Evolutionary Neural Architecture Search for High-Dimensional Skip-Connection Structures on DenseNet Style Networks

Damien O'Neill, *Member, IEEE*, Bing Xue, *Senior Member, IEEE*, and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Convolutional neural networks hold state-of-the-art results for image classification, and many neural architecture search algorithms have been proposed to discover high performance convolutional neural networks. However, the use of neural architecture search for the discovery of skip-connection structures, an important element in modern convolutional neural networks, is limited within the literature. Furthermore, while many neural architecture search algorithms utilize performance estimation techniques to reduce computation time, empirical evaluations of these performance estimation techniques remain limited. This work focuses on utilizing evolutionary neural architecture search to examine the search space of networks, which follow a fundamental DenseNet structure, but have no fixed skip connections. In particular, a genetic algorithm is designed, which searches the space consisting of all networks between a standard feedforward network and the corresponding DenseNet. To design the algorithm, lower fidelity performance estimation of this class of networks is examined and presented. The final algorithm finds networks that are more accurate than DenseNets on CIFAR10 and CIFAR100, and have fewer trainable parameters. The structures found by the algorithm are examined to shed light on the importance of different types of skip-connection structures in convolutional neural networks, including the discovery of a simple skip-connection removal, which improves DenseNet performance on CIFAR10.**

***Index Terms*—Convolutional neural networks (CNNs), evolutionary neural architecture search, genetic algorithms, neural architecture search (NAS), skip connections.**

I. Introduction

CLASSIFICATION is among the most important machine learning tasks. In particular, much recent work has been focused on machine learning for image classification, with significant advances being found, which improve accuracy and efficiency in this task. Improvements to convolutional neural networks (CNNs) have been a key route through which these benefits have been found, and CNNs accordingly hold state-of-the-art results on several benchmark image classification datasets. In particular, the introduction of skip-connections has allowed for CNNs to achieve previously unattainable results in both handcrafted and automatically designed networks [1]–[5].

Within the CNN design, the recent work in neural architecture search (NAS) has demonstrated that automatically designed networks can match, and even surpass, the performance of CNNs designed by human experts, and has led to the discovery of novel state-of-the-art CNN architectures with modules, which are unlikely to be developed by human experts [5]. However, the most competitive NAS algorithms utilize predefined, simple, skip-connection structures, despite the apparent importance of skip connections and the use of distinct, competing styles of skip connection appearing in state-of-the-art works [3]–[5].

Furthermore, the high computational cost of NAS algorithms has led to many NAS algorithms utilizing performance estimation techniques to reduce computational cost. However, empirical justifications of the performance estimation techniques used are frequently lacking, leaving much NAS research in the dark with respect to the ramifications of these choices and possible alternatives to the chosen techniques [6].

In works where the skip-connection structure of networks has been optimized, networks have either been shallow [7] or have failed to improve the performance of baseline networks [8], likely due to the fact that the optimization of skip connections is NP-hard [9]. In particular, the selection of skip connections is similar to feature selection, but with the added complication that the removal of a skip connection will vary the values of all weights in the trained network, and accordingly, the network usage of all other connections. The mechanisms behind these interactions occur both in changing the gradient information used in backpropagation, and in changing the information passed through the network in forward propagation.

A. Motivations

While much NAS work has utilized various performance estimation techniques and measures of possible models in order to reduce computation time [6], [10], it has been

Manuscript received May 7, 2020; revised August 26, 2020 and February 23, 2021; accepted May 7, 2021. Date of publication May 24, 2021; date of current version December 1, 2021. This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1913 and Contract VUW1914; in part by the Science for Technological Innovation Challenge (SfTI) Fund under Grant E3603/2903; in part by the University Research Fund at Victoria University of Wellington under Grant 223805/3986; in part by the MBIE Data Science SSIF Fund under Contract RTVU1914; and in part by the National Natural Science Foundation of China (NSFC) under Grant 61876169. *(Corresponding author: Damien O'Neill.)*

The authors are with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: damien.oneill@ecs.vuw.ac.nz; bing.xue@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2021.3083315.

Digital Object Identifier 10.1109/TEVC.2021.3083315

noted that minimal work has been conducted into directly studying the effects of these techniques, parameters and configurations for these techniques, and measures of model performance [6]. Thus, while a given performance estimation technique may be suitable for the work in which it is used, researchers are without clear demonstration of the generality of the used performance estimation technique, and there is a lack of context around how different choices relating to performance estimation techniques may affect the final outcome in a given work. As a result, researchers need to either copy a performance estimation technique exactly, potentially a technique that generalizes poorly to a new problem space, or evaluate many performance estimation techniques from scratch, likely resulting in a duplication of efforts.

Furthermore, given the stochastic nature of training modern ANNs via backpropagation, NAS has an inherently stochastic search space, where the evaluations of an ANN structure will vary between random seeds. This presents a challenge for the NAS algorithm design, including evolutionary NAS, in that it is unclear exactly how this stochasticity affects the search for good candidate neural structures, and what design decisions need to be made for evolutionary NAS algorithms to best manage the stochasticity of evaluation outcomes.

With regards to the breadth of scope of NAS algorithms, minimal research has been conducted with a focus on the automatic discovery of skip-connection structure in CNNs, and no published work has demonstrated an algorithm, which can successfully optimize skip-connection structure when the possible number of skip connections is large.

Potentially, as a result of the difficulties outlined above, the roles of specific skip connections in different parts of CNNs are still relatively poorly understood, despite their ongoing usage in state-of-the-art networks [3]–[5] and works being published, which formalize some of the ways in which skip connections improve network accuracy [11], [12]. This minimality of understanding is evident when comparing the divergence in skip-connection utilization between different high performance architectures (e.g., between [4] and [5]).

In practical applications, the discovery of novel skip-connection structures in deep networks provides the opportunity to optimize networks to be both smaller and more accurate for a given task, and do so in a way that does not conflict with existing methods which optimize the layers used in networks. This is of particular importance when considering the use of artificial neural networks in computation limited devices, such as mobile phones and autonomous vehicles [13].

B. Goals

The overall goal of this work is to create a novel evolutionary NAS algorithm for the discovery of high performance skip-connection structures given a baseline network, and use discovered networks to illuminate how the importance and roles of skip connections can change depending on which parts of a CNN they connect. In constructing the novel algorithm, we seek to characterize several relatively unexamined behaviors of this search space, including the effects of different configurations of performance estimation techniques (namely, lower fidelity estimation), and how algorithm construction can be informed by analyzing the stochasticity in a given problem space. A summary of specific contributions of this article is presented as follows.

1) A common and effective form of lower fidelity estimation trains models on a subset of available data for a limited number of epochs. Further choices in lower fidelity estimation include whether to utilize training performance or holdout performance when estimating test performance and which measure to use for estimation (e.g., accuracy or cross-entropy). This article evaluates and presents a wide range of different configurations using the above options, and provides an analysis of correlation between these configurations and test accuracy, as well as an analysis regarding the effect of chosen configurations on the stochasticity of estimations.
2) Once a lower fidelity estimation configuration has been chosen using the above analysis, a novel NAS algorithm can be created where key choices are made using known characteristics of that lower fidelity estimation configuration. As this work is concerned with the efficiency of the discovered skip-connection structures, the novel NAS algorithm aims to find networks that outperform a selected base network and, as a result of the removal of skip connections, have fewer trainable weights. To demonstrate efficiency, discovered networks, thus, have both their performance and number of trainable weights compared to the base network on two competitive benchmark image datasets, with the aim of increased performance and fewer trainable weights.
3) The resulting networks from the novel NAS algorithm should be more efficient than the network on which they are based. Thus, an analysis of the skip-connection structures in these discovered networks can yield insight into the characteristics of efficient skip-connection structures, and how these structures change throughout deep CNNs. To provide these insights, resulting structures are examined in depth, with further investigation provided regarding specific skip connections according to notable patterns.

II. Background

A. Convolutional Neural Networks

Many current state-of-the-art models for visual classification are variants of CNNs [5], [14]–[18], which use weight sharing in the form of convolutional filters to utilize the spatial relationships inherent to visual data [19].

Convolutional filters, in addition to allowing networks to find local patterns in visual data, reduce the number of weights being optimized, making more efficient models. Formally, a convolutional filter is typically an $N \times N$ trainable weight matrix, with N user specified. DenseNets, which are being used as a baseline in this work, for instance, primarily utilize $N \in \{1, 3\}$ [4]. These learned convolutions are iterated over an image, and return a 2-D matrix, consisting of convolved features for different spatial positions in the input.

1) Training Issues in Deep CNNs: As deep neural networks, including deep CNNs, have become state-of-the-art in many machine learning tasks, difficulties have been noted with regards to the training of these networks. Notably, the vanishing gradient problem [20] and issues related to singularities in networks (see [11], [12]) have been shown to drastically reduce the performance of deep CNNs in terms of efficiency in training and the final accuracy of trained models. One of the key ways in which these issues are addressed is the use of skip connections [17], whereby a layer in a network is directly connected to more than one later layer in the network. DenseNets, in particular, utilize skip connections heavily, and show the benefit of these connections.

2) Cross-Entropy: Cross-entropy [21] is a measure of difference between two probability distributions, and is used as a loss function in multiclass classification problems, being optimal when the loss is 0.0. In a multiclass classification problem, the true probability distribution $p(x)$ for a given instance is a vector with 1.0 for the true label, and 0 for all other labels. For example, in a 3-class classification problem, if the true class is the second class, $p(x) = [0.0, 1.0, 0.0]$. Importantly, if cross-entropy is 0.0, the network necessarily has perfect accuracy as well.

Specifically, given $p(x)$ and an estimated probability distribution over classes for an instance $q(x)$, cross-entropy is defined by the following function, where x refers to the indices in a probability distribution for an instance

$$H(p, q) = -\sum_x p(x) \log_2(q(x)).$$

As $p(x)$ is zero for all but the true label, this loss is equivalent to $-\log_2(q(x))$ where x is the probability estimate for the true label. It is, thus, optimal exactly when $q(x) = 1.0$ for the true class.

