of a specified small number of genes, of which all genes except the last, can each take a value in the range [0, 8], inclusive. The possible values represent a particular layer type or operation, as presented in Table I.

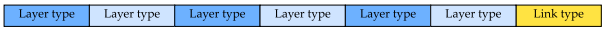


Fig. 2. GA-XCGP: Genotype for micro-architecture representation.

Each gene represents a layer connected to its neighbours, implying that this encoding will facilitate the representation of micro-architectures with linearly-chained topologies. Since each chromosome is limited to a small number of genes, the generated micro-architectures are small and similar in depth to those of expert-designed micro-architectures, as in ResNet [9], DenseNet [10], and VDCNN-Convolution [11]. Three commonly used kernel sizes are catered for, namely 3, 5, and 7. These kernel values are effective sizes for CNN applied to image and text classification tasks, as seen in ResNet [9], InceptionNet [12], DenseNet [13], and VDCNN-Convolution [11] architectures. The padding values are chosen to ensure that the resultant feature maps retain their dimension size to reduce the pooling operations needed when concatenating feature maps on the macro-architecture level as is implemented by GP-Dense [5].

TABLE I
Layer Types and Values.

Gene Value	Layer Type	Kernel Size	Stride	Padding
0	Convolution	3	1	1
1	Convolution	5	1	2
2	Convolution	7	1	3
3	ReLU	-	-	-
4	Leaky ReLU	-	-	-
5	Batch Normalisation	-	-	-
6	Max Pooling	2	2	1
7	Average Pooling	2	2	1
8	Ignore	-	-	-

Two activation functions are available, ReLU, and its modern variant, LeakyReLU. Catering for both activation functions will allow the evolutionary process to determine which variants are favoured for the task at hand. Batch Normalisation is catered for, and two pooling functions are provided: Max

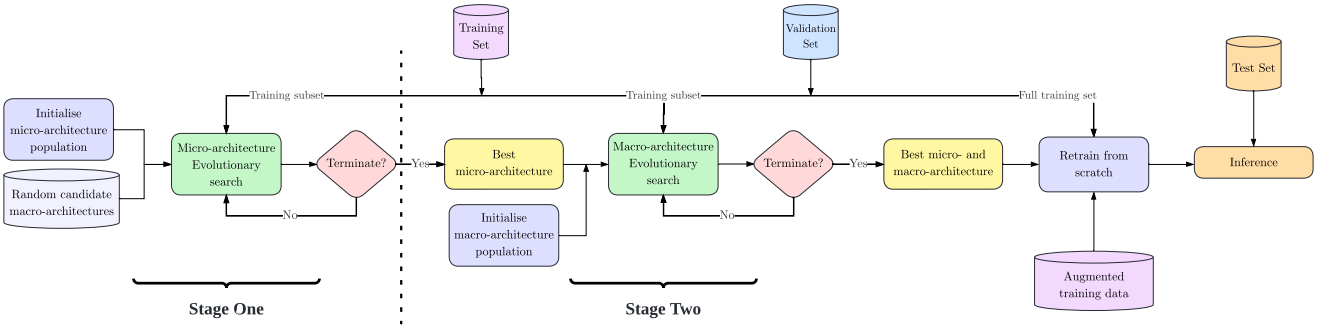


Fig. 1. Framework of GACE-NAS.

and Average Pooling. These pooling operations are commonly used in the literature for text and image classification tasks. The pooling operations use a kernel size of two and a stride of two, as frequently used in the literature. The final available gene value is an *ignore* layer, represented by the number eight, which allows the construction of micro-architectures of varying depths.

The last gene can take a value in the range [0, 2] as listed in Table II, representing the type of skip link to be used or if a skip link is absent. A value of 0 indicates that the micro-architecture has no skip link. A value of 1 or 2 indicates that a skip link is present with a connection operation of either concatenation or summation, respectively. The summation operation, as ResNet uses, allows the number of channels to remain fixed, resulting in a compact micro-architecture [9]. The concatenation operation, as used in InceptionNet and DenseNet, allows feature reuse; however, this has the disadvantage that the micro-architecture requires larger memory sizes in which to be hosted [12]. Note that each skip link type has a 33.33% chance of being selected, meaning that there is a 66.66% chance that a micro-architecture will have a skip link. This approach is deliberate as a skip link is more valuable than not, especially in deeper CNNs [14].

