Improved Facial Attribute Classification through Selective Test Data Augmentation

Master Thesis

Yuang CHENG
18-751-701

Submitted on
May 15 2021

Thesis Supervisor
Prof. Dr. Manuel Günther

University of Zurich^UZH
Department of Informatics

Master Thesis

Author: Yuang CHENG, yuang.cheng@uzh.ch

Project period: Nov. 26 2020 - May 15 2021

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

I want to express sincere gratitude for supervisor Prof. Dr. Manuel Günther for his continuous support and patience. He provided me with the necessary resources for the research. And without his insights and suggestions, this Master Thesis would not have been possible.

I am thankful to the members of system administration team for the maintenance of the computation server *Rolf*. Whenever there was a problem, they fixed it quickly.

I can not thank enough to my family for the support for my Master's study, especially during the COVID-19 pandemic.

Abstract

Applying data augmentation on test images can improve facial attribute classifications. The Alignment-Free Facial Attribute Classification Technique (AFFAT) shows that performing 162 transformations on test images can improve the prediction of facial attributes on the CelebA dataset. However, this process costs a substantial amount of time, and the effectiveness of each transformation is not considered. This research aims to find the best combination of transformations that will be applied as test-time data augmentation, such that the prediction results of the AFFACT model can be improved. Genetic algorithms are employed for the optimization task. As a result, even though the overall prediction accuracy does not improve, the number of transformations applied as test-time data augmentation has been decreased dramatically.

Contents

Chapter 1

Introduction

The task of facial attribute classification is to predict the attributes of a face, such as gender and whether having beard. It can be further employed in other applications such as image search (Kumar et al., 2011) and retrieval (Siddiquie et al., 2011). After Liu et al. (2015) introduced a large-scale benchmark dataset for this problem - CelebFaces Attributes (CelebA), facial attribute classification using deep learning methods have been improved.

Günther et al. (2017) propose Alignment-Free Facial Attribute Classification Technique (AFFACT) which has achieved state-of-the-art results on non-aligned faces. They pretrained a ResNet-50 network on the ImageNet dataset and add a fully connected layer before the model is fine-tuned on the CelebA dataset. Perturbations (shifts, scaling, horizontal flip and blurring) are applied to training images, and it results in the AFFACT network. The best prediction accuracy is obtained when 162 transformations are applied to detected bounding boxes as test-time data augmentation.

Compared with the commonly used test-time data augmentation technique, ten-crops (Krizhevsky et al., 2012; He et al., 2016), which yields only shifts to images, the 162 transformations employed in the AFFACT paper include shifts, rotation, scaling and horizontal flip.

However, to obtain the prediction results of a single image, the technique requires the DCNNs to make predictions on that image 162 times, which is a drawback. On the one hand, the running time of 162-transformations is much longer than that of ten-crops. The effectiveness of each transformation, on the other hand, is not taken into consideration. The transformations are assigned with equal weight. Some transformations may play a more important role while some are trivial, which means some transformations applied to images may not be necessary.

The main goal of this work is to build an optimization algorithm to search for a combination of transformations whose

- number of transformations is far smaller than that applied in the AFFACT paper *i.e.* 162 and
- the performance is better, or at least not much worse than the baseline.

I define the search space of the problem-domain of the optimization as all combinations of 162 transformations and the objective is to obtain the best prediction results of the AFFACT model on the CelebA dataset. Images may be better represented by transformations that do not belong to 162 AFFACT transformations. This paper also investigates a larger search space.

An immediate approach is to search the whole search space using brute-force methods. But the attempt to exhaust all combinations of 162 transformations in accepted time is destined to fail, given the fact that the number of all combinations of 162 transformations is an astronomical figure, let alone the larger search space. Therefore, I either implement brute-force search on a subset of the search space or employ a more efficient and faster searching mechanism.

An evolutionary algorithm (EA) is a feasible optimization approach. It mimics the natural process of evolution, based on *'Survival of the fittest'* from Darwinian evolutionary theory. The candidate solution *i.e.* a combination of transformations, to the problem of optimization, is denoted as an individual and its quality is assessed by a fitness function. An EA begins with a randomly initialized population. Offspring are generated by the way of recombination between selected individuals (crossover), alternation of elements of individuals (mutation). Then a proportion of the fittest individuals form a new population that replaces the old one and is used in the next iteration of the EA. The process is repeated until the termination criteria are met. In the end, the solution to the optimization will emerge.

A combination of transformations can be easily encoded into a string. Therefore, genetic algorithm (GA) (Goldberg, 1988; Holland, 1975), a subclass of EA is used in this paper. An individual taking part in a GA is a string that represents parameters from the problem-domain. The number of transformations is an undefined value. So the length of a genetic string is not fixed. It is adaptive instead.

To achieve the research goal, this paper implements GAs with different hyperparameters. GAs will be evaluated in terms of the performance of the AFFACT model on images processed by the best transformations they discover.

It is shown that while the overall prediction accuracy is not improved, the GAs have discovered better solutions on several attributes. Additionally, the solutions have far less transformations compared to the 162 transformations used in the AFFACT paper.

Chapter 2

Related Work

2.1 Facial Attribute Classification

The problem of facial attribute classification was introduced to the computer vision community by Kumar et al. (2008). They trained Support Vector Machines(SVMs) on a collection of low-level features to build a face search engine FaceTracer that involved facial attributes. Ehrlich et al. (2016) utilized a Restricted Boltzmann Machine(RBM) that served the objective of multitask learning.

Researchers have developed many deep learning models based on Deep Convolutional Neural Networks (DCNNs) after Liu et al. (2015) introduced a large-scale benchmark, CelebA. CelebA contains more than 200k images with 40 attributes attached to each image. Along with the datasets, Liu et al. (2015) presented three networks, two Localization Networks(LNets) and Attribute Network(ANet). LNets were built based on AlexNet and were used for locating faces. The ANet was built on Alexnet architecture as well. It was pretrained on the ImageNet dataset and then fine-tuned on the CelebA benchmark. The facial attribute classifications were obtained from SVMs. Wang et al. (2016) pretrained their DCNNs on external data obtained from body cameras and fine-tuned the model on CelebA before they employed SVMs to make the final prediction of attributes. These works required a single model to be learned for each attribute, which was inefficient.

Rudd et al. (2016) addressed this issue and they introduced the Mixed Objective Optimization Network(MOON) that was able to learn all attributes at once. They also handled the label imbalance during training. MOON was built on VGG-16 architecture (Simonyan and Zisserman, 2014). As a result, MOON outperformed all previously mentioned models.

The research of Günther et al. (2017) is the first to combine data augmentation techniques with unaligned faces. They applied perturbations to images during training phase and used data augmentation on test images as well. They have achieved the highest classification accuracy on the CelebA dataset to date.

2.2 Test-time Data Augmentation

Research shows that the performance of NN models on images can be improved by augmenting data at test-time. Test-time data augmentation is analogous to ensemble learning techniques (Shorten and Khoshgoftaar, 2019). It is denoted as data distillation that ensembles predictions to get a better representation of an image (Radosavovic et al., 2018).

AlexNet (Krizhevsky et al., 2012), which is CNNs trained on the ImageNet dataset, can obtain higher prediction accuracy from the average of outputs of CNNs of ten cropped patches. These ten crops are center crop, four corner crops as well as those of the horizontally flipped image.

The same ten-crop technique is used in the evaluation of Residual Networks (He et al., 2016) on ImageNet and CIFAR-10. Huang et al. (2017) compare the validation error of DenseNets using ten-crops with that using single-crop. Ten-crops is reported to outperform single-crop.

While ten-crops yield only shifts to images, other augmentation techniques can help with the prediction. Günther et al. (2017) train DCNNs on the CelebA dataset. During the test phase, they apply 162 transformations on test images and obtain state of the art from the average of CNN outputs. The 162 transformations are all combinations of rotation, shifts $\in \{-10, 0, 10\}$, scaling $\in \{0.9, 1.0, 1.1\}$ and horizontal flip. Perez et al. (2018) investigate the effectiveness of augmentation techniques including geometric transformations, elastic mixing and color transformations on the skin lesion classification task.

Test-time data augmentation is not heavily explored. As pointed out in Shorten and Khoshgoftaar (2019), assigning weight to each augmented prediction is likely to improve the performance of prediction on augmented images.

2.3 Genetic Algorithms For Optimization

Genetic algorithm (GA) (Goldberg, 1988; Holland, 1975) is a search-based optimization mechanism. It is one of the variants of EA. The entity that plays a role in a vanilla GA is formulated by a fixed-length numeric string that represents variables and parameters. This characteristic makes GAs suitable for optimization problems where they are plenty of parameters. It has been shown that GAs can handle problem-domain with millions of parameters (Whitley et al., 2019).

The population is randomly initialized, and it evolves more and more towards the better region of the search space. The individuals in the population are evaluated by an objective function that maps an individual to a real number. After selection, the selected individuals will give birth to children via genetic operators *i.e.* crossover and mutation.

Selection is the process to pick individuals out of the population and then to send them to the next phases of evolution. Researchers have developed several selection strategies, such as ranking selection, roulette wheel selection (Zhong et al., 2005), linear ranking selection (Baker, 1985) and tournament selection (Miller and Goldberg, 1995). Ranking selection is the simplest implementation where the individuals are ranked according to their fitness and the individuals with higher rankings are selected. Roulette wheel selection and linear ranking selection are similar. Individuals are assigned with probabilities to be selected based on their fitness and the total fitness of the population. In a tournament selection, pairs of individuals are picked out and compete with each other and the winners are selected. It is observed that tournament selection is the best among these selection strategies in terms of convergence rate and time complexity (Shukla et al., 2015).

