An evolutionary search algorithm for efficient ResNet-based architectures: a case study on gender recognition

André Ramos Fernandes da Silva[†], Lucas Marcondes Pavelski[†], Luiz Alberto Queiroz Cordovil Júnior[†], Paulo Henrique de Oliveira Gomes[†], Layane Menezes Azevedo[†] and Francisco Erivaldo Fernandes Junior[†,*]
[†]*Retail Management System Research Group*
Sidia R&D Institute, Sidia Amazon Tower, Darcy Vargas Avenue, 654, 69055-035, Manaus, AM, Brazil
Email: {andre.fernandes, lucas.pavelski, luiz.cordovil, paulo.henrique, layane.azevedo, francisco.junior}@sidia.com
([*]Corresponding Author. All authors had equal contribution)

Abstract—**Neural Architecture Search (NAS) is a busy research field growing exponentially in recent years. State-of-the-art deep neural networks usually require a specialist to fine-tune the model to solve a specific problem. NAS research aims to design neural network architectures automatically, thus easing the need for machine learning specialists to spend a lot of effort on hand-crafted attempts. As artificial intelligence applications are becoming ubiquitous, there is also a growing interest in efficient applications that could be deployed to smartphones, smart wearable devices, and other edge devices. Gender recognition in unfiltered images — such as those we find in real-world situations like pictures taken with smartphones and video shots from surveillance cameras — is one of such challenging applications. In this work, we developed an evolutionary NAS algorithm that consistently finds efficient ResNet-based architectures, named RENNAS, which have a good trade-off between classification accuracy and architectural and computational complexities. We demonstrate our algorithm's performance on Adience dataset of unfiltered images for gender recognition.**

Index Terms—**Evolutionary Neural Architecture Search, Residual Neural Networks, Genetic Algorithms, Computer Vision, Gender Recognition**

I. Introduction

Artificial intelligence applications on smartphones and edge devices are becoming popular. One of these applications is gender recognition [1], [2]. Gender classification or gender recognition tasks aim to identify the gender of a person based on their characteristics, which is generally done by differentiating between two genders: male and female. There are plenty of motivations for gender recognition. For example, demographic factors like gender and age can be used by retailers to recommend products through customized advertising [3]. Besides that, it is also applicable on security: this type of information can be used to identify criminals and terrorists through surveillance cameras [4].

Furthermore, one can notice the need to develop assessment measures that take into account the limited computational resources of these devices. On the other hand, deep neural networks (DNNs) became the state-of-the-art for many machine learning tasks [5]. Particularly, convolutional neural networks (CNNs) perform well in computer vision applications [6]. As the models become deeper and more complex, embedding them on edge devices is a challenging task.

State-of-the-art network models show that networks with a high number of layers (depth) are able to achieve good performance on several tasks [7]. However, by increasing the number of nonlinear transformations, the accuracy gets saturated, making it harder to train. Residual Neural Networks (ResNet) add skip connections between layers to cope with the training accuracy degradation issue, allowing deeper models to get better performance and lower complexities than previous models [8].

However, the design process of DNN architectures demands a lot of trial-and-error work that consume large computational resources, and it still can be sensible to a specific task and the data distribution [9]. Neural Architecture Search (NAS) emerged to ease the design of neural networks in a completely automatic way [10]. In order to solve this optimization problem, evolutionary algorithms like Genetic Algorithms (GA) are widely used [11], [12]. GA allows us to find the best network in a search space for a specific problem, but if the search space is too large it could take a long time to converge. Aiming to reduce the search space, the use of blocks has been promising [13], [14]. Thus, the search space is limited by the blocks representation which allows faster results.

For applications on edge devices, accuracy should not be the only performance indicator, since the network size and prediction time are also a concern. Therefore, we should consider them while evaluating CNN architectures in the GA fitness function. The NetScore was designed to assess the balance between classification accuracy, network architecture complexity and computational complexity [15]. In this work, we show that if we use the NetScore with the default coefficients, the Evolutionary Neural Architecture Search (ENAS) will prioritize tiny networks at the expense of classification accuracy. However, it is possible to reformulate the function in a way that it penalizes the fitness value if the accuracy decreases below a certain threshold. Our results show how this new NetScore configuration improves the search towards more efficient networks that preserve a competitive classification accuracy. Moreover, we validate the proposed NetScore configuration on the well-known CIFAR-10 dataset [16].

Here, we test the proposed ENAS for gender recognition task. The search was performed using the Adience dataset [17], which consists of unfiltered images (captured from smartphone cameras) for gender recognition and age estimation. One of the most challenging computer vision tasks consists on learning from unfiltered images that can be seen in real-world applications, e.g., photographs taken from cellphones, video shots from surveillance cameras and others. These images may have great variation in lighting conditions, posture, occlusion, angle, resolution etc.

Our main contributions are: a reformulation of the NetScore function that allows finding simpler ResNets keeping competitive accuracy; how to select a sound configuration of NetScore's accuracy and network complexities coefficients for a specific application according to the user's preferences; a set of custom mutation operators for searching ResNet-based architectures with ENAS. To the best of our knowledge, no proposal has been made for automatic neural architecture search to address the problem of gender recognition in unfiltered images.

The remainder of this paper is organized as follows. Section II reviews NAS methods. Section III details our proposed algorithm. Section IV introduces the experiment settings. Section V presents the results. Finally, Section VI concludes with a summary regarding achievements, contributions, limitations and an outline of future works.

II. Related Works

There is a growing interest in ENAS. The first works in this area date back to 1980s, e.g. adapting weights using evolutionary algorithms. These former strategies are surveyed in [18]. Recently, the number of publications in the area doubles every year since 2017, with later studies using modern DNN architectures for image and text applications. We briefly review some works using NAS. For in-depth surveys see [9], [10].

As the NAS development progresses, block-based techniques are being used to define the search space, combining various types of layers as the basic unit [19]. These block-based techniques present good performance while finding suitable architectures with fewer parameters.

Zhong et al. [20] proposed BlockQNN, the first block-wise NAS. Defining basic block structures — such as inception-block or residue-block — and replicating them along the network layers was a successful design strategy for many hand-crafted architectures. BlockQNN follow this strategy and generates automatically-designed blocks using the well-known Reinforcement Learning (RL) technique Q-Learning. It uses a reward function that seeks a trade-off by rewarding accuracy and penalizing the number of FLOPs (floating points operations) and network density. However, they do not show the implicit trade-off represented by the FLOPs and network density penalty coefficients.

Xue et al. [19] proposed a GA with adaptive mutation to search mixed architectures containing ResNet, DenseNet and max pooling blocks. The authors introduce a variable-length strategy for individuals encoding and block-based genetic operators. The search is guided considering the accuracy on the validation set as fitness function. The optimization is devoted to the type of blocks and do not embraces internal parameters such as number of filters, for instance.

