

Improved Face Recognition Through Image Perturbation

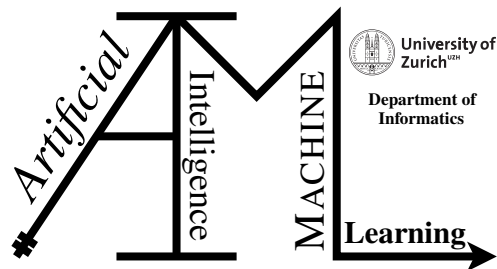
Master's Thesis

Yun Wang

18-745-232

Submitted on
October 27 2021

Thesis Supervisor
Prof. Dr. Manuel Günther



Master's Thesis

Author: Yun Wang, yun.wang@uzh.ch

Project period: 28th. April 2021 - 27th. October 2021

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

Acknowledgements

I would like to thank Prof. Dr. Manuel Günther who is the supervisor of my master's thesis. He guided me all the way with his professional knowledge and patience during the work of this thesis. He provided me with clear road map in the thesis project, and each time I came across with confusions during experiments, he always pointed out the right directions.

Thanks to the spirit of sharing in Computer Science community, I, as a previous student of chemistry and physics, can quickly grasp the knowledge and programming skills I need. I appreciate the scientists, researchers and engineers who bring Computer Science and Internet technology to the world. To extent the appreciation, I thank all scientists, researchers and engineers who work for the good.

I also would like to appreciate UZH for the resource given to me. Last but not the least, I thank my family for their support and for the joy they bring to me.

Abstract

Face alignment is considered a crucial process in the pipeline of face recognition. However, to perform face alignment precisely is not an easy job. [Günther et al. \(2017\)](#) proposed AFFACT technique to train a network that is robust to facial misalignment, while the original work is for attribute classification problem. In this thesis, I apply the AFFACT technique to train a target face recognition model on AFFACT-perturbed images, and a baseline model on aligned images. The results of the two models are compared to show how AFFACT technique works for face recognition model. Furthermore, I put the emphasis on studying the image perturbation in the test stage. Single perturbations are experimented and combination of perturbations are searched by different optimization algorithms. To combine different perturbations, majority voting for the final predicted labels and averaging the predictions of probability are often used in different works. Combining the image similarities is also mentioned in previous research. In this thesis, I test the combination of perturbations by averaging the (weighted) features extracted from the perturbed images.

Zusammenfassung

Die Gesichtsausrichtung gilt als ein entscheidender Prozess in der Pipeline der Gesichtserkennung. Die genaue Gesichtsausrichtung ist jedoch keine leichte Aufgabe. [Günther et al. \(2017\)](#) schlugen die AFFACT-Technik vor, um ein Netzwerk zu trainieren, das robust gegenüber Gesichtsfehlausrichtungen ist, während die ursprüngliche Arbeit für das Attributklassifizierungsproblem gilt. In dieser Arbeit wende ich die AFFACT-Technik an, um ein Zielgesichtserkennungsmodell auf AFFACT-gestörten Bildern und ein Basismodell auf ausgerichteten Bildern zu trainieren. Die Ergebnisse der beiden Modelle werden verglichen, um zu zeigen, wie die AFFACT-Technik für das Gesichtserkennungsmodell funktioniert. Darüber hinaus lege ich den Schwerpunkt auf die Untersuchung der Bildstörung in der Testphase. Einzelne Störungen werden experimentiert und Kombinationen von Störungen werden durch verschiedene Optimierungsalgorithmen gesucht. Um verschiedene Störungen zu kombinieren, werden in verschiedenen Arbeiten häufig Mehrheitsentscheidungen für die endgültigen vorhergesagten Labels und Mittelung der Wahrscheinlichkeitsvorhersagen verwendet. Das Kombinieren der Bildähnlichkeiten wird auch in früheren Forschungen erwähnt. In dieser Arbeit teste ich die Kombination von Störungen durch Mittelung der (gewichteten) Merkmale, die aus den gestörten Bildern extrahiert wurden.

