Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2022.020975
Article

Tech Science Press

An Improved Genetic Algorithm for Automated Convolutional Neural Network Design

Rahul Dubey* **and Jitendra Agrawal**

School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, India
*Corresponding Author: Rahul Dubey. Email: rdofficials05@gmail.com
Received: 17 June 2021; Accepted: 27 August 2021

Abstract: Extracting the features from an image is a cumbersome task. Initially, this task was performed by domain experts through a process known as handcrafted feature design. A deep embedding technique known as convolutional neural networks (CNNs) later solved this problem by introducing the feature learning concept, through which the CNN is directly provided with images. This CNN then learns the features of the image, which are subsequently given as input to the further layers for an intended task like classification. CNNs have demonstrated astonishing performance in several practicable applications in the last few years. Nevertheless, the pursuance of CNNs primarily depends upon their architecture, which is handcrafted by domain expertise and type of investigated problem. On the other hand, for researchers who do not have proficiency in using CNNs, it has been very difficult to explore this topic in their problem statements. In this paper, we have come up with a rank and gradient descent-based optimized genetic algorithm to automatically find the architecture design of CNNs that is vigorously competent in exploring the best CNN architecture for maneuvering the tasks of image classification. In the proposed algorithm, there is no requirement for handcrafted pre- and post-processing, which implies that the algorithm is fully mechanized. The validation of the proposed algorithm on conventional benchmarked datasets has been done by comparing the run time of a graphics processing unit (GPU) throughout the training process and assessing the accuracy of various measures. The experimental results show that the proposed algorithm accomplishes better and more persistent 'classification accuracy' than the original genetic algorithm on the CIFAR datasets by using fifty percent less intensive computing resources for training the individual CNN and the entire population.

Keywords: Convolutional neural network (CNN); genetic algorithm (GA); differential architecture search (DART)

1 Introduction

Convolutional neural networks (CNNs) are one of the leading techniques of 'deep learning' [1] and have exhibited superior performance in several real-world problems over the various customary machine learning algorithms [2]. The first CNN was developed in 1998, known as LeNet5 [3]. Subsequently, various versions

of CNN, namely AlexNet [4], VGG [5], GoogleNet [6], and ResNet [7], have been proposed on the basis of variation in architectures. The reasoning behind such designs is that a deeper CNN typically is able not only to handle more complex problems but also to deal with large amounts of data. The state-of-the-art CNN is specifically designed by domain experts who possess knowledge of datasets and CNN architecture. However, such a combination is rare, as researchers having proficient knowledge of data may not be well adroit with CNN architecture design or vice versa. Consequently, there is a vast demand to come up with an algorithm that not only allows researchers to explore the best CNN architecture but also use fewer computational resources. In fact, in the literature, distinct algorithms for automation purposes have been stated over the past few years. These algorithms can be classified into two groups depending on their base techniques. The first group is 'evolutionary algorithms' [8] like the genetic CNN algorithm. The second group is based on reinforcement learning like neural architecture search [9]. However, these algorithms have demonstrated very optimistic classification accuracy against onerous benchmark datasets like CIFAR 10 and CIFAR 100 [10]. However, certain limitations exist in this regard. Firstly, both category algorithms suffer from the usage of high computational resources. Secondly, manual assistances based on domain expertise are desirable for most of the algorithms in both categories. As a result, the development of such an algorithm is exigent to perform the following tasks:

- Automatically best CNN architecture design for given data
- Use of limited computational resources
- No manual assistance

In this paper, a proposition of rank and a gradient descent-based optimized genetic algorithm have been given to find the architecture design of CNNs automatically using optimized computational resources. The main benefits of presenting this paper are as follows:

1) It optimized the well-known evolutionary method known as the genetic algorithm for automation and efficient computational resource utilization.
2) It converted a discrete set of candidate architecture searches into a continuous architectural search by using DARTS or the gradient-based architecture search method [11].
3) It has demonstrated that gradient-based architectural search achieves extremely promising results on CIFAR-10 and CIFAR-100 through massive image classification studies. This is an exciting conclusion given that the most effective architectural search approaches to date have relied on non-differentiable search approaches such as RL and evolutionary approaches.

This article is divided into the following sections. Section 2 presents the background of the proposed methodology. Section 3 shows the proposed contribution. Then, Sections 4 and 5 discuss the experimental design and results. Conclusion and the scope for future work are stated in Section 6.

2 Background

In this section, we will discuss CNNs in general, the different types of CNNs, the genetic algorithm, and the principle of network architecture searches. These topics are part of the background of the proposed algorithm and will also help the reader understand the proposed algorithm.

2.1 Convolutional Neural Networks

In this subsection, we will discuss the basics of CNNs. The CNN is an important invention in the world of computer vision. It is a multidisciplinary concept that combines mathematics and biology with computer science. In 1988, Fukushima discovered the architecture of a neural network known as CNN [3]. This was the first CNN, which later became the basis for all subsequent CNNs. In CNNs, there are two important layers: convolution and pooling. The convolutional layer uses filters for convolution operation. The output of this

convolutional layer is called a feature map, which gives the parameters of CNN. Sometimes a padding operation is also used along with a convolution operation. After the convolution operation, the pooling layer performs the pooling operation. There are two types of pooling: average and max pooling. The number of parameters of pooling depends on the size of the kernel. The fully connected layer [FC] comes after the last convolutional layer. Although the power of a CNN mainly depends upon the manner of filters' usage and the way the layers are connected, gradient back propagation is the main learning algorithm for all types of CNNs. To design an optimized CNN architecture, it is mandatory to know the parameter calculation of each layer. According to [12], the size of a CNN layer is calculated based on the below equation

$$C_w = C_h = (I + 2 \times P - f)/s + 1 \quad (1)$$

where I is the size of the input image, P is padding, f is the size of the filter, and s is stride (which defines how far the filter will move from one position to the next position Cw), and Ch is the size of the convolution layer. Fig. 1 demonstrates the working of a CNN with its number of parameters. The following steps are taken to find parameters mathematically.