Hasançebi and Erbatur (2000) summarize commonly used crossover approaches to two categories. One is performing crossover between subelements of two individuals. An element that represents an individual is divided into multiple subelements and crossover is performed locally on corresponding subelements of two elements. The other category does not concern partition. Elements or subelements can be divided by a single point or multiple points.

Mutation is the crucial process to restore lost genetic information. Crossover is a recombination of information and it does not bring new information into the population. After the initialization of a population, new parameters are introduced only through mutation.

Typical configurations of GAs use high crossover rate and low mutation rate (De Jong and Spears, 1990). But the choice of optimal control hyperparameters is an open issue. It involves the trade-offs between different perspectives of the quality of GAs. For example, increasing the mutation rate helps with restoring genetic information but it introduces a lot of randomness to the search process (Srinivas and Patnaik, 1994).

A healthy population with high diversity is believed to be important for discovering a good solution (Simon, 2013). This can be achieved by introducing great amount of variations to the individuals in a population or using local selection schemes under which evolution happens at different rates across individuals. Tournament selection is one of the local selection schemes.

To ensure the quality of the solution emerging does not drop over iterations, selection with elitism (Baluja and Caruana, 1995) is introduced. The new population consists of a proportion of children and parents.

Chapter 3

Approach

In this chapter, I describe the manipulation of the dataset, the image preprocessing procedure, and the genetic algorithm implementation.

3.1 Dataset, general optimization procedure

3.1.1 Dataset

The experiments were conducted on the CelebA dataset (Liu et al., 2015). CelebA consists of more than 200k images of celebrities with 19,867 images in the validation set and 19,962 in the test set. There are 40 binary codes representing attributes attached to each image. And each image is also labeled with five landmarks which can be used for alignment.

3.1.2 Optimization and evaluation procedure

In this paper, GAs with different hyperparamters are implemented. The optimizers firstly run separately for each attribute as well as the overall metric on the validation set in order to find the best combination of transformations that lead to the best prediction performance. Subsequently, the discovered best combination of transformations is applied to the test images before a prediction is made on the test set. Therefore, there are 41 independent results for each GA.

3.2 Image Preprocessing Pipeline

The CelebA dataset provides hand-labeled coordinates of five landmarks, both eyes, left and right mouth corners and the nose tips. These hand-labeled landmarks are used to calculate estimated bounding boxes of faces using the same approach in the AFFACT paper (Günther et al., 2017).

Specifically, the center of eyes $\vec{t}_e$ and the center of mouth $\vec{t}_m$ are calculated given the labels of both eyes $\vec{t}_{e_l}, \vec{t}_{e_r}$ and both mouth corners $\vec{t}_{m_l}, \vec{t}_{m_r}$ respectively.

$$\vec{t}_e = \frac{\vec{t}_{e_l} + \vec{t}_{e_r}}{2}, \quad \vec{t}_m = \frac{\vec{t}_{m_l} + \vec{t}_{m_r}}{2} \tag{3.1}$$

Then the mouth-eye-distance d is obtained: $d = ||\vec{t}_e - \vec{t}_m||$. The width w and the height h of the bounding box are estimated based on the distance $d: w = h = 5.5 \cdot d$. Finally, the four coordinates (left x_l, right x_r, top y_t, bottom y_b) of the bounding box are computed:

$$\begin{aligned} x_l &= x_e - 0.5 \cdot w & x_r &= x_e + 0.5 \cdot w \\ y_t &= y_e - 0.45 \cdot h & y_b &= y_e + 0.55 \cdot h \end{aligned} \tag{3.2}$$

The original rotation angle α_0 is calculated from eye coordinates and the scaling factor r_0 is drawn. W and H represent the resolution of the image after transformation.

$$\alpha_0 = \text{arc tan} \frac{y_{e_r} - y_{e_l}}{x_{e_r} - x_{e_l}} \qquad r_0 = \frac{W}{w} = \frac{H}{h} \tag{3.3}$$

A transformation is marked by five different geometric manipulations of an image. For a transformation, its parameters, rotation angle α , shifts on x- and y-axis x, y and scaling factor s, are added to the coordinates. Additionally, the image is horizontally flipped if the flip indicator is $True$. In this paper, a transformation is represented by (α, x, y, s, f).

$$\begin{aligned} \tilde{x}_l &= x_l + x & \tilde{x}_r &= x_r + x \\ \tilde{y}_t &= y_t + y & \tilde{y}_b &= y_b + y \\ \tilde{\alpha} &= \alpha_0 + \alpha & \tilde{s} &= s_0 + s \end{aligned} \tag{3.4}$$

Using these formulas, the image is loaded, transformed and cropped into an image with resolution $W = H = 224$, before extrapolation using nearest neighbor technique is performed on each color channel of the image. Only invalid pixels *i.e.* pixels lay outside the original bound are extrapolated.

Image transformations are handled by Bob[1] (Anjos et al., 2012). To speed up this process, I use *multiprocessing.Pool*[2] to parallelize the image preprocessing on the CPU.

An image is processed with a combination of multiple transformations. The AFFACT model makes attribute predictions on each transformed image. Then a final prediction is made based on the weighted average of the outputs of the AFFACT model (see Figure 3.1).

---

[1] `https://www.idiap.ch/software/bob/docs/bob/bob.ip.base/stable/py_api.html`

[2] `https://docs.python.org/3.8/library/multiprocessing.html`

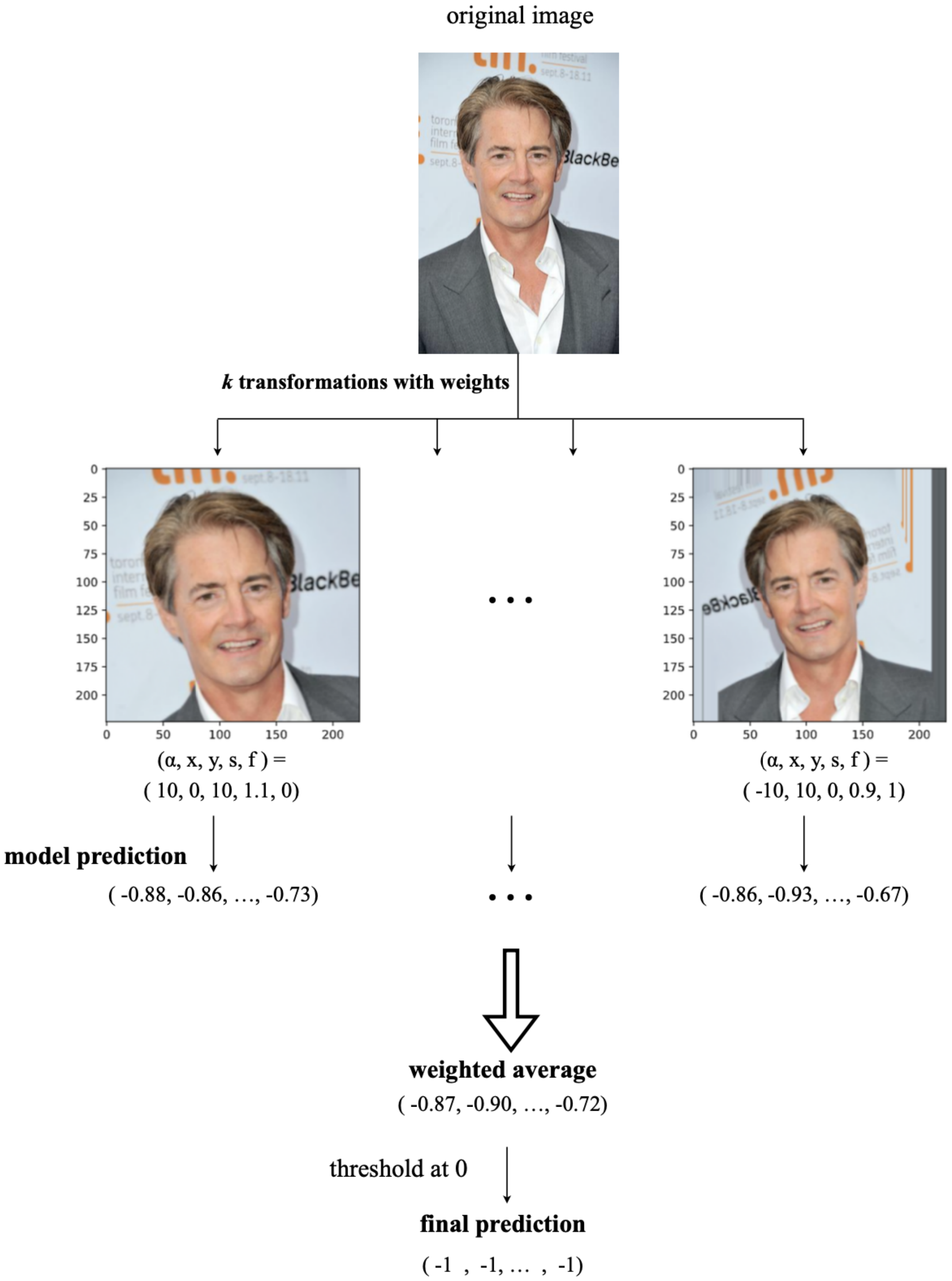


Figure 3.1: AN EXAMPLE OF IMAGE TRANSFORMATION. The image is processed with k transformations. Each transformation is assigned with a weight. The AFFACT mode makes predictions on k transformed images. Then the weighted average of DCNNs outputs is obtained. The average score is thresholded at 0 to take final prediction on attributes.

transformation parameter	search space 1	search space 2
rotation $\alpha \in$	$\{-10, 0, 10\}$	$[-20, 20] \cap \mathbb{Z}$
shifts $x, y \in$	$\{-10, 0, 10\}$	$[-20, 20] \cap \mathbb{Z}$
scaling $s \in$	$\{0.9, 1.0, 1.1\}$	$\{0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$
flip $f \in$	{0,1}	{0,1}
weight $w \in$	$\mathbb{N}^+ \cup 0$	$\mathbb{N}^+ \cup 0$

Table 3.1: SEARCH SPACE. The ranges of transformation parameters and weight in two search spaces. During mutation, the parameters are bounded.