Loni et al. [21] proposed DeepMaker for automatic design of DNNs for embedded devices. Combined with a pruning strategy, the network search is formulated as a multi-objective optimization problem that considers accuracy and network size. The parameters' optimization is performed by Non-Dominated Sorting Genetic Algorithm (NSGA-II). However, the search space for each parameter is pre-defined which limits the candidate DNNs and the number of filters is not covered in the search. Loni et al. [22] also proposed the FastStereoNet for searching CNNs efficiently in a non-block-based search space for resource-limited devices. It employs a late acceptance hill climbing (LAHC), followed by simulated annealing (SA). A transferred weights mechanism improves training and evaluation of CNNs. However, it limits the number of filters and network depth.

Zoph and Le [23] proposed a NAS to search networks that can have skip connections between layers as in ResNet. They trained a Recurrent Neural Network (RNN) with RL to automatically design CNNs. Accuracy is used as reward signal in the design of CNNs which includes the definition of number of filters, kernel size, stride and activation function type. It achieves 3.65% error rate on CIFAR-10, but with CNN that are too large considering edge devices (up to 37M parameters). Pham et al. [24] follow a similar search strategy and proposed the Efficient Neural Architecture Search. They use parameter sharing for candidate models obtained during the search to significantly reduce the computational cost of the NAS. However, the CNN parameters are pre-set. Accuracy is also used as reward function, which results in large CNNs.

Javaheripi et al. [25] proposed GeneCAI, a generic solution based on evolutionary algorithms to compress pre-trained DNNs (ResNet-50, AlexNet and VGG-16) while maximally preserving model accuracy using a multi-objective score based on accuracy and number of FLOPs. GeneCAI exploits parallelism to reduce optimization time and an auxiliary application programming interface (API) was developed to allow multi-GPU execution. Nevertheless, GeneCAI does not design an optimized network from scratch, since it needs a pre-trained DNN as a starting point.

Regarding gender recognition tasks, one can find different approaches of CNNs in literature. In this paper, we compare our proposal with architectures applied to gender recognition in Adience Benchmark [17]. For example, some authors propose the use of GoogleNet and CaffeNet [26], ResNet-50 [27], CNNs with saliency maps [28], deepwise separable CNNs [29], and other conventional CNNs [30]–[32]. They achieve 80.8% to 96.2% accuracy on this gender recognition benchmark. However, these architectures were manually de-

Algorithm 1: RENNAS – ResNet-based Evolutionary Neural Architecture Search

Input	:	$\mathcal{T}$ (training dataset), $\mathcal{V}$ (validation dataset)
Parameters	:	α (accuracy coefficient), β (parameters coefficient), γ (MACs coefficient), C (population size), G (number of generations), p_m (mutation probability), p_c (probability of changing base values), δ (noise range) ϵ (probability of noise falling out of the range), η_{NAS} (NAS training epochs), η_{ref} (refinement training epochs)
Output	:	$\mathcal{X}_f$ (fittest individual), $a(\mathcal{X}_f)$ (validation accuracy), $p(\mathcal{X}_f)$ (number of parameters), $m(\mathcal{X}_f)$ (MACs), $\Omega(\mathcal{X}_f)$ (NetScore)

```
1  $\mathcal{P}_0 \leftarrow$ GENERATE_INITIAL_POPULATION($C$);                    // as detailed in Section III-B
2  TRAIN($\mathcal{P}_0,\ \mathcal{T},\ \eta_{\text{NAS}}$);
3  EVALUATE($\mathcal{P}_0,\ \mathcal{T},\ \alpha,\ \beta,\ \gamma$);                 // as detailed in Section III-C
4  for $t \leftarrow 1$ to $G$ do
5      $\mathcal{Q}_t \leftarrow$ GENERATE_OFFSPRING($\mathcal{P}_{t-1},\ p_m,\ p_c,\ \delta,\ \epsilon$);   // as detailed in Section III-D
6      TRAIN($\mathcal{Q}_t,\ \mathcal{T},\ \eta_{\text{NAS}}$);
7      EVALUATE($\mathcal{Q}_t,\ \mathcal{T},\ \alpha,\ \beta,\ \gamma$);
8      $\mathcal{U}_t \leftarrow \mathcal{P}_{t-1} \cup \mathcal{Q}_t$;
9      $\mathcal{P}_t \leftarrow$ ENVIRONMENTAL_SELECTION($\mathcal{U}_k,\ C$);          // as detailed in Section III-E
10 end
11 $\mathcal{X}_f \leftarrow$ fittest individual from $\mathcal{P}_G$;         // fittest individual at the last generation
12 TRAIN($\mathcal{X}_f, \mathcal{T},\ \eta_{\text{ref}}$);                              // as detailed in Section III-F
13 EVALUATE($\mathcal{X}_f,\ \mathcal{V},\ \alpha,\ \beta,\ \gamma$);
14 return $\mathcal{X}_f,\ a(\mathcal{X}_f),\ p(\mathcal{X}_f),\ m(\mathcal{X}_f),\ \Omega(\mathcal{X}_f)$
```

signed.

ResNets are CNNs composed by linear stacking of residual blocks [8]. A residual block is a sequence of convolutions connected by a shortcut connection. While the convolution sequence performs a series of non-linear operations, the shortcut connection adds the input signal to the output of the block's last convolution. DenseNet [33] also employs shortcut connections. In order to maximize information flow, they connect all layers with shortcut connections so that each layer receives the outputs of all previous ones. ResNeXt [34] generalizes the ResNet blocks by adding parallel convolutional branches with the same architecture. Due to the block structure, the block's cardinality (number of parallel branches) is a hyperparameter that can be easily optimized for different learning tasks.

In this work, we propose to use the NetScore as fitness function and evaluate its effects on a NAS algorithm.

III. RENNAS: ResNet-based Evolutionary Neural Architecture Search

Our proposed evolutionary neural architecture search is based on Genetic Algorithm (GA), which explores the search space seeking a neural network with the best trade-off between classification accuracy and architecture complexity, i.e., number of parameters and Multiply-Accumulate Operations (MACs) as defined in Algorithm 1.

ResNets [8] performs well in several computer vision tasks. It is known that some tasks may benefit from deeper architectures [35], [36], different numbers of building blocks and number of filters [37]. In this way, ResNet's search space is still underexplored in NAS literature. In [38] one can see recent initiatives on designing ResNet-based design spaces to parameterize populations of architectures. In this context, we designed a search space with variable network depth, based on ResNet building blocks, and number of filters. The proposed algorithm is a population based that automatically evolves ResNet-based architectures. Individuals are encoded to represent a sequence of ResNet blocks with different numbers of filters.

First, an initial population containing random variable-length individuals is set. Then, the usual steps of GA are performed: evaluation of each individual according to a fitness function, offspring generation by crossover and mutation, and selection of promising architectures that will survive and pass to the next generation. These procedures are executed for a given number of iterations until reaching the pre-set maximum number of generations. Finally, the last generation's fittest individual is trained to its full capacity. In the end of this process, one expect to find a ResNet-based network with a good balance between classification accuracy and architecture complexity. In the following sections, we give details about each of these modules.

A. Individual's Encoding Scheme

Let $\mathcal{P}$ be a population composed by C individuals and let $\mathcal{X}_i$ be the i-th individual belonging to $\mathcal{P}$. We define that

$$\mathcal{X}_i = [\chi_{i,1},\ \chi_{i,2}, \dots,\ \chi_{i,j}, \dots,\ \chi_{i,N}] \tag{1}$$

where $|\mathcal{X}_i| = N$ is the number of genes and $\chi_{i,j} = (f_{i,j},\ b_{i,j})$ is the j-th gene representing a list of $b_{i,j}$ ResNet building blocks with $f_{i,j}$ filters per layer.