Contents

1	Introduction	1
2	Related Work	3
2.1	Image Perturbation	3
2.2	Image Perturbation for Face Recognition	6
2.3	Test Time Image Perturbation	8
2.4	Optimization Algorithm	10
3	Dataset	13
3.1	VGGFace2	13
3.2	LFW	13
4	Model Training	15
4.1	Architecture of the Neural Network	15
4.2	Procedure of AFFACT	15
4.3	Image Preprocessing	17
4.3.1	Image Perturbation	17
4.3.2	Normalization	18
4.4	Hyperparameters	18
4.5	Metrics	22
4.5.1	Metrics in Training	22
4.5.2	Metrics in validation	22
5	Perturbation in Test Stage	25
5.1	Single Perturbations	25
5.1.1	Perturbation Space	25
5.1.2	Experiment Process	25
5.1.3	Results	27
5.2	Non-weighted Combination of Perturbations	29
5.2.1	Greedy Algorithm	29
5.2.2	Results	29
5.3	Weighted Combination of Perturbations	31
5.3.1	General SimpleGA	31
5.3.2	SimpleGA Variant Approach	32
5.3.3	Results	37
5.4	Results Comparison and Test on View 2 Dataset	38

6	Discussion	43
7	Conclusion	49
A	Attachments	51

Introduction

In 2012, Deep Convolutional Neural Network (DCNN) demonstrated an outstanding performance in image recognition when Krizhevsky and his colleges won the competition of ImageNet (Krizhevsky et al., 2012). In 2014, DeepFace (Taigman et al., 2014) manifested the promising future of DCNN for face recognition. Since then, the effort to improve the face recognition accuracy mainly focuses on three aspects:

The first is to design the architecture of DCNN for better feature extraction. Many DCNN architectures were proposed, such as VGG (Simonyan and Zisserman, 2015), GoogleNet (Szegedy et al., 2015), ResNet (He et al., 2016), SENet (Hu et al., 2018), etc.

The second is to create better loss function to enlarge the inter-person differences and in the meantime reduce the intra-person variance. Different loss functions were designed, including the Euclidean-distance-based loss such as triplet loss (Schroff et al., 2015), center loss (Wen et al., 2016), range loss (Zhang et al., 2017), etc., and cosine-margin-based loss such as Cosface loss (Wang et al., 2018), Arcface loss (Deng et al., 2019a), etc.

The third is to collect higher quality and larger amount of annotated data, as research has revealed that the performance of deep learning model strongly relies on the size of training dataset (Zhou et al., 2015; Hestness et al., 2017). Deepface from Facebook was trained on 4 million images of 4000 people (Taigman et al., 2014), and FaceNet from Google was trained on an even larger face database, including 200 million images of 3 million people (Schroff et al., 2015). However, these large scale databases are not public. Some other databases, such as VGGFace2 and MS-Celeb, have been available publicly only for some time. Since annotating large scale of face data is tedious and laborious, data augmentation emerged as another research direction in face recognition. Data augmentation is transformation or perturbation technique applied on training or test images without changing the labels. Many research show that data augmentation can enlarge the data and advance the performance of networks (Howard, 2013a; Lv et al., 2016). Most of the studies use different data transformations or perturbations to conduct data augmentation in the model training stage, while a few studies were in the test stage.

The general procedure of face recognition consists of (1) face alignment; (2) feature extraction; and (3) face recognition. Face alignment was considered a crucial procedure for face recognition in the last few decades. Many face recognition works were based on the face alignment (Huang et al., 2007; Nguyen and Bai, 2010; Taigman et al., 2014), and some works were on exploring the approach to achieve high-quality face alignment (Hasan and Pal, 2011; Xiong and De la Torre, 2013; Deng et al., 2019b). Wang et al. (2014) and Jin and Tan (2016) summarized the research on the methods of face alignment and its difficulties in the unconstrained environment in their surveys.