3.3 Genetic Algorithms For Searching The Best Combination of Transformations

GAs are employed to search for the best combination of transformations per attribute well as the overall. In the beginning, the population is initialized with n individuals. And it evolves via selection, crossover, gene-level mutation, individual-level mutation and survival in each generation until the termination criteria are met.

An individual is decided by the expression of its genes combined and assigned with a fitness. A gene represents a set of five single transformations together with a weight (weigh w, rotation angle α, x-axis shift x, y-axis shift y, scaling factor s, flip f). An individual is a weighted combination of transformations. In this paper, a combination refers to multiple transformations.

One of the purposes of the research is to lower down the number of transformations performed as test-time augmentation. Therefore, a length constraint is applied, which forces the number of genes in an individual to be no greater than **twenty** at any time during optimization.

3.3.1 Search space

GAs implemented in this paper perform optimization on two search spaces respectively. One is all combinations of 162 transformations used in the AFFACT paper (search space 1). The other one is a much larger space (search space 2). In detail, parameters in a transformation can vary in the ranges shown in Table 3.1.

Using search space 1, the results of optimization can be compared with the 162 AFFACT transformations. And the DCNNs output for each transformation is able to be easily stored, which accelerate processing. Search space 2 contains more genetic variations but it requires more time for the optimization to converge, and it is impractical to cache DCNNs output for all transformations.

3.3.2 Initialization

The population is initialized with n individuals. In all implementations of GA, the population size is set to 100. An individual has k genes with $k \in [2, 7], k \in \mathbb{N}$. The weight of each gene is set to be an integer between 2 and 5. For GAs running on search space 1, a gene randomly picks a transformation from the pool of 162 transformations. For the GA running on search space 2, transformation parameters are randomly generated in the range shown in Table 3.1.

Algorithm 1 Fitness function

```
Input: an individual $p = (gene_1, \cdots, gene_k)$ with gene$_i = (\mathrm{w}_i, \alpha_i, x_i, y_i, s_i, f_i)$
       attribute $attr$, a dataset of $Q$ images
$avg\_output \leftarrow 0_{Q\times 40}$
$W \leftarrow \sum w$
for each $gene$ in $p$ do
    $\lambda \leftarrow w/W$
    if $\lambda > 0$ then
        load $images$
        $transformed\_images \leftarrow$transform$(images, \alpha, x, y, s, f)$
        $output_{Q\times 40} \leftarrow$ AFFACT_model$(transformed\_images)$
        $avg\_output \leftarrow \lambda \cdot output + avg\_output$
$prediction \leftarrow$ threshold$(avg\_output, \tau = 0)$
obtain $accuracy_{1\times 40}$
if $attr$ is overall then
    $fitness \leftarrow$ average$(accuracy)$
else
    $fitness \leftarrow accuracy[attr]$
return $fitness$
```

3.3.3 Fitness function

An individual is evaluated by a fitness function, which takes the genes of an individual, the attribute for which the optimization is carried out as input, and outputs the prediction accuracy of the AFFACT model on the transformed images from the validation set (Algorithm 1). Given an individual with k genes, for each gene, the AFFACT model makes predictions on images that are transformed corresponding to the parameters in it. It results in k DCNNs outputs. Then the weighted average of the k outputs is calculated. The averaged score is thresholded at 0 to take the final prediction. The prediction accuracy of the input attribute or the overall prediction accuracy is assigned as the fitness to the individual. I use the AFFACT model trained with euclidean loss.

After optimization, the solutions found out by GAs are evaluated on the test set.

$$\begin{aligned}
&avg\_output_{Q\times 40} = \sum_{i=1}^{k} w_i \cdot \text{AFFACT\_model}(transformed\_images) \\
&\text{with} \quad w \text{ refers to the weight of a transformation} \\
&prediction = \theta(avg\_output, \tau = 0) \\
&\text{where} \quad \theta(x, \tau) = \begin{cases} 1 & \text{if } x > \tau \\ -1 & \text{if } x \leq \tau \end{cases}
\end{aligned} \tag{3.5}$$

3.3.4 Genetic operations

The population will undergo tournament selection, crossover, parameter-level mutation and gene-level adaptive length mutation to create offspring. At the end of each iteration, a proportion of the parent population and the offspring will be selected to form the new population that goes to the next iteration.

Algorithm 2 tournament selection and crossover

```
Input: population $P = \{p_1, p_2, ..., p_n\}$, Fitness function $F(\cdot)$, Offspring $OS$
$Winner \leftarrow \emptyset$
for $i \leftarrow 0$ to $n/2$ do
    randomly pick $p_s, p_t \in P$
    if $F(p_s) > F(p_t)$ then
        $Winner \leftarrow Winner \cup p_s$
    else
        $Winner \leftarrow Winner \cup p_t$
for $i \leftarrow 0$ to $\|Winner\|/2$ do
    randomly pick $p_a, p_b \in Winner$
    if $p_a \neq p_b$ then
        $child \leftarrow$ crossover$(p_a, p_b)$
        $OS \leftarrow OS \cup child$
```

tournament

Tournament (Miller and Goldberg, 1995) is designed to select individuals out of the parents and these selected individuals will mate and give birth to new individuals through crossover. In a tournament, individuals with higher fitness have a higher probability to be selected.

Two random individuals are picked and compared with each other. The one with higher fitness is regarded as the winner. This process is repeated $\frac{n}{2}$ times and results in $\frac{n}{2}$ winners (see Algorithm 2).

In a tournament crossover, not only the individuals with high fitness can have offspring but also the individuals with low fitness. It preserves the diversity of the population, which is argued to be very important for discovering good solutions (Simon, 2013).

crossover

Crossover is a recombination of elements,*i.e.* genes between two parent individuals. A modified one-point crossover is performed. In a canonical one-point crossover, cut points of two individuals are at the same location. Because individuals do not have fixed lengths, the cut point is randomly drawn for each individual separately.

Two parents are randomly selected from winners. A parent is divided into two parts given a cut point, so as the mating partner. Then elements are swapped to form two children (Figure 3.2). A spawned individual is discarded if the number of its genes exceeds the limit, twenty. The crossover between two parents is repeated until at least one child is acceptable.

Crossover is performed on individual-level. A cut point lies between genes and it does not disrupt the structure of a gene.

mutation

All individuals within the parent population will mutate on parameter-level (Figure 3.3). For GAs optimizing on search space 1, a gene that represents a transformation in 162-transformations will be altered nominally to a random transformation and the weight score will change as well.

For GAs optimizing on search space 2, parameters including the weight in a gene will be altered numerically by a randomly generated value. Parameters are bounded according to Table 3.1. A mask is randomly drawn for each gene. It gates the mutation process. Only parameters

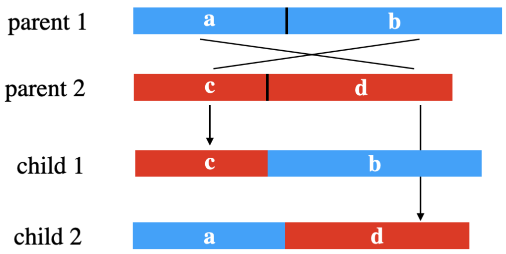
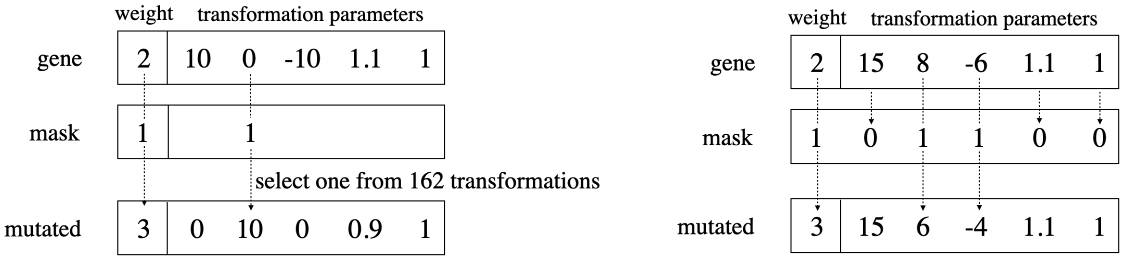


Figure 3.2: CROSSOVER. A child is accepted if its length does not exceed twenty



(a) Search space 1 (b) Search space 2

Figure 3.3: THE MUTATION OF A GENE. Parameters are randomly altered.

whose corresponding values in the mask are *True* will change by a random value x following the discrete uniform distribution over the set:

$$\begin{array}{ll} \{-1, 1\} & \text{if the parameter is } w \\ ([5, 0) \cap (0, 5]) \cup \mathbb{Z} & \text{if the parameter is } \alpha, x, y \\ \{-0.1, 0.1\} & \text{if the parameter is } s \end{array}$$

For horizontal flip f, if $f = 0$, then it is mutated to be 1. Otherwise, f changes to 0. For GAs that do not assign weights to genes, the weighting factors are set to be 1 constantly. The mutated genes are regarded as newborn children.