The individual defined in (1) is encoded to represent a ResNet-based convolutional neural network as one can see in Figure 1. The first convolutional layer has $f_{i,1}$ filters, then it is followed by a sequence of ResNet blocks defined by the genes. The network has two types of ResNet building blocks: ordinary RB and RB*, as shown in Figure 2. We use the ordinary ResNet Block (ordinary RB) when we want the block's output tensor size to be equal to the input tensor size, i.e., when the input and output of each convolutional layer have equal dimensions. The ordinary RB is shown in Figure 2a. However, when we need to reduce a block's output tensor's width and height and change the number of filters in the subsequent block, dimension compatibility is necessary to keep the shortcuts. The building block RB* is used to perform such dimension compatibility through a downsampling operation composed of a convolutional layer with a stride equal to 2, as shown in Figure 2b. We connect the ResNet blocks in the following way: for the first gene we connect $b_{i,1}$ ordinary RB blocks, each with $f_{i,1}$ filters; for the following genes we start with one RB* followed by $b_{i,j}-1$ ordinary RB blocks. All filters have size equals to 3×3. The last convolutional layer is connected to a Global Average Pooling (GAP) layer [39]. Finally, the GAP is connected to a fully-connected layer whose output will predict the image's class.

Since, for $2 \leq j \leq N$, adjacent genes $\chi_{i,j-1}$ and $\chi_{i,j}$ are connected with downsampling RB* blocks, we impose a limit on individual's size in order to avoid creating unnecessary layers. We define that the minimum number of genes is $\texttt{min}_{|\mathcal{X}|} = 2$ and the maximum number of genes is $\texttt{max}_{|\mathcal{X}|} = \log_2\left(\min(\texttt{input}_{\texttt{width}}, \texttt{input}_{\texttt{height}})\right) - 1$, where $\texttt{input}_{\texttt{width}}$ and $\texttt{input}_{\texttt{height}}$ are the input image's width and height, respectively.

B. Population Initialization

The initial population $\mathcal{P}_0$ is composed by C individuals, which are generated considering a variable-length strategy for encoding. For $i = 1, \ldots, C$, individual $\mathcal{X}_i$ will have N_i genes such that it is uniformly chosen from $N_i \sim U(\texttt{min}_{|\mathcal{X}|}, \texttt{max}_{|\mathcal{X}|})$.

As for the genes' base values, let $\text{Pois}(\lambda)$ be a Poisson random variable such that $\text{E}\left[\text{Pois}(\lambda)\right] = \lambda$. A gene's base values are initialized as:

$$\begin{aligned} f_{i,j} &= 1 + \text{Pois}(\lambda = 2^{i+3} - 1) \\ b_{i,j} &= 1 + \text{Pois}(\lambda = 2) \end{aligned}$$

for $j = 1, \ldots, N_i$. Then, $\text{E}[f_{i,j}] = 2^{i+3}$ and $\text{E}[b_{i,j}] = 3$. In this way, the initial population will have distribution of parameters similar to typical values for ResNet blocks as one can see in the literature [8].

C. Evaluation of Individuals

Let $\mathcal{N}$ be a neural network with accuracy $a(\mathcal{N})$, number of parameters $p(\mathcal{N})$ and multiply-accumulate (MAC) operations $m(\mathcal{N})$. These indicators compose the NetScore [15] as follows

$$\Omega(\mathcal{N}) = 20 \log\left(\frac{a(\mathcal{N})^\alpha}{p(\mathcal{N})^\beta m(\mathcal{N})^\gamma}\right) \qquad (2)$$

where $\alpha,\ \beta,$ and γ are the coefficients of accuracy, number of parameters and MACs, respectively. $p(\mathcal{N})$ is scaled in millions of parameters and $m(\mathcal{N})$ in billions of MACs.

The NetScore function can be very useful to guide the evolutionary search towards efficient neural architectures. From equation (2), one can notice that number of parameters and MACs are inversely proportional to NetScore insofar as accuracy is directly proportional. Using NetScore as a fitness function, one can suspect that the evolutionary search will favor neural networks with as few parameters as possible at the expense of accuracy, since

$$\lim_{p(\mathcal{N})\to 0} \Omega(\mathcal{N}) = \infty$$

The same is true when $m(\mathcal{N}) \to 0$. Consequently, the simpler the network, the higher the NetScore. Nonetheless, this drawback can be avoided in practice if we set values for α, β, and γ to impose a strict trade-off relation between network accuracy and network complexities. We performed experiments with different values of α, β, and γ (sec. IV-B) and obtained insights about their effects on NAS (sec. V-A).

Consider two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ such that

$$\begin{aligned} a(\mathcal{N}_2) &= (1 - \Delta_a) a(\mathcal{N}_1) \\ p(\mathcal{N}_2) &= (1 - \Delta_p) p(\mathcal{N}_1) \\ m(\mathcal{N}_2) &= (1 - \Delta_m) m(\mathcal{N}_1) \end{aligned}$$

where $\Delta_a,\ \Delta_p,\ \Delta_m \in\ ]0, 1[$ denote the attenuation rates. We could allow the search algorithm to reduce the number of parameters by Δ_p and MACs by Δ_m but only if it resulted in a small Δ_a loss of classification accuracy. If we believe these trade-offs are acceptable, then we can say that $\mathcal{N}_1$ and $\mathcal{N}_2$ are equivalent in terms of NetScore. Making $\Omega(\mathcal{N}_1) = \Omega(\mathcal{N}_2)$, it follows that the relation between the attenuation rates $\Delta_a,\ \Delta_p,\ \Delta_m$ and the NetScore's coefficients $\alpha,\ \beta,$ and γ is:

$$\alpha \log\left(1 - \Delta_a\right) - \beta \log\left(1 - \Delta_p\right) - \gamma \log\left(1 - \Delta_m\right) = 0 \quad (3)$$

The practical meaning of (3) is that it becomes unlikely that the ENAS will drastically drop the number of parameters (or MACs) at the expense of classification accuracy from one generation to another. Also, this trade-off relation allows the ENAS the opportunity to consider networks with lower complexities.

It goes without saying that computing $a(\mathcal{N})$ is a demanding task, because we need to train the network. Notwithstanding, during the architecture search phase we are more interested in selecting promising architectures. In order to save search time, we perform training for η_{NAS} epochs just enough to obtain a reasonable $a(\mathcal{N})$ estimate.

D. Offspring Generation

Let $\mathcal{P}_t$ be the population in the t-th GA generation. For each GA iteration, a set of offspring $\mathcal{Q}_t$ will be generated from $\mathcal{P}_t$ such that $|\mathcal{P}_t| = |\mathcal{Q}_t| = C$. The offspring are generated using the following operators.

a) Mate Selection: This operator selects two individuals from $\mathcal{P}$ that will mate and reproduce. The binary competition strategy is employed for mate selection [40]. By randomly selecting two individuals in the population, the one with a highest NetScore wins the mate competition and is selected to be the first parent. In the sequel, the same procedure is repeated for the second parent. It is noteworthy that an individual can be selected more than once.

b) Crossover: For a given crossover point randomly chosen from a uniform distribution, such that $c_p \sim U(1, \min(|\mathcal{X}_1|, |\mathcal{X}_2|) - 1)$, the crossover is carried out as follows. Let $\mathcal{X}_1$ and $\mathcal{X}_2$ be two parents and $\mathcal{X}_3$ and $\mathcal{X}_4$ be

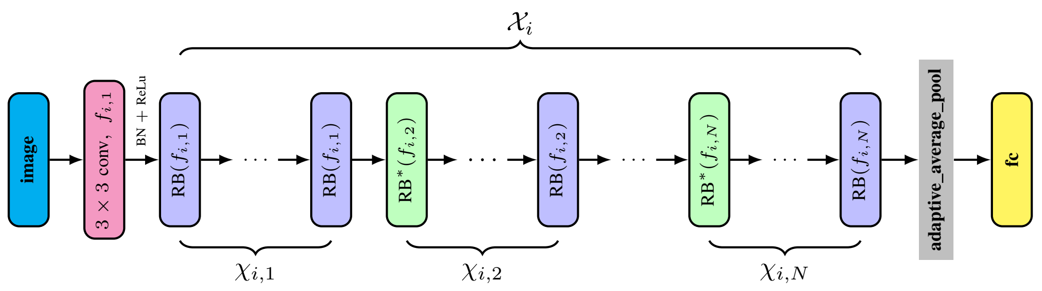

Fig. 1. ResNet-based architecture. '**fc**' denotes the fully-connected layer. '**adaptive_average_pool**' denotes the global average pooling layer.

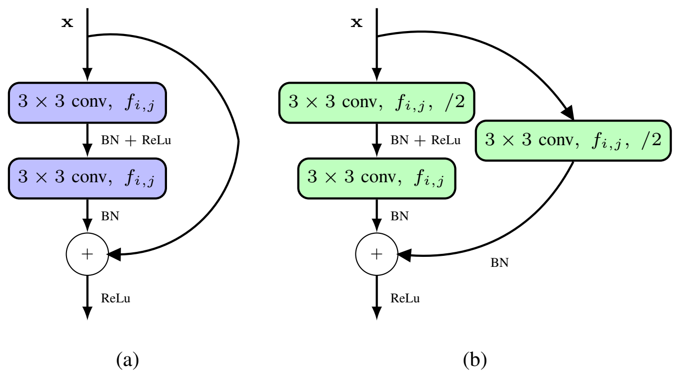
x
3×3 conv, $f_{i,j}$
BN + ReLu
3×3 conv, $f_{i,j}$
BN
ReLu
(a)
x
3×3 conv, $f_{i,j}$, /2
3×3 conv, $f_{i,j}$
3×3 conv, $f_{i,j}$, /2
BN
(b)

Fig. 2. ResNet building blocks. (a) ordinary RB. (b) RB*: an RB with downsampling (stride 2). The notation '$/2$' represents downsampling with stride 2. 'BN' represents batch normalization which is performed right after each convolutional layer and before ReLu activation.

two offspring to be generated [41]. It is noteworthy that this operator accepts parents with different number of genes.

$$\mathcal{X}_3 = \left[\chi_{1,1},\dots,\chi_{1,c_p},\chi_{2,c_p+1},\dots,\chi_{2,|\mathcal{X}_2|}\right]$$
$$\mathcal{X}_4 = \left[\chi_{2,1},\dots,\chi_{2,c_p},\chi_{1,c_p+1},\dots,\chi_{1,|\mathcal{X}_1|}\right]$$

c) Mutation: The mutation operator is applied on offspring with p_m probability, which is defined previously. If an individual is chosen for mutation, there will be 3 possible outcomes: adding a new gene, removing a gene, or changing a gene's base values (adding/removing filters and ResNet building blocks):

(i) *Adding a new gene*: Let $m_a \sim U(1, |\mathcal{X}_i|+1)$ be an insertion position randomly chosen from a uniform distribution. A new gene (f,b) will be inserted in $\mathcal{X}_i$ at position m_a. Its number of filters f and number of ResNet building blocks b will be randomly set according to the following distribution. Let Pois(λ) be a Poisson random variable such that $\mathrm{E}\left[\mathrm{Pois}(\lambda)\right]=\lambda$. Then, we make $b = 1+\mathrm{Pois}(\lambda=2)$ and, the number of filters

$$f = \begin{cases} 1+\mathrm{Pois}(\lambda = \lceil 0.5 f_{i,1}\rceil - 1), & \text{if } m_a = 1 \\ 1+\mathrm{Pois}(\lambda = 2 f_{i,N} - 1), & \text{if } m_a = |\mathcal{X}_i|+1 \\ 1+\mathrm{Pois}(\lambda = \sqrt{f_{i,m_a} f_{i,m_a-1}} - 1), & \text{otherwise.}\end{cases}$$

Making it so will result in $E[b]=3$ and $E[f]$ having a compatible order of magnitude depending on its adjacent genes.

(ii) *Removing a gene*: Let $m_r \sim U(1, |\mathcal{X}_i|)$ a random gene index. The m_r-th gene will be removed from the chromosome.

(iii) *Changing genes' base values*: In this mutation, each gene is updated with a probability p_c. Suppose that a gene $(f_{i,j}, b_{i,j})$ is chosen for mutation. Then, a noisy vector (α_f, α_b) is added to the gene's base values

$$f_{i,j} := \max(1, f_{i,j}+\alpha_f)$$
$$b_{i,j} := \max(1, b_{i,j}+\alpha_b)$$

The noisy terms have a Gaussian distribution $\alpha_f \sim N\left(0,\sigma_f^2\right)$ and $\alpha_b \sim N\left(0,\sigma_b^2\right)$, such that

$$\sigma_f = \frac{0.5\delta f_{i,j}}{\sqrt{2}\mathrm{erf}^{-1}(1-2\epsilon)}, \quad \sigma_b = \frac{0.5\delta b_{i,j}}{\sqrt{2}\mathrm{erf}^{-1}(1-2\epsilon)}$$

where $\mathrm{erf}^{-1}(\cdot)$ is the inverse error function. δ and ϵ were chosen such that

$$\mathrm{P}\left(-0.5\delta f_{i,j} \le \alpha_f \le 0.5\delta f_{i,j}\right) = 1-\epsilon$$
$$\mathrm{P}\left(-0.5\delta b_{i,j} \le \alpha_b \le 0.5\delta b_{i,j}\right) = 1-\epsilon$$

This means that δ defines a range of noise values proportional to the gene's base values and ϵ defines the probability of α_f or α_b falling outside this range.

Each mutation strategy has the same probability to occur. If an individual is chosen to be mutated, it must be mutated one way or another, considering the following special cases:

(i) A mutated individual's length cannot be greater than $\mathtt{max}_{|\mathcal{X}|}$. If $|\mathcal{X}_i| = \mathtt{max}_{|\mathcal{X}|}$, then mutation by removing a gene or changing genes' base values is applied.

(ii) A mutated individual's length must not be reduced below $\mathtt{min}_{|\mathcal{X}|}$. If $|\mathcal{X}_i| = \mathtt{min}_{|\mathcal{X}|}$, then mutation by adding a new gene or changing a genes' base values is applied.

E. Environmental Selection

Environmental selection chooses the individuals that will survive until the next generation. We use a semi-complete binary competition strategy to avoid the elimination of the excellent individuals [19]. Let $\mathcal{P}_t$ and $\mathcal{Q}_t$ be the parent and offspring population at the $t-$th generation, respectively. The next population $\mathcal{P}_{t+1}$ will be composed according to the following steps:

(i) Put $\mathcal{P}_t$ and $\mathcal{Q}_t$ in the same set $\mathcal{U}_t = \mathcal{P}_t \cup \mathcal{Q}_t$.
(ii) Generate a random number k such that $1 \leq k \leq C/2$.
(iii) Let $\mathcal{F}_k \subset \mathcal{U}_t$ be the k individuals with the highest fitness values from $\mathcal{U}_t$. Put $\mathcal{F}_k$ in $\mathcal{P}_{t+1}$.
(iv) Select $C - k$ individuals from $\mathcal{U}_t - \mathcal{F}_k = \mathcal{S}_k$ using binary competition and put then in $\mathcal{P}_{t+1}$. The binary competition picks two random individuals and select the fittest.

F. Final Model Refinement

After G generations, we expect that the fittest individual $\mathcal{X}_f$ has a good architecture automatically tuned for the problem represented by the dataset. Then, it is chosen as the final model and it is fully trained with more η_{ref} epochs than in the search phase.

IV. Methodology

This section details the experiment settings to evaluate RENNAS. We introduce the benchmarks used for image classification, the parameters assignments for NetScore trade-off and the assessment strategies for each benchmark. All experiments were run on NVIDIA Tesla V100 GPUs with 16GB of memory.

A. Benchmarks Description

In this work we test the RENNAS in two benchmark datasets: CIFAR-10 [16] and Adience Benchmark [17].

Several NAS proposals consider CIFAR-10 as a benchmark [9]. This dataset contains 32x32 pixels natural images of animals, plants and objects [1]. The CIFAR-10 dataset contains $60,000$ images labeled into 10 classes. Our NAS uses $50,000$ images for training and the remaining $10,000$ are reserved as test set for the final comparison. To improve the network generalization, we use the following data augmentation steps during training: adding 4 white padding pixels, flipping horizontally with probability 0.5, random cropping back to size 32x32 pixels.

Another database we use in the experiments concerns a gender classification task and it is named Adience Benchmark [17]. This database consists of faces from Flickr albums. It has several challenging real-world conditions like different angles, lighting, pose, and others. There are a total of $19,487$ photos from $2,284$ subjects classified into male and female [2]. The experiments follow the 5-fold cross-validation procedure described in [17]. Regarding images settings, we used the frontalized images resized to 256×256 pixels. Each image channel (Red, Green and Blue) was normalized with mean $[0.30767522,\ 0.34428763,\ 0.4452923]$ and standard deviation $[0.19725059,\ 0.21831386,\ 0.25025621]$. During training, we employ the following data augmentations: random horizontal and vertical flips and a random crop of 224×224 pixels. The test folds are center cropped with 224×224 pixels.

[1]The full set of images and labels are available in the python torchvision library at https://pytorch.org/vision/stable/datasets.html

[2]Adience gender and age dataset is available for download at https://talhassner.github.io/home/projects/Adience/Adience-data.html

B. Parameters Assignments

Regarding the GA's parameters, we set the population size to $C = 20$ and the number of generations to $G = 20$ (as suggested in [19]). The mutation probability is set to $p_m = 0.25$ so that $1/4$ of the generated offspring have genetic diversity in relation to their parents. The probability of changing genes' base values is set to $p_c = 0.25$ so that the expected number of changed genes is at least 1 when this mutation operator is selected to occur. The parameters ϵ and δ are set to 0.15 and 0.2, respectively so that 85% of the noise added to the base values are within a $\pm 10\%$ range of the original values. As detailed in Section III-D, each mutation (adding a gene, removing a gene, changing genes' base value) has the same probability to occur. We use stochastic gradient descent for network weights optimization. For both experiments on CIFAR-10 and Adience, each architecture is trained for $\eta_{\text{NAS}} = 10$ epochs during ENAS phase. In the sequence, we provided more details.

1) CIFAR-10 dataset: We use this dataset to validate different NetScore configurations. For the trade-off equation in (3), one can assign the attenuation rates ($\Delta_a,\ \Delta_p,\ \Delta_m$) and coefficients ($\alpha,\ \beta,\ \gamma$) according to the user preferences. The experiments have been conducted considering four configurations:

Default coefficients. This configuration uses the default NetScore coefficients suggested by [15], i.e., $\alpha = 2$, $\beta = 0.5$, and $\gamma = 0.5$.

Lax trade-off. This configuration accepts a trade-off of 5% of accuracy if it results in half the network complexities, that is, the accuracy attenuation rate is set to $\Delta_a = 5\%$ and the number of parameters and MACs attenuation rates are set to $\Delta_p = \Delta_m = 50\%$. We use (3) to translate these deltas into coefficients α, β, and γ. First, we set $\alpha = 4$ and make $\gamma = \beta$. Regarding these assignments, it follows from (3) that $\beta = \gamma = -2\log_2(0.95) \approx 0.15$.

Strict trade-off. This configuration accepts to reduce the network complexities by half but only if it results in a 1% loss of accuracy, that is, $\Delta_a = 1\%$ and $\Delta_p = \Delta_m = 50\%$. Using the same reasoning, we set $\alpha = 4$ and $\gamma = \beta$. It follows from (3) that $\beta = \gamma = -2\log_2(0.99) \approx 0.03$.

Only accuracy. This configuration uses accuracy as a fitness function and totally ignores the network complexities during evolution.

After the search phase, the fittest architecture was trained for $\eta_{\text{ref}} = 80$ epochs. The learning rate is set to 0.025 for the batch size of 512 images, and it is decayed by 1/3 every 30 epochs.

2) Adience dataset: In the search phase, we used the strict trade-off parameters configuration for NetScore to evaluate the candidate architectures. After that, each architecture obtained for each fold is trained for $\eta_{\text{ref}} = 140$ epochs. We use L2 normalization (weight decay 10^{-5} for all parameters except batch normalization layers), Mixup [42] and Cutmix [43] (both using parameter $\alpha = 0.8$) to reduce overfitting. Mixup mixes two samples by interpolating both the image and labels, while

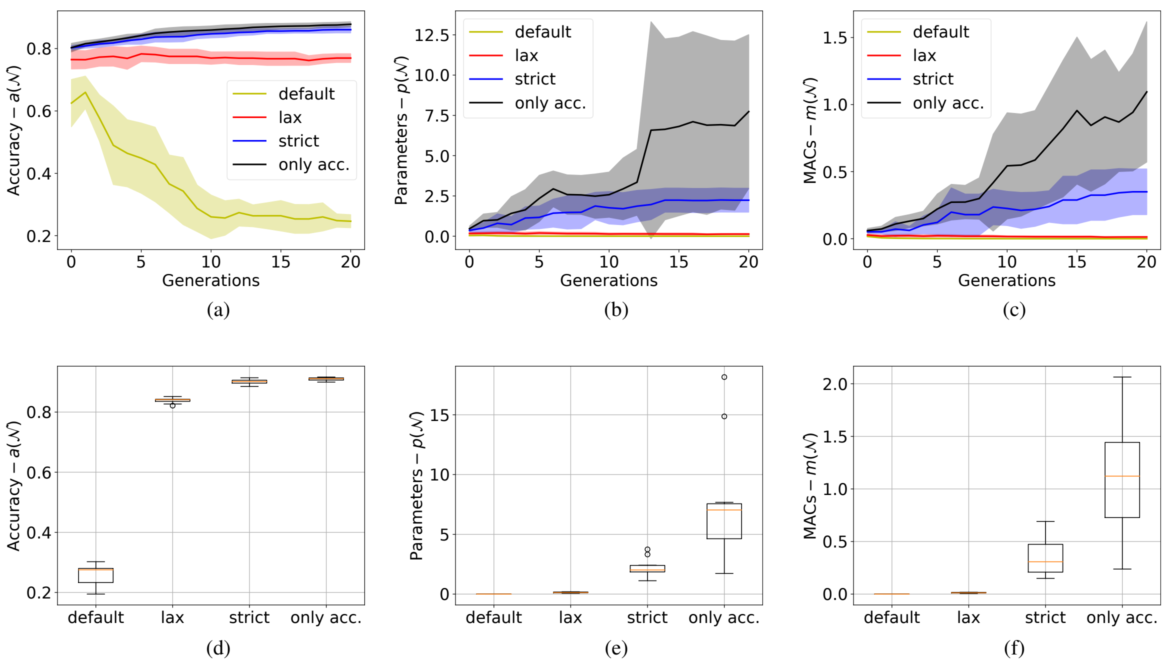

Fig. 3. NetScore configurations' performance on CIFAR-10. Each configuration was run 10 times. $p(\mathcal{N})$ is scaled to millions of parameters and $m(\mathcal{N})$ to billions of MACs. (a), (b) & (c): fittest individuals' average score (solid lines) and standard deviation (shaded areas). (d), (e) & (f): score distribution for the refined final models.

in Cutmix patches are cut and pasted among training images and labels are also mixed proportionally to the area of the patches. Using a batch size of 16, the learning rate uses Cosine Annealing with Warm Restarts schedule [44] with maximum learning rate automatically defined as the steepest descent on the training set [45]. We also show results where a pre-trained model is fine tuned on Adience. The pre-training uses Images of Groups Dataset [46] containing $28,205$ faces detected using Facenet's MTCNN [47] and the fine tuning uses the same parameters with learning rate scaled by 1%.

C. Performance Assessment

For each benchmark dataset, we use the following assessment schema:

a) CIFAR-10: The objective of this experiment is to validate the trade-off configurations. We run RENNAS ten times for each NetScore configuration. After the evolutionary search, the accuracy of the final model is estimated with cross-validation.

b) Adience: The experiments follow the 5-fold cross-validation scheme according to [17]. RENNAS produces a different neural architecture for each cross-validation iteration. We average the refined fittest individuals' scores.

In both scenarios, the final architectures are evaluated in terms of accuracy, number of parameters and MAC operations.

V. Results and Analysis

In this section, we present and analyze RENNAS experimental outcomes. The NetScore trade-off analysis is provided considering the default coefficients, lax, strict and only accuracy configurations on CIFAR-10. In the sequel, the outcomes on gender recognition are presented detailing the architectures found in the search phase and their respective performances.

A. NetScore trade-off analysis

Figure 3 shows that the NetScore with its default coefficients are not useful as a fitness function. It falls into a pitfall, trading a lot of accuracy to obtain a minimal network. As a result, it was the only configuration that ended up with an accuracy worse than the initial population. The lax trade-off configuration was not that destructive. Notwithstanding, it could not lead the initial population to better accuracy. The other two configurations achieved accuracy improvement. As expected, they have a small difference in accuracy, whereas their difference in the number of parameters and MACs is huge. The strict trade-off configuration imposes a selection pressure that makes the population accuracy improve generation by generation and results in cheaper networks than the accuracy-only configuration. Therefore, we can find efficient neural architectures by leveraging NetScore with a suitable configuration.

In order to obtain a better insight about the implicit accuracy trade-off imposed by the NetScore's coefficients, let us write Δ_a as a function of α, β, and γ. Equation (3) defines an infinite set of equivalent networks, i.e., individuals that have the same fitness value. It follows from (3) that

$$\Delta_a = 1 - (1 - \Delta_p)^{\beta/\alpha}(1 - \Delta_m)^{\gamma/\alpha} \tag{4}$$

Figure 4 shows the accuracy trade-off profiles for the aforementioned NetScore configurations. We observe the values of Δ_a as a function of Δ_p and Δ_m according to (4), for

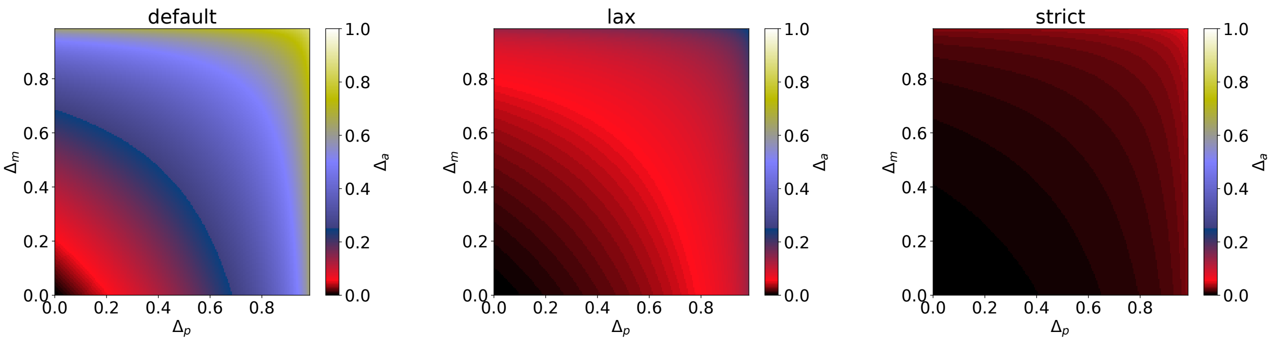

Fig. 4. Accuracy trade-off profiles for all equivalent networks $\mathcal{N}_1$ and $\mathcal{N}_2$ such that $\Omega(\mathcal{N}_1) = \Omega(\mathcal{N}_2)$, for $0 \leq \Delta_p,\ \Delta_m < 1$. (a) **Default configuration:** it accepts a 90% accuracy attenuation rate in the worse-case scenario if $\Delta_p = \Delta_m = 99\%$. (b) **Lax configuration:** it accepts $\Delta_a = 29\%$ if $\Delta_p = \Delta_m = 99\%$. (c) **Strict configuration:** it accepts Δ_a up to 7%.

$0 \leq \Delta_p, \Delta_m < 1$. The default configuration has an unsuitable accuracy trade-off because it would give the same NetScore to accuracy attenuation rates varying from 0% (if $\Delta_p = \Delta_m = 0$) to 90% (if $\Delta_p = \Delta_m = 0.99$) (Figure 4.(a)). The lax configuration accepts an accuracy trade-off varying up to 29% in the worst-case scenario if $\Delta_p = \Delta_m = 0.99$ (Figure 4.(b)). This explains why this configuration cannot lead the initial population to better accuracies. The strict configuration has a much better profile because it would accept to trade at most 7% of its accuracy in the worst-case scenario if $\Delta_p = \Delta_m = 0.99$ (Figure 4.(c)). Thus, the strict trade-off configuration will likely avoid falling into the pitfall of minimizing the network complexities at the expense of classification accuracy.

B. Gender recognition on unfiltered images

Table I shows the best architectures found by RENNAS on gender recognition task in Adience dataset. The networks have 20 to 38 layers. As one can see, the configuration of groups of blocks differ from the original ResNets: both in terms of number of filters per convolutional layer and in terms of number of residual blocks per group. Moreover, RENNAS found new ResNets with lower complexities than ResNet-34 [8].

Table II shows the results in terms of accuracy, number of parameters and MACs for different CNNs proposed for Adience, standard ResNet baselines and RENNAS best architectures. Compared to the baseline ResNets, RENNAS networks are better in terms of accuracy and cost. This shows that, for gender recognition in Adience, tuning the architecture is advantageous for performance and efficiency.

TABLE I
ARCHITECTURES FOUND BY RENNAS FOR ADIENCE.

Fold	$\mathcal{X}_f = [..., (f_i, b_i), ...]$, where f_i no. filters and b_i no. blocks
#1	$[(17,1),(14,2),(19,1),(59,5),(90,5),(101,1)]$
#2	$[(17,1),(33,1),(64,3),(32,1),(81,5),(98,2)]$
#3	$[(14,1),(21,1),(44,5),(57,2),(100,5),(115,1)]$
#4	$[(14,3),(32,3),(56,1),(151,4),(115,4),(150,3)]$
#5	$[(18,1),(43,1),(31,3),(78,3),(115,3),(195,2)]$

We also see that the architectures found by RENNAS show competitive performance to state-of-the-art results. We report the classification accuracy as originally presented in the papers [26]–[32]. However, some references do not specify the number of parameters and MACs. For those, we compute these indicators using Thop[3]. It is worth noting that some methods in the literature use pre-processing procedures — like face detection and alignment — and pre-trained networks on other datasets. The cost of these extra procedures, as well as training time, are not accounted in our evaluation.

Regarding classification accuracy, with no pre-processing nor pre-training, RENNAS outperformed all the three approaches: [29], [31], [32]. Moreover, with pre-traning, RENNAS rivals [27] and is the overall second best approach losing only to [30]. As for network complexities, the average number of parameters is smaller than all of the compared approaches, except [29] that achieves 87.1% accuracy with less than a million parameters. Finally the number of MACs is better than approaches by [26], [27], but similar or higher than the remaining networks.

These results show that RENNAS is able to find networks that are comparable to the state-of-the-art for the gender recognition task. The architectures are found within 3 GPU days each, which is low considering other ENAS proposals [9]. Considering the Adience gender recognition task, the models are shown to be efficient (mainly in terms of number of parameters), and competitive with other models for this task.

VI. CONCLUSION

In this paper, we introduced an evolutionary neural architecture search algorithm to automatically obtain efficient ResNet-based architectures. The proposed algorithm includes a variable-length strategy for individual's encoding into a search space directly parameterized by the input image size. In addition, custom mutation operators are presented to evolve the ResNet-based architectures. Furthermore, this algorithm could

[3]Available at https://openbase.com/python/thop. Thop is an open source package for computation of parameteres and MACs of PyTorch models.

TABLE II
COMPARISON WITH STATE-OF-THE-ART METHODS ON LITERATURE – ADIENCE DATASET.

Method			Accuracy$^{(a)}$	Parameters$^{(b)}$	MACs$^{(c)}$	Pre-processing$^{(d)}$	Pre-training$^{(e)}$	Design
ResNet-18 baseline [8]			89.0% (+)	11 (+)	2.1 (+)	No	No	
ResNet-34 baseline [8]			89.5% (+)	21 (+)	4.2 (+)	No	No	
ResNet-50 baseline [8]			87.9% (+)	24 (+)	4.7 (+)	No	No	
Agbo-Ajala and Viriri [30]			96.2% (−)	2.6 (+)	0.45 (−)	Yes	Yes	
Zhou et al. [27]			93.2% (+)	24 (+)	4.7 (+)	Yes	No	
Gurnani et al. [28]			91.8% (+)	74 (+)	21 (+)	Yes	Yes	Hand-crafted
Lapuschkin et al. [26] - GoogleNet [36]			91.7% (+)	4.0$^{(f)}$ (+)	1.5 (−)	Yes	Yes	
Lapuschkin et al. [26] - CaffeNet [48]			90.6% (+)	40 (+)	0.70 (−)	Yes	Yes	
Vu et al. [29]			87.1% (+)	0.90$^{(f)}$ (−)	0.051 (−)	No	No	
Levi and Hassncer [31]			86.8% (+)	11 (+)	0.71 (−)	No	No	
Ekmekji [32]			80.8% (+)	7.9 (+)	0.65 (−)	No	No	
RENNAS	Fold	#1	94.0%	1.3	1.5			
		#2	91.9%	1.3	2.8			
		#3	91.4%	1.6	0.73	**No**	**No**	**Automatic**
		#4	93.6%	4.2	0.99			
		#5	89.7%	2.6	1.9			
		Average	**92.1%**	**2.2**	**1.6**			
	Fold	#1	95.3%	1.3	1.5			
		#2	92.6%	1.3	2.8			
		#3	93.3%	1.6	0.73	**No**	**Yes**	**Automatic**
		#4	95.2%	4.2	0.99			
		#5	91.6%	2.6	1.9			
		Average	**93.6%**	**2.2**	**1.6**			

$^{(a)}$ Accuracy reported in the original papers. $^{(b)}$ Millions of parameters. $^{(c)}$ Billions of MACs. $^{(d)}$ Anything more sophisticated than resizing. $^{(e)}$ Pre-training on other datasets. $^{(f)}$ Values reported in the original papers. The remainder of parameters and MACs were estimated using Thop[3].

be easily modified to search for neural architectures based on different block-based designs such as DenseNet [33] and ResNeXt [34]. Future works could also compare it with other NAS, using known search spaces like DARTS [49].

We demonstrated that the NetScore evaluation metrics can be a useful fitness function, inasmuch as we established a strict trade-off between accuracy and network complexities. RENNAS is assessed on the challenging Adience gender benchmark. When compared to state-of-the-art algorithms, it presented better or competitive results on accuracy and network complexities. Although we tested NetScore on an ENAS, we hypothesise that it could also be applied to reinforcement-learning-based NAS (as a reward signal) and to gradient-based NAS (as a regularization component).