TABLE II
LAYER TYPES AND VALUES.

Gene Value	Skip link Type
0	No Skip Link
1	Concatenation Skip Link
2	Summation Skip Link

B. Macro-architecture Representation

This work uses a modified version of the CE representation implemented in the GP-Dense algorithm [5]. To capture the general workings of cellular encoding, the sequential cell division (**S**) and parallel cell division (**P**) operations are implemented. Four new modifier operations are introduced, namely, Increment Depth (**ID**), Decrement Depth (**DD**), Increment Width (**IW**), and Decrement Width (**DW**). These modifiers can be applied with the S and P operations, for example, SID, SDDIW, PDW, PIDDW, etc. The modifier operations apply a transformation to the micro-architecture blocks contained in a cell that is currently being operated on. Note that the modifiers ID and DD can not be applied together. Neither can IW and DW. This means that there are 18 possible combinations for the proposed encoding.

The workings of the implemented operations and modifiers are explained below:

1) **S**: A child cell is constructed that inherits the mother cell's number of filters and depth. The child cell is inserted into the network in series after the mother cell. The mother cell's output channels are connected to the child's input channels. The child cell's output channels will be connected to the mother cell's original output destination.
2) **P**: A child cell is constructed that inherits the mother cell's number of filters and depth. The child cell is connected in parallel with the mother cell in the network. The mother and child cells receive the same input channels, and their output channels are concatenated in a channel-wise format and connected to the mother cell's original output destination.
3) **ID**: The depth of a child's cell is twice as deep as the inherited mother's cell. i.e. the child cell will contain twice as many stacked micro-architecture copies as the mother's cell contains.
4) **DD**: A child cell's stacked micro-architecture blocks are half that of the mother cell's number of stacked micro-architecture blocks. A child cell will always contain at least one micro-architecture block.
5) **IW**: The child cell inherits twice as many filters as the mother cell.
6) **DW**: The child cell inherits half the number of filters from the mother cell or a minimum specified number of filters, whichever is the largest.

C. Population Initialisation

1) Micro-architecture initialisation: the population initialisation method ensures that every individual contains at least one convolutional layer. The values in the

chromosome of an individual are randomly chosen from the available layer types until a predefined chromosome length between a specified minimum and maximum size has been attained. After an individual is generated, it is checked for at least one convolutional layer. If none are found, a random position in the individual is selected to be converted to a convolutional layer.

2) Macro-architecture initialisation: Individuals between specified minimum and maximum depths are randomly generated, with each possible combination of operator and modifiers in the function set, selected based on a uniform distribution basis.

D. Mutation

1) Micro-architecture mutation: A random gene in the chromosome is selected. If the selected gene is a skip link type, then a different, randomly selected skip link type is chosen to replace the gene's value. Suppose the gene represents a convolutional layer, and it is the only convolutional layer type in the chromosome. In that case, the gene can only be mutated to another convolutional layer with different kernel sizes and corresponding padding values. This design decision ensures that a micro-architecture will always contain a convolutional layer. On the other hand, if the gene does not represent the only convolutional layer, then the gene can be mutated to any other layer type as presented in Table I. Note that all initially created genotypes will always have at least one convolutional layer, as previously discussed.
2) Macro-architecture mutation: A uniform single-point mutation operation is chosen for the proposed algorithm. The operation involves randomly selecting a node in the genotype that is replaced by a randomly generated subtree of a specified size. The operation is based on that used by GP-Dense [5].

E. Crossover

1) Micro-architecture crossover: A random position in each of the parent chromosomes is selected. The point represents the position indices where the first and second parents exchange genes. Once the exchange of gene information has been completed, the results are two new offspring genotypes. Both offspring genotypes are checked to ensure each contains at least one convolutional layer. If either does not include a convolutional layer, the crossover operation is considered a failure and the offspring are discarded.
2) Macro-architecture crossover: The single-point crossover operation is chosen for its simplicity of operation and implementation. The single-point crossover operation works by randomly selecting a point in each parent genotype, and the sub-trees formed below these two points are crossed over between the parents to create two offspring. The operation is based on the same operation used in GP-Dense [5].