Face alignment requires the accurate detection of facial landmarks, while to obtain precise coordinates of landmarks is a tough job. It is common that the detected landmarks differ from the ground truth, causing the misalignment of faces. To mitigate the impact of misalignment of

faces, [Lv et al. \(2016\)](#) presented a landmark perturbation method to augment training data and introduce misalignment into the model training. [Günther et al. \(2017\)](#) proposed Alignment-Free-Facial-Attribute-Classification-Technique (AFFACT). By using the AFFACT technique to preprocess input images during model training, the obtained model can still perform well on misaligned images.

Face recognition can be categorized into two different scenarios, one is face verification and the other is face identification. In both scenarios, a face recognition model is trained, then the obtained model is used to extract the deep features of face images. In the task of face verification, images pass through the model to obtain the features of the images, then to verify whether two of the images represent the same person by using the cosine or Euclidean distance of the features. Whereas, face identification requires to compare the distances between the probe face image and the other images in the database, then determine the identity of the probe face.

In this thesis, I apply AFFACT technique, which was originally for facial attribute classification, to face recognition. A target face recognition model is trained with the AFFACT technique which is slightly modified to suit the face recognition situation, and a baseline model is trained on the aligned images. In the test stage, face verification is performed with the two obtained models. Verification performance of the two models is compared to show how AFFACT technique works for face recognition. Furthermore, image perturbation in the test stage is studied. Single perturbations are experimented and combination of perturbations are searched by different optimization algorithms. The experiment results reveal that the face recognition model trained with AFFACT technique is more robust to facial variation than the baseline model, and the proper combination of perturbations in the test time can boost the performance of both AFFACT and baseline model. To combine different perturbations, majority voting for the final predicted labels and averaging the predictions of probability are often used in different works. Combining the image similarities is also mentioned in previous research. The contribution of my work is that I test the combination of perturbations by averaging the (weighted) features extracted from the perturbed images, and I adopt different image transformations including scaling, rotation and shifting for each single perturbation.

The main content of this thesis consists of six chapters. In Chapter 2, I will introduce some related work. I will give an overview about the image transformations I use in this thesis, and introduce some related work about the image perturbation for face recognition. I will also refer to some works about image perturbation in the test time. In addition, some works that adopted optimization algorithms to search different transformations for data augmentation are also introduced. In Chapter 3, I will briefly introduce the datasets I use in my work. Then, I will introduce how I train the baseline model and face recognition model with AFFACT technique in Chapter 4. In Chapter 5, I will describe the test-time perturbation experiments. Next, I will discuss some limitations and possible improvements. And In the last chapter, I will summarize the results and propose some possible directions to improve the work.

Related Work

2.1 Image Perturbation

The generic image perturbation or transformation can be categorized into geometric transformation and photometric transformation. Geometric transformation maps pixel values of an image to new positions to change the geometry of an image. It includes elastic distortion, lens distortion, rotation, shifting, scaling, cropping, flipping, etc. Photometric transformation changes the RGB channels by altering the pixel colors. This kind of transformation includes grayscaling, color jittering, noise adding, contrast changing, erasing, sharpening, blurring, etc. For image classification, object detection, face recognition and some other object recognition related tasks in which objects of the same class may have varying appearance, these perturbations can not only enrich the training data, but also reduce overfitting and improve the performance of models. The geometric transformation can reduce the model's dependence on an object's geometric properties, such as location, angle, shape, size, etc. The photometric transformation can relieve the model from relying on color, illumination, sharpness and other photometric properties.

In this thesis, the image perturbation focuses on flipping, cropping, scaling, rotation, shifting, and Gaussian blurring.

Flipping Flipping generally includes horizontal flipping and vertical flipping (Figure 2.1). Vertical flipping is rarely seen in image classification and face recognition, while horizontal flipping is commonly adopted to augment the data in image classification (Paulin et al., 2014; Krizhevsky et al., 2012; Howard, 2013b). Many works show horizontal flipping improves model's performance. However, Yang and Patras (Yang and Patras, 2015) studied horizontal flipping in object detection tasks and an interesting finding was reported: the state-of-the-art methods predicted better on original images than their mirrored ones, even though these state-of-the-art methods are trained on augmented data with mirrored images. Specially in face recognition, horizontally mirroring faces might not be identity preserving, for instance, a man has a birthmark on his left face, by using horizontal flipping, a mirrored birthmark appears on the right face. In reality, two faces with landmarks on different sides can't be the same person. Besides, faces are not perfectly symmetric. However, horizontal flipping is a common approach for training face recognition networks.

Cropping Cropping is to remove the unwanted area of an image. By defining a crop size and a crop location, the other area are removed from the image. Figure 2.2 illustrates the visual effect of cropping.

Image cropping can help the models learn to identify the object with the subsets of it. The commonly used cropping is Center Crop whose location is in the center of the image, and Five

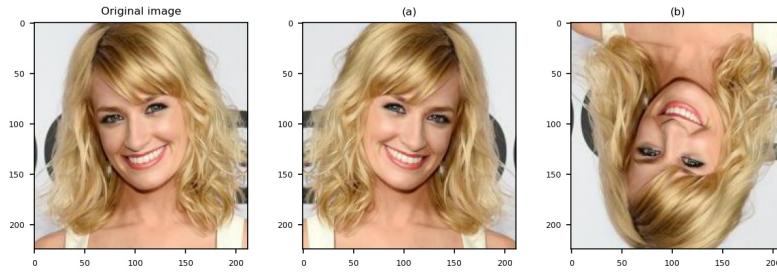


Figure 2.1: Illustration of Flipping. (a) is horizontal flipped image, and (b) is the vertical flipped image.

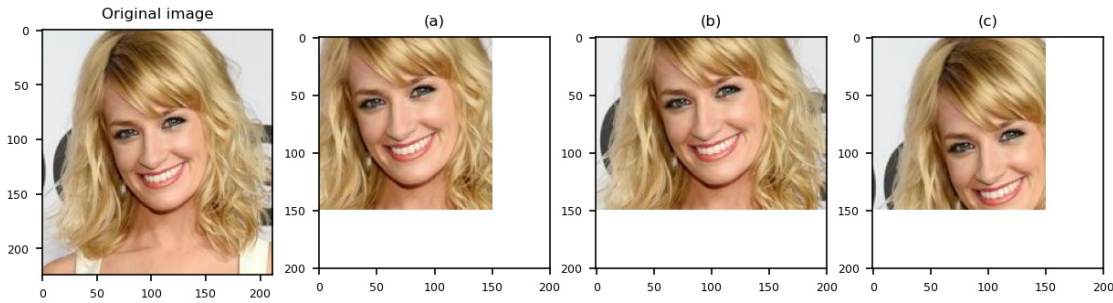


Figure 2.2: Illustration of Cropping. (a) and (b) show the center crop with different crop size. (d) is a corner crop.

Crop whose locations are at the four corners and the center, which is only used in validation. Random Located Crop is also used in research to increase training data. For example, [Krizhevsky et al. \(2012\)](#) took randomly located crops and their horizontal reflections to enlarge the training data for combating the overfitting of the neural network. In addition, the five crops and their horizontal flipped ones are used for the test in their work.

Scaling Scaling is to resize an image. The same object may have different sizes in images because of different shooting distances. Scaling or rescaling input helps the models not to rely on sizes. [Paulin et al. \(2014\)](#), [Sun et al. \(2014\)](#), and [Mash et al. \(2016\)](#) adopted scaling to augment training data in their works.

Figure 2.3 illustrates the visual effect of scaling: Figure 2.3(a) is the upscaled image cropped with the original image size; Figure 2.3(b) is the downscaled images and the black area on the image shows the pixels are missed. A constant value or values generated by certain interpolation methods can be filled in the missing pixels. Figure 2.3(c) shows the interpolation effect. The interpolation is implemented by the function `extrapolate_mask` provided by bob library ([Anjos et al., 2017](#)).

Rotation Rotation is to rotate an image with a certain angle as it is shown in Figure 2.4. Objects exhibit different angles in images and faces are often not aligned. Rotation is used as a data augmentation method to mitigate the misalignment ([Mash et al., 2016](#); [Xie and Tu, 2017](#)).

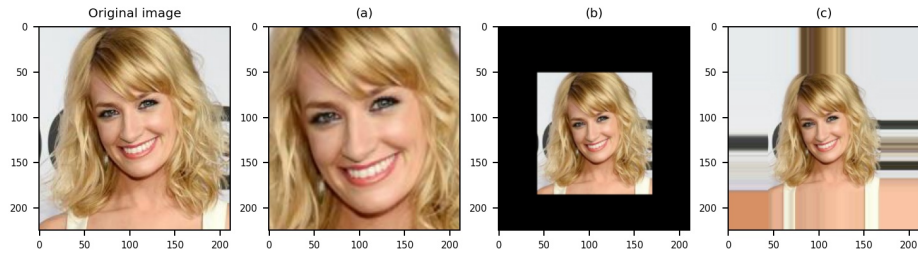


Figure 2.3: Illustration of Scaling. (a) is an upscaled image cropped with the original image size. (b) is a downscaled image. (c) is the downscaled image with interpolation.

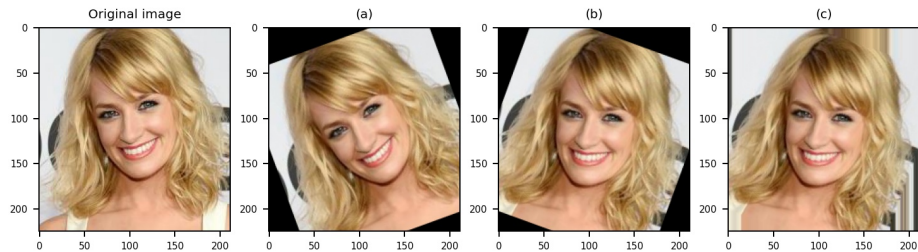


Figure 2.4: Illustration of Rotation. (a) is a counter clock-wise rotated image. (b) is a clock-wise rotated image. (c) is the clock-wise rotated image with interpolation.

Shifting Shifting is to move the image in different directions, so the position of an object in an image changes (Figure 2.5). By varying the object position with shifting, positional bias can be reduced. Shifting is similar as cropping, and they basically have the same characteristics.

Gaussian Blurring Gaussian blurring, also known as Gaussian smoothing, is used to blur an image and remove noise by applying 2D Gaussian function to each pixel. The visual effect of Gaussian blurring is shown in Figure 2.6. The standard deviation of the Gaussian distribution is important, when the standard deviation is larger, the Gaussian distribution has a larger peak, leading to greater blurring in the image. Recognizing a blurred image is challenging, however blurred images are commonly shot when the object is moving in high speed or the camera is out of focus. Although image blurring is widely studied (Fiche et al., 2010; Balas et al., 2018), using it as a data augmentation method is rarely seen in research.

Taylor and Nitschke (2017) evaluated different generic data transformations in image classification task on Caltech-101 database. They added the transformed images into training data to train neural networks, then compared the Top-1 and Top-5 classification accuracy with the baseline whose training data did not include the transformed images. The generic image transformations they evaluated were flipping, rotation, cropping, color jittering, edge enhancement and PCA. They found that by enlarging the training data with transformed images, model's performance was improved. The geometric image transformations outperformed the photometric transformations when training on the Caltech-101 dataset. In addition, cropping improved the performance

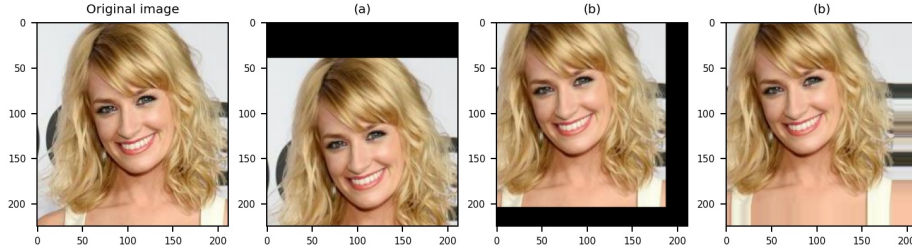


Figure 2.5: Illustration of Shift. (a) is a vertical shifted image. (b) is shifted along both axis. (c) is the image (b) with interpolation.

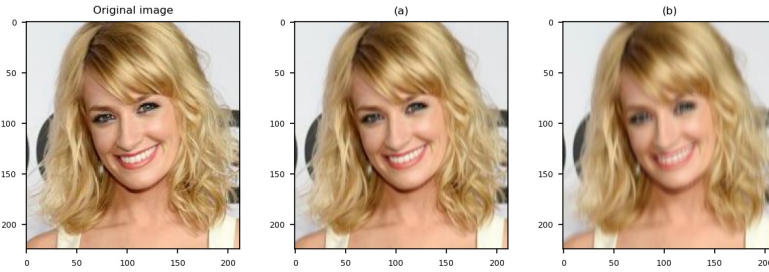


Figure 2.6: Illustration of Blurring. (a) and (b) are blurred images with different Gaussian filters.

the most, increasing the top-1 accuracy by 13.82%.

2.2 Image Perturbation for Face Recognition

Generic image perturbation methods mentioned above are also commonly used for data augmentation in face recognition.

Xu et al. (2014) proposed a novel scheme to generate mirror images. After augmenting the training samples with their mirror versions, they tested the performance of several representation-based face recognition models on FERET, Yale B and ORL databases. The results informed that the models trained on augmented data produced lower error rate than models trained on original images. They further studied the rationales behind the improvement in the paper.

Lv et al. (2016) came to similar conclusion: models trained on augmented data by horizontal flipping can extract features that are more robust to misalignment. Except for transformation of the horizontal flipping, they explored and experimented 6 more data perturbation methods. One perturbation method combines contrast adjustment, Gaussian blurring, noise injection and color casting, which improves the model's performance as well. In their research, the transformation that achieves the best verification accuracy is the landmark perturbation. They first detected the face area by a Viola-Jones face detector (Viola and Jones, 2001), and adopted Supervised Descent Method (Xiong and De la Torre, 2013) to detect the facial landmarks including the two eyes' centers and mouth center. Then, they randomly perturbed the positions of the landmarks with $P_i^* = P_i + \gamma$, where P_i^* is the perturbed position, P_i is the detected position, and the perturbation parameter is Gaussian distributed $\gamma \sim N(\mu, \Sigma)$. Here, they used Gaussian distribution to model

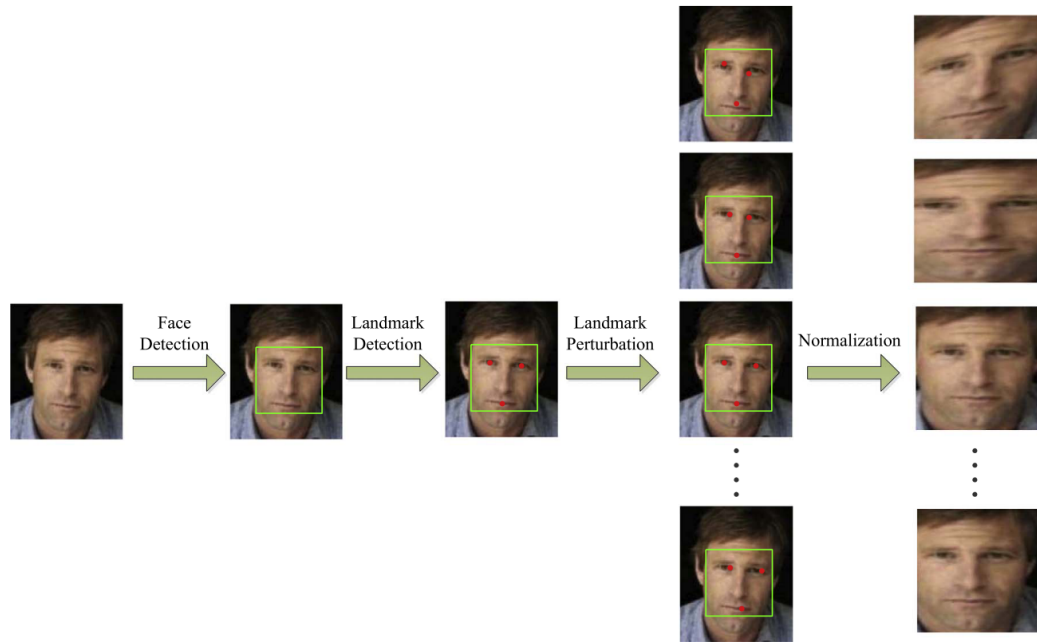


Figure 2.7: Process of Landmark Perturbation (Wang et al., 2019).

the perturbation range for the fact that the alignment error satisfies a Gaussian distribution (Shan et al., 2004). The process of landmark perturbation for one image is shown in Figure 2.7. A large number of misaligned training images can be generated by applying sets of landmark perturbation. They reported the landmark perturbation method reduced the face verification error rate on LFW by 50.6% compared with the baseline of no augmentation.

Günther et al. (2017) presented Alignment-Free-Facial-Attribute-Technique (AFFACT). The original work is for facial attribute classification, not face recognition. However, good performance of the classification indicates good feature extraction of facial attributes, which are also necessary information for face recognition. In their research, they perturbed training images with random scaling, rotation, shifting, horizontal flipping and Gaussian blurring. Except for horizontal flipping which is performed with probability of 0.5, the other perturbation ranges are Gaussian distributed. Figure 2.8 shows the visual effect of AFFACT perturbation. The obtained AFFACT network is impressively robust to misalignment. It can achieve state-of-the-art results even when the faces are not aligned.

In addition to generic image transformations, other face specific data transformations such as makeup transfer, accessory synthesis, pose synthesis and expression synthesis are also explored in face recognition. Lv et al. (2016) tried glasses synthesis and pose synthesis in their research, and the face verification accuracy on LFW outperforms the baseline of no data augmentation. Masi et al. (2016) did ablation study among no augmentation, pose synthesis, 3D shape variations, and expression synthesis on WebFace database. With the augmented training data, performance of the face recognition model improves and pose synthesis contributes the most in the improvement.

Figure 2.9 is an overview of different face perturbations summarized by Wang et al. (2019). These image perturbations are generally performed in the training stage to augment training data for face recognition models. The effect of perturbations in test time for face recognition are not often studied. However, many works about image perturbation in the test time for image

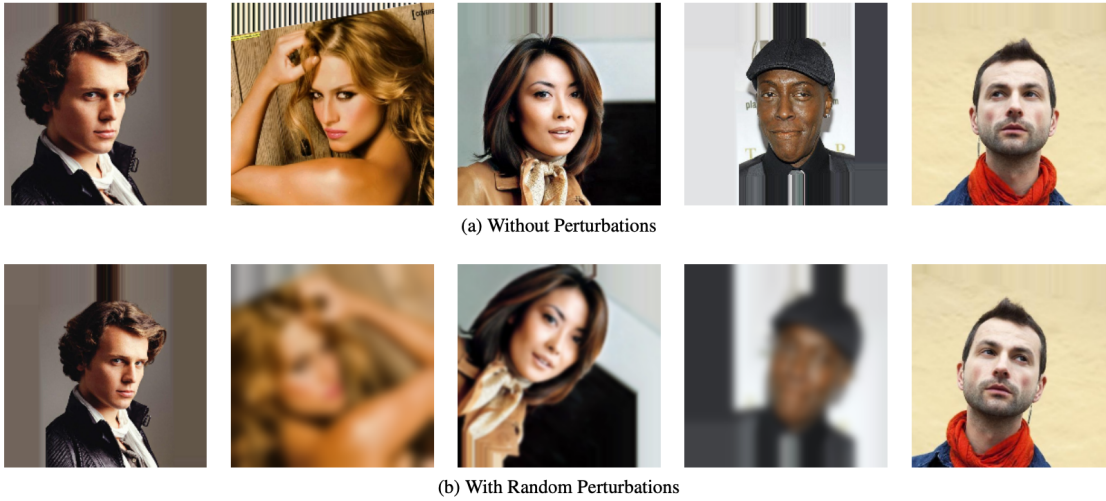


Figure 2.8: Random Perturbations on CeleA. (a) are the aligned images, (b) are AFFACT-perturbed images (Günther et al., 2017).

classification were presented.

2.3 Test Time Image Perturbation

Cropping and flipping are commonly used image perturbations in the test phase. Krizhevsky et al. (2012) generated five crops from the test images and produced the horizontal flipped images of these five crops. Then they input the 10 transformed images into the neural network, and the output of the last layer of the network is the predictions of probability. The mean-probability were obtained by averaging the 10 predictions of probability, and the predicted label was the category corresponding to the largest mean-probability.

Except for the commonly used test-time image transformations like cropping and flipping, the same data perturbation method used in the training stage is also directly adopted in the test time in some research. Qinghe et al. (2020) transformed each test image by the same data augmentation method used in training phase. Labels were predicted for each transformed image by the neural network, then the final label was chosen by majority voting. If a balanced vote appeared, the category with the highest probability was selected as the predicted label.

Masi et al. (2016) exploited pose synthesis in the test time for face recognition. They used pose synthesis images and in-plane aligned images in the test stage on IJB-A and LFW databases. The similarity of each image pair was calculated for both pose synthesis images and aligned images, and they designed a softmax operator to compute the fusion similarities to combine pose synthesis and aligned images. They found that the best ROC was obtained when combining both images.

Paulin et al. (2014) presented a conclusion that it is important to apply perturbation at both training and test time for the best performance. They generated different virtual samples for the test images, computed an independent score for each perturbed test sample, and tried to aggregate the scores by averaging, maximum selection and softmax aggregation. They found that applying perturbations on both training and test stages outperformed applying them on one of or neither of the two stages, and the softmax aggregation is better than averaging and maximum














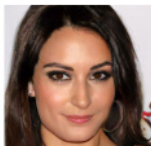



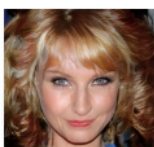
Generic Transformation	Geometric	 →  
	Photometric	 →  
Component Transformation	Hairstyle	 → 
	Makeup	 → 
	Accessory	 → 
Attribute Transformation	Pose	 → 
	Expression	 → 
	Age	 → 

Figure 2.9: An Overview of Face Transformations (Wang et al., 2019).

score selection in their cases.

2.4 Optimization Algorithm

There are various of image transformation methods, and the selection of an effective set of transformations is crucial. How to explore a satisfactory set of transformations for model training and testing (e.g. crop+scale)? How to select the set of transformation parameters (e.g. a set of standard deviation for Gaussian blurring)? Most works depend on domain knowledge and experience of the researchers. What if there is no prior experience? Then, the optimization algorithm is a possible solution.

An optimization algorithm is a procedure that iteratively compares various solutions until an optimