Evolutionary Simulated Annealing for Transfer Learning Optimization in Plant-Species Identification Domain

Gusti Ahmad Fanshuri Alfarisy
School of Digital Science
Universiti Brunei Darussalam
Brunei Darussalam
20h8562@ubd.edu.bn

Owais Ahmed Malik
School of Digital Science
Universiti Brunei Darussalam
Brunei Darussalam
owais.malik@ubd.edu.bn

Ong Wee Hong
School of Digital Science
Universiti Brunei Darussalam
Brunei Darussalam
weehong.ong@ubd.edu.bn

***Abstract*—The reuse of the pre-trained deep neural network models has been found successful in improving the classification accuracy for the plant species identification task. However, most of these models have a large number of parameters, and layers and take more storage space which makes them difficult to deploy on embedded or mobile devices for real-time classification. Optimization techniques, such as Simulated Annealing (SA), can help to reduce the number of parameters and the size of these models. However, SA can easily get trapped into local optima when dealing with such complex problems. To solve this problem, we propose a new technique, namely Evolutionary Simulated Annealing (EvoSA), which optimizes the process of transfer learning for the plant-species identification task. We incorporate the genetic operators (e.g., mutation and recombination) on SA to avoid the local optima problem. The technique was tested using the MNetV3-Small as a pre-trained model due to its efficiency on mobile for two plant species data sets (MalayaKew and UBD botanical garden). As compared to the standard SA and Bayesian Optimization techniques, the EvoSA provides the least cost value with a similar number of objective evaluations. Moreover, the EvoSA produces approximately 14x and 6x less cost compared to SA for MalayaKew and UBD botanical data sets, respectively. The results show that the EvoSA can generate solutions with higher test accuracy than typical transfer learning with a competitive number of parameters.**

***Index Terms*—transfer learning, evolutionary algorithm, simulated annealing, plant-species identification**

I. Introduction

Plant biodiversity plays an important role in prolonging human life. A variety of plants contribute to our nutritional needs and medicinal materials. In the short-term effect, the exposure to biodiversity in the urban areas could mitigate pollution-related problems and improve human health both physically and mentally [1]. Unfortunately, plant species are at risk of extinction and human overexploitation has been proven to aggravate the situation [2]. Conserving biodiversity in plants becomes necessary to provide similar benefits for our descendants.

In order to achieve effective conservation, a monitoring system has become essential to maintain biodiversity, which could give a critical signal with regard to the ecological conditions. With this system, humans can make a decision to manage it. One of the important bioindicators is the species metrics that measures the number of species and their diversity in a community [3]. Therefore, species identification is the first step to achieving this objective.

Manual identification of plant species needs meticulous observations from the experts and is a much more difficult task for non-experts. Morphological and morphometric analyses are used by experts to understand the characteristics of plants and to distinguish them, which makes the identification of a single plant a laborious task. On the other hand, automating plant species identification is a promising trend. It can reduce the gap between experts and non-experts, which may trigger seamless communication between them in order to achieve the same objective, i.e., identifying plant species. Furthermore, it also stimulates cross-disciplinary research between computer science and ecology that could accelerate the common goal of biodiversity monitoring [4].

In existing computer vision approaches for the automatic identification of plant species, feature extraction plays an essential role in deriving important characteristics from plant images. The most commonly used features for plant species identification are shape, color, texture, and vein of leaves [4]. The problem with traditional deterministic machine learning is the need for feature engineering as one of the main components of the plant species identification pipeline. It requires human experts to identify and extract the important features of a specific family/species of plants. The process may need modification when applied to a different set of species or for different settings of the environment used during the image acquisition stage. Thus, using hand-crafted features may not be an ideal approach for plant species identification.

Currently, utilizing a pre-trained model for solving downstream tasks has become a gold standard for real-world problems [5]. It preserves the parameters that implicitly act as a knowledge trained on a large number of images from data sets like ImageNet [6]. It also alleviates the overfitting problem since the trained weights have knowledge of general images

resulting in better accuracy at the target task.

The knowledge in the pre-trained model is transferred to the target task by fine-tuning the hyper-parameters or merely learning the connections between the last layer and the classification layer. Many studies in plant species identification have used the same technique where the parameters/layers were frozen for extracting the features using the pre-trained models [7], [8]. However, when the parameters for the previous layers are frozen and cannot be trained, some of the important information that needs to be adjusted may be lost. Moreover, most of the literature uses a large network with a high number of parameters. In [9], the features derived from MobileNet and DenseNet121 are concatenated to predict the plant species. The complexity of the networks was increased by adding more parameters. In [10], Inception-ResNet-v2 is employed with more than 50 million parameters. Furthermore, approximately 138 million parameters are used in [11] by using VGG16.