F. Selection

Tournament selection is used for both the micro-architecture and macro-architecture evolutionary processes. Tournament selection is chosen as it reasonably balances exploration and exploitation, which the tournament size can control. Furthermore, tournament selection is a popular method in evolutionary NAS algorithms in the literature.

G. Fitness Function

During stage one, each candidate micro-architecture is evaluated in three predefined macro-architectures. The fitness of the micro-architecture is calculated as the average validation accuracy attained by each predefined macro-architecture using the candidate micro-architecture. In stage two, a new population of macro-architectures is evolved using the best micro-architecture produced during stage one. The validation accuracy determines the fitness of each candidate's macro-architecture during stage two.

H. Data Augmentation

Data augmentation is standard practice in the NAS literate, particularly for image classification tasks. However, using data augmentation when designing character-level CNN architectures for text classification tasks is not common practice. To ensure that the performance of the proposed algorithm is fairly compared against peer competitors, data augmentation is only used by the proposed algorithm when evolving CNNs image classification tasks.

IV. Experiment Design

A. Benchmark Datasets and Peer Competitors

Four datasets are chosen to test the utility of the proposed algorithm for classification tasks. Two datasets are from the image domain, and the other two are from the text domain. Fashion-MNIST and CIFAR-10 are chosen to benchmark image classification tasks. Similarly, AG's News and Yelp Reviews Full are commonly used benchmark datasets to evaluate CNNs for text classification tasks and, therefore, used in this work. The algorithm is evaluated across thirty independent experiment runs per dataset. To accelerate the training times of candidate CNN architectures, the experiment runs are distributed across four NVIDIA RTX 2080 GPUs. For image classification tasks, the manually designed ResNet [9], InceptionNet [12], and DenseNet [13] are selected. Furthermore, the peer competitor NAS algorithms EvoCNN [15], FPSO [10], CGP-CNN-ConvSet [16], EIGEN [17] and CE-GeneExpr [7] are chosen as peer competitors for image classification tasks. Two manually designed CNN architectures, Zhang-Full-Convolution [18] and VDCNN-Convolution [11], and the state-of-the-art GP-Dense [5] NAS algorithm are chosen as peer competitors for text classification tasks.

B. Parameter Settings

The parameter settings are based on community conventions and those used in CE-based NAS algorithms [5], [7]. Table III lists the parameters for the GA algorithm used in stage one

to evolve candidate micro-architectures. A small population and the number of generations are chosen to reduce the required computational time and costs associated with ENAS algorithms.

TABLE III
PARAMETERS FOR MICRO-ARCHITECTURE EVOLUTION.

Parameter	Setting
Population Size	10
Generations	10
Chromosome Length	[4,7]
Minimum filter count	32
Elitism	0.1
Mutation Probability	0.01
Mutation Type	Custom single point
Crossover Probability	0.50
Crossover Type	Custom single point

Table IV lists the parameter settings used for the GP-based CE algorithm used in stage two to evolve macro-architectures. Since the GP-based algorithm used in stage two is based on the representation used by GP-Dense [5], the same parameter and evolutionary operation settings are also used. The number of generations and population size are both set to twenty. This number is chosen to reduce the amount of computation time required and is based on current CE-based ENAS algorithms [5], [7].

TABLE IV
PARAMETERS FOR MACRO-ARCHITECTURE EVOLUTION.

Parameter	Setting
Number of generations	20
Population size	20
Elitism	0.1
Mutation probability	0.01
Mutation growth type	Grow
Mutation tree growth size	[1,3]
Crossover probability	0.5
Crossover type	Single point
Tournament selection size	3
Function set (Operations)	SEQ,PAR
Terminals	{END}
Max tree depth	17
Initial tree depth	[1, 6]
Initial tree growth	Half and Half

V. RESULTS AND DISCUSSIONS

A. Test Accuracies

The best test results produced from the thirty independent experiment runs of the proposed algorithm are presented in Table V and VI for the image and text datasets, respectively. On the image datasets, GACE-NAS has attained a test accuracy of 95.83% on the Fashion-MNIST dataset, outperforming all other competitors. On the CIFAR-10 dataset, GACE-NAS has performed well, outperforming all peer competitors, except the current state-of-the-art CE-GeneExpr NAS algorithm and the manually designed DenseNet architecture. It is interesting to note that the top-performing NAS algorithms are both CE-based, highlighting the utility of using CE to evolve CNN architecture for image classification tasks.