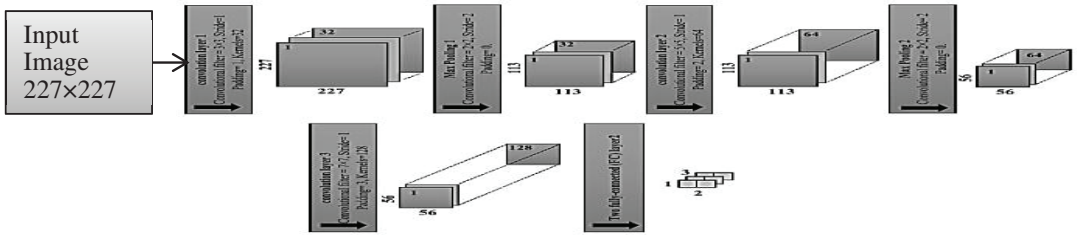

Figure 1: Convolutional neural network

1. As shown in Fig. 1, for Convolution-I layer, (I = 227, f = 3, p = 1, s = 1, and total filter = 32). This means

$$C_w = C_h = \frac{227 + 2 \times 1 - 3}{1} + 1 = 227$$

As there are 32 filters, the total number of neurons in the feature map of the first convolutional layer is $227 \times 227 \times 32 = 1{,}648{,}928$

2. The size of this layer is the previous layer's input size divided by the size of stride. As shown in Fig. 1, Stride is 2, and the previous layer's input size is 227. So, $\text{maxpooling}_1 = \frac{227}{2} \approx 113$.
3. As shown in Fig. 1, for Convolution-II layer, I = 113, f = 5, p = 2, s = 1, and total filter = 64. This means

$$C_w = C_h = \frac{113 + 2 \times 2 - 5}{1} + 1 = 113$$

As the number of fitters is 64, the total number of neurons for the feature map of the second convolutional layer is $113 \times 113 \times 64 = 817{,}216$

4. As shown in Fig. 1, stride is 2 and the previous layer's input size is 113. Therefore, $\text{maxpooling}_2 = \frac{113}{2} \approx 56$
5. As shown in Fig. 1, for Convolution-III layer, I = 56, f = 7, p = 3, s = 1, and total filter = 128. This means

$$C_w = C_h = \frac{56 + 2 \times 3 - 7}{1} + 1 = 56$$

As the number of fitters is 128, the total number of neurons for the feature map of the third convolutional layer is 56 × 5 ×6 128 = 401,408

6. As shown in Fig. 1, the last layer (the fully connected layer) is responsible for calculating the class score. The input size for this layer is 401,408.

In this paper, the primary center of attention is the convolution and pooling layers. Fundamental objects are encoded by the proposed algorithm to represent CNNs. In the upcoming two subsections, we will discuss the types of CNNs that have been used as population and peer competitors for our proposed algorithm.

VGG-19. VGG stands for the Visual Geometry Group (of the University of Oxford) [5]. VGG came after Alex Net. It took ideas from old CNNs and changed them to achieve a high level of accuracy. It is mainly trained on Image Net dataset, which has 1.2 million images. The architecture of VGG is shown below in Fig. 2.

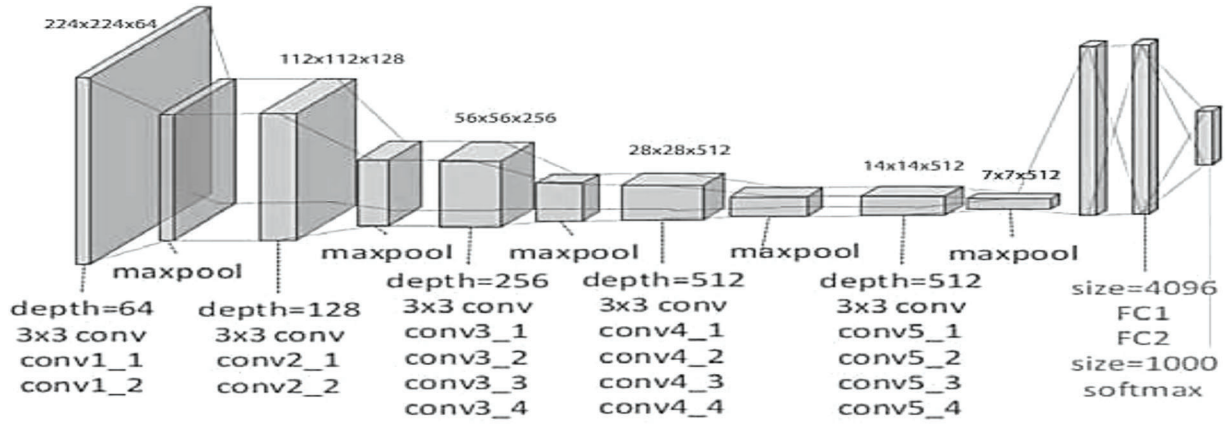

Figure 2: VGG-19 architecture

Important points about VGG are as follows: The size of the input is always fixed at 224 × 224. It does not require any special processing, as only normalization is applied. The kernel size is 3 × 3. A rectifier linear unit is used to provide nonlinearity during processing. Three fully connected layers of 4096 neurons are used. The last layer is the classification layer in which the softmax function is used.

ResNet. [13] Because of the vanishing gradient problem, it is extremely difficult to train a very deep neural network (a neural network with many layers). Skip connection is a solution that allows the user to take the activation from one layer and instantly feed it to another layer that is further deeper in the network, allowing them to train networks with more than 100 layers. The residual block is formed by these skip connections or shortcuts. Residual networks are made up of residual blocks. These networks may grow deeper without compromising performance. The residual block structure is depicted in Fig. 3 below.