B. DenseNet

DenseNets [4] form the base structure for this work, with the algorithm presented seeking to take DenseNets and reduce the number of connections present in them. DenseNets are chosen to illustrate the algorithm as their structure can be easily encoded using an adjacency matrix [22], and the way in which they are constructed leads to guarantees that the number of trainable parameters is reduced if anything less than the maximum number of skip connections is utilized.

DenseNets are a form of CNN defined by three parameters: 1) the depth of the network ℓ; 2) the initial convolution width k_0; and 3) the growth rate k [4]. The depth of a network is considered in terms of the number of trainable layers, and transition blocks, basic blocks, the initial convolution, and densely connected output all contribute to depth, as they each contain a trainable layer. DenseNets, as presented in the original work, contain three dense blocks, which are made up of basic blocks, with dense blocks connected to each other by transition blocks. The overall DenseNet structure used in this work contains 40 trainable layers and is visualized in Fig. 1(c).

Basic Blocks: Basic blocks are a low-level structure from which DenseNets are constructed. Basic blocks consist of four layers used on a concatenated input: 1) a batch normalization [23] layer; 2) ReLU activation [24], [25]; 3) a 3 × 3 convolutional layer with step size one; and 4) zero padding, and dropout [26] with a dropout rate of 0.2. These blocks utilize k filters in their convolutional layer, and are visualized in Fig. 1(a).

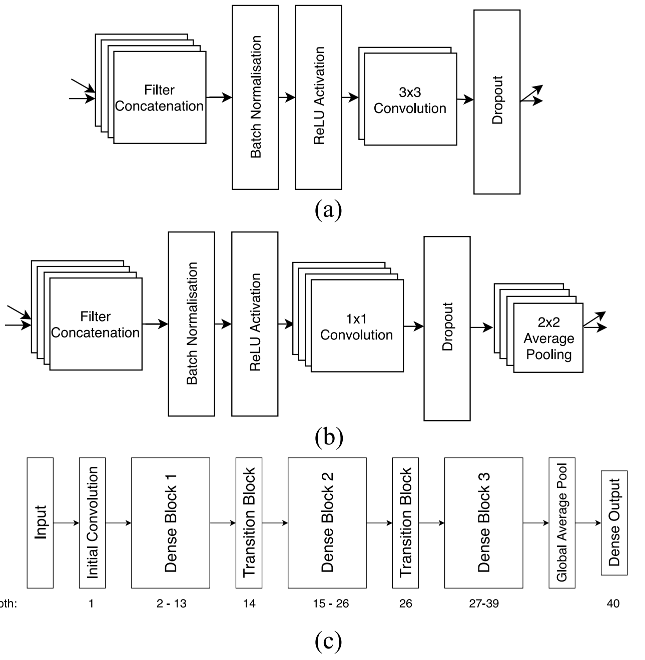

Fig. 1. DenseNet small components (adapted from [22]). (a) Basic block. (b) Transition block. (c) DenseNet structure.

Transition Blocks: Transition blocks connect dense blocks to each other, and are constructed from the following sequence of layers: batch normalization, ReLU activation, 1 × 1 convolution with step size 1, and the number of output filters equals to the size of the input to the preceding dense block plus k times the number of basic blocks per dense block. After convolution, dropout is applied with at a rate of 0.2, and then 2 × 2 average pooling with stride two (reducing the dimensionality of outputs). Transition layers are visualized in Fig. 1(b).

Dense Blocks: Dense blocks consist of an initial input, and several basic blocks. Within a dense block, all permissible skip connections are utilized. The DenseNets utilized in this work contain three dense blocks; thus, with a depth of ℓ, each dense block contains $[(\ell - 4)/3]$ basic blocks (calculated as total depth minus trainable layers not in dense blocks, divided by the number of dense blocks).

C. Neural Architecture Search

Evolutionary NAS has been applied successfully to many different problems, and is emerging as a key paradigm within NAS (see [6], [10]). While much recent work has investigated the use of NAS for the discovery of novel networks, it has primarily searched for high performance modules to be used in a network of prespecified structure (e.g., [5], [27]–[29]), or the automatic generation of networks from scratch, using some predefined layer types and possible connections (e.g., [30]–[38]).

We acknowledge the importance of the contributions to the study of NAS from these works, e.g., AmeobaNet from [5] achieving state-of-the-art results, but will focus discussions on the small amount of prior work focused on the automatic discovery of skip-connection structures on fixed networks, an area that we believe is underexplored.

1) Automated Search for Skip Connections: Two notable works have investigated the automatic selection of skip connections and are strongly related to the current paper.

Genetic CNN: An evolutionary method, named Genetic CNN [7], utilized a Boolean string to represent feedforward connections and skip connections in a network styled after LeNet [3], and used a standard GA to evolve promising candidate structures. Individuals were of size 17, representing a search space of size 2^{17}. While the maximal number of layers was fixed during experiments, for evaluation on the ILSCRC2012 dataset, the researchers increased the width of the networks until a comparable complexity to the state of the art was reached, resulting in slight improvements over the state of the art for the time. However, on CIFAR10 and CIFAR100, where the width of layers was not increased to offset the lower model complexity, the networks did not reach state-of-the-art results.

In addition, Genetic CNN was only considered on networks with a depth of up to 22, with a corresponding search space of 2^{237} bits, and as such was not demonstrated as viable for the selection of skip connections in deeper networks. The current work, for instance, deals with a search space size of 2^{237} (i.e., a space 2^{220} times larger than that considered by Genetic CNN [7]). Furthermore, even within this relatively small search space, Genetic CNN was highly computationally expensive.

Genetic CNN also used accuracy on holdout data as fitness, without demonstrating a consideration of other fitness measures that may be more appropriate for the given search space, and used crossover that swapped contiguous blocks of information between individuals, despite the fact that not all sequential bits in the individuals would be close (as considered by either Manhattan or Euclidean distance) in a genuine adjacency matrix. Furthermore, in [7], no clear analysis of weight complexity is presented, obscuring the benefit of the algorithm and missing a potential opportunity to provide clarity on properties of the considered models.

Probabilistic Gradient Approach: The work done on using gradient approximation to create a probability distribution from which the presence of a skip connection is determined has also been conducted [8]. In this work, Monte Carlo sampling is used to approximate the importance of given skip connections, with gradient descent applied using this information. The research utilizes a user-specified penalty term that determines how heavily the number of skip connections is penalized. The work found that their method was able to reduce the number of skip connections contained in models relative to DenseNet($\ell = 40, k_0 = 16, k = 12$) with less performance degradation than would occur with random connection removal.

The work done in [8] did not find models that outperformed standard DenseNet($\ell = 40, k_0 = 16, k = 12$) on CIFAR10, implying that potentially the search space is sufficiently large such that approximating the relevant multivariate Bernoulli distribution directly is impractical. Furthermore, the algorithm utilized did not remove any skip connections when no error penalty was utilized, and was highly sensitive to this user-specified parameter. In particular, sensitivity to this error penalty, and a lack of ability to determine this parameter prior to experimentation, demonstrate that the algorithm is unable to determine good network structures without significant computational cost in tuning this parameter.

2) Lower Fidelity Estimation Techniques: Due to the high computational cost of fully training CNNs, and the fact that NAS algorithms typically require many evaluations, it has become necessary to find ways of evaluating CNNs with a reduced computational cost. While many different performance estimation techniques exist in the literature, the most common, intuitive, and, in our opinion, broadly applicable performance estimation technique is lower fidelity estimation [6]. Lower fidelity estimation tends to train on only a subset of the training data, train for fewer epochs, or both, reducing the computation time for evaluations. However, despite the prevalence of this method, it is under documented in the research, frequently being used to aid training, but lacking an explanation or thorough analysis of different configurations of lower fidelity estimation on a given task [6].

D. Relationship Between Network Pruning and NAS

Network pruning algorithms have a similar goal to some NAS algorithms, namely, the discovery of small and efficient networks. It should be noted, however, that NAS differs from network pruning in some key ways.

First, network pruning fully trains a parent network, and then seeks subnetworks from this trained network [39]. However, matrix evolution for high-dimensional skip-connection structure (ME-HDSS), as is typical in NAS algorithms, does not utilize the concept of a parent network. Specifically, all architectures considered in ME-HDSS are trained from a random initialization, and the base network is given no preference as a candidate solution (i.e., it will only be evaluated if discovered stochastically by ME-HDSS).

Second, network pruning tends to be an iterative process on a single network, taking a well-trained network and removing filters or layers using some local criteria (e.g., magnitude of trained weights) [39]; however, it should be noted that some recent works diverge from this trend (e.g., performing NAS and network pruning jointly [40] or using an EA to find trained subnetworks stochastically [41]). ME-HDSS, however, does not initialize its architecture search from a given network. Instead, as is typical of NAS algorithms, ME-HDSS begins its search of possible architectures from a random population, and searches for high-performance architectures stochastically from this random initialization.