TABLE V
IMAGE CLASSIFICATION: TEST ACCURACIES (%) COMPARED TO PEER COMPETITORS. (M = MANUAL, N = NAS, '-' = NOT AVAILABLE)

Algorithm	Type	Fashion-MNIST	CIFAR-10
GACE-NAS (**ours**)	N	**95.83**	95.66
EvoCNN [15]	N	94.53	-
FPSO [10]	N	95.07	93.72
CGP-CNN-ConvSet [16]	N	-	93.25
EIGEN [17]	N	-	94.60
CE-GeneExpr [7]	N	-	96.26
ResNet [9]	M	93.40	93.57
InceptionNet (GoogLeNet) [12]	M	92.74	93.64
DenseNet [13]	M	93.91	**96.54**

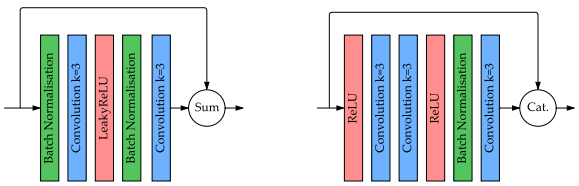
On the text datasets, GACE-NAS has attained a test accuracy of 92.44% for the AG's News dataset, outperforming all peer competitors. GACE-NAS has outperformed the current state-of-the-art GP-Dense NAS algorithm for the Yelp Reviews Full Full dataset. VDCNN-Convolution remains the top-performing architecture, highlighting the difficulty of NAS algorithms to model the challenging Yelp Reviews Full dataset. However, it should be remembered that VDCNN-convolution is a human-expert, manually designed architecture. GACE-NAS is performant across both image and text classification domains and, to date, is the only CE-based NAS algorithm effective in more than one classification domain.

TABLE VI
TEXT CLASSIFICATION: TEST ACCURACIES (%) COMPARED TO PEER COMPETITORS. (M = MANUAL, N = NAS)

Algorithm	Type	AG's News	Yelp Reviews Full
GACE-NAS (**ours**)	N	**92.44**	61.96
GP-Dense [5]	N	89.58	61.05
Zhang-Full-Convolution [18]	M	90.15	61.60
VDCNN-Convolution [11]	M	91.27	**64.72**

B. Best Evolved Micro-architectures

Fig. 3 displays the best micro-architectures evolved for the image datasets. For the Fashion-MNIST dataset, the micro-architecture consists of a LeakyReLU activation layer, two convolutional layers, two batch normalisation layers, and a skip link using a summation operation. The evolved micro-architecture is similar to a ResNet convolutional block, demonstrating that GACE-NAS can evolve highly competitive ResNet-like micro-architectures.



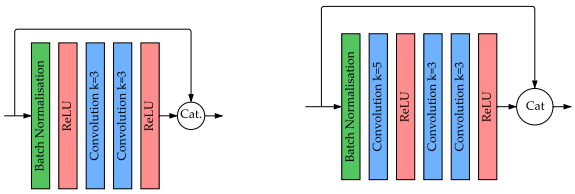
(a) Fashion-MNIST (b) CIFAR-10

Fig. 3. Best micro-architectures for image datasets.

The evolved micro-architecture for CIFAR-10 consists of six layers. CIFAR-10 is more challenging to model than

Fashion-MNIST. This would explain why the evolved micro-architecture is more complex than the one for Fashion-MNIST. The evolved micro-architecture consists of two ReLU activation layers, three convolutional layers with a kernel size of three and one batch normalisation layer. In addition, a skip link is present using a concatenation operation. Interestingly, the evolutionary process has favoured a concatenation operation over a summation operation, in contrast to the micro-architecture evolved for the Fashion-MNIST dataset. This observation implies a DenseNet-like approach should be considered when modelling CIFAR-10.