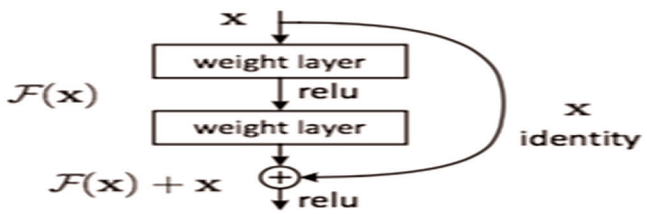

Figure 3: Residual block

Residual Network Equation:

$$y_l = h(x_l) + f(x_l,\ w_l)x_l \tag{2}$$

$$x_{l+1} = f(y_l) \tag{3}$$

In these equations, $f(y_l)$ and $h(x_l)$ are 'identity mappings' (i.e., the signal could be directly propagated from one unit to any other unit, either forward or backward). This discussion is enough to establish the benefit of ResNet over other networks and is also helpful to understand the proposed algorithm because this is one of the networks we used in our population.

2.2 Genetic Algorithm

The genetic algorithm [14,15] flowchart is given in Fig. 4. First, the initial population of the individual is triggered randomly (regarding our proposal algorithm, the convolutional neural network is the individual of the population with its variable architecture), and subsequently, the fitness of the individual is evaluated. To measure this fitness, we use a novel fitness function. In the case of our proposed algorithm, the fitness of the individual is measured based on its performance on image classification then the individuals whose fitness is the best are selected to generate offspring. Offspring are generated by crossover and mutation. After this, we merge these offspring into the population, and the above process continues until we get the optimized solution.

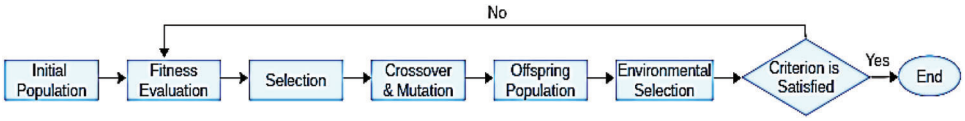

Figure 4: Flow chart of genetic algorithm

2.3 Principal of Network Architecture Search

In recent years, CNN architectures have shown remarkable performance in classification, localization, video classification, segmentation, and captioning. Examples of their applications include DCNN for hyper spectral imaging segmentation [3], image registration [16], handwritten character recognition [17], and optical image classification [18]. As the manual design of CNN classification is time-consuming—and because the deep learning frameworks are heavily data-dependent and deal with versatile data—this type of classification requires frequent architecture changes. Therefore, it is mandatory to develop an algorithm that can find the best CNN architecture automatically and quickly.

Suitable architecture designing can be considered as a search problem [9]. As shown in Fig. 5, this system is divided into three parts: the search space, searching strategy, and performance estimation. Concerning this work, the collection of CNN architectures are a search space. Here, one point is very important: the proper search space definition can reduce the complexity of a system and save time. The search strategy is the core of this NAS system, with better strategies generating better CNN architectures. However, there is always a back-and-forth in between the classification performance and speedup. In this work, because we use a single GPU, fast convergence is required. Performance estimation is the final important component used by the strategy to evaluate the performance of a system. In our case, the classification accuracy and execution time of a CNN to find the best architecture for a validation data set are the main performance measures considered.

Figure 5: Automatic architecture design principle

3 Proposed Methodology

This section mainly deals with the proposed work. Subsection 3.1 introduces the framework of the proposed algorithm, and Subsections 3.2–3.5 provide the details of each step. To provide a better understanding of the work, we will give remarks with justifications for each section. The proposed algorithm's framework is shown in 'Algorithm 1.'

Algorithm 1:

Input: CNN building blocks, size of the population, number of generations (G), data set for classification.

Output: best CNN architecture.

1 $I_0 \leftarrow$ initialize the population with a defined size by generating individuals randomly

2 for ($i = 0$; $i < G$; $i = i + 1$) **do**

- **Fast** fitness evaluation of each individual in I_i using **proposed Algorithm 3**
- $J_i \leftarrow$ select the best parents per fitness and generate the offspring by using genetic operations.
- $I_{i+1} \leftarrow$ new population selection from the joint pool of I and J

3 Return the Best Individual According to Fitness I_i

Algorithm 1 introduces the framework of the proposed algorithm. It starts by working with the initial population of the CNN. After several evolutionary steps, it discovers the best architecture of CNN for the classification of the image. During progressions, the population is initialized instantly with the size defined beforehand, and individuals are encoded using the bit representation encoding technique [6] (line 1). There is one counter, i, that is initialized with zero for the current generation.

During evolution, the fitness of an individual is evaluated on a given image dataset using the proposed algorithm. After that, the best parents are selected according to fitness and rank values, and then new offspring are generated using crossover and mutation genetic operations. Then, a new population is selected from the joint pool of the surviving individuals and the current population, which becomes the population for the next generation. This process continues with the maximum range of generations as shown in Fig. 4.

The proposed algorithm has a standard pipeline of GA. However, one point to be noted here is that GA provides a unified framework to solve optimization problems. When this algorithm is used for a particular problem, its components must be redesigned. In the proposed algorithm, we carefully followed the bit encoding strategy, genetic operations. With this, we also proposed the novel "individual fitness evaluation algorithm" for individual fitness evaluation and the calculate_rank algorithm to accelerate convergence. We also used a differential architecture search to train individuals to make the CNN architectures automated and time-efficient.

Algorithm 2: Individual Training:

Input: learning rates β_1 and β_2, number of epochs, w, and α

Output: Best CNN architecture

1. For every epoch:
    a. For every training and validation batch,
        i. $\alpha \leftarrow \alpha - \beta_2 \nabla_{\alpha} L_v((w - \beta_1 \nabla_w L_t(w,\ \alpha),\ \alpha)$
        ii. $w \leftarrow w - \beta_1 \nabla_w L_t(w,\ \alpha)$
2. Choose the best α^* according to the performance on the validation dataset.
3. $o_{i,j} = maximum(o \in O\ \ \alpha_{i,j}^{o\,*})$ find the best CNN

Algorithm 2 explains the gradient-based architecture search method (also known as the differential architecture search method) [11]. To find the best solution for any problem, it is always important to define the search space precisely. The incessant relaxation of the architecture representation is allowed by the gradient-based search. It does not search a discrete set of candidate architectures but relaxes the search space to be incessant so that the architecture can be optimized regarding its validation set performance by 'gradient descent.'

Although the idea of searching in continuous space is not new, this method is different from the previous one. It discovers efficient performance building blocks with complex graph topologies within rich search space. Moreover, this method is not restricted to any particular family of neural networks; it is a generalized method. The efficiency of gradient-based optimization, unlike the random hidden search, allows it to exhibit tremendous performance with relatively few computational resources.

This method searches [19,20] a computation cell (shown in Fig. 6) as a building block of the final architecture. Searched optimal cells are stacked together to form an efficient CNN architecture. Each cell is like a direct acyclic graph and consists of two inputs and one output; the inputs are given by the previous two cells. Let us consider N_i as the node of the cell and $O_{i,\ j}$ as an operation set (like pooling, convolution padding, and so on) between N_i and N_j. Thus, the output of N_i is calculated based on all previous nodes per Eqs. (4) and (5). The cell output, Cout, is calculated as the concatenation of the output of each intermediate node.

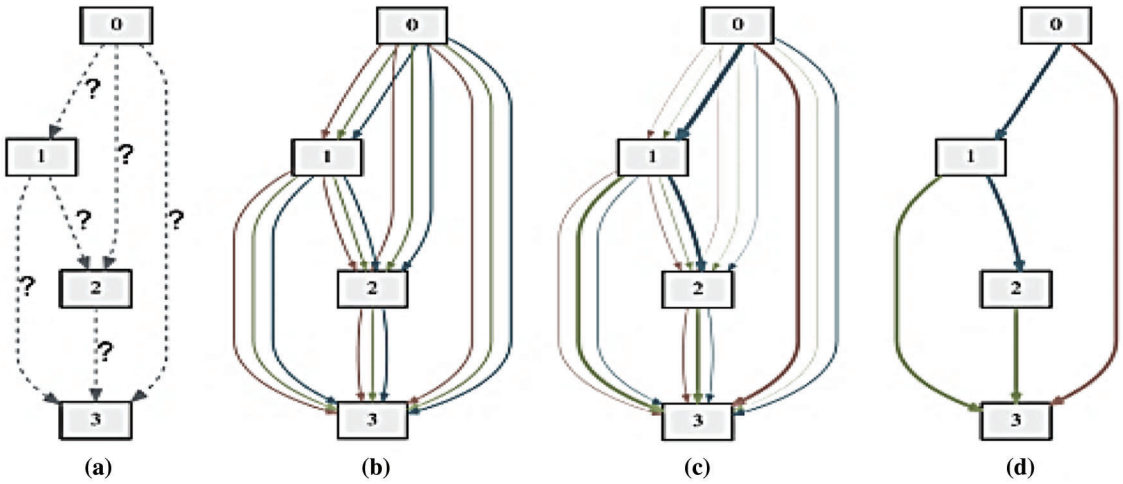

Figure 6: Scrutiny of DARTS: (a) unknown operations initially (b) continuous relaxation of the search space (c) Bi-level optimization, and (d) final architecture

$$N_i = \sum_{j<i} o_{i,j}(N_j) \tag{4}$$

$$Cout = \sum concatnate(N_i) \tag{5}$$

The search procedure can be accelerated by considering the gradient-based optimization method. It is well-established that gradient-based approaches always require a continuous search space. So, by using a soft-max function, this categorical space can be converted into a continuous search space. The output of an operation set is 'the weighted sum of the outputs of each operation in the operation set,' which is described by Eqs. (6) and (7).

$$C^o_{i,j} = \frac{expo(\alpha^o_{i,j})}{\sum_{o' \in O} expo(\alpha^o_{i,j})} \tag{6}$$

$$o_{i,j}(N_j) = \sum_{o \in O} C^o_{i,j} \; o(N_j) \tag{7}$$

where $C^o_{i,j}$ is operation o's coefficient under operation set O between node N_i, the N_j coefficient is obtained by the softmax operation, and $\alpha^o_{i,j}$ is a parameter optimized by the gradient descent method. Upon completion of the search, the most appropriate operation, according to $o_{i,j} = maximum(o \in O \; \alpha^o_{i,j})$, is selected as the final operation, and then the network architecture is determined. Due to the above-described method, the discrete search space is changed to continuous search space on which gradient descent can be applied. After this continuous relaxation, the goal is to find the optimal α and w, which can be obtained by minimizing the losses through training and validation. Let L_t and L_v be training and validation losses, respectively. Then optimal parameter can be obtained by the bi-level optimization algorithm, which can be understood by using Eq. (8).

$$\min_{\alpha} L_v(w, \; \alpha) \; such \; that \; w = \min_{w} L_t(w, \; \alpha) \tag{8}$$

where α is the gradient descent parameter for CNN architecture and w is the weight in CNN. After the bi-level optimization shown in line 1.a.i, the optimized values α* and w* are used to design the CNN architecture. Instead of finding the best architecture through human or manual expertise, this work uses the gradient descent parameter to find the best CNN architecture.