III. Proposed Approach

This section outlines several key methods utilized in this work. Section III-A first introduces Sparse DenseNets, the

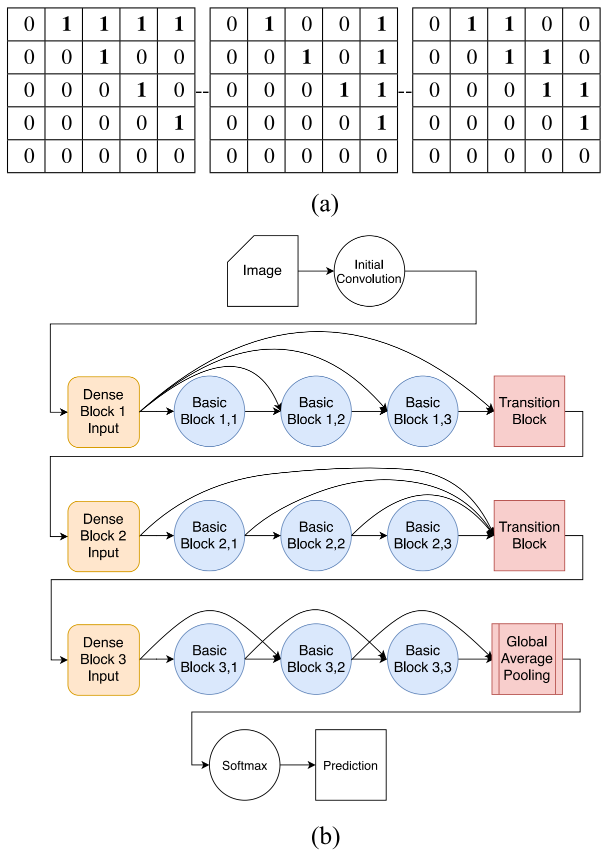
$$\begin{array}{|c|c|c|c|c|}\hline 0&1&1&1&1\\\hline 0&0&1&0&0\\\hline 0&0&0&1&0\\\hline 0&0&0&0&1\\\hline 0&0&0&0&0\\\hline\end{array} - \begin{array}{|c|c|c|c|c|}\hline 0&1&0&0&1\\\hline 0&0&1&0&1\\\hline 0&0&0&1&1\\\hline 0&0&0&0&1\\\hline 0&0&0&0&0\\\hline\end{array} - \begin{array}{|c|c|c|c|c|}\hline 0&1&1&0&0\\\hline 0&0&1&1&0\\\hline 0&0&0&1&1\\\hline 0&0&0&0&1\\\hline 0&0&0&0&0\\\hline\end{array}$$

Fig. 2. Full example individual and interpretation. (a) 3×5×5 representation. (b) Interpretation.

networks which form a core component of this work, including how these networks are represented within the evolutionary NAS algorithm. Section III-B then introduces the proposed evolutionary NAS algorithm, named ME-HDSS. ME-HDSS is a GA-based algorithm designed to directly optimize adjacency matrices of possible skip-connection structures for deep CNNs, allowing for the discovery of more efficient and effective networks given an underlying structure.

A. Individual Representation and Interpretation

This work utilizes DenseNets [4] as both benchmark and the base structure used to demonstrate ME-HDSS, due to their unique suitability for the study of skip connections and the fact that they are among the most accurate and efficient CNNs in the current literature.

Each individual is represented as a 3×N×N Boolean adjacency matrix, where N is the number of layers that can have skip connections attached to them within a dense block, and 3 is the number of dense blocks in the network, which is fixed for all DenseNets. A full example of an individual and resulting network is displayed in Fig. 2.

Thus, each individual is represented as a 3×N×N matrix, where $N = [(\ell - 4)/3] + 2$, with ℓ being a specified parameter for the number of trainable layers in the original DenseNet. The formulation for N is derived from the structure of DenseNet, with four layers in the networks not appearing in Dense modules, the number of layers being evenly distributed across dense modules in the network, and there being two extra possible connections in each dense block corresponding to the input and output of the dense block.

The Boolean values in these matrices represent whether a skip connection between basic blocks is present or not. When interpreting a matrix as an adjacency matrix for use within a Sparse DenseNet, each entry, $M_{k,i,j}$, is interpreted as representing the presence of a directed connection from the output of block i to the input of block j within the dense block k. For example, the outgoing connections from the input layer in the first dense block are all entries associated with the first row (i.e., $M_{1,1,j}, j \in \{1, N\}$), and all incoming connections to the third basic block in the second dense block are the entries associated with the third column (i.e., $M_{2,i,3}, i \in \{1, N\}$). Under this representation, the first row and the first column of a submatrix encode the input of a dense block, the final row and the final column of a submatrix encode the output of a dense block, and each intermediary row and column of a submatrix encode a basic block within the dense block.

B. ME-HDSS Design

1) Fitness Function: Within each generation, the training data are split into two disjoint subsets, one of which is used as subtrain data and the other which is used as holdout data. The same data split is used for all individuals in a given generation, such that comparisons within a generation are fairer.

Once this split has been performed, discovered individuals are used to construct a Sparse DenseNet, and are trained on the subtrain data using stochastic gradient descent. After this training, the individual is evaluated on the holdout data, and the cross-entropy of the individual on the holdout data is used as fitness for the genetic algorithm.

Here, the size of the training data subset, which is used to train the model and the number of epochs used for training are both used to balance the training time of models with the accuracy of the evaluation of the individual. Both of these parameters are specified in Section V-A, where the analysis of lower fidelity estimation configurations is presented.

2) Population Initialization: Individuals are generated from a user specified initialization probability matrix. Given this probability matrix, when a random individual is generated, each bit is set to true with probability corresponding to relevant entry in the probability matrix. For example, if one wanted to generate a random population of individuals where there was a 90% chance that the skip connection from the first input to the second basic block was present, the probability matrix P would have $P_{1,1,3} = 0.9$.

In the current work, these probability matrices are constructed such that when interpreted as a sparse dense block, all feedforward connections are present (i.e., have probability 1), and all other possible skip connections present with a user specified probability. For example, a matrix with shape 2×5×5 (representing a DenseNet with two dense blocks, each containing three basic blocks) with skip-connection probability 0.75 would have the initialization probability matrix shown

0	1	0.75	0.75	0.75
0	0	1	0.75	0.75
0	0	0	1	0.75
0	0	0	0	1
0	0	0	0	0

0	1	0.75	0.75	0.75
0	0	1	0.75	0.75
0	0	0	1	0.75
0	0	0	0	1
0	0	0	0	0

Fig. 3. Example initialization probability matrix.

in Fig. 3. In the current work, this initialization probability matrix has shape 3×14×14.

Initialization probability matrices are used to ensure that ME-HDSS is easily adjustable to precise initialization criteria. For example, if it were known that models had higher performance when skip connections in the third dense block were more prevalent, the initialization can be adjusted such that these skip connections are more likely to be used.

A skip-connection initialization probability of 0.75 is used in this work, and is uniform for all skip connections, as it was found in our preliminary work [22] that when small DenseNets are being used as a base structure, better networks tend to be found with more than half of skip connections utilized. However, the preliminary work did not show any other clear patterns that justify a more fine tuned initialization. Thus, this initialization probability is expected to aid the evolutionary search by providing an initial population with appropriate model complexity, without being overly tuned.

3) Evolutionary Operators: ME-HDSS utilizes standard tournament selection and elitism, but unique and new crossover and mutation operators. The crossover and mutation operators are designed such that all individuals (with the exception of those selected by elitism) always have both operators applied to them, and the crossover and mutation rates controlling how much individuals are changed by each of these operators on average.

Crossover, shown in Algorithm 1, changes an individual by choosing a random mate for it, and with probability $Pr(Crossover)$ swapping the bits of the individual to the corresponding value of the chosen mate. Thus, $Pr(Crossover)$, instead of controlling the probability that crossover occurs, controls how much genetic material, on average, individuals will take from a randomly selected mate.

Mutation, shown in Algorithm 2, randomly changes bits of an individual with $Pr(Mutate)$ chance to a value drawn from the relevant entry in the initialization probability matrix. Mutation, in this way, can be considered analogous to crossover, but instead of making an individual more like another network in the population, it becomes more like a randomly initialized individual. This allows mutation to utilize prior information about the search space (codified in the initialization probability matrix), while still providing diversity to the population.

The selection operation is tournament selection, while the overall population update algorithm, using selection, mutation, and crossover, is shown in Algorithm 3.

Population Size and Number of Generations: In order to perform experiments reasonably quickly, a small population size (i.e., 30) and a small number of generations (i.e., 10) are utilized for evolution. An initial investigation found that with a small population size, the training performance tended tends to plateau at around generation 10, so using more generations with these parameters is a poor use of computational resources for the current experiments. The population size of 30 in this work is determined largely by computational restraints.

Elitism: The number of best individuals transferred to the subsequent generation directly is fixed at 4 throughout the evolutionary process, as it provides a high probability of keeping the best individual through generations (demonstrated in Section V-A). It is important to note that higher elitism rates decrease the amount of exploration conducted during the evolutionary process, and thus, this elitism rate is chosen to balance exploration and exploitation in the algorithm.

Tournament Size, Crossover Probability, and Mutation Probability: To determine tournament size, crossover probability, and mutation probability, the performance of the novel algorithm was assessed on a task that is computationally inexpensive, but demonstrates the ability of the algorithm to find optima in a search space similar to the target NAS search space. The details of this method are presented in Section IV-B, and the full results from this method are presented in Section V-B. The final selected parameters were a tournament size of 3, crossover probability of 0.5, and a mutation probability of 0.05. The overall evolutionary parameters for ME-HDSS are shown in Table I.

Algorithm 1 Crossover

```
   Input: $Ind_1$, $Ind_2$, $Pr(Crossover)$, $N$
1: for index in {0,1,2} do:
2:     $M_1 \leftarrow Ind_1$.matrices[index];
3:     $M_2 \leftarrow Ind_2$.matrices[index];
4:     for $i, j$ in $1, ..., N$ do
5:         if Random.random() < Pr(Crossover) then
6:             $M_1[i, j] \leftarrow M_2[i, j]$;
7:         end if
8:     end for
9: end for
```

Algorithm 2 Mutation

```
   Input: $Individual$, $Initialisation\_Matrix$, $Pr(Mutation)$
1: $random\_ind \leftarrow$ individual generated using $Initialisation\_Matrix$;
2: Crossover($Individual$, $random\_ind$, $Pr(Mutation)$);
```

Algorithm 3 Updating a Population

```
   Input: $Population$, $Elitism$, $Pr(Crossover)$, $Pr(Mutate)$
   Require: $Population$ is sorted
1: $New\_Pop = Selection(Population)$;
2: for $Ind$ in $New\_Pop$ do:
3:     $Crossover(Ind, random\_partner(New\_Pop))$
4:     $Mutate(Ind)$;
5: end for
6: $New\_Pop[:Elitism] = Population[:Elitism]$;
```

TABLE I
ME-HDSS EVOLUTIONARY PARAMETERS

Parameter	Value
Number of generations	10
Population size	30
Tournament size	3
Elitism	4
Crossover probability	0.5
Mutation probability	0.05

TABLE II
LOWER FIDELITY ESTIMATE CONFIGURATION

Parameter	Value
Fitness function	Cross entropy on $Holdout\_subset$
$\lvert Training\_subset \rvert$ during evolutionary procedure	5,000
$\lvert Holdout\_subset \rvert$ during evolutionary procedure	45,000
$\lvert Training\_subset \rvert$ for final evaluation	10,000
$\lvert Holdout\_subset \rvert$ for final evaluation	40,000

Algorithm 4 ME-HDSS

```
    Input: Initialisation_Matrix, Training_Data
 1: Gen_Count ← 0;
 2: Initalise Population;
 3: while Gen_Count ≤ Generations do
 4:     Training_Subset, Holdout_Subset ← split Training_Data s.t.
        |Training_Subset| = 5,000 and |Holdout_Subset| = 45,000;
 5:     for Individual in Population do:
 6:         Network ← Sparse DenseNet generated from Individual;
 7:         train Network on Training_Subset for 300 Epochs;
 8:         set fitness of Individual using Fitness_Function;
 9:         if Gen_Count ≠ Generations then
10:             Population ← update using Algorithm 3;
11:         end if
12:     end for
13:     Gen_Count++;
14: end while
15: Considered_Networks ← the four Individuals in Population with best fitness values;
16: Training_Subset, Holdout_Subset ← split Training_Data s.t. |Training_Subset| =
    10,000 and |Holdout_Subset| = 40,000;
17: for Individual in Considered_Networks do:
18:     Network ← Sparse DenseNet generated from Individual;
19:     train Network on Training_Subset for 300 Epochs;
20:     set fitness of Individual using Fitness_Function;
21: end for
    return best individual in Considered_Networks;
```

4) Overall ME-HDSS Algorithm: Using the above evolutionary operators and parameters, the final ME-HDSS algorithm is displayed in Algorithm 4. This is displayed in full to provide clarity, particularly regarding how the training data are utilized within evolution, and how the final network is selected from the final generation. Notably, the final network is selected by taking the top four individuals in the final generation, and evaluating them with higher fidelity than is used in the evolutionary process. The specific performance estimation configuration used in ME-HDSS is displayed in Table II, and the experimental results leading to this selected configuration are presented in Section V-A.

5) Mutation Analysis: This analysis focuses how likely mutation is to add or remove skip connections in ME-HDSS, as measured by expected change. The full derivation of these formulas can be found in the online supplementary material.

Given the probability $P(C) = [\text{Count}(\text{bits} = 1)/\text{Count}(\text{bits})]$, an equal likelihood of a selected bit set to 1 or 0, and defining change $= \text{bit}_{t-1} - \text{bit}_t$ for a random *bit* at time t, we note that $\mathbb{E}(\text{change}) = 0.5 - P(C)$. This is zero (i.e., unbiased) when $P(C) = 0.5$ and scales linearly otherwise, indicating some bias within the search space toward half of possible skip connections being selected.

It should be noted that when a similar analysis is performed on the mutation operator used in Genetic CNN [7], which selects a bit and deterministically flips it, $\mathbb{E}(change) = 1 - 2P(C)$. This is again zero when $P(C) = 0.5$; however, it is approximately twice as biased toward half of the possible skip connections being selected as the mutation operation used in ME-HDSS, scaling by $-2P(C)$ rather than $-P(C)$.

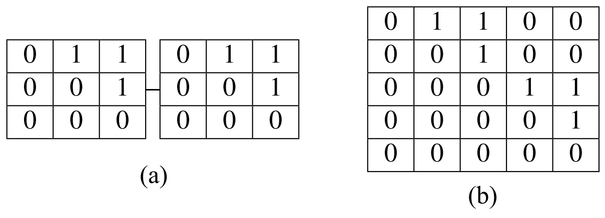

Fig. 4. Different possible representations of the same network. (a) 2×3×3 representation. (b) 5×5 representation.

Rationale of Not Enforcing that $\mathbb{E}(change) = 0$: Where the bit flipping probability is not constant, but rather the probability of flipping a bit to 1 equals $1 - P(C)$, the expected difference is 0 for all values of $P(C)$. However, this leads to issues for extreme values of $P(C)$. For example, under this possible formulation where $P(C) = 1.0$ (i.e., all connections selected) or $P(C) = 0.0$ (i.e., all connections not selected) mutation will never change the value of a bit, removing any benefit of the mutation operator.

Thus, the mutation operator used in ME-HDSS can be said to allow, in likelihood, more diversity of solutions during search than prior mutation operations for similar problems (i.e., [7]), while avoiding issues that can arise from using dynamic mutation rates to directly overcome bias.

6) On the Generality of the Proposed Representation: While the current work seeks to show that novel skip-connection structures can improve the efficiency and accuracy of networks using DenseNet [4] as a base structure, the representation and algorithm are easily extendable to other networks, as shown below. A specific note on application to ResNet [17] style networks can be found in the online supplementary material.

Generalization to Removing Layers: In the current work, ME-HDSS enforces that all feedforward connections are utilized through the initialization probability matrix using these connections with probability 1.0. If this probability was less than 1.0, it would be possible to remove layers from the corresponding network entirely.

Generalization to Nonblock-Based Networks: As the DenseNets considered in this work utilize dense blocks, creating individuals as networks with shape $B \times N \times N$, where B equals the number of dense blocks, reduces some memory overhead and allows for greater clarity in analysis, but there is a functionally equivalent representation with shape $\ell \times \ell$ (e.g., Fig. 4). Importantly, because our initialization probability matrix itself matches the shape of our desired individual, under the $\ell \times \ell$ representation, if an operation was introduced, which made skip connections between blocks permissible, these could be trivially introduced into the architecture search.

Generalization to Recurrent Networks: In the current work, recurrent networks are not considered, and this is enforced in ME-HDSS by ensuring that the diagonals of matrices are zero. If one wished to generalize ME-HDSS to standard recurrent networks, they need only add a nonzero probability to the diagonals of the adjacency matrices and, where layers are self-connected, replace them with a chosen recurrent layer.

Algorithm 5 Assessing Estimation Configuration

```
Input: Network
1: Training_Dataset ← CIFAR100 training data;
2: Epochs ← {150, 200, 300};
3: Training_Sizes ← {2500, 5000, 10000};
4: Estimation_Cases ← ['training_accuracy', 'training_entropy', 'holdout_accuracy', 'holdout_entropy'];
5: Pseudo_Train, Pseudo_Test ← split Training_Dataset s.t. $|Pseudo\_Train| = 40,000$ and $|Pseudo\_Test| = 10,000$;
6: train Network on Pseudo_Train for 300 Epochs;
7: store Network accuracy on Pseudo_Test;
8: reset Network weights;
9: for all combinations of Epochs and Training_Sizes do:
10:     Training_Subset, Holdout_Subset ← split Pseudo_Train s.t. $|Training\_Subset| = Train\_Size$ and $|Holdout\_Subset| = 40,000 - Train\_Size$;
11:     train Network on Train_Subset for Epoch;
12:     store Network accuracy and entropy on Train_Subset and Holdout_Subset;
13:     reset Network weights;
14: end for
```

Furthermore, for a generalization of recurrent networks, the lower triangle of matrices could be initialized with nonzero connection probability, with these connections being interpreted as passing the output from a given layer at time $t-1$ as input to a preceding layer at time t.

IV. Experiment Design

This section details how the three connected experiments in the current work are designed. Section IV-A details how, using the Sparse DenseNets introduced in Section III-A, lower fidelity estimation configurations will be assessed. Section IV-B details the approximate parameter tuning method used to tune parameters for ME-HDSS. Section IV-C details the experimental setup for ME-HDSS, also using the Sparse DenseNets introduced in Section III-A, including an overview of the datasets used.

A. Assessing Lower Fidelity Estimation Configurations

To examine the expected performance of lower fidelity estimates for network performance, a random population of 60 Sparse DenseNets($k_0 = 16, k = 12, \ell = 40$) are created, using the initialization probability of 1.0 for feedforward connections and 0.75 for skip connections within dense modules, and their performance is considered with respect to a pseudotest set, using Algorithm 5. Networks are trained using the training schedule outlined in the original DenseNet paper [4].

Once Algorithm 5 has been used to gather data on the different possible configurations for lower fidelity estimation, all of the considered configurations will have their Pearson correlation to pseudotest set accuracy calculated. This correlation will be used to assess how well different configurations are likely to predict test performance, and thus, which configuration should be selected for use in ME-HDSS.

B. Approximate ME-HDSS Parameter Tuning

The computationally inexpensive task used for parameter tuning is the ability of the algorithm to optimize toward arbitrary points in the search space, using distance from this arbitrary point as a fitness function to be minimized. Intuitively, as it is not known which structures are likely to be optimal when performing NAS, the ability of the algorithm to discover arbitrary optima in the space provides information as to how well the algorithm can generically search, given a fixed population size and fixed number of generations. Specifically, using a population size of 30 and evolving the population for ten generations using the evolutionary operators outlined in Section III-B, the following steps are performed in 100 independent trials.

1) A random target matrix with shape 3×14×14 is generated.
2) For each combination of mutation rate, crossover rate, and tournament size, the evolutionary process is run using the Manhattan distance between individuals and the target matrix as fitness, aiming to minimize fitness.
3) The best Manhattan distance found in the final generation is recorded.

The considered values of parameters are: Mutation Probabilities $= \{0.01, 0.02, 0.05, 0.10, 0.20\}$, Crossover Probabilities $= \{0.05, 0.10, 0.15, \ldots, 1.00\}$, and Tournament Sizes $= \{2, 3, 4\}$. The results from this process will then be used to inform parameter selection for ME-HDSS, with adjustments made to account for how this simplified search problem may bias parameter tuning away from exploration.

C. Assessing ME-HDSS

Benchmark Network Choice: The DenseNet published in the original work with the fewest layers, namely, DenseNet ($k_0 = 16, k = 12, \ell = 40$), is selected as both the base network and benchmark in this work. The smaller network is selected for two reasons: 1) so that ME-HDSS can be demonstrated as able to improve CNNs that are not obviously overparameterized, and 2) to increase the number of trials that are able to be conducted. It is to note that while CNNs in some respects are highly parallelizable, their time complexity still increases linearly with the number of sequential layers used, and thus, a CNN with twice the number of sequential layers of another network will also have approximately twice the time complexity.

Overall Evaluation Procedure: ME-HDSS will be evaluated on the CIFAR10 and CIFAR100 datasets, with five independent trials being run for each of these datasets. The analysis will then be conducted on the resulting networks (five optimized for CIFAR100 and five optimized for CIFAR10). The analysis will first consider the classification performance and number of trainable weights of the returned networks against the benchmark DenseNet ($k_0 = 16, k = 12, \ell = 40$), aiming to achieve better classification performance using fewer trainable weights. It is important to note that removing skip connections from DenseNets necessarily reduces the number of trainable weights in the network, and that the number of trainable weights in a network is a measure of network complexity.

Then, the evolutionary behavior of ME-HDSS will be assessed, and returned networks from the independent trials are analyzed. The analysis of these networks seeks to find skip-connection patterns, which provide insight into efficient networks. If simple patterns are found, then further experiments will be performed to assess the possible benefits of the discovered patterns.

V. Results From Parameter Tuning

This section details the results of the first two experiments outlined in Section IV. In particular, Section V-A presents the results and analysis of lower fidelity estimation configurations, which informs the choice of lower fidelity estimation configurations and elitism rate used in testing the ME-HDSS algorithm. Section V-B presents the results of the approximate parameter tuning algorithm.

A. Lower Fidelity Estimation Configurations

1) Grid Search Over Lower Fidelity Estimation Configurations: A grid search was performed over epochs and training data subset sizes, using the parameters Epochs $\in \{150, 200, 300\}$, $|Training\_Subset| \in \{2500, 5000, 10000\}$. Fig. 5 shows the correlations for this grid search.

In this grid search, a more stable relationship between parameters and correlation can be seen, with higher epochs and more training data appearing beneficial (although more expensive). In particular, several of these configurations achieve higher than 0.8 Pearson correlation, indicating that cross-entropy on holdout training data, under these parameter settings, offers a good lower fidelity performance estimation configuration for these models on this dataset. Furthermore, the settings Epochs $= 300$, $|Training\_Subset| = 5000$ offer a Pearson correlation of 0.82, as shown in Table III, which is ten times less expensive to train than the procedure used for CIFAR10 and CIFAR100 in the original DenseNet paper [4], and appears to distinguish high performing individuals fairly well. Also, it is notable that the configuration Epochs $=$ 300, $|Training\_Subset| = 10000$ is particularly well correlated for higher performance individuals. Thus, the selection of Epochs $= 300$, $|Training\_Subset| = 5000$ with holdout cross-entropy as fitness function as a lower fidelity estimation of the performance of individuals during the evolutionary process. The configuration Epochs $= 300$, $|Training\_Subset| = 10000$ using cross-entropy on holdout data as a fitness function is utilized when selecting the final solution from the evolutionary process, as by the final generation individuals should be relatively high performance, and this configuration should have advantage in choosing a single solution from a pool of high performance candidates. Table III shows all correlations to the accuracy on the pseudotest set over the considered measures under the refined grid search, with the highest Pearson correlation across measures for a given configuration bolded.

2) Stochasticity of Repeated Evaluation: Evolutionary computation methods are sensitive to stochasticity in candidate evaluation due to the importance of candidate ranking in these algorithms (e.g., when fitness is close between individuals, small amounts of noise can change the outcome of elitism and selection operations).

To quantify this effect, the stochasticity of evaluation under two lower fidelity estimation configurations is examined, and the likelihood of elitism selecting the best individual under these two lower fidelity estimation configurations is assessed. Namely, $|Training\_Subset| = 5000$ and #Epochs $\in \{150, 300\}$ are selected as lower fidelity estimation configurations, with $|Training\_Subset| = 5000, \#Epochs = 300$ being the lower fidelity estimation configuration selected for use in the novel ME-HDSS, and $|Training\_Subset| = 5000, \#Epochs = 150$ being used to show how stochasticity can increase when the fidelity of estimates becomes even lower than that used in ME-HDSS.

Specifically, a population of 30 random individuals are generated and trained 30 times on different random subsets of the training data, where $|Training\_Subset| = 5000$, with negative holdout cross-entropy recorded for each of these trainings. Thus, for each individual, the distribution of returned cross-entropy under lower fidelity performance estimation techniques on CIFAR100 can be observed. Fig. 6 shows the plot of negative cross-entropy for the 30 individuals, where the individuals are ordered by μ_{loss} over the repeated evaluations, for both 150 and 300 epochs of training.

Table IV shows a comparison of standard deviations between these two lower fidelity estimation configurations. In particular, relative to 150 epochs of training, 300 epochs of training slightly lowers the stochasticity of a given individuals evaluation [i.e., intraindividual standard deviation, or

TABLE III
Pearson Correlation to Pseudotest Accuracy for All Considered Fitness Functions

		Pearson Correlation with Pseudo-Test Accuracy			
#Train	#Epochs	Train Acc	-Train Loss	Holdout Acc	-Holdout Loss
2500	150	**0.73**	0.01	0.02	-0.07
2500	200	**0.71**	-0.01	0.25	0.53
2500	300	0.57	0.58	0.31	**0.67**
5000	150	**0.74**	0.67	0.62	0.71
5000	200	0.75	0.81	0.68	**0.81**
5000	300	0.77	0.81	0.61	**0.82**
10000	150	**0.81**	0.74	0.67	0.77
10000	200	0.8	0.71	0.60	**0.83**
10000	300	0.77	0.82	0.62	**0.87**

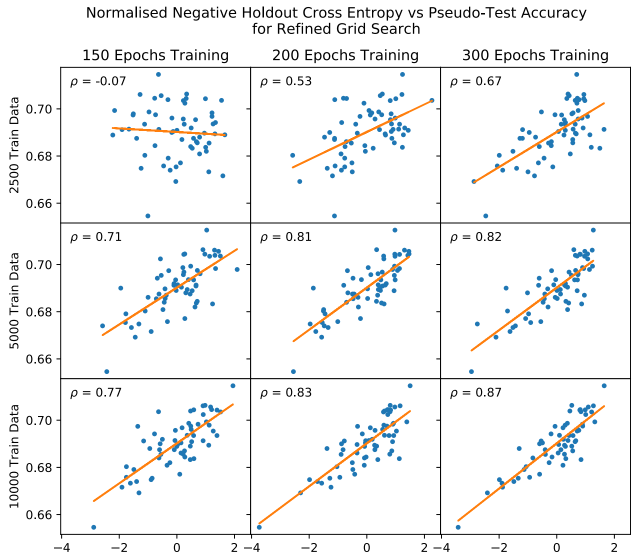

Fig. 5. Holdout loss correlation plot.

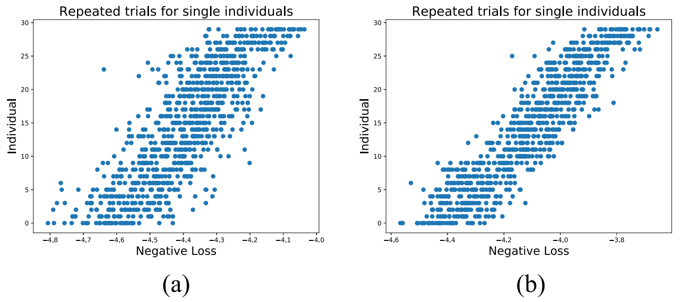

Fig. 6. Stochasticity of loss under low-fidelity performance estimation. (a) 150 epoch training. (b) 300 epochs training.

TABLE IV
MEAN INTRAINDIVIDUAL STANDARD DEVIATION OF CROSS-ENTROPY VERSUS OVERALL STANDARD DEVIATION OF CROSS-ENTROPY

	$\mu(\sigma_{individuals})$	σ_{all_trials}
150 epoch	0.08	0.12
300 epoch	0.07	0.18

TABLE V
PROBABILITY OF ELITISM KEEPING BEST INDIVIDUAL

Elitism rate	1	2	3	4	5
150 epoch	0.35	0.59	0.78	0.88	0.93
300 epoch	0.55	0.73	0.84	0.91	0.97

$\mu(\sigma_{individuals})$], while increasing the average range of evaluation over all groups (i.e., overall standard deviation, or σ_{all_trials}), meaning that it is generally easier to distinguish a good individual from a poor individual during the evolution with 300 epochs of training under this lower fidelity estimation configuration. To quantify the effect that this difference can have during evolutionary processes, and perform analysis to inform elitism rate selection, the application of the elitism operator on these individuals is simulated.

Table V shows the probability of elitism selecting the best individual (measured by mean fitness) when sampling a performance estimation of the 30 random individuals. The elitism rate denotes how many of the top individuals are copied from a generation directly into the next generation. As shown by Table V, when selecting evaluation strategies in evolutionary algorithms, not only is the relationship between the fitness function and goal important but also stochasticity of evaluation has measurable flow on effects to the rest of algorithm design. In this case, the chance that elitism will retain the best individual throughout generations can be seen to differ by up to 20% due to changes in stochasticity. With respect to the current work, Table V shows that an elitism rate of 4 provides a 90% chance of retaining the best candidate between generations, as ranked by mean performance. As this elitism rate is also small enough that diversity throughout generations should not be too adversely affected, it is used as the value for the elitism rate in ME-HDSS.

B. Parameter Tuning Results

The results of the parameter tuning method outlined in Section IV-B can be seen in Fig. 7, which shows the mean normalized Manhattan distance from the optimal points for

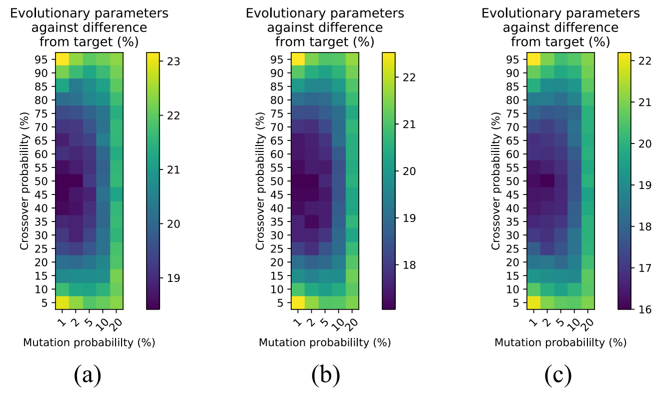

Fig. 7. Mean of parameter tuning results. (a) Tournament size = 2. (b) Tournament size = 3. (c) Tournament size = 4.

each combination of parameters over 100 runs. In this simplified search space, the following parameters are optimal: *Tournament Size* = 4, *Crossover Probability* = 0.5, and *Mutation Probability* = 0.02. The standard deviation of these results was fairly uniform over different parameter combinations, with $\sigma \approx 1.5\%$. However, this search space for parameter tuning is not only free of noise but also entirely convex, and thus, will likely devalue mutation and overvalue tournament size, as diversity tends to be less important when there are clear routes to global optima. Hence, the parameter selection for ME-HDSS is adjusted toward higher mutation and lower tournament size, giving the selected evolutionary parameters for ME-HDSS: *Tournament Size* = 3, *Crossover Probability* = 0.5, and *Mutation Probability* = 0.05.

C. Summary

This section discussed the results of the empirical assessment of low-fidelity performance estimation techniques, fitness functions, and how characteristics of a low-fidelity performance estimation technique can be used to inform evolutionary algorithm design. In particular it was found that as indicated on the CIFAR100 dataset, utilization of small training sets (i.e., 5000–10 000 training examples) with a high amount of holdout data was beneficial, but a reasonably high number of epochs was still required to judge performance. For fitness functions, it was determined that holdout cross-entropy was beneficial, although training accuracy also appears viable under several configurations.

VI. ME-HDSS RESULTS

This section presents results verifying the effectiveness of the novel ME-HDSS algorithm, and includes sections relating to the evolutionary behavior of ME-HDSS, discovered network analysis, and an investigation into a unique skip connection highlighted by ME-HDSS.

A. Performance

Over five independent trails, ME-HDSS consistently outperformed the baseline DenseNet ($k_0 = 16, k = 12, \ell = 40$) in terms of the classification error rate and the network size on both the CIFAR10 and CIFAR100 datasets.

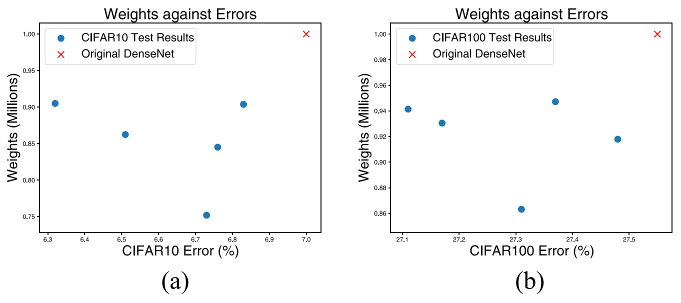

Fig. 8. ME-HDSS test result plots. (a) ME-HDSS(C10) networks on CIFAR10. (b) ME-HDSS(C100) networks on CIFAR100.

On CIFAR100, networks evolved by ME-HDSS had on average 0.32% better classification error rate than the baseline DenseNet, while being on average ~8% smaller in terms of the number of trainable parameters. Furthermore, ME-HDSS produced a model that outperforms the baseline by 0.24% classification error rate while having 13.58% fewer trainable parameters. On CIFAR10, networks evolved by ME-HDSS had on average 0.37% better classification error rate than the baseline DenseNet, while being on average ~15% smaller in terms of the number of trainable parameters. Furthermore, ME-HDSS produced a model that outperforms the baseline by 0.24% classification error rate while having 24.82% fewer trainable parameters. The exact performance and weights over these five trials are presented in Fig. 8 (where points closer to the origin are more optimal and in each subplot, the lowest blue point is the smallest final model discovered for that dataset).

The five networks (from independent trials) that are evolved by ME-HDSS for CIFAR10 [denoted as ME-HDSS(C10)] are also tested on the CIFAR100 data, and the five networks (from independent trials) that are evolved by ME-HDSS for CIFAR100 [denoted as ME-HDSS(C100)] are also tested on the CIFAR10 dataset. This is to investigate the generality of the models evolved by ME-HDSS, although the primary goal of this work is to create an algorithm that can optimize skip-connection structure for a given problem. The results of these trials showed that ME-HDSS(C100) networks, when applied to CIFAR10, roughly matched the performance of the ME-HDSS(C10) networks, although being larger, which makes them less efficient for the CIFAR10 dataset. The converse did not hold, however, and when applied to CIFAR100, the ME-HDSS(C10) networks underperformed both the baseline and the ME-HDSS(C100) networks, indicating that a wider variety of networks is suitable for CIFAR10 compared to CIFAR100, and thus, ME-HDSS may only generalize when transferring networks optimized for more complicated datasets to simpler datasets. A summary of all results, including these additional evaluations, can be seen in Table VI.

B. Evolutionary Behavior

The evolutionary behavior of the ME-HDSS algorithm showed stable training behavior overall, but with different behaviors between the two datasets. Fig. 9 shows the mean fitness and number of trainable weights for individuals over all trials in ME-HDSS on CIFAR10, while Fig. 10 does the same for ME-HDSS on CIFAR100.

TABLE VI
OVERALL ME-HDSS TEST RESULTS

Network/s	#Parameters	CIFAR10 Test Error	CIFAR100 Test Error
DenseNet	1.0M	7.00%	27.55%
ME-HDSS(C10)	$\mu = \mathbf{0.85}$M, $\sigma = 0.06$, $min = \mathbf{0.75}$M	$\mu = \mathbf{6.63\%}$, $\sigma = 0.19\%$, $best = \mathbf{6.32\%}$	$\mu = 28.07\%$, $\sigma = 0.31\%$, $best = 27.73\%$
ME-HDSS(C100)	$\mu = \mathbf{0.92}$M, $\sigma = 0.03$, $min = \mathbf{0.86}$M	$\mu = \mathbf{6.65\%}$, $\sigma = 0.25\%$, $best = \mathbf{6.35\%}$	$\mu = \mathbf{27.23\%}$, $\sigma = 0.13\%$, $best = \mathbf{27.11\%}$

Note: Improved results are bolded.

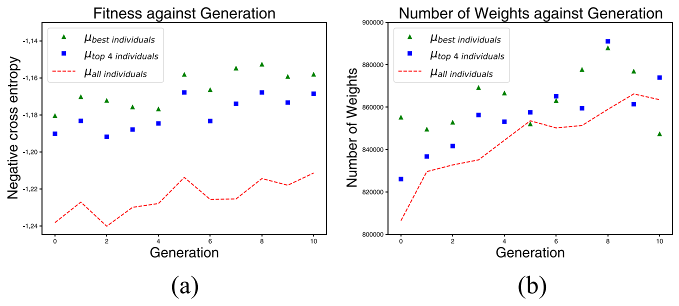

Fig. 9. ME-HDSS evolutionary behavior on CIFAR10. (a) Fitness by generation. (b) Weights by generation.

CIFAR100 appears to have fewer high performance networks associated with it, with relatively low initial population performance for ME-HDSS on CIFAR100 that quickly improves, before more stable improvements are seen. ME-HDSS on CIFAR10, however, shows relatively consistent improvement over all generations. The observation that fewer high performance networks exist for CIFAR100 is reinforced by the results shown in Table VI, where the ME-HDSS(C100) networks generalize to CIFAR10, but the ME-HDSS(C10) networks do not appear to generalize as well to CIFAR100.

Likewise, when observing the size of models, the larger models consistently produced higher performance on CIFAR100, and that the average number of weights in the models trends consistently higher throughout generations, although the number of weights in the best performing individuals appears relatively stable from generation 5 onward. This behavior can be interpreted as ME-HDSS successfully moving enough individuals to an optimal weight range to find good models, with the improved performance of the final ME-HDSS networks relative to baseline further evidence of the benefit of this weight range. ME-HDSS on CIFAR10, to contrast, quickly finds top performing models that contain fewer trainable parameters than the average individuals in a given generation, indicating that it found an optimal weight range more easily.

C. ME-HDSS(C100) Network Analysis

Training behavior across all the five final networks (corresponding to one network for each independent trial) was very similar, showing that the discovered skip-connection structures are all similarly effective with regards to gradient propagation

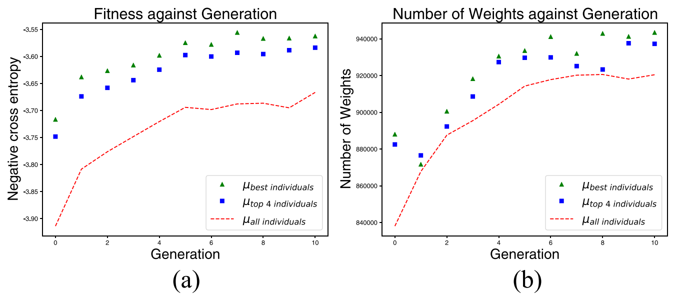

Fig. 10. ME-HDSS evolutionary behavior on CIFAR100. (a) Fitness by generation. (b) Weights by generation.

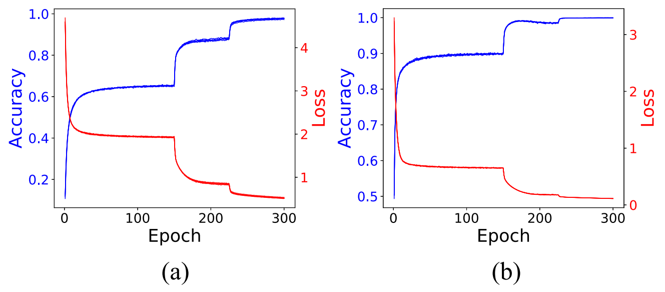

Fig. 11. Network training accuracy and loss. (a) ME-HDSS(C100) networks on CIFAR100. (b) ME-HDSS(C10) networks on CIFAR10.

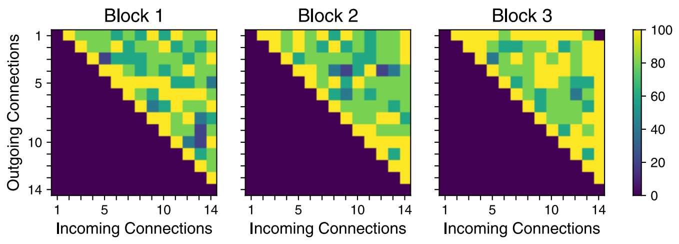

Fig. 12. ME-HDSS(C100) networks selected skip connections by percent occurrence.

during training, despite model size differences. Fig. 11 shows a plot of the training accuracy and training cross-entropy loss for the 5 ME-HDSS(C100) networks on CIFAR100; however, the accuracy and loss of all networks overlap heavily for each epoch of training.

Fig. 12 shows the percentage of the final 5 ME-HDSS(C100) networks in which a given connection is present within the three dense blocks [i.e., the number of ME-HDSS(C100) networks in which given skip connections were present, divided by the number of independent trials], and Table VII shows the average number of skip connections present within a given dense block. Overall, the final ME-HDSS(C100) networks showed consistent patterns across independent trials for both skip-connection structures and the overall number of skip connections within specific blocks. It should be noted that the exact relationship between skip-connection structures and parameters is nontrivial, and is explored in the online supplementary material.

As shown in Table VII, dense block 3 reliably has more skip connections than dense blocks 1 and 2, indicating that these networks benefit from more connections between deeper layers in networks.

TABLE VII
ME-HDSS(C100) NETWORKS AVERAGE NUMBER OF SKIP CONNECTIONS BY BLOCK

Dense block	1	2	3
μ skip-connections	59.60	61.20	66.00
σ skip-connections	1.96	1.72	1.10

It is also of note that the probability of a connection within dense block 3 being used in the final ME-HDSS(C100) networks appears almost negatively correlated to the $L1$ norms of connection weights shown in Fig. 5 from the original DenseNet paper [4]. However, despite this apparent relationship within dense block 3, other dense blocks less appear to have less correlation between the probability of a connection being selected and this original $L1$ norm analysis. This implies that the magnitude of final weights is not necessarily a reliable indicator of the importance of given connections on trained networks. Furthermore, where the $L1$ norm of trained weights could be used as an indicator of the importance of a given skip connection in a network it may, counter intuitively, be the skip connections with the smallest magnitudes of trained weights that are the most important to network performance. This reinforces the notion that the primary benefit of many skip connections may come from improvements to gradient propagation, rather than direct contributions to inference.

Dense Blocks 1 and 2: As can be seen from Fig. 12, skip connections are relatively inconsistent in the first dense block across independent trials, with the exception of a large number of skip connections from basic blocks 4 and 5 [seen in Fig. 12 as the yellow streak under outgoing connections 4 and 5 in block 1]. For basic block 4, these skip connections are always to immediately proceeding blocks, in the style of ResNet [17] connections, while for basic block 5, there are skip connections to basic blocks that are further away. Furthermore, in both dense block 1 and dense block 2, a connection from the input directly to the output is present, implying that in earlier dense blocks, this unique connection is consistently important for network performance. This connection is unique because it directly passes information from one input to the next without any 3×3 convolutions, normalizations, activations, or dropout being applied to it.

Dense Block 3: Dense block 3 shows clearer patterns than the preceding dense blocks. This block, which directly precedes the output layer, has the most heavy utilization of skip connections, and consistently, connects the input layer to almost all later layers and almost all layers directly to the output layer. Interestingly, connections directly to the output in dense block 3 increase the number of features used in the final classification, and are the only connections in the network with the ability to do this, making them unique in the context of DenseNets. Contrary to the above patterns, however, the connection directly from the input layer to the output layer [the upper right connection under block 3 in Fig. 12] is absent in all trials. Further analysis on this specific skip connection (i.e., $M_{3,1,14}$) is conducted in later in this section.

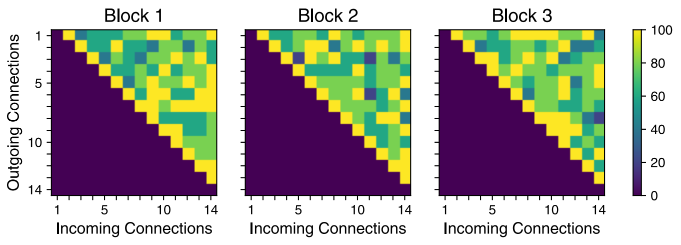

Fig. 13. ME-HDSS(C10) networks selected skip connections by percent occurrence.

TABLE VIII
ME-HDSS(C10) NETWORKS AVERAGE NUMBER OF SKIP CONNECTIONS BY BLOCK

Dense block	1	2	3
μ skip-connections	59.60	58.60	58.80
σ skip-connections	3.87	3.61	2.78

D. ME-HDSS(C10) Network Analysis

As with the ME-HDSS(C100) networks, training behavior of the ME-HDSS(C10) networks was similar across all independent trials, which shows that the discovered skip-connection structures are all similarly effective with regards to gradient propagation during training, with the smallest network discovered showing no discernible visual decrease in the efficacy of the backpropagation algorithm.

However, the final ME-HDSS(C10) networks showed less consistent patterns in skip-connection structures compared to the ME-HDSS(C100) networks. Fig. 13 shows the percentage of final networks in which a given connection is present within the three dense blocks [i.e., the number of ME-HDSS(C10) networks in which given skip connections were present, divided by the number of independent trials], and Table VIII shows the average number of skip connections present within a given dense block. It should again be noted that the exact relationship between skip-connection structures and parameters is nontrivial, and is explored in the online supplementary material.

Unlike the ME-HDSS(C100) networks, the dense blocks in the ME-HDSS(C10) networks tended to have a similar number of skip connections between blocks, but also had a higher standard deviation for the number of skip connections within dense blocks. This implies that on the CIFAR10 dataset, dense blocks themselves may not require as much specialization as they do for the CIFAR100 dataset. Furthermore, whereas for the ME-HDSS(C100) networks, the skip connection directly from the input to the output for dense block 3 was always absent; for ME-HDSS(C10), it is only absent in three out of five trials. This again indicates that there are a broader range of appropriate networks for CIFAR10, as it is a simpler dataset, and note that overall, the ME-HDSS(C10) networks were smaller than the ME-HDSS(C100) networks, providing further evidence for this claim.

E. $M_{3,1,14}$

As noted, $M_{3,1,14}$, the connection directly from the input to the output in the final dense block, is absent in all of the discovered ME-HDSS(C100) networks and three out of five of the discovered ME-HDSS(C10) networks, implying that this connection can be detrimental to network performance. While this is possibly true, the proposed ME-HDSS algorithm, by design, considers the performance of skip connections in the context of all other selected skip connections at, and thus, the removal of this skip connection may only be beneficial where other specific skip-connection structures already exist in the network. Thus, the effect of the removal of this connection in isolation should be considered, noting that $M_{3,1,14}$ is unique in a number of ways.

First, in all of the networks discovered, both dense blocks 1 and 2 have information passed directly from the input to the output; thus, $M_{3,1,14}$ includes nearly unprocessed information from the input images. Second, $M_{3,1,14}$ contributes the single largest number of filters to the output layer, due to the way in which transition blocks get wider throughout a DenseNet. Third, $M_{3,1,14}$ directly connects an average pooling operation to the global average pooling operation. This could lead to detrimental learning in the network, with filters being optimized to output the presence of features in parts of the image where they do not appear.

To look at the effect of the removal of this skip connection in isolation, a network is created that reflects the original DenseNet (i.e., all skip connections present), with the exception that $M_{3,1,14}$ is removed. This network is then trained on both CIFAR10 and CIFAR100, before evaluating the final test performance of the model.

Results of Removing Only $M_{3,1,14}$ From DenseNet: As can be seen in Table IX, the network has a nearly unchanged number of trainable parameters. This network outperforms the original DenseNet on CIFAR10 while underperforming the original DenseNet on CIFAR100. This indicates that while this skip connection may be detrimental to performance on some datasets, it is not universally the case that removing this connection will improve results.

It is of note that despite the benefit of removing this skip connection from DenseNet in isolation, it was absent in only three of five of the ME-HDSS(C10) networks, and the test classification error of the trained DenseNet without skip connection $M_{3,1,14}$ is effectively equal to the mean classification error of the five ME-HDSS(C10) networks. This indicates that there are a number of different skip-connection structures, which improve the performance and efficiency of this DenseNet on CIFAR10, and that removing this connection is simply one pathway to performance improvement.