Fig. 4 displays the best micro-architectures evolved for the text datasets. The micro-architecture for AG's News contains a batch normalisation layer, two ReLU activation layers and two convolutional layers with a kernel size of three. The state-of-the-art architecture, VDCNN-Convolution [11], contains a micro-architecture that consists of the same type and quantity of layers except for an additional batch normalisation layer. GACE-NAS has successfully evolved a micro-architecture with properties similar to the state-of-the-art VDCNN-Convolution micro-architecture, which demonstrates the effectiveness of GACE-NAS for evolving micro-architectures on par with those designed by human experts.



(a) AG's News (b) Yelp Reviews Full

Fig. 4. Best micro-architectures for text datasets.

For the Yelp Reviews Full dataset, the evolved micro-architecture is relatively complex and has no similarities to known expert-designed micro-architectures. Regardless, the micro-architecture has achieved competitive results. This observation means an evolutionary approach to evolving unorthodox micro-architectures appears promising on the Yelp Reviews Full dataset.

VI. Conclusions

This paper introduces an EC based NAS algorithm that automatically evolves high-performance micro- and macro-architectures to synthesise CNN architectures for classification tasks. The proposed algorithm uses a hybrid GA and CE approach to evolve the micro- and macro-architectures, respectively. Experiment results show that the algorithm can evolve highly competitive CNN architectures. Our best-evolved networks defeated all state-of-the-art peer competitors for image and text classification tasks across the AG's News and Fashion-MNIST datasets and achieved competitive results on the Yelps Reviews Full and CIFAR-10 datasets. Further research will apply the algorithm to larger image datasets, such as ImageNet, and innovate a novel approach to evolve the micro- and macro-architectures simultaneously but in isolation to further improve classification performance.

References

[1] Z.-H. Zhan, J.-Y. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomputing*, pp. 42–58, 2022.

[2] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient Neural Architecture Search via parameter Sharing," in *35th International Conference on Machine Learning, ICML 2018*, vol. 9, pp. 6522–6531, International Machine Learning Society (IMLS), 2 2018.

[3] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[4] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A Survey on Evolutionary Neural Architecture Search," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[5] T. Londt, X. Gao, and P. Andreae, "Evolving Character-Level DenseNet Architectures Using Genetic Programming," *Lecture Notes in Computer Science*, pp. 665–680, 2021.

[6] F. Gruau, F. Gruau, L. C. B.-l. I, O. A. D. De Doctorat, M. J. Demongeot, E. M. M. Cosnard, M. J. Mazoyer, M. P. Peretto, and M. D. Whitley, "Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm.," 1994.

[7] C. Broni-Bediako, Y. Murata, L. H. B. Mormille, and M. Atsumi, "Evolutionary NAS with Gene Expression Programming of Cellular Encoding," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 2670–2676, 2020.

[8] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statistics and Computing 1994 4:2*, vol. 4, pp. 87–112, 6 1994.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 12 2016.

[10] J. Huang, B. Xue, Y. Sun, and M. Zhang, "A Flexible Variable-length Particle Swarm Optimization Approach to Convolutional Neural Network Architecture Design," *2021 IEEE Congress on Evolutionary Computation, CEC 2021 - Proceedings*, pp. 934–941, 2021.

[11] A. Conneau, H. Schwenk, Y. L. Cun, and L. Barrault, "Very deep convolutional networks for text classification," in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, vol. 2, pp. 1107–1116, Association for Computational Linguistics (ACL), 2017.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 10 2015.

[13] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2261–2269, 11 2017.

[14] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[15] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving Deep Convolutional Neural Networks for Image Classification," *IEEE Transactions on Evolutionary Computation*, pp. 394–407, 4 2020.

[16] M. Suganuma, S. Shirakawa, and T. Nagao, "A Genetic Programming Approach to Designing Convolutional Neural Network Architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pp. 497–504, 2017.

[17] J. Ren, Z. Li, J. Yang, N. Xu, T. Yang, and D. J. Foran, "Eigen: Ecologically-inspired genetic approach for neural network structure searching from scratch," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 9051–9060, 6 2019.

[18] X. Zhang, J. Zhao, and Y. LeCun, "Character-Level Convolutional Networks for Text Classification," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, (Cambridge, MA, USA), p. 649–657, MIT Press, 2015.