Algorithm 3: (Proposed Algorithm) Fitness Evaluation

Input: Population Ii

Output: Population Ii contains individual with fitness values.

```
1 if i == 0 then
2       lookup = {};
3       Make look_up as global variable;
4 end
5 for every individual in Ii do
6       if the index of individual in look_up then
7       f ← get the fitness using index from look_up;
8             rank←find rank using Proposed Algorithm calculate_rank (individual, f)
```

(Continued)

Algorithm 3: (continued)

```
8              Set f and rank to individual;
9        else
10               while GPU is available do
11               evaluate individual on GPU asynchronously using proposed algorithm Fast Individual
                   Fitness evaluation
12               end
13         end
14 end
15 Return Ii.
```

The method of fitness evaluation is described by Algorithm 3 to evaluate the fitness of each individual in population Ii and ultimately return the population whose fitness has been evaluated. Explicitly at the initial population level, a global cache signified as 'look_up' is generated. This cache stores individual fitness with obscured architectures (lines 1–3). After that, the fitness of every individual found is taken from this memory, and their ranks are calculated according to the proposed Algorithm 6, which is described in a later section (lines 5–8). Otherwise, an individual is assigned to the available GPU for fitness evaluation. The whole process is done asynchronously (lines 10–12). One point to be noted here is that from look_up, each individual is accessed based on its index.

Algorithm 4: (Proposed Algorithm) Individual Fitness Evaluation

Input: The individual, GPU availability, training epochs, the global look_up buffer, the training data, the validation data,

Output: The individual and its fitness.

```
1 Build CNN using encoding strategy over given data set;
2 valid_fit ← 0;
3 for every epoch in the given training epochs do
4       Train the CNN using Algorithm 3 on training data by using the given GPU;
5       v_f ← Calculate the error rate and time taken to process the image on validation data;
6       if v_f > valid_fit then
7       valid_fit ← v_f;
8       end
9 end
10 mark valid_fit as the fitness of individual;
11 set the index of the individual into look_up buffer;
12 Return individual
```

Algorithm 4 presents the logic of the evaluation of individual fitness. At the outset, the CNN is decoded from the individual, and a classifier is added to this CNN (line 1) based on the image classification given in

the dataset. In the proposed algorithm, the softmax classifier [15] is used, and classes are derived from the benchmark data set. After decoding, various operations like batch normalization, dropout, and stride are added to perform the training of CNN. All configurations are managed per the literature to achieve optimized performance.

Training is performed using Algorithm 2 (line 4), which is the best available differentiable architecture search method to reduce the time of training and achieve fast automation. When the training phase is completed based on the time taken to train, error rate, and the accuracy fitness values assigned to individuals, the index of an individual with its fitness is finally set in the look_up cache (lines 10–11).

The next reason for designing DART-based, asynchronous, and cache components, for example, is given here. First, the training of the CNN is time-consuming. So, it is better to make a decision based on the best architecture design as discussed in Algorithm 2. DART does not completely run the CNN until it finds the best architecture. Therefore, it only considers the positive CNN and kills individuals that underperform. So, it is the fastest way to decide the best architecture since multiple training sessions are not required to find the best performance.

Second, deep learning architectures reach their full potential with large amounts of data, which necessitates a lengthy training period. However, most architectures are based on gradient-based optimization, which can be performed in parallel and asynchronously—and, in this work, fitness evaluation also supports this technique. Indeed, there are various existing libraries such as tensor flow and Pytorch that also support this technique. The cache component is also used to provide the memorization based on the following assumptions.

1. Fitness evaluation is not required again for individuals whose architectures have not changed and that have survived until the next generation.
2. After the crossover and mutation operation, the regeneration of the same architectures may be probable. One problem that can arise is that duplicate key generation is also handled by dictionary concept, which is an index and value mapping. For example, a record such as "index1 = 88.12," which denotes the index, is "index1," and its fitness value is "88.12."

Algorithm 5: Offspring Generating

Input: Ii Population set and fitness, crossover probability cp, the mutation operation probability mp, list of mutation operation ml, and mutation operation selection probability pl.

Output: offspring set Ji.

```
1: Ji = { }
2 while |Ji| < |Ii| do
3        I1 ← randomly select two points and then select the best one from Ii
4        I2 ← Repeat Line 3;
5 while I2 == I1 do
6        step 4;
7 end
8 r ← random value in between (0, 1);
9 if r < cp then
10        randomly divide the initial population into P1 & P2 find O1 & O2 using crossover
11        Ji ← Ji U O1U O2;
```

(Continued)

Algorithm 5: (continued)

```
12 else
13      Ji ← Ji U P1 U P2;
14 end
15 end
16 for each individual p in Ji do
17      r ← random value from (0, 1);
18      if r < mp then
19      based on randomly chosen place t do mutation from operation chosen from
            The list of operations based on probability pl
20      end
21 end
22 Return Ji
```

The process of offspring generation is shown in Algorithm 5. It contains two main parts: crossover and mutation. During the crossover operation, two parents are selected, one from each list of randomly generated individuals based on their fitness evaluation (lines 1–4). This selection process is called binary tournament and is often used as a single-point optimization technique. As a parent is selected, one random number is generated to help determine whether the crossover will occur. If the randomly regenerated number does not fall below the crossover probability, then both individuals are placed on the Ji list. Otherwise, the parent individuals are split into dual parts, and both parts are interchanged, which creates offspring (lines 9–15).

Furthermore, during mutation, a random number is again generated performed according to (lines 16–20). As it relates to the proposed work, the following operations are carried out.

- Randomly changing the value of parameters of building blocks
- Adding a pooling layer
- Adding a skip layer
- Removing a layer from a particular position

All the configurations of this algorithm are based on the best performance of the system and are recommended in the literature. One difference in the proposed algorithm is that a higher probability is given to the addition of a skip layer because it increases the depth of architecture, which improves the performance. However, it is the opposite of a pooling layer. For other operations, equal probabilities are given.