In this research, we developed an Evolutionary Simulated Annealing (EvoSA) method for finding the optimal architecture from the feature extractor to the output layers. We improved the Simulated Annealing (SA) with a search strategy inspired by an evolutionary algorithm. Mutation and recombination were employed without sacrificing the performance of SA. Furthermore, one layer in the pre-trained model was selectively chosen that could improve the performance.

To the best of our knowledge, this is the first study in optimizing transfer learning in CNNs using the meta-heuristic approach with a selective trainable layer for plant species identification. The rest of the paper is organized as follows. Section II describes our proposed methodology, section III presents the empirical analysis and results, and section 4 concludes the paper.

II. The Proposed Methodology

In this section, we propose a novel technique named Evolutionary Simulated Annealing (EvoSA) for optimizing transfer learning using MobileNetV3-Small.

A. MobileNetV3-Small

MobileNetV3-Small is a CNN model from Google with roughly 2.5 M parameters that was designed for mobiles. It is the extension of MobileNetV2 with the Squeeze-and-Excite technique employed in the residual block [12]. The hard swish activation has been introduced for non-linearity in the network that decreases the latency cost [13].

B. Representation of Candidate Solution

The solution is represented by an 8 dimensional vector with real values denoted as $s = (s_1, \ldots, s_8)$ as shown in Figure 1. The first three elements $(s_1 - s_3)$ are the number of neurons in each layer while the next three elements $(s_4 - s_6)$ represent the activation function for each layer. For instance, the third layer has 115 neurons with the activation function at index zero which is the Elu activation. The seventh element (s_7) shows the patience level of the early stopping mechanism. The final element (s_8) shows the selective trainable layer index from the pre-trained models. This candidate solution is projected to generate a deep neural network with its corresponding hyper-parameters.

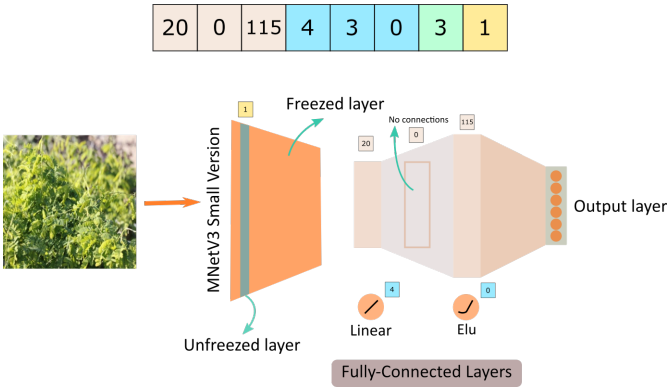

Fig. 1. Illustration of the representation

C. Cost Function

To steer the proposed algorithm into an optimal solution, the cost function is defined by accounting for the mean test accuracy of the k-fold cross-validation, the number of parameters of the networks, the average latency, and the availability of neurons on the classification part. This cost function will be minimized to compute the best solution. To represent the cost function we used the following notations: X is a set of images such that $X = \{x_1, \ldots, x_i, \ldots, x_N\}$, N is the total number of images, Y is a set of target labels associated with X such that $Y = \{y_1, \ldots, y_i, \ldots, y_N\}$, D is the set of supervised data points, where $D = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_N, y_N)\}$, p denotes the total parameters of the network, m denotes the number of times the latency is calculated using one particular image, and s is the candidate solution of the hyper-parameters that are being optimized. Then cost function can be defined as shown in (1).