Algorithm 3 parameter-level mutation on search space 2

```
Input: population $\mathbf{P} = \{p_1, p_2, ..., p_n\}$, offspring $OS$
  for $i \leftarrow 0$ to $n$ do
    $child$ =copy($p_i$)
    for $j \leftarrow 0$ to $\|child\|$ do
      $gene \leftarrow child[j]$
      $mask := gene \rightarrow \{0, 1\}^6$
      for $k \leftarrow 0$ to 6 do
        if $mask[k]$>0 then
          $gene[k] \leftarrow gene[k] + x$                ▷ x is a random number
    $OS \leftarrow OS \cup child$
```

gene-level adaptive length mutation

In addition to the commonly used genetic operations, a proportion of parents will mutate on the gene-level. An individual will be either added a randomly drawn gene to it or be deleted one gene from it. If the individual has twenty genes, it will not expand. The random gene is generated using the same approach in initialization.

ranking selection

Not all parents are discarded. Instead, the fittest parents from the old population are carried over to the next. This method refers to elitism (Baluja and Caruana, 1995). It guarantees the solution obtained by a GA does not decrease over generations.

After the reproduction of offspring, the best $0.3n$ individuals from the parents and the best $0.7n$ individuals from the offspring are picked out to form a new population, which replaces the old one. All the other individuals are discarded. The survival mechanism drives the population to evolve towards the better direction.

Termination of GA

In some GA implementations, optimization terminates when the highest fitness of offspring is not greater than that of the parent. Because the search space is enormous, it is of high possibility that the highest fitness does increase in several consecutive generations. The early stopping is likely to take place in the very beginning. In this paper, the algorithms do not stop until the maximum generation is reached.

3.3.5 Dataset split

As a result of optimization with GA, the combination of transformations with highest fitness (lowest predict error) on the whole validation set is selected as the best individual. Thanks to the nature of GAs, the population becomes more and more adaptive to the data (the whole validation set) as the optimization goes. Therefore, the algorithm will always pick the best individual from the last survived generation. And this individual is the fittest one, which means it may have low generalizability. The prediction performance of the AFFACT model on the test set may suffer from overfitting.

To mitigate the potential problem of overfitting during optimization, I implement modified GAs with a different picking strategy in experiment 3 and 4. I simply expose the GAs to only half of the data and make another half unavailable. The validation set is randomly split into two parts, the seen set and the unseen set. The optimization is carried out on the seen set. Then the best individual from last several generation is selected and compared with each other. Because the population in the beginning may not be adaptive to the data, they are not considered. The algorithms pick the individual whose prediction accuracy is the highest on the unseen set (see Figure 3.4). Hopefully, the dataset-splitting strategy will select an individual whose fitness is among the best as well as the generalizability. Experiments (experiment 3 and 4) that employed this strategy share the same dataset partition.

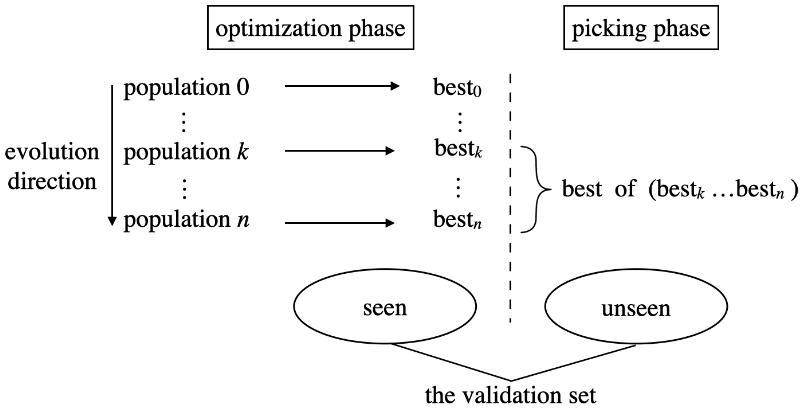


Figure 3.4: GA Optimization On The Split Validation Set. After optimization, the best individuals from last several generations are compared with each other on the unseen set.

Chapter 4

Experiment

4.1 Baseline

Two baselines are established in this paper. One baseline is obtained after performing 162 transformations on test images from the CelebA datatset. It is denoted as baseline-T. Since this paper does not use an automatic bounding box detector as it does in the AFFACT paper, prediction errors of the baseline are not comparable to those reported in the AFFACT paper. The other one is the attribute classification results on aligned faces and it is denoted as baseline-A.

4.2 Experiment 1: Brute-force search

The research starts with a brute-force search to investigate the effectiveness of all transformations that are used in AFFACT paper (162-transformations). I design an that optimization for all combinations of k transformations with $k \in [1, 20], k \in \mathbb{N}$ separately. The search is performed on the validation set and then the discovered solutions are evaluated on the test set.

The running time increases dramatically as the number of transformations k increases. The time complexity of the brute-force search algorithm is $O(n^k)$. For this reason, exhaustive search is only employed for optimization on the overall metric (average prediction of 40 attributes), not per attribute.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} = \frac{(n-k+1) \cdot (n-k+2) \cdots n}{k!}$$

$$T(n,k) = T(\frac{n!}{k!(n-k)!}) = O(n^k), \text{ with } k << n$$

Nevertheless, it is still hardly feasible to perform a greedy search for all k. By the time I submitted the paper, results of $k = 1, 2, 3, 4$ were obtained. It is shown that none of the best combinations of 1,2,3,4 transformations led to a better prediction performance with respect to the baseline. As the number of transformations k increases, the prediction result gets better (Table 4.1). The results indicate that to get better overall prediction performance, the number of transformations applied on an image should not be smaller than 4.

Table 4.2 shows the best combinations for $k = 3, 4$. It is observed that the aligned transformation (no additional transformation is applied to the aligned faces) is included in the solutions for neither $k = 3$ nor $k = 4$.

	baseline-T	baseline-A	$k = 1$	$k = 2$	$k = 3$	$k = 4$
OVERALL error rate	8.098	8.251	8.211	8.156	8.168	8.142

Table 4.1: Results of brute-force search for best combinations of transformations

	The best combination of transformations
$k = 3$	(-10, -10, 10, 1.0, 1), (0, -10, 10, 1.1, 0), (10, 10, 0, 1.0, 1)
$k = 4$	(-10, -10, 10, 1.1, 1), (0, 0, 0, 1.1, 0), (0, 10, 0, 0.9, 1), (10, 10, 10, 0.9, 0)

Table 4.2: The solutions found out by the brute-force approach

4.3 Experiment 2: Genetic optimization on the whole validation set

The brute-force-search-based optimization can not complete the task. It is necessary to implement GAs, which do not exhaust all the possible combinations and run faster and more efficiently. The running time is solely dependent on the population size instead of the size of the searching space.

The first GA (GA1) proposed in this paper serves the same purpose as the brute-force search. It aims to find the best combination from 162-transformations without considering the importance of each transformation.

In addition, the paper investigates the effectiveness of each set of transformations. On the basis of GA1, GA2 assigns weights to transformations.

In this experiment, the search space is the combinations of 162-transformations.

4.3.1 setup

GA2 assigns weights to genes in an individual, while GA1 does not. The size of the population of all algorithms is 100. The GAs terminate when they reach maximum generation. The whole validation set is used during optimization.

The maximum generation of GA1 is thirty (GA1-30). I run GA2 two times with maximum generation to be thirty and a hundred respectively(GA2-30, GA2-100). For all GAs, the parameter-level mutation rate is 1.0 and the adaptive length mutation rate is 0.2.

Temporary storage caches the DCNNs output of each set of transformations. Since the optimization is carried out in search space 1, there are at maximum 162 entries stored in the cache. Before the AFFACT model makes a prediction, the GAs firstly check whether there is a corresponding entry to the set of transformations in the gene. If so, the DCNNs output will be obtained directly, and the step of model prediction will be skipped.

The implementation of cache allows the algorithms not to run DCNNs model repeatedly and therefore running time can be decreased.

4.3.2 results

prediction error rates

On overall error rate, GA1-30 (8.098%) outperformed the other two GAs and GA2-30(8.107%) was better than GA2-100 (8.110%). However, the differences were minimal. GA1-30 was the only one that obtained no worse overall classification result with respect to the baseline (see Table 4.3).

Out of forty attributes and the overall metric, GA1-30 performed no worse on 9 (22%) compared to the baseline. The numbers of GA2-30 and GA2-100 are 14 (34%) and 11 (27%).

All three GAs obtained better prediction results compared to baseline-A on nearly all attributes.

Independent T-tests were carried out to check whether there were significant differences between the error rates of baselines and the results of GAs. Overall error rates were not included in the t-test. Because the overall error rate of the baseline is the average of the error rates of 40 attributes while the overall error rates were obtained independently by GAs. The statistical analysis showed that none of the three GAs was significantly different from the baseline-T or baseline-A (all *p-value*>0.05).

While GA1-30 achieved the lowest overall prediction error rate, it was beaten by GA2-30 in terms of the number of improved attributes.

number of transformations

The solution discovered by GA1-30 on the overall classification result has eighteen different transformations (Table A.1). The numbers of GA2-30 and GA2-100 are 18 and 14 respectively. The dominant transformation in the solution of GA2-30 ($(\alpha, x, y, s, f) = (-10, -10, 10, 1.0, 1)$) contributed 11.3% to the prediction (Table A.2) while the heaviest transformation in the solution of GA2-100 ($(10, 10, 0, 0.9, 0)$) had a weighting factor of 14.6% (Table A.3).

The genes in the solutions of GA1-30 are not necessarily to be different from each other, because crossover may cause duplicates in an individual. There were duplicate transformations in the solutions for all forty attributes. Surprisingly, the solution for the overall metric was the only one that did not contain repeated transformations. This helps explain why the overall error rate of GA1-30 was lower than GA2-30 while GA1-30 was outperformed by GA2-30 in terms of the number of improved attributes.

convergence and overfitting

Convergence and overfitting are two major issues concerned with GAs. Figure 4.1 shows the variations of the overall error rates of the best individual over iterations for three GAs on the validation set. The population of three GA2 was initialized to have poor quality. After few generations, they evolved into a better region. And the best solution kept improving as the optimization went on. This was a result of combining high reproduction rates (mutation and crossover) and elitism. High reproduction rates introduced a great amount of new genetic information into the population and elitism led to the non-decreasing quality of the best solution.

On the validation set where optimization took place, the quality of the solution found by GA2-100 was better than those discovered by the other two. Since GA2-100 had more generations, the population had more chance to evolve and therefore more adaptive to the validation set.

However, results were opposite on the test set where GA2-100 was outperformed by both GA1-30 and GA2-30. As stated in section 3.1.1, GA2-100 suffered from overfitting. During optimization, the population evolved within the environment pictured by the validation set. An individual that is extremely fit on this environment is likely to not be adaptive to a new environment. Experiment 3 addressed this issue.

Not only a large number of generations incurred overfitting, but also the usage of the weighting factor. On the validation set, the solution of GA1-30 had the poorest quality. But GA1-30 achieved the best overall result on the test set. The only difference between GA1-30 and GA2-30 was the presence of weights. There are three transformations with heavy weight (>10%) in the solution of GA2-30.

Attribute	baseline-A	baseline-T	GA1-30	GA2-30	GA2-100	GA3	GA4
5 o Clock Shadow	5.245	5.130	5.110	5.100	5.115	5.085	5.170
Arched Eyebrows	15.695	14.989	15.364	15.189	15.239	15.209	15.334
Attractive	17.012	16.772	16.877	16.907	16.862	16.917	16.862
Bags Under Eyes	14.713	14.633	14.427	14.467	14.498	14.462	14.518
Bald	0.957	0.917	0.937	0.952	0.952	0.907	0.917
Bangs	3.847	3.792	3.827	3.857	3.927	3.832	3.827
Big Lips	27.192	27.207	27.342	27.362	27.292	27.437	27.397
Big Nose	15.630	15.514	15.555	15.600	15.645	15.740	15.509
Black Hair	9.568	9.378	9.423	9.398	9.453	9.408	9.358
Blond Hair	3.917	3.807	3.877	3.822	3.872	3.817	3.837
Blurry	3.572	3.537	3.492	3.482	3.487	3.517	3.487
Brown Hair	10.605	10.209	10.295	10.470	10.360	10.310	10.405
Bushy Eyebrows	7.159	6.908	6.913	6.968	6.918	7.028	6.943
Chubby	4.323	4.338	4.428	4.318	4.288	4.368	4.368
Double Chin	3.447	3.502	3.517	3.437	3.487	3.547	3.577
Eyeglasses	0.361	0.311	0.306	0.321	0.346	0.331	0.336
Goatee	2.405	2.425	2.425	2.470	2.400	2.440	2.425
Gray Hair	1.738	1.648	1.688	1.728	1.723	1.718	1.688
Heavy Makeup	8.145	7.900	7.805	7.950	7.795	7.875	7.850
High Cheekbones	11.853	11.832	11.848	11.802	11.918	11.898	11.832
Male	1.483	1.302	1.333	1.323	1.312	1.318	1.338
Mouth Slightly Open	5.901	5.646	5.691	5.646	5.751	5.631	5.646
Mustache	2.845	2.655	2.705	2.730	2.745	2.700	2.685
Narrow Eyes	12.233	12.148	12.293	12.198	12.163	12.178	12.158
No Beard	3.577	3.587	3.527	3.557	3.582	3.562	3.507
Oval Face	23.059	22.778	22.809	22.873	22.939	22.808	22.793
Pale Skin	2.820	2.680	2.820	2.745	2.735	2.770	2.700
Pointy Nose	22.252	21.932	21.967	21.967	22.077	22.012	22.027
Receding Hairline	5.966	5.936	5.871	5.891	5.961	6.021	5.936
Rosy Cheeks	4.764	4.614	4.629	4.709	4.629	4.664	4.599
Sideburns	2.239	2.104	2.119	2.089	2.134	2.109	2.094
Smiling	6.818	6.442	6.618	6.467	6.512	6.547	6.512
Straight Hair	14.528	14.272	14.357	14.427	14.387	14.427	14.427
Wavy Hair	13.556	13.260	13.491	13.295	13.290	13.355	13.051
Wearing Earrings	9.142	8.822	8.967	8.912	8.882	8.847	8.962
Wearing Hat	0.822	0.812	0.837	0.827	0.837	0.842	0.822
Wearing Lipstick	6.132	5.996	6.006	6.077	6.036	6.117	6.127
Wearing Necklace	10.715	10.640	10.194	10.114	10.144	10.159	10.194
Wearing Necktie	2.705	2.670	2.695	2.645	2.655	2.605	2.640
Young	11.081	10.871	10.891	10.785	10.971	10.851	10.846
OVERALL	8.251	8.098	8.098	8.107	8.110	8.113	8.104
Number of attributes on which the error rate is not higher than the baseline			10	11	9	11	17

Table 4.3: ATTRIBUTE CLASSIFICATION ERROR RATES. The numbers are error rates in percentage. Underlined numbers refer to the lowest error rates among GAs.

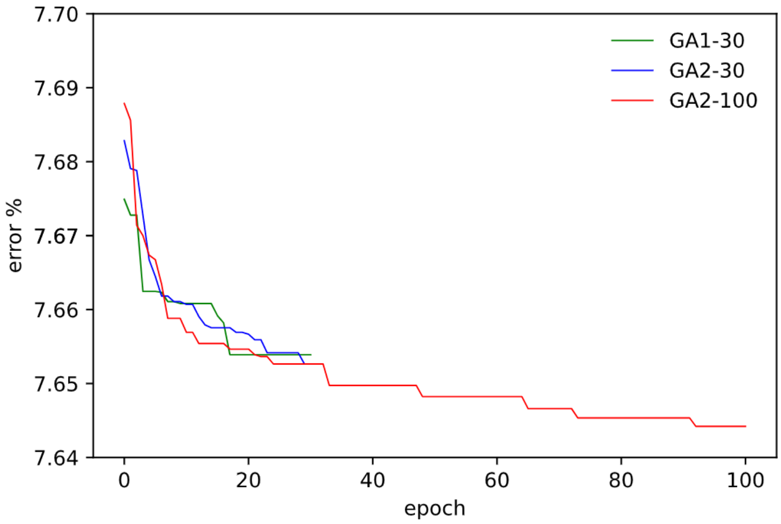


Figure 4.1: OVERALL Error Rate of The Best Individual Over Iterations

4.4 Experiment 3: Use split data to mitigate overfitting

It is observed that the population gets overfit as it evolves in experiment 2. To alleviate this problem, the data-splitting strategy is used in the optimization with a genetic algorithm (GA3). The validation set is randomly split into the seen set and the unseen set. I firstly perform optimization with GA3 on the seen set with 100 iterations. Secondly, the best individuals from the last eighty generations are picked out and their fitness on the unseen set. Finally, the individual with the highest fitness on the unseen set is selected as the best one.

If two individuals have the same error rate on the unseen set, the one with a lower error rate on the seen set will be picked.

Results of experiment 2 also show that applying weights to genes slightly impaired the overall prediction result. In this experiment, I implement a genetic algorithm (GA4) optimizing on the split validation set without applying weights.

4.4.1 setup

The search space is the same as experiment 2, the combinations of 162-transformations. In this way, the effectiveness of data-splitting can be evaluated. In GA3, transformations are assigned with weights. The population size is 100, the parameter-level mutation rate is 1.0, the individual-level length mutation rate is 0.2. GA4 is identical to GA3 except that GA4 does not apply weights.

As in GA1 and GA2, cache is implemented in GA3 and GA4.

4.4.2 results

prediction error rates

GA3 was slightly outperformed by GA2-100 on the overall error rate. GA3 achieved an overall error rate of 8.113%, higher than GA2-100 (8.110%) with the difference to be 0.003%. While the overall error rate was still higher than the baseline, GA3 had lower error rates on 11 attributes.