Algorithm 6: Calculate_rank (individual, Past_rank_index) (Proposed Algorithm)

Input: The individual, individual fitness that is Past_rank_index

Output: Individual Current rank that is Curr_rank_index

```
1 curr_rank_index = past_rank_index
2 if past_rank_index == −1
      curr_rank_index = 0
```

(Continued)

Algorithm 6: (continued)

```
3 find acc, loss, r2_score, and weight update count n_weight and time n_seconds
  With respect to current individual
4     curr_rank_index /= acc # acc and r2_score gives positive correlation then
5     curr_rank_index /= r2_score
6     if loss > 1:
7             curr_rank_index *= loss
8     else:
9             curr_rank_index /= loss
10            curr_rank_index /= n_weights/n_seconds
11    return(curr_rank_index)
```

The process of the rank calculation of individuals is exhibited in Algorithm 6. This algorithm assigns ranks to all individuals based on their performance according to the following parameters: accuracy, r2_score error, weight updates, and time taken to pass through the allotted images. Accuracy, R2 score, and past rank have a positive effect on rank calculations. Other factors, such as loss, number of weights updates (n_weights), and estimated time (n_seconds), have an inverse effect on a rank calculation. In other words, the higher the value, the lower the rank of this particular CNN will be. This calculated rank helps in selecting the next generation's population, as our goal is to optimize the time taken by the convergence algorithm. So, this function rank calculation varies substantially according to the time taken if other parameters remain constant. Initially, current_rank is set to the past rank (line 1). If the CNN architecture is new, then the current rank is set to 0; otherwise, current_rank varies according to the acc, loss, r2_score, weight, update count (n_weight), and time (n_seconds). These parameters are calculated in line 3. The next step is to find curr_rank_index, which is updated with accuracy and r2 score (line 4–5). If the loss is greater than 1, curr_rank_index is multiplied by the loss (lines 6–7); otherwise, curr_rank_index is updated by n_weigh and n_times (lines 8–10). Finally, the current rank is returned (line 11).

4 Experimental Design

Numerous image classification experiments have been carried out in order to assess the performance of the proposed algorithm. Peer competitors were specifically chosen to be compared against the proposed algorithm. Specific peer competitors have been introduced in Subsection 4.1 to be compared against the proposed algorithm. Subsequently, the benchmark datasets are deployed as elaborated in Subsection 4.2. Eventually, the parameter settings of the proposed algorithms are exhibited in Subsection 4.3.

4.1 Peer Competitors

State-of-the-art algorithms were selected as peer rivals to demonstrate the efficacy and efficiency of the suggested algorithm. Specific peer rivals have been picked from among the three distinct categories. The first relates to state-of-the-art CNNs, which are manually developed, such as ResNet [7] and VGGNet [5]. Specifically, ResNet version 2 achieves excellent classification accuracy while having few parameters of any of its variations. It is worth noting that algorithms from this category have dominated large-scale visual recognition tasks in recent years [21]. The second and third sections comprise CNN architectural design methods from the "partial tuning" and "automated" categories, respectively. CNN-GA with cutout

falls into the second group, while CNN-GA falls into the third. The term "cutout" refers to a "regularization procedure" [22] employed in CNN training that may be able to revive the final performance. A noteworthy characteristic is that the proposed method primarily focuses on providing a "time-efficient automatic" technique for users who do not have significant domain expertise in adjusting CNN structures to create competent CNN structures. According to the "no free lunch" theorems [23], CNNs with architectures developed with manual tuning should have higher classification accuracies than those with "automatic" tuning, including the proposed approach. Clearly, comparing the proposed method to CNN architectural ideas from the "automatic" category is unfair. We would still want to provide detailed comparisons to CNN architectures created with domain knowledge in this experiment to demonstrate the efficiency and efficacy of the proposed method among all existing state-of-the-art CNNs.

4.2 Experimental Dataset

Here, we use the CIFAR10 and CIFAR100 benchmark datasets for the image classification task [3]. These datasets were selected for the following reasons: 1) both datasets are stringent in terms of picture sizes, classification categories, noise, and rotation in each picture, and 2) they are widely used to assess the performance of deep learning algorithms, and the curacy of the majority of comparable methods have been publicly revealed. The CIFAR10 dataset, in particular, is an image classification benchmark for detecting 10 kinds of natural things such as airplanes and birds. It is made up of 60,000 RGB pictures, each with a 32×32 pixel resolution. In addition, the training and testing sets include 50,000 photos each. There is an equal number of photographs in each category. Nonetheless, the CIFAR100 dataset is comparable to the CIFAR10 dataset, with the exception of 100 classes. In order to get a fast overview of both datasets, we randomly picked three classes from each benchmark dataset, followed by a random selection of 10 photos from each class. Fig. 7 depicts a selection of these photos. Specifically, Fig. 7a shows photos from CIFAR10, whereas Fig. 7b shows photos from CIFAR100. The class names of the photos in the same rows are shown in the left column of Fig. 7. It can be seen that the items to be categorized within these benchmark datasets typically occupy discrete portions of the whole image, and their placements differ in different photos.

Figure 7: Examples of (a) CIFAR-10 and (b) CIFAR-100 images

The classification algorithm's modifications are also difficult. The training set is divided into two portions in the experiments. The first 90 percent of the photos serve as the training set for training the participants, while the remaining photos serve as the fitness evaluation set for measuring fitness. Due to the enormous number of classes in the CIFAR100 dataset, most architecture discovery methods have not performed tests on it to demonstrate the ascendant.

4.3 Parameter Setting

The basic goals of this study, as has been described previously, are to create an autonomous architecture and uncover algorithms for researchers without CNN domain expertise. Similarly, to increase the application of the proposed technique, we designed the architecture so that adept users do not have to be experts in

evolutionary algorithms. As a result, only the parameters of the suggested algorithms have been established based on conventions. Specifically, the crossover and mutation probabilities are set to 0:9 and 0:2, respectively, as indicated in [15]. Meanwhile, with a batch size of 48, the Adam optimizer is used to train 100 epochs at a learning rate of 0.025, a momentum of 0.2, and a weight decay of 0.0003. We also utilized a dropout of 0.5 and a grad clip of 5 to avoid over fitting and gradient explosion, and for the loss function, cross-entropy is employed. During the differential architecture search for the optimal CNN, an arch learning rate of 0.0003 and an arch weight decay' of 0.001 are employed. Indeed, the majority of peer competitors follow a similar training regimen. When the proposed algorithm concludes, we select the one with the greatest fitness value and train it for 100 epochs on the original training set instead of 350 epochs [15], as we avoid over fitting by employing early stopping regularization. Finally, the classification accuracy on the testing set is summarized for comparison with peers.

In theory, any probability of adding mutation may be set by keeping it greater than the others. Furthermore, the population size and number of generations are set to 20, and the peer competitors use comparable parameters. A greater population size and a higher maximum generation number should result in improved performance, as well as the engrossment of additional computing resources. Nonetheless, such ideal settings are beyond the scope of this research because the existing settings, as used by other publications [15], easily outperform the majority of peer rivals, as shown in Tab. 1.

Table 1: Comparison of results

Name	CIFAR10	CIFAR 100	Parameters	GPU Days	Manual work
Base CNN	71.93	34.81	4,528,970	N/A	Yes
VGG19	94.71	89.10	21,240,010	N/A	Yes
VGG19 + cutout	89.79	73.38	20,024,384	N/A	Yes
ResNetV2	86.96	57.12	42,292,222	N/A	Yes
ResNetV2 + cutout	77.42	41.55	38 million	N/A	Yes
CNN-GA *(*Yanan Sun et al. arxiv:2020*)*	**95.22**	–	**2.9 M**	**35**	**Not needed**
CNN-GA *(*Yanan Sun et al. arxiv:2020*)*	–	**77.97**	**4.1 M**	**40**	**Not needed**
CNN-GA-DARTS	**94.24**	**87.20**	**6 million**	**14**	**Not needed**
CNN-GA-DARTS-Cutout	**90.19**	**85.34**	**5 million**	**13**	**Partial**

It is worth noting that the proposed algorithm's tests are carried out on a single GPU card model named RTX 2060 Super-8 GB rather than three GPU cards with the same model of NVidia GeForce GTX 1080 Ti, which is a less powerful configuration.

5 Experimental Results and Analysis

This section provides an overview of the comparison of results between the suggested algorithm and the selected rivals. In studying the best CNN architecture, the reader will be able to understand the suggested algorithm and the evolutionary paths. In this way, the algorithms will be understood better.

5.1 Overall Results

As the state-of-the-art CNN pairs have to be constructed manually, the primary task in studying related CNNs is to compare the classification accuracy and the number of parameters and "GPU days." Explanatory

words include the GPU Day unit since the algorithm has been executed one day on a single GPU. After the method concludes, the reflection of the computer resources contained inside is apparent. In turn, the name of the architecture that identifies the algorithm was used as the name of the identified CNN while comparing the accuracy of the classification and the number of parameters among the pairs.

For instance, the algorithm offered is called CNN-GA. Tab. 1 shows the comparison between the method presented and the peer competition after 20 epochs, respectively. In the table, the name of competitors is shown in the first column. In addition, the last column shows how much manual compliance the associated CNN needs during the discovery of the CNNs' structures. In the second and third columns, the CIFAR10 and CIFAR100 data sets are classified precisely, while in the fourth column, the numbers of parameters are shown in the respective CNNs. The fifth column shows the use of GPU days solely for the semi-automatic and automated methods.

CNN-GA-DARTS achieves 7.72% and 4.7% improvement in CIFAR 10 data settings over Resnet-V2 and VGG-Cutouts, having a manual category, respectively, while utilizing just 14% and 30% of their respective paramae. CNN-GA-DARTS achieves the best classification precision with both the CIFAR-10 and the CIFAR-100. In the automated category, CNN-GA-DARTS yields 94.24% accuracy, which is 1.02% less than CNN-GA over CIFAR-10% and 87.20% over CIFAR-100, which is 11.88% higher than CNN-GA. In short, CNN-GA-DARTS overrides the majority of the state-of-the-art CNN algorithms built manually and automatically in terms of classifications, the number of parameters, and the number of computer resources used. Although some semi-automatic algorithms exhibit better classification accuracy, CNN-GA-DARTS is a completely automatic algorithm and does not necessitate any human proficiency in the course of solving real-world tasks, which is the main pursuit in this paper.

5.2 Evolutionary Trajectories

Fig. 8 depicts the evolutionary trajectory of the proposed algorithm on the CIFAR10 dataset; specifically to attain better convergence of the proposed algorithm 20 generations are considered. In line with this, the initiation is, to collect the individuals selected in each generation. With the use of the dashed line, the most efficient and median classification accuracy is also being connected in the meantime. The number of generations is represented by the horizontal axis, and the classification accuracy is signified by the vertical axis as depicted in Fig. 8. The evolution progress is accelerated by the best and median classification accuracies depicted in Fig. 8. It can be observed height of each box is continuously decreasing i.e., in exploring the architectures of CNNs on the CIFAR10 dataset, the evolution is moving towards a stationary state The up-gradation of the classification accuracy transpires unceasingly from the second generation to the eighteenth generation. Henceforth, not much change is seen in the classification accuracy until the termination of the evolution. This implies that since the proposed algorithm is linked up well with this setting, then in this specific case, the setting of 20 generations is rational.