Other fitness functions could be designed, inspired by the NetScore's principles. It considers number of parameters and MACs as indicators of network complexities, but others might be more suitable to different applications (e.g. taking training time into account). It is noteworthy that number of parameters and MACs are hardware-independent indicators. However, a specific application for edge devices with limited computational resources will certainly require hardware-dependent indicators such as network size in memory and computing time. In addition, one could also formulate the trade-off on network performance and complexities as a multi-objective problem and obtain a Pareto front instead of a single solution. All of these new assessment metrics could also be useful as fitness functions provided that a reasonable trade-off between accuracy and model complexity is imposed. Here we showed how to analyze the trade-off both theoretically and experimentally, and we expect that similar analyses are useful to other optimization applications.

ACKNOWLEDGMENT

This work was partially supported by Samsung Eletrônica da Amazônia Ltda. under the Brazilian Informatics Law #8.387/91.

REFERENCES

[1] K. Z. Haider, K. R. Malik, S. Khalid, T. Nawaz, and S. Jabbar, "Deepgender: real-time gender classification using deep learning for smartphones," *Journal of Real-Time Image Processing*, vol. 16, no. 1, pp. 15–29, Sep 2017.

[2] A. Rattani, N. Reddy, and R. Derakhshani, "Gender prediction from mobile ocular images: A feasibility study," in *Proceedings of the 2017 IEEE International Symposium on Technologies for Homeland Security (HST)*, 2017, pp. 1–6.

[3] M. M. Islam and J.-H. Baek, "Deep learning based real age and gender estimation from unconstrained face image towards smart store customer relationship management," *Applied Sciences*, vol. 11, no. 10, 2021.

[4] F. Becerra-Riera, A. Morales-González, and H. Méndez-Vázquez, "A survey on facial soft biometrics for video surveillance and forensic applications," *Artificial Intelligence Review*, vol. 52, no. 2, pp. 1155–1187, Jun 2019.

[5] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, 2017.

[6] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021.

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

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 770–778.

[9] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A Survey on Evolutionary Neural Architecture Search," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2021, Conference name: IEEE Transactions on Neural Networks and Learning Systems.

[10] T. Elsken, J. H. Metzen, and F. Hutter, "Neural Architecture Search: A Survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[11] X. Zhou, A. K. Qin, Y. Sun, and K. C. Tan, "A survey of advances in evolutionary neural architecture search," in *Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 950–957.

[12] J. Huang, B. Xue, Y. Sun, and M. Zhang, "A flexible variable-length particle swarm optimization approach to convolutional neural network architecture design," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 934–941.

[13] F. E. Fernandes and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm and Evolutionary Computation*, vol. 49, pp. 62–74, 2019.

[14] ——, "Automatic searching and pruning of deep neural networks for medical imaging diagnostic," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5664–5674, 2021.

[15] A. Wong, "NetScore: Towards universal metrics for large-scale performance analysis of deep neural networks for practical on-device edge usage," in *Lecture Notes in Computer Science*. Springer International Publishing, 2019, pp. 15–26.

[16] A. Krizhevsky, G. Hinton, and others, "Learning multiple layers of features from tiny images," 2009, Publisher: Citeseer.

[17] E. Eidinger, R. Enbar, and T. Hassner, "Age and Gender Estimation of Unfiltered Faces," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 12, pp. 2170–2179, Dec. 2014.

[18] Xin Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep. 1999.

[19] Y. Xue, Y. Wang, J. Liang, and A. Slowik, "A self-adaptive mutation neural architecture search algorithm based on blocks," *IEEE Computational Intelligence Magazine*, vol. 16, no. 3, pp. 67–78, 2021.

[20] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2423–2432.

[21] M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, and M. Sjödin, "Deepmaker: A multi-objective optimization framework for deep neural networks in embedded systems," *Microprocessors and Microsystems*, vol. 73, p. 102989, 2020.

[22] M. Loni, A. Zoljodi, A. Majd, B. H. Ahn, M. Daneshtalab, M. Sjödin, and H. Esmaeilzadeh, "Faststereonet: A fast neural architecture search for improving the inference of disparity estimation on resource-limited platforms," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, pp. 1–13, 2021.

[23] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[24] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4095–4104.

[25] M. Javaheripi, M. Samragh, T. Javidi, and F. Koushanfar, "GeneCAI: genetic evolution for acquiring compact AI," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, ser. GECCO '20. New York, NY, USA: Association for Computing Machinery, Jun. 2020, pp. 350–358.

[26] S. Lapuschkin, A. Binder, K.-R. Müller, and W. Samek, "Understanding and comparing deep neural networks for age and gender classification," 2017.

[27] Y. Zhou, H. Ni, F. Ren, and X. Kang, "Face and gender recognition system based on convolutional neural networks," in *Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2019, pp. 1091–1095.

[28] A. Gurnani, K. Shah, V. Gajjar, V. Mavani, and Y. Khandhediya, "SAF-BAGE: Salient approach for facial soft-biometric classification - age, gender, and facial expression," in *Proceedings of the 2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019, pp. 839–847.

[29] D.-Q. Vu, T.-T.-T. Phung, C.-Y. Wang, and J.-C. Wang, "Age and gender recognition using multi-task CNN," in *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, 2019, pp. 1937–1941.

[30] O. Agbo-Ajala and S. Viriri, "Deeply learned classifiers for age and gender predictions of unfiltered faces," *The Scientific World Journal*, vol. 2020, pp. 1–12, Apr 2020.

[31] G. Levi and T. Hassncer, "Age and gender classification using convolutional neural networks," in *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2015, pp. 34–42.

[32] A. Ekmekji, "Convolutional neural networks for age and gender classification," *Stanford University*, 2016.

[33] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2261–2269.

[34] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 5987–5995.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[36] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[37] S. Kuruvayil and S. Palaniswamy, "Emotion recognition from facial images with simultaneous occlusion, pose and illumination variations using meta-learning," *Journal of King Saud University - Computer and Information Sciences*, 2021.

[38] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, "Designing network design spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 10 428–10 436.

[39] M. Lin, Q. Chen, and S. Yan, "Network in network," 2014.

[40] B. L. Miller, D. E. Goldberg *et al.*, "Genetic algorithms, tournament selection, and the effects of noise," *Complex systems*, vol. 9, no. 3, pp. 193–212, 1995.

[41] D. Goldberg, *Genetic Algorithms*. Pearson Education, 2006.

[42] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," in *Proceedings of the 2018 International Conference on Learning Representations*, 2018.

[43] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6023–6032.

[44] I. Loshchilov and F. Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," in *Proceedings of the 2017 International Conference on Learning Representations*, 2017.

[45] L. N. Smith, "Cyclical learning rates for training neural networks," 2017.

[46] A. Gallagher and T. Chen, "Understanding images of groups of people," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[47] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[48] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "CAFFE: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM International Conference on Multimedia*, ser. MM '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 675–678.

[49] M. Ruchte, A. Zela, J. Siems, J. Grabocka, and F. Hutter, "Naslib: A modular and flexible neural architecture search library," https://github.com/automl/NASLib, 2020.