Figure 4.2 shows the variations of the error rates over iterations. For GA3, the lowest error rate on the unseen set is achieved at the very beginning. However, the best individuals from the first twenty generations are not taken into consideration because the population has hardly evolved and the individuals are not adaptive to the environment. This is evidenced by the high error rates on the seen set of the first few generations. The best individual is picked from the 46th epoch (from the 46th to the 53rd epoch, the best individual does not change).

The overfitting problem emerged in the last 50 generations of GA3. After the 50th epoch, as the error rate on the seen set decreased, the error rate on the unseen set went in the opposite direction. This pattern can be observed in the late stages of optimizations for all 40 attributes. GA4 showed a similar pattern (Figure 4.2b) besides the drop of error rate on the unseen set at the very end. It indicates that the data-splitting strategy can mitigate overfitting and therefore improve the quality of the solution obtained by the GAs.

GA4 obtained better overall prediction than GA3 (Table 4.3). Although the overall error rate of GA4 (8.104%) was slightly higher than GA1-30, it had most number of improved attributes (17) among GAs.

However, the independent t-tests did not show significant differences between GA3 and the baseline-T, or between GA4 and the baseline-T (all p-value>0.05).

number of transformations

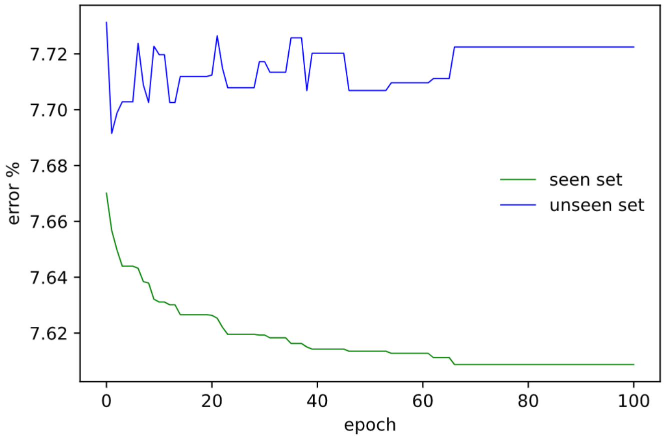
For optimization on the overall metric, the best combination discovered by GA3 contains 17 different transformations, which is the larger than any other attribute. The heaviest transformation ($(\alpha, x, y, s, f) = (10, 10, 0, 0.9, 0)$) contributes 13.6% to the average DCNNs output (see TableA.4). Apart from this transformation, the weights of other transformations are very close to each other.

In general, the number of the best combination of transformations per attribute ranges from 2 to 16. The average number is 8.25 (± 3.22). The solution for *Bald* has the minimum number of transformations, 2. However, the classification result of it (0.907%) on the test images was better than the baseline. It may result from the high performance of the AFFACT model on predicting *Bald*. The error rate of the baseline is already very low. And the evaluation metric could not tell big differences from the individuals. Evolution did not happen to a great extend as it is shown in Figure 4.3. This is similar to the natural world where all individuals enjoy themselves in a good environment.

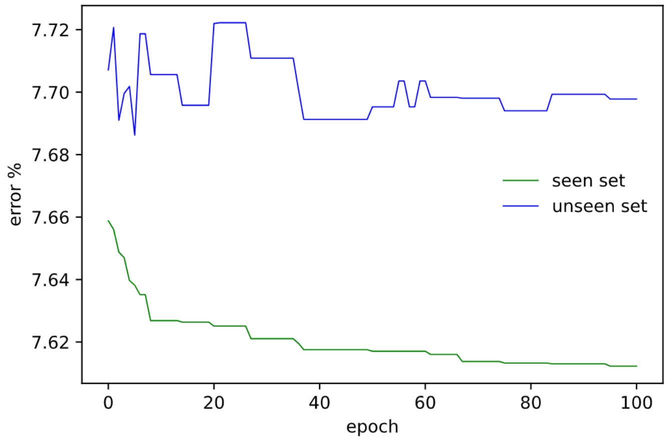
The solutions discovered by for forty attributes and the overall metric cover 102 different transformations. And the number becomes 62 if the solutions that are worse than the baseline are excluded.

As mentioned above, the GAs not applying weights still attached weights to transformations by producing duplicates in an individual. As a result of GA4, duplicates were observed in the solutions for 27 attributes and the overall metric. Out of the solutions for 17 improved attributes, duplicated are found in 13 solutions. In the best combination of transformations for the overall error rate, there was a transformation occurred twice which corresponds to a weighting factor of 11.7% (see Table A.5). Hence, it is hardly confident to say applying weights is harmful for the optimization.

In total, GA4 covers 104 different transformations in the solutions. Only regarding the solutions for the improved attributes, there are 76 transformations. It shows that at least 47% of the



(a) GA3



(b) GA4

Figure 4.2: THE VARIATION OF OVERALL ERROR RATE OF GA3 AND GA4. The blue line represents the error rate on the unseen set and the green line is the error rate on the seen set.

162 AFFACT transformations are effective in improving the classifications.

diversity of the population

In this experiment, I track the diversity of the population by measuring the standard deviation of fitness as well as the fitness of the worst individual of GA3 (Figure 4.4). The initial population had the highest variance. The variance immediately dropped to a low point after the initialization and then, it showed a tendency to increase as the optimization carried on.

Since the population is totally randomly initialized, it is not surprising that the variance of the first population is the highest one. Then selection took place and the individuals with low quality were discarded, which led to the immediate decrease of variance in the first few generations. As the process of evolution went on, more and more genetic information was introduced to the population by mutation. Therefore, the variance of the population rose.

It is hard to predict the behavior of the variance if the optimization keeps running indefinitely. From the observation that the quality of the worst individual did not deteriorate, the variance is

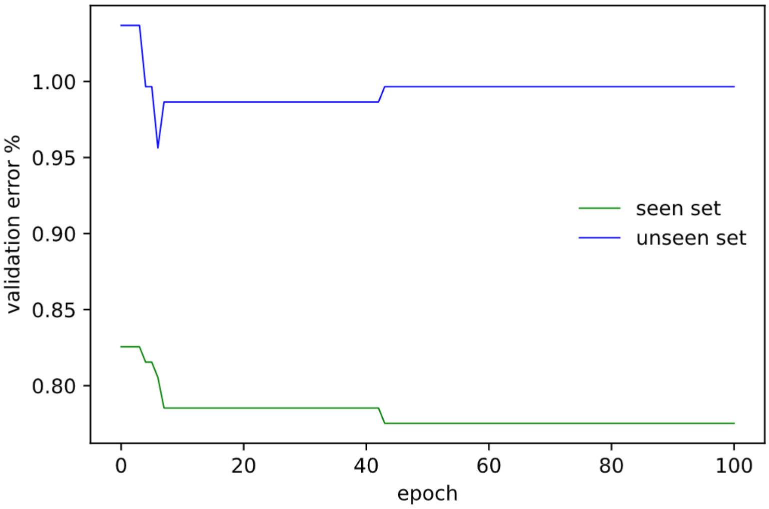


Figure 4.3: THE VARIATION OF ERROR RATE ON *Bald* OF GA3.

likely to have an upper bound. This was a result of elitism.

4.5 Experiment 4:Optimization on a larger search space

Previous GAs did not improve the overall classification performance compared to baseline-T. It is possible that the best combination of transformations has transformation parameters that lay outside the range of 162 AFFACT transformations. In experiment 4, the genetic algorithm (GA5) runs on a larger search space.

4.5.1 setup

GA5 performs optimization on search space 2. The data-splitting strategy is employed. Using only the seen set for optimization can not only avoid overfitting but also reduce running time. Weight is applied in GA5 for the purpose of investigating the effectiveness of each transformation. From another perspective, there is no good reason to abandon the usage of weight because GAs not applying weight still assign weights to transformations by producing duplicates.

Unlike previous GAs, GA5 does not use temporary storage to record DCNNs outputs of all transformations. Instead, the cache has a limited size. If the cache is full, the oldest entry will be deleted before new entries are written into the cache.

To reduce processing time, the maximum generation is twenty. And the mutation rates are the same as previous GAs.

4.5.2 results

Because GA5 is time-consuming, only the optimization for *5 o Clock Shadow*, *Arched Eyebrows* and overall metric have completed. GA5 achieved an overall error rate of 8.099%, which outper-

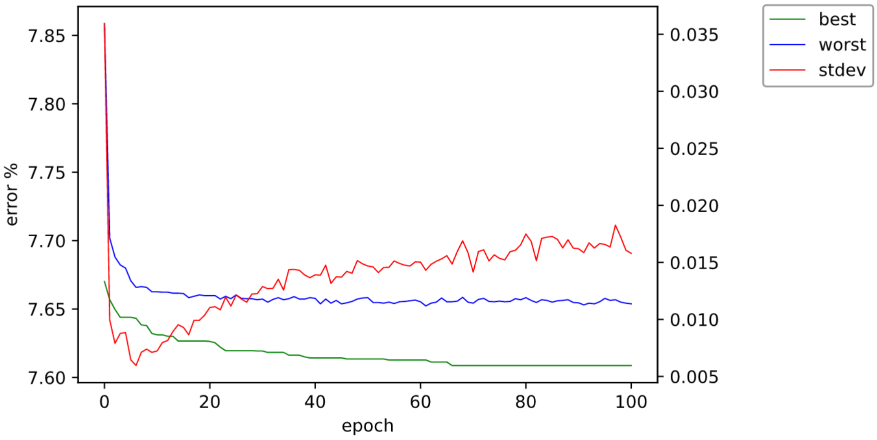


Figure 4.4: ERROR RATES AND STANDARD DEVIATION. Two error rates are given in the figure. One belongs to the best individual from each generation (green) and the other one belongs to the worst (blue).

attribute	baseline-A	baseline-T	GA5
5 o Clock Shadow	5.245	5.130	5.030
Arched Eyebrows	15.695	14.989	15.019
OVERALL error rate	8.251	8.098	8.099

Table 4.4: ERROR RATES OF GA5. Only error rates on two attributes and the overall error rate are given.

formed all the other GAs except GA1-30. And the difference between GA5 and the baseline-T is only 0.001%. The prediction on *5 o Clock Shadow* was better than the baseline-T.

Figure 4.5 shows the variation of overall error rate. After the ninth generation, the error rate of the best individual did not change. The optimization might fall into a local optima. Or the optimization did not converge because insufficient number of generations was provided. Even so, GA5 has obtained almost the best classification results.

The solution for overall metric contains 11 different transformations and the weighting factor of the heavies one is 14.7%. The number of transformations of the solution discovered by GA5 is the lowest among GAs.

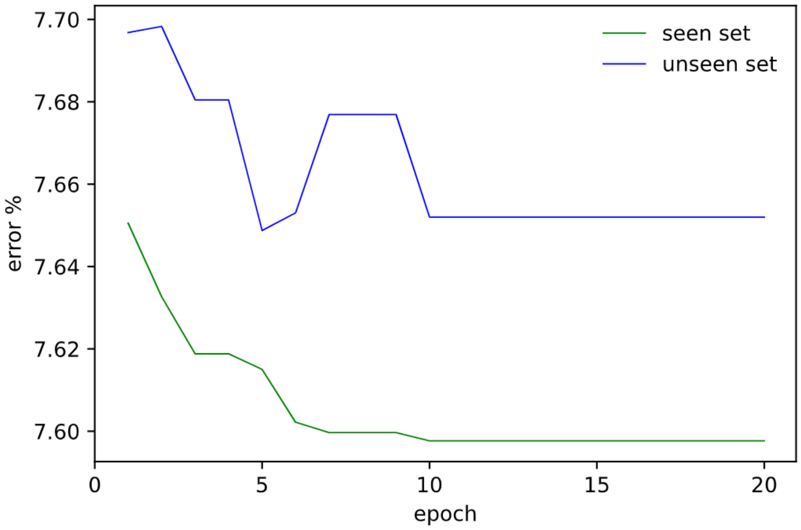


Figure 4.5: THE VARIATION OF OVERALL ERROR RATE OF GA5 .

Chapter 5

Discussion

analysis of GAs on search space 1

None of the GA shows statistical differences from the baseline-T or baseline-A. It exhibits the robustness of the AFFACT model. Some researchers believe that the impact of test-time augmentation on classification is a measurement of the robustness of a classifier (Shorten and Khoshgoftaar, 2019). The predictions of a robust model should not be greatly affected by augmentations.

Moreover, the high performance of the AFFACT model on the CelebA dataset makes it difficult for the GAs to outperform the baseline. In fact, the AFFACT model has achieved highest accuracy among studies. The environment in which the population evolves is good enough, and the differences in gene codes are played down when they express themselves in such environment. The performance on the CelebA dataset has been in plateau (Thom and Hand, 2020). Since the introduction the benchmark, only 4% improvement in accuracy has been achieved on average. The baseline can be imagined as Mount Everest on the Tibetan Plateau.

GA1-30 has discovered the best solution on the overall classification results. But the number of improved attributes of GA1–30 is the lowest among all GAs. The most probable explanation for this observation is that GA1-30 found the best combination of transformations for the overall metric by chance. GAs rely on guided stochastic processes, which means the paths taken are non-deterministic and unrepeatable (Sloss and Gustafson, 2019). In the meanwhile, I introduce a lot of randomness to the algorithms. For example, the changes to the parameters are randomly drawn during mutation. The nature of randomness of GAs makes it difficult to explain some results. Nevertheless, since the task is to discover a solution of high quality, how the GA paves the way for this solution is not concerned and is not able to be explored.

The overall error rate of GA1-30 is equivalent to the baseline-T. It is concluded that the solution of GA1-30 is better than the ensemble of 162 transformation because it decreases the number of transformations applied to images from 162 to 18 (-89%). Based on the same reasoning, the results of this research is satisfactory. All GAs deliver better results on several attributes (9 - 17).

In terms of the number of improved attributes, GA4 outperformed other GAs. Despite the arguable usage of weight, the optimization process benefit from both data-splitting strategy and not applying weight.

analysis of GA5

Even though GA5 ran for only twenty generations, it has achieved almost the best overall classification results. The solution on overall metric found out by GA5 contains fewest number of transformations. The results show that large search space is somehow an advantage for optimization. GA5 brings more genetic variations and therefore may increase the diversity of the population. Unfortunately, the results for all forty attributes are not able to be presented in this paper.

Some parameters of the solution on overall metric lay outside the range of search space 1. An interesting finding is that none of the rotation angles of the transformations is 0, which means titled images may be desirable for the AFFACT model.

problems with the CelebA dataset

In addition to plateauing, the CelebA dataset have imbalance in its labels. This problem is addressed in the work of Rudd et al. (2016). Optimization with GAs showed imbalance across attributes as well. Some attributes improved in all GAs (*e.g.* 5_o_Clock_Shadow, Wearing_Necklace, Bags_Under_Eyes), while some did not improve at all (*e.g.* Arches_Eyebrows, Attractive, Big_Lips).

Moreover, there are mislabeled images in the dataset, which may hinder the improvement of classifications.

control parameters, settings and the health of GAs

The configurations of GAs are trade-offs between different perspectives.

A diverse population is vital for delivering good solutions. To serve this goal, I use a large population size,100 and high mutation rates. Expanding the population size will indeed increase the diversity of the population and in the meanwhile decrease the chance to fall into local optima. But it will also increase the converge time. Using high mutation rates helps with introducing new genetic information to the population. Figure 4.4 indicates that the diversity of the population is preserved even in the end of evolution.

Elitism ensures the optimization good solutions. It enlarges the problem of overfitting as it keeps the most adaptive individuals alive in the population. However, overfitting can be alleviated by using data-split strategy as it is shown in experiment 3.

all-at-once optimization

In this paper, the GAs perform optimization separately for each attribute as well as the overall metric. In another word, the fitness function serves a single objective. The process is cumbersome. Like 162 transformations and ten-crops, a universal augmentation technique for all attributes is desirable.

The key is to build a multi-objective fitness function. The overall error rate, *i.e.* the arithmetic mean of error rates on 40 attributes, captures general performance of the facial attribute classifications but it does not consider the imbalance of labelse. Including information of weight of attributes in the genetic codes is a feasible approach.

data splitting strategy

The data splitting strategy implemented in this paper effectively mitigates the problem of overfitting. By selecting the individual with the best quality on the unseen set, the generalizability of the solution is improved. However, the GA with this strategy did not show better classifications on test images. This is either due to the population's less adaptive to the data or the imbalance of the CelebA dataset. The former possibility is ruled out because the overall validation error of the solution of GA3 was lower than GA2-100. A better optimization solution might be achieved if external data were used.

aggregation of predictions

Even though the usage of weight does not lead to better overall predictions of facial attributes, it can be used in applications to allow faster processing by aggregating predictions of different transformations one by one. For example, a confident prediction can be obtained firstly on aligned faces and presented to users immediately. While the users are using the application, the back-end model updates the prediction by adding augmentation techniques according to the order of weights. This is another motivation to use weight in GA5.

Chapter 6

Conclusion and Future Works

6.1 Conclusion

In this paper, I implemented five genetic algorithms to search for the best combination of transformations from 162 AFFACT transformations that lead to better classification results. While the overall prediction is not improved, the solutions discovered by GAs have achieved lower error rates on many attributes. More importantly, the solutions have considerably lower number of transformations compared to the 162 transformations used in the AFFACT paper.

In addition, a GA is employed to perform optimization on a larger search space. Although the solutions do not show any improvement, it decreases the number of transformations

6.2 Future works

GA5 did not complete search for all attributes because of the lengthy processing time. If given more time, GA5 might have obtained better results.

During optimization, weights will be assigned to transformations by producing duplicates through crossover, even if the GA does not attach a weighting factor to a transformation. It is worth to research on how the optimization will behave if duplicate transformations are removed.

In this research, bounding boxes are estimated based on hand-labeled. Differently, automatic bounding box detector is used in the AFFACT paper. This is the reason why the baseline-T in this paper is not comparable to the baseline reported in the AFFACT paper. Results might be improved if bounding box detector is used in the optimization.

Appendix A

Attachements

The best combination of transformations

transformation
(-10, 0, 10, 1.0, 0)
(0, 10, 10, 1.1, 0)
(-10, -10, 10, 1.0, 1)
(-10, 10, 0, 0.9, 0)
(0, 10, 0, 0.9, 0)
(-10, -10, 10, 1.1, 0)
(10, 10, 0, 1.1, 1)
(10 ,10, 10, 0.9, 1)
(10, 0, 10, 0.9, 1)
(0, -10, 10, 0.9, 0)
(10, 10, 0, 0.9, 1)
(0, 0, 10, 1.1, 0)
(0, -10, 0, 1.1, 0)
(-10, 0, 10, 0.9, 1)
(-10, 0, 0, 1.0, 1)
(-10, -10, 0, 0.9, 1)
(10, -10, 10, 0.9, 0)
(-10, -10, 0, 1.1, 0)

Table A.1: SOLUTION OF GA1 ON OVERALL METRIC.

weight	transformation
11.3%	(-10, -10, 10, 1.0, 1)
9.9%	(-10, 10, 0, 0.9, 0)
7.0%	(-10, 0, 10, 0.9, 1)
7.0%	(0, 0, 1.0, 0.9, 1)
5.6%	(-10, -10, 0, 1.0, 1)
5.6%	(10, -10, 10, 1.0, 0)
5.6%	(-10, 0, 0, 0.9, 1)
5.6%	(0, 10, 0, 1.1, 0)
5.6%	(0, -10, 0, 0.9, 0)
5.6%	(-10, 0, 0, 0.9, 0)
5.6%	(-10, 0, 0, 1.1, 0)
4.2%	(-10, -10, 10, 1.1, 1)
4.2%	(10, 0, 0, 0.9, 1)
4.2%	(0, -10, 10, 1.0, 0)
4.2%	(0, 10, 10, 1.0, 0)
2.8%	(0, 10, 10, 1.1, 0)
2.8%	(10, -10, 0, 0.9, 1)
2.8%	(10, -10, 10, 1.0, 1)

Table A.2: SOLUTION OF GA2-30 ON OVERALL METRIC. Weights are translated into ratio in percentage.

weight	transformation
14.6%	(10, 10, 0, 0.9, 0)
12.5%	(-10, 0, 0, 0.9, 1)
12.5%	(0, 10, 0, 1.1, 0)
8.3%	(0, 0, 0, 0.9, 1)
8.3%	(-10, 0, 10, 1.1, 0)
6.2%	(0, 10, 10, 1.0, 1)
6.2%	(-10, -10, 10, 1.0, 1)
6.2%	(0, -10, 10, 1.0, 1)
6.2%	(-10, 10, 0, 1.1, 0)
4.2%	(10, 10, 0, 1.0, 1)
4.2%	(10, -10, 0, 1.1, 1)
4.2%	(10, 10, 0, 0.9, 1)
4.2%	(10, 10, 10, 0.9, 0)
2.1%	(10, 0, 0, 0.9, 1)

Table A.3: SOLUTION OF GA2-100 ON OVERALL METRIC. Weights are translated into ratio in percentage.

weight	transformation
13.6%	(10, 10, 0, 0.9, 0)
6.8&	(10, 0, 0, 1.1, 1)
6.8%	(10, 10, 10, 1.1, 1)
6.8%	(0, 10, 10, 1.1, 1)
6.8%	(10, 0, 0, 1.0, 1)
6.8%	(10, 10, 0, 1.1, 1)
6.8%	(-10, 10, 0, 0.9, 0)
5.1%	(10, 10, 10, 1.1, 0)
5.1%	(-10, 0, 0, 1.0, 1)
5.1%	(0, -10, 10, 1.0, 0)
5.1%	(0, -10, 10, 1.0, 1)
5.1%	(-10, 0, 10, 1.0, 0)
5.1%	(0, ,0, 0, 1.1, 0)
5.1%	(10, 0, 10, 0.9, 1)
3.4%	(0 ,0 ,10, 1.1, 0)
3.4%	(-10, 0, 0, 0.9, 1)
3.4%	(10, 10, 10, 1.0 ,1)

Table A.4: SOLUTION OF GA3 ON OVERALL METRIC. Weights are translated into ratio in percentage.

transformation
(0, 0, 10, 1.1, 1)
(0, 10, 0, 1.1, 0)
(0, -10, 10, 1.0, 0)*
(0, -10, 10, 1.0, 0)*
(-10, 10, 10, 1.1, 1)
(-10, 10, 10, 1.0, 1)
(-10, 10, 0, 0.9, 1)
(-10, 10, 0, 0.9, 0)
(-10, 10, 0, 1.1, 1)
(10, 10, 10, 1.0, 1)
(10, 10, 10, 1.0, 0)
(10, -10, 10, 0.9, 0)
(10, -10, 0, 0.9, 1)
(10, 0 ,0, 1.0, 0)
(-10, 10, 0, 1.0, 1)
(-10, 0, 0, 0.9, 1)
(-10, 0, 0, 0.9, 0)

Table A.5: SOLUTION OF GA4 ON OVERALL METRIC. The starred transformations refer to duplicates

weight	transformation
14.7%	(-16, -3, -15, 1.3, 0)
11.8%	(9, 4, -19, 0.9, 0)
11.8%	(-15, -10, 0, 1.0, 0)
8.8%	(6,-8,-6, 0.7,1)
8.8%	(16, -3, -20, 1.2, 1)
8.8%	(-2, 15, -17, 1.0, 0)
8.8%	(9, -1, -19, 1.3, 1)
8.8%	(7, -11, 10, 0.8, 1)
5.9%	(-20, -8, -15, 1.1, 0)
5.9%	(-15, -6, 14, 1.2, 0)
5.9%	(20, -7, 20, 0.7, 0)

Table A.6: SOLUTION OF GA5 ON OVERALL METRIC. Weights are translated into ratio in percentage.

List of Figures

List of Tables

List of Listings

Bibliography

Anjos, A., Shafey, E., Wallace, R., Günther, M., Mccool, C., and Marcel, S. (2012). Bob: a free signal processing and machine learning toolbox for researchers.

Baker, J. E. (1985). Adaptive selection methods for genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and their applications*, volume 101, page 111. Hillsdale, New Jersey.

Baluja, S. and Caruana, R. (1995). Removing the genetics from the standard genetic algorithm. In *Machine Learning Proceedings 1995*, pages 38–46. Elsevier.

De Jong, K. A. and Spears, W. M. (1990). An analysis of the interacting roles of population size and crossover in genetic algorithms. In *International Conference on Parallel Problem Solving from Nature*, pages 38–47. Springer.

Ehrlich, M., Shields, T. J., Almaev, T., and Amer, M. R. (2016). Facial attributes classification using multi-task representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 47–55.

Goldberg, D. (1988). Genetic algorithms in search, optimization, and machine learning. *Ethnographic Praxis in Industry Conference Proceedings*, 9(2).

Günther, M., Rozsa, A., and Boult, T. E. (2017). Affact: Alignment-free facial attribute classification technique. In *2017 IEEE International Joint Conference on Biometrics (IJCB)*, pages 90–99. IEEE.

Hasançebi, O. and Erbatur, F. (2000). Evaluation of crossover techniques in genetic algorithm based optimum structural design. *Computers & Structures*, 78(1-3):435–448.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Holland, J. (1975). Adaptation in natural and artificial systems. *University of Michigan Press*.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Kumar, N., Belhumeur, P., and Nayar, S. (2008). Facetracer: A search engine for large collections of images with faces. In *European conference on computer vision*, pages 340–353. Springer.

Kumar, N., Berg, A., Belhumeur, P. N., and Nayar, S. (2011). Describable visual attributes for face verification and image search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(10):1962–1977.

Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

Miller, B. and Goldberg, D. (1995). Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, 9.

Perez, F., Vasconcelos, C., Avila, S., and Valle, E. (2018). Data augmentation for skin lesion analysis. In *OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis*, pages 303–311. Springer.

Radosavovic, I., Dollár, P., Girshick, R., Gkioxari, G., and He, K. (2018). Data distillation: Towards omni-supervised learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4119–4128.

Rudd, E. M., Günther, M., and Boult, T. E. (2016). Moon: A mixed objective optimization network for the recognition of facial attributes. In *European Conference on Computer Vision*, pages 19–35. Springer.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48.

Shukla, A., Pandey, H. M., and Mehrotra, D. (2015). Comparative review of selection techniques in genetic algorithm. In *2015 international conference on futuristic trends on computational analysis and knowledge management (ABLAZE)*, pages 515–519. IEEE.

Siddiquie, B., Feris, R. S., and Davis, L. S. (2011). Image ranking and retrieval based on multi-attribute queries. In *CVPR 2011*, pages 801–808. IEEE.

Simon, D. (2013). *Evolutionary optimization algorithms*. John Wiley & Sons.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Sloss, A. N. and Gustafson, S. (2019). 2019 evolutionary algorithms review.

Srinivas, M. and Patnaik, L. M. (1994). Genetic algorithms: A survey. *computer*, 27(6):17–26.

Thom, N. and Hand, E. M. (2020). Facial attribute recognition: A survey. *Computer Vision: A Reference Guide*, pages 1–13.

Wang, J., Cheng, Y., and Feris, R. S. (2016). Walk and learn: Facial attribute representation learning from egocentric video and contextual data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2295–2304.

Whitley, D., Chicano, F., Ochoa, G., Sutton, A., and Tinós, R. (2019). Next generation genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1113–1136.

Zhong, J., Hu, X., Zhang, J., and Gu, M. (2005). Comparison of performance between different selection strategies on simple genetic algorithms. In *International conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce (CIMCA-IAWTIC'06)*, volume 2, pages 1115–1121. IEEE.