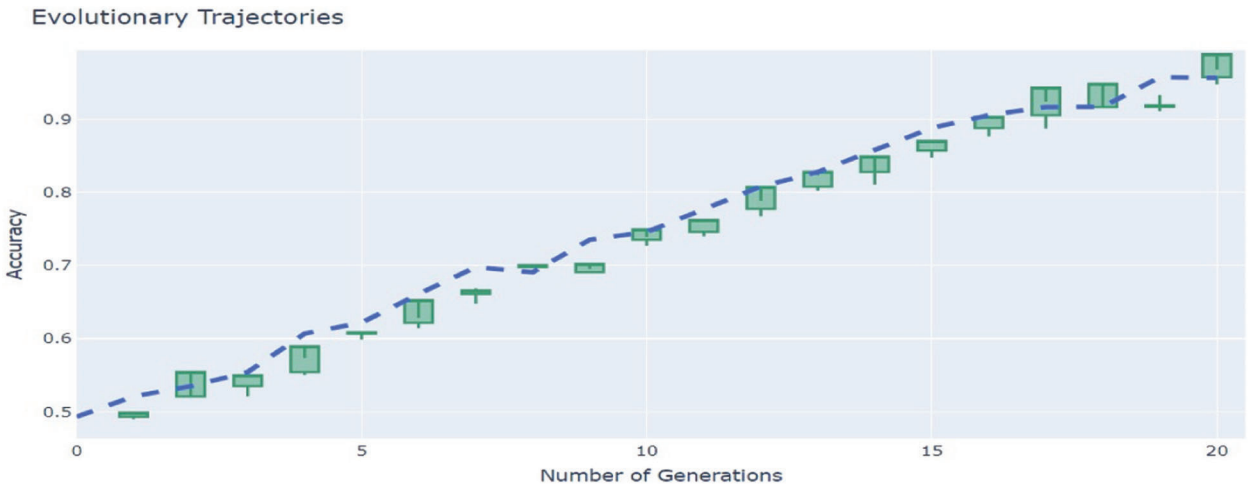

Figure 8: The evolutionary trajectory of the proposed algorithm generated during the discovery of the best CNN architecture for the CIFAR 10 dataset

6 Conclusions

The goal of this research was to develop an autonomous architecture design method for CNNs using the updated GA (abbreviated CNN-GA-DARTS), which is efficient for finding the best CNN architecture in tackling image classification challenges for users who do not have expertise in adjusting CNN architectures. This goal was achieved by providing a DART-based individual training and rank-based individual fitness assessment function, as well as by establishing a parallel and a lookup cache component, to significantly accelerate the fitness evaluation supplied in a limited computational resource. The proposed approach was tested using two daunting benchmark datasets and compared against five state-of-the-art rivals, five personally constructed CNNs, and one automated method investigating the architectures of CNNs.

According to the results, the CNN-GA-DARTS outperforms most manually developed CNNs and artificial peer rivals in terms of classification accuracy. Also, in comparison to most of its peers, the CNN found by CNN-GA-DART has few parameters. Furthermore, significantly fewer computing resources are used by CNN-GA-DART than by the majority of its automated and manually tuned peer rivals. Regardless of whether they are familiar with CNNs or GAs, users can utilize CNN-GA-DARTS to solve picture categorization problems because this method is fully automated. Furthermore, the CNN architecture built by CNN-GA-DARTS on CIFAR10 performs well. Three components have been devised to speed up the fitness evaluation while conserving many computing resources. Future research works may consider using various combinations of optimization algorithms to generate automated architectures.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

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

[2] Y. Bengio, "Learning deep architectures for AI," Now publisher. 2009. [Online]. Available: shorturl.at/drCU2.

[3] F. Fukushima and K. Neocognitron, "A hierarchical neural network capable of visual pattern recognition," *Neural Network*, vol. 1, pp. 119–130, 1988.

[4] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Image net classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 1, pp. 1097–1105, 2012.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Int. Conf. on Machine Learning*, Lille, France, 2015.

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed *et al.,* "Going deeper with convolutions," in *IEEE Conf. on Compute Vision and Pattern Recognition*, Boston, MA, USA, pp. 1–9, 2015.

[7] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 770–778, 2016.

[8] T. Back, "Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming," *In Genetic Algorithms*, England, UK: Oxford University Press, 1996. [Online]. Available: https://bit.ly/35isX28.

[9] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Int. Conf. Learning Representations*, Toulon, France, 2017.

[10] A. Krizhevsky and G. Hinton, "*Learning multiple layers of features from tiny images*," *M.Sc. dissertation*, University of Toronto, Canada, 2009.

[11] L., Hanxiao, K. Simonyan and Y. Yang, "Darts: Differentiable architecture search," arXiv preprint arXiv: 1806.09055, 2018.

[12] A. M. Hasan, H. A. Jalab, F. Meziane, H. Kahtan and A. S. Al-Ahmad, "Combining deep and handcrafted image features for MRI brain scan classification," *IEEE Access*, vol. 7, pp. 79959–79967, 2019.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, pp. 770–778, 2016.

[14] X. Lingxi and A. Yuille, "Genetic CNN," in *IEEE Int. Conf. on Computer Vision*, Venice, Italy, pp. 1379–1388, 2017.

[15] S. Yanan, B. Xue, M. Zhang, G. Yen and L. Jiancheng, "Automatically designing CNN architectures using the genetic algorithm for image classification." *IEEE Transactions on Cybernetics*, vol. 50, pp. 3840–3854, 2020.

[16] D. Vos, B. D. Berendsen, F. F. Viergever, M. A. Staring and M. Išgum, "End-to-end unsupervised deformable image registration with a convolutional neural network," *In Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*, Cham, Switzerland: Springer, pp. 204–212, 2017.

[17] M. Z. Alom, P. Sidike, M. Hasan, T. M. Taha and V. K. Asari, "Handwritten bangla character recognition using the state-of-the-art deep convolutional neural networks," *Comput. Intell. Neurosci.*, vol. 6747098, pp. 1–13, 2018.

[18] M. Z. Alom, A. A. S. Awwal, R. Lowe-Webb and T. M. Taha, "Optical beam classification using deep learning: A comparison with rule-and feature-based classification," *In Optics and Photonics for Information Processing XI*, San Diego, CA, USA: SPIE, vol. 10395, 2017.

[19] Z. Barret, V. Vasudevan and J. Shlens, "Learning transferable architectures for scalable image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City USA, pp. 8697–8710, 2018.

[20] R. Esteban, A. Aggarwal, Y. Huang and V. L. Quoc, "Regularized evolution for image classifier architecture search," in *AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4780–4789, 2019.

[21] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh *et al.,* "Image net large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[22] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," arXiv preprint arXiv:1708.04552, 2017.

[23] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.