Algorithm 1 Evolutionary Simulated Annealing

Input: an initial temperature T_0, a threshold mutation λ, a list of lower bound (lb) and upper bound (ub) of solution
Output: the best encoded solution found so far s^*

$s \leftarrow$ initialize random solution
$c \leftarrow evaluate(s)$ using (1)
$s^* \leftarrow s$ (*best solution so far)
$c^* \leftarrow c$ (*best cost so far)
$T \leftarrow T_0$; $time \leftarrow 0$
while stop condition is not meet **do**
  $r_1, r_2 \leftarrow$ random uniform $\in [0, 1]$
  $mutated \leftarrow False$
  if $r_1 < \lambda$ **then**
    $s^{'} \leftarrow$ mutate random position (s) using (8)
    mutated $\leftarrow$ True
  end if
  if $r_2 < \lambda$ **then**
    $s^{'} \leftarrow$ mutate flip position (s) using (9)
    mutated $\leftarrow$ True

end if
if not mutated **then**
$s^{'} \leftarrow neighbour(s, \delta)$ using (6)
$s^{'} \leftarrow$ clip $s^{'}$ using lb and ub
end if
update temperature using (7)
$c^{'} \leftarrow evaluate(s^{'})$ using (1)
if $c^{'} < c$ **then**
$s \leftarrow s^{'}$; $c \leftarrow c^{'}$
if $c^{'} < c^{*}$ **then**
$s^{*} \leftarrow s^{'}$; $c^{*} \leftarrow c^{'}$
end if
else if $e^{(c-c^{'})} > rand()$ **then**
$s \leftarrow s^{'}$; $c \leftarrow c^{'}$
end if
end while
parents, costs $\leftarrow$ initiate new parents based on the solution s^{*} using **Algorithm 2**
$s^{*} \leftarrow$ evolutionary process on the parents using **Algorithm 3**
return s^{*}

$$cost(D, p, s) = \alpha \,.\, cost_{acc}(D, s) + \beta \,.\, cost_{param}(p) + \gamma \,.\, cost_{latency}(s) + cost_{nl}(s) \quad (1)$$

The first term in the cost function shows the penalty of the accuracy controlled by the pre-defined weight α. As shown in (2), $k(D, s)$ is a k-fold cross-validation function outputting the mean test average accuracy. At first, the model was trained by Adam optimizer with a learning rate of 0.001 using the representation of s with early stopping. Then the pre-defined number of k-fold is used to get the mean test accuracy.

$$cost_{acc}(D, s) = 1 - k(D, s) \quad (2)$$

The second term in the expression describes the penalty of the parameters weighted by β. Since the MobileNetV3-Small version has roughly 2.5 million parameters for ImageNet classes (1000 output layer), it is divided by one million and subtracted by σ to balance with other terms as shown in (3).

$$cost_{param}(p) = \frac{p}{1,000,000} - \sigma \quad (3)$$

In the third term, the average latency is calculated for m times using a sample x_a weighted by γ as shown in (4). Min-max normalization is utilized for latency cost to balance the value with other terms. In the final term, the penalty is given when no layer is added before the output layer as shown in (5). This will prevent the search algorithm from staying at the typical transfer learning procedure, which may be considered as local optima and strategically could provide a potential search to find fewer parameters than the standard transfer learning approach if the total classes are large such as in the plant species domain. Further, adding more layers also contributes to the accuracy of the trained models.

$$cost_{latency}(s) = \frac{\frac{1}{m}\sum_{i=1}^{m} latency(x_a, s) - min}{max - min} \quad (4)$$

$$cost_{nl}(s) = \begin{cases} 1 & if\ no\ additional\ layer\ added \\ 0 & otherwise \end{cases} \quad (5)$$

D. Evolutionary Simulated Annealing for Transfer Learning

The proposed algorithm for optimizing the transfer learning with selective trainable layers is presented in Algorithm 1. SA is the main search algorithm that helped the evolutionary process. Two simple mutation techniques are proposed for avoiding the local optima. The first one is to randomly set the position for all elements in the hyper-parameter representation. The index in the representation is randomly selected, and a random integer value within the lower and upper bound of that index is generated to replace the current value. Another mutation flips the neurons of the first three elements (number of neurons) of the representation. This mutation is performed when two uniform random numbers $r_1, r_2 \in [0, 1]$ are less than the predefined mutation threshold. When a mutation is used to find the solution, the standard neighborhood search is not used. The acceptance of a bad solution is determined by a uniform random number $\in [0, 1]$ as shown in Algorithm 1. The initiation of the neighbor of the solution is performed by adding the current solution using some random value in the range of $[-\delta, \delta]$ as shown in (6). Each element of the representation has its own δ value that is pre-defined by the user. This gives the user control over the movement during the search process. Afterward, the neighbor solution is clipped to a pre-defined bound.