On CIFAR100, despite being absent from all ME-HDSS(C100) networks, the removal of this skip connection by itself decreases the performance of DenseNet. This implies that on CIFAR100, there are likely fewer, more specific, skip-connection structures, which improve both performance and efficiency of the baseline network. The removal of this skip connection, thus, appears to reliably form just a part of resulting good skip-connection structures, with actual benefits coming from interactions between the removal of $M_{3,1,14}$ and other skip connections. This highlights the importance of considering interactions between all skip connections when designing and evaluating networks in ME-HDSS.

TABLE IX
PERFORMANCE AFTER REMOVAL OF $M_{3,1,14}$

Model	#Parameters	CIFAR10 Test Error	CIFAR100 Test Error
Original DenseNet	1.00M	7.00%	27.55%
DenseNet $\setminus M_{3,1,14}$ (C10)	0.97M	**6.66%**	-
DenseNet $\setminus M_{3,1,14}$ (C100)	0.96M	-	28.50%

Note: Improved results are bolded.

F. Summary

The performance of the novel evolutionary NAS algorithm, ME-HDSS, was demonstrated on the CIFAR10 and CIFAR100 datasets, with analysis provided on discovered skip-connection structures. ME-HDSS was found to be able to consistently find skip-connection structures, which outperformed the base model on these datasets, while making models smaller by up to 24.82%. These results further indicate that the lower fidelity estimation experiment yielded appropriate design choices.

When analyzing the networks discovered by ME-HDSS, it was found that skip-connection structures optimized for CIFAR100 generalized to CIFAR10, but that the reverse was not true. Further analysis of discovered skip-connection structures showed that CIFAR100 appears to require more specialized skip-connection structures than CIFAR10, with more consistent and unique skip-connection patterns being found by ME-HDSS on the CIFAR100 dataset.

An analysis of a specific skip connection, which was frequently absent from networks, revealed that removal of just this skip connection from DenseNet($k_0 = 16$, $k = 12$, $\ell = 40$) improved its performance on CIFAR10, but not on CIFAR100. The identification of the removal of this skip connection from DenseNet as a simple way to improve DenseNet performance on simpler datasets is of possible benefit to some machine learning practitioners.

VII. CONCLUSION AND FUTURE WORK

This work has developed and presented a new Evolutionary NAS algorithm, ME-HDSS, which is the first algorithm, to our knowledge, to improve accuracy on both the CIFAR10 and CIFAR100 datasets relative to a baseline CNN through only the removal of skip connections. In addition, discovered skip-connection structures were analyzed and found to share many common patterns. Notably, a skip connection with unique characteristics was highlighted, and the removal of this skip connection was shown to improve DenseNet performance on CIFAR10, but not CIFAR100.

This work also assessed many lower fidelity performance estimation configurations, being among the first works, to our knowledge, to extensively examine configurations over a class of models. It was found that using a small subset of data for weight training using stochastic gradient descent and a comparatively large amount for data for performance evaluation during the evolutionary process appeared to correlate more highly to test performance while significantly reducing the computational cost of evaluation, especially when performance was evaluated using cross-entropy. The stochasticity introduced by lower fidelity performance estimation was also assessed, and it was shown how algorithm design can be informed analytically by this information.

There are several avenues for possible future work. In particular, the adaptation of ME-HDSS to discover novel structures related to other networks, as described in Section III-B, is clearly an important extension to the current work. Furthermore, the use of an evolutionary multiobjective framework should be used to explicitly document the tradeoff between model complexity and performance under this skip-connection structure framework. Finally, a continued investigation into other forms of performance estimation techniques (e.g., weight sharing between networks), which may further reduce computation time and improve results, should be investigated.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 818–833.

[3] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[4] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2017, pp. 2261–2269.

[5] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

[6] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," 2019. [Online]. Available: arXiv:1905.01392.

[7] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1379–1388.

[8] S. Saito and S. Shirakawa, "Controlling model complexity in probabilistic model-based dynamic optimization of neural network structures," in *Proc. Int. Conf. Artif. Neural Netw.*, 2019, pp. 393–405.

[9] E. Amaldi and V. Kann, "On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems," *Theor. Comput. Sci.*, vol. 209, nos. 1–2, pp. 237–260, 1998.

[10] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[11] A. E. Orhan and X. Pitkow, "Skip connections eliminate singularities," 2017. [Online]. Available: arXiv:1701.09175.

[12] R. Yasrab, "SRNET: A shallow skip connection based convolutional neural network design for resolving singularities," *J. Comput. Sci. Technol.*, vol. 34, no. 4, pp. 924–938, 2019.

[13] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017. [Online]. Available: arXiv:1704.04861.

[14] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015. [Online]. Available: arXiv:1511.07289.

[15] B. Graham, "Fractional max-pooling," 2014. [Online]. Available: arXiv:1412.6071.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 1026–1034.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[18] Z. Liao and G. Carneiro, "On the importance of normalisation layers in deep learning with piecewise linear activation units," 2015. [Online]. Available: arXiv:1508.00330.

[19] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[20] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[21] J. Shore and R. Johnson, "Axiomatic derivation of the principle of maximum entropy and the principle of minimum cross-entropy," *IEEE Trans. Inf. Theory*, vol. IT-26, no. 1, pp. 26–37, Jan. 1980.

[22] D. O'Neill, B. Xue, and M. Zhang, "The evolution of adjacency matrices for sparsity of connection in DenseNets," in *Proc. IEEE Int. Conf. Image Vis. Comput. New Zealand (IVCNZ)*, 2019, pp. 1–6.

[23] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015. [Online]. Available: arXiv:1502.03167.

[24] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Stat.*, 2011, pp. 315–323.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "DropOut: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] H. Liu, K. Simonyan, and Y. Yang, "DARTs: Differentiable architecture search," in *Proc. ICLR*, 2019, pp. 1–9.

[28] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 1294–1303.

[29] X. Wang, Y. Jin, and K. Hao, "Evolving local plasticity rules for synergistic learning in echo state networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1363–1374, Apr. 2020.

[30] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," 2017. [Online]. Available: arXiv:1711.00436.

[31] E. Real *et al.*, "Large-scale evolution of image classifiers," 2017. [Online]. Available: arXiv:1703.01041.

[32] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 350–364, Apr. 2020.

[33] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, Apr. 2020.

[34] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020.

[35] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 921–934, Dec. 2019.

[36] G. Kyriakides and K. Margaritis, "Regularized evolution for macro neural architecture search," in *Proc. IFIP Int. Conf. Artif. Intell. Appl. Innov.*, 2020, pp. 111–122.

[37] C. Li *et al.*, "Block-wisely supervised neural architecture search with knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1989–1998.

[38] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search," 2020. [Online]. Available: arXiv:2007.10396.

[39] J. Liu, S. Tripathi, U. Kurup, and M. Shah, "Pruning algorithms to accelerate convolutional neural networks for edge applications: A survey," 2020. [Online]. Available: arXiv:2005.04275.

[40] T. Wang *et al.*, "APQ: Joint search for network architecture, pruning and quantization policy," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2078–2087.

[41] Y. Zhou, G. G. Yen, and Z. Yi, "Evolutionary compression of deep neural networks for biomedical image segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 8, pp. 2916–2929, Aug. 2020.

Damien O'Neill (Member, IEEE) received the B.A. degree in psychology from the University of Otago, Dunedin, New Zealand, in 2011, the graduate certificate in business from the Auckland University of Technology, Auckland, New Zealand, in 2014, and the B.Sc. degree (Hons.) in computer science from the Victoria University of Wellington, Wellington, New Zealand, in 2017, where he is currently pursuing the Ph.D. degree in artificial intelligence and machine learning.

He is a Data Lead for Analytics with Contact Energy Ltd., Wellington. His primary research interest is the intersection of evolutionary computation and artificial neural networks.

Bing Xue (Senior Member, IEEE) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science from the Victoria University of Wellington (VUW), Wellington, New Zealand, in 2014.

She is currently a Professor of Computer Science and the Program Director of Science with the School of Engineering and Computer Science, VUW. She has over 300 papers published in fully refereed international journals and conferences and her research focuses mainly on evolutionary computation, machine learning, classification, symbolic regression, feature selection, evolving deep neural networks, image analysis, transfer learning, and multiobjective machine learning.

Dr. Xue is currently the Chair of IEEE Computational Intelligence Society (CIS) Task Force on Transfer Learning and Transfer Optimization, the Vice-Chair of IEEE CIS Evolutionary Computation Technical Committee, an Editor of IEEE CIS Newsletter, and the Vice-Chair of IEEE Task Force on Evolutionary Feature Selection and Construction and IEEE CIS Task Force on Evolutionary Deep Learning and Applications. She served as an Associate Editor of several international journals, such as *IEEE Computational Intelligence Magazine* and IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.

Mengjie Zhang (Fellow, IEEE) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2000.

He is currently a Professor of Computer Science, the Head of the Evolutionary Computation Research Group, and an Associate Dean (Research and Innovation) with the Faculty of Engineering, Victoria University of Wellington, Wellington, New Zealand. He has published over 600 research papers in refereed international journals and conferences. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job shop scheduling, and transfer learning.

Prof. Zhang was the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee and IEEE CIS Emergent Technologies Technical Committee, and the Evolutionary Computation Technical Committee and a member of the IEEE CIS Award Committee. He is the Vice-Chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction and Task Force on Evolutionary Computer Vision and Image Processing and the Founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is a Committee Member of the IEEE NZ Central Section. He is a Fellow of Royal Society of New Zealand, and has been a Panel Member of the Marsden Fund (New Zealand Government Funding) and a member of ACM.