$$neighbour(s_i, \delta_i) = s_i + r \in [-\delta_i, \delta_i] \quad (6)$$

The temperature is decreased gradually using Boltzman annealing (7) [14]. The lower temperature provides a low probability that a bad solution is taken for the next solution, mimicking the annealing process. Meanwhile, the higher temperature permits SA to take a bad solution to avoid getting stuck in local optima in the early phase. The initial temperature is denoted as T_0 while t is the current iteration.

$$temp(t) = \frac{T_0}{log(1 + t)} \quad (7)$$

Algorithm 2 Crossover-based Parents Initialization

Input: current best solution s^{*} and current best cost value c^{*}
Output: list of solution s^{*} and list of cost c^{*}

$parents \leftarrow s^{*}$; $costs \leftarrow c^{*}$
for $i \leftarrow 1$ to 3 **do**
$s \leftarrow$ initialize random solution
$ch_1, ch_2 \leftarrow crossover(s^{*}, s)$
$cost_1 \leftarrow evaluate(ch_1)$ using (1)
$cost_2 \leftarrow evaluate(ch_2)$ using (1)
if $cost_1 < cost_2$ **then**
$parents \leftarrow parents \cup ch_1$

```
    costs ← costs ∪ cost_1
  else
    parents ← parents ∪ ch_2
    costs ← costs ∪ cost_2
  end if
end for
return parents, costs
```

E. Mutation Techniques

The mutation is employed in SA to replace the neighbor selection with a small probability. The first type of mutation is performed by randomly selecting index j of the candidate solution S and replacing it with a value between the lower and upper bounds as presented in (8). The second type of mutation is performed by flipping the value of neuron representation as shown in (9) where η is the maximum number of neurons in a particular layer. The value is flipped between zero and a non-positive value.

$$s_j = round(r \in [lb_j, ub_j]) \tag{8}$$

$$s_j = \begin{cases} 0 & s_j > 0 \\ round(r \in [1, \eta]) & otherwise \end{cases} \tag{9}$$

F. Crossover-based Parents Initialization

Once SA is terminated based on a stopping condition, new individuals act as the parent for the later evolutionary search and are initialized based on the best solution found so far. The details can be seen in Algorithm 2. We only initiate three new parents to make the evolution process faster since the evaluation of the objective function is expensive. Hence, the total number of individuals in a population is only four.

Algorithm 3 Recombination Process

```
Input: parents, costs
Output: best solution s*
for i ← 1 to φ do
  for j ← 1 to 2 do
    p_1, p_2 ← pick two random individuals from parents
    ch_1, ch_2 ← crossover(p_1, p_2)
    cost_1 ← evaluate(ch_1) using (1)
    cost_2 ← evaluate(ch_2) using (1)
    children ← children ∪ ch_1 ∪ ch_2
    ccost ← ccost ∪ cost_1 ∪ cost_2
  end for
  population ← population ∪ children
  pcost ← pcost ∪ ccost
  population, pcost ← elitism(population, pcost)
end for
return best solution from population (s*)
```

For each iteration, a random solution is generated. This non-optimal solution is then recombined with the best solution from SA that produces two children using the extended intermediate crossover operation [15]. The best child is selected as the new parent.

G. Evolutionary Process

The standard evolutionary process has been followed in the proposed model. Using the solution from Algorithm 2, two random parents are chosen, then an extended intermediate crossover is performed on them to produce two children [15]. The details are shown in Algorithm 3. In our case, we set ϕ to 10, which contributes to the 40 objective evaluations.

III. Empirical Analysis

A. Data sets

We experimented with two plant species data sets namely MalayaKew [16] and UBD botanical garden [17]. MalayaKew data set contains 44 species collected in Kew, England. This data set consists of only the leaves' data without any background. The suggested split of train and test sets was used. Since our approach used an early stopping mechanism, 10% of the train set was used as the validation set. The other data set (UBD botanical garden) contains data for 45 plant species found in the Borneo region. This data set is divided into training (80%), validation (10%), and the remaining 10% for testing.

B. Experimental Settings

We used 10-fold cross-validation to compute the performance of the models. The train and test subsets were combined as a single set for each data set, while the validation sets were utilized for the early stopping mechanism. For parameters in our objective function, σ was set to 1.5, which is approximately the number of the smallest possible parameters in MobileNetV3-Small in TensorFlow 2.4.1. The value of m was set to 10 and η set to 2000. The preference of each component of the main objective function, α, β, and γ was set to 0.5, 0.25, and 0.25, respectively. These parameters may depend on the problem at hand and should be adjusted based on the user's preference. For min-max normalization in the latency function, we set the minimum value to $1 \times 10^{-}5$ and the maximum value to 2. The experiments were run on two machines with GPUs (RTX 2060 for UBD botanical garden and RTX 2080 super for MalayaKew). We ensured that the experiments for each data set were run on the same machine for a fair comparison. For stopping criteria, we used 100 iterations for SA followed by 10 iterations of the evolutionary process.

EvoSA was compared with standard Simulated Annealing (SA) and Bayesian Optimization (BO) algorithms. Both SA and BO searched for the optimal solution using 150 iterations to have a roughly similar number of objective function evaluations as for EvoSA. A batch size of 32 was used with 100 epochs. Due to the stochastic nature of the algorithm, we run all algorithms for 10 trials for a fair comparison. Furthermore, we compare the performance of the best solution found using EvoSA with a typical transfer learning-based method where a pre-trained model is connected directly to the output layers. Furthermore, to provide more insight about the cost function results, EvoSA was re-run with a higher value of α (EvoSA*) as α = 0.9, β = 0.05, and γ = 0.05.

TABLE I
PERFORMANCE COMPARISON OF DIFFERENT HYPER-PARAMETER OPTIMIZATION TECHNIQUES

Data set	Metrics	EvoSA	SA	BO
MalayaKew	Cost	$\mathbf{0.058 \pm 0.004}$	0.799 ± 0.500	0.072 ± 0.012
	Top-1 Train	$99.655 \pm 99.655\%$	$98.893 \pm 0.817\%$	$99.587 \pm 0.962\%$
	Top-1 Val	$94.367 \pm 0.768\%$	$94.432 \pm 1.351\%$	$94.721 \pm 1.802\%$
	Top-1 Test	$94.148 \pm 0.798\%$	$93.691 \pm 1.139\%$	$93.935 \pm 1.960\%$
	Top-5 Train	100%	$99.996 \pm 0.006\%$	$99.998 \pm 0.007\%$
	Top-5 Val	$99.983 \pm 0.023\%$	$99.930 \pm 0.055\%$	$99.991 \pm 0.028\%$
	Top-5 Test	99.668 ± 0.105	$99.772 \pm 0.053\%$	$99.768 \pm 0.075\%$
	Parameters	1.577 ± 0.019 M	$4.533 \pm 1.992\%$ M	1.626 ± 0.029 M
	Latency	0.075 ± 0.002 s	$0.077 \pm 0.003\%$ s	0.079 ± 0.007 s
UBD Botanical	Cost	$\mathbf{0.099} \pm 0.025$	0.569 ± 0.364	$0.134 \pm \mathbf{0.016}$
	Top-1 Train	$99.865 \pm 0.202\%$	$99.954 \pm 0.133\%$	$99.919 \pm 0.150\%$
	Top-1 Val	$85.078 \pm 1.515\%$	$86.172 \pm 2.212\%$	$84.609 \pm 2.126\%$
	Top-1 Test	$88.287 \pm 1.144\%$	$88.869 \pm 2.352\%$	$87.521 \pm 1.933\%$
	Top-5 Train	$99.993 \pm 0.013\%$	$100 \pm 0\%$	$99.999 \pm 0.003\%$
	Top-5 Val	$97.227 \pm 0.558\%$	$97.063 \pm 0.615\%$	$96.977 \pm 0.556\%$
	Top-5 Test	$98.108 \pm 0.412\%$	$97.283 \pm 0.467\%$	$97.873 \pm 0.328\%$
	Parameters	1.626 ± 0.099 M	3.517 ± 1.450 M	1.748 ± 0.066 M
	Latency	0.070 ± 0.003 s	0.072 ± 0.004 s	0.076 ± 0.007 s

C. Results and Discussion

Table I presents a comparison of EvoSA with standard Simulated Annealing (SA) and Bayesian Optimization (BO) techniques based on different metrics for both data sets. It can be noticed that EvoSA provides the smallest cost for both MalayaKew and UBD botanical garden data sets. For the MalayaKew data set, the most stabilized results were generated from EvoSA as indicated by the standard deviation value of the cost metric. In contrast, for UBD botanical garden data set, BO offers more stabilized performance than the EvoSA. The other scores (e.g., train accuracy) were not compared as the cost function considers only the test accuracy, a number of parameters, latency, and a number of layers in the optimization computation. For the running time, EvoSA took roughly 4.6 and 5.3 days for 10 times running on UBD botanical and MalayaKew data sets, respectively. Meanwhile, for one run, EvoSA took roughly 11.2 hours on UBD botanical and 12.7 hours on MalayaKew.

The resulting consistency of EvoSA may depend on the data set and the inability of SA to avoid local optima. The smaller and imbalanced data sets may contribute to the difficulty of the search space for EvoSA. Moreover, if the candidate solution is trapped far from global optima, the evolutionary process through recombination may not be capable of reaching near the desired solution. Additional techniques for recombination may be needed to help EvoSA jump out from the local optima with a more stabilized solution (the solution with a reduced standard deviation score).

The best solution found by EvoSA was compared with the results of the standard transfer learning met hod without a trainable pre-trained layer as shown in Table II. EvoSA* represents the EvoSA with a higher value of α. EvoSA$^-$ is the same model as EvoSA, but it does not use a selective trainable layer in the pre-trained part found during the search process. EvoSA produces a solution that picks one of the pre-trained layers to be trained. In contrast, EvoSA$^-$ freezes all the layers and uses the same architecture and other hyper-parameters.

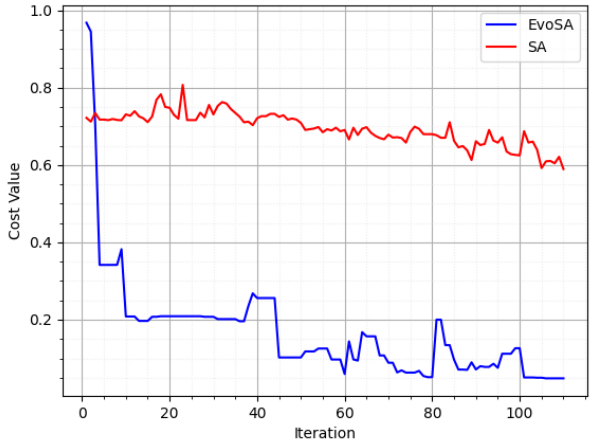

Fig. 2. Cost value over time

Increasing the value of α in the cost function (EvoSA*) can provide higher test accuracy by sacrificing small space in the parameter size. For the MalayaKew data set, it only increased the parameters by 0.1 million. As compared to EvoSA$^-$, the accuracy for both data sets experienced a consistent drop for EvoSA* model. The best solution found by EvoSA without the selective trainable layer did not provide higher accuracy than the standard transfer learning model. In contrast, the EvoSA solution provided higher accuracy with fewer parameters compared to the standard transfer learning model for both data sets. These results exhibit the potential savings on the parameters using a selective trainable layer. By unfreezing a particular layer for the pre-trained model, this layer's parameters are updated to the target task, requiring minor adjustments through the classification layer's parameters. Hence, it will require fewer neurons during the optimization phase.

TABLE II
COMPARISON TO TYPICAL TRANSFER LEARNING

Data set	Metrics	EvoSA	EvoSA*	EvoSA⁻	Standard
MalayaKew	Top-1 Train	99.549%	99.936%	97.535%	98.613%
	Top-1 Val	94.716%	95.502%	93.144%	93.625%
	Top-1 Test	95.013%	95.439%	92.229%	93.004%
	Top-5 Train	100%	100%	99.966%	99.991%
	Top-5 Val	100%	100%	99.956%	100%
	Top-5 Test	99.691%	99.923%	99.614%	99.614%
	Parameters	1.569 M	1.628 M	1.569 M	1.575 M
	Latency	0.075 s	0.075 s	0.029 s	0.027 s
UBD Botanical	Top-1 Train	99.960%	100%	99.336%	99.618%
	Top-1 Val	84.531%	88.281%	83.438%	86.016%
	Top-1 Test	88.399%	91.938%	86.416%	87.311%
	Top-5 Train	100%	100%	100%	100%
	Top-5 Val	97.5%	98.594%	96.797%	97.109%
	Top-5 Test	98.007%	98.822%	97.464%	97.914%
	Parameters	1.561 M	2.324 M	1.561 M	1.576 M
	Latency	0.072 s	0.069 s	0.069 s	0.068 s

The effectiveness of EvoSA in avoiding local optima is depicted in Figure 2. It clearly shows that SA is easily trapped in local optima and hardly jumps out from the local solution. Little improvement was observed at 100 iterations. In contrast with EvoSA, which starts with a much worse solution, the proposed mutation could help EvoSA to avoid local optima that are shown in a lower cost value during its movement. In addition, recombination in EvoSA (above the 100 iterations) could find a lower-cost value.

IV. CONCLUSION

A combination of simulated annealing and genetic operators along with a selective trainable layer in the representation was found to be helpful to find the optimal architecture for the plant species identification task. We compared the proposed approach with Simulated Annealing and Bayesian Optimization techniques and showed that EvoSA produced the least cost evaluation value with fewer parameters and reduced latency for transfer learning. In addition, we presented the importance of selectively choosing the layers for the pre-trained models during the training phase that can enhance the performance of the searched model.

REFERENCES

[1] R. Aerts, O. Honnay, and A. V. Nieuwenhuyse, "Biodiversity and human health: mechanisms and evidence of the positive health effects of diversity in nature and green spaces," *British Medical Bulletin*, vol. 127, no. 1, pp. 5–22, Jul. 2018. [Online]. Available: https://doi.org/10.1093/bmb/ldy021

[2] A. C. de Souza and J. A. Prevedello, "The importance of protected areas for overexploited plants: Evidence from a biodiversity hotspot," *Biological Conservation*, vol. 243, p. 108482, Mar. 2020. [Online]. Available: https://doi.org/10.1016/j.biocon.2020.108482

[3] J. Niemelä, "Biodiversity monitoring for decision-making," 2000.

[4] J. Wäldchen and P. Mäder, "Plant species identification using computer vision techniques: A systematic literature review," *Archives of Computational Methods in Engineering*, vol. 25, no. 2, pp. 507–543, Jan. 2017. [Online]. Available: https://doi.org/10.1007/s11831-016-9206-z

[5] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, "Pre-trained models: Past, present and future," *AI Open*, Aug. 2021. [Online]. Available: https://doi.org/10.1016/j.aiopen.2021.08.002

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009.

[7] M. Lasseck, "Image-based plant species identification with deep convolutional neural networks," in *CLEF*, 2017.

[8] M. M. Ghazi, B. Yanikoglu, and E. Aptoula, "Plant identification using deep neural networks via optimization of transfer learning parameters," *Neurocomputing*, vol. 235, pp. 228–235, Apr. 2017. [Online]. Available: https://doi.org/10.1016/j.neucom.2017.01.018

[9] A. P. S. S. Raj and S. K. Vajravelu, "DDLA: dual deep learning architecture for classification of plant species," *IET Image Processing*, vol. 13, no. 12, pp. 2176–2182, Sep. 2019. [Online]. Available: https://doi.org/10.1049/iet-ipr.2019.0346

[10] J. Krause, K. Baek, and L. Lim, "A guided multi-scale categorization of plant species in natural images," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, Jun. 2019. [Online]. Available: https://doi.org/10.1109/cvprw.2019.00320

[11] G. He, Z. Xia, Q. Zhang, H. Zhang, and J. Fan, "Plant species identification by bi-channel deep convolutional networks," *Journal of Physics: Conference Series*, vol. 1004, p. 012015, Apr. 2018. [Online]. Available: https://doi.org/10.1088/1742-6596/1004/1/012015

[12] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, Jun. 2018. [Online]. Available: https://doi.org/10.1109/cvpr.2018.00745

[13] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. [Online]. Available: https://doi.org/10.1109/iccv.2019.00140

[14] N. Siddique and H. Adeli, "Simulated annealing, its variants and engineering applications," *International Journal on Artificial Intelligence Tools*, vol. 25, no. 06, p. 1630001, Oct. 2016. [Online]. Available: https://doi.org/10.1142/s0218213016300015

[15] H. Mühlenbein and D. Schlierkamp-Voosen, "Predictive models for the breeder genetic algorithm i. continuous parameter optimization," *Evolutionary Computation*, vol. 1, no. 1, pp. 25–49, Mar. 1993. [Online]. Available: https://doi.org/10.1162/evco.1993.1.1.25

[16] S. H. Lee, C. S. Chan, P. Wilkin, and P. Remagnino, "Deep-plant: Plant identification with convolutional neural networks," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, Sep. 2015. [Online]. Available: https://doi.org/10.1109/icip.2015.7350839

[17] O. A. Malik, M. Faisal, and B. R. Hussein, "Ensemble deep learning models for fine-grained plant species identification," *2021 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)*, pp. 1–6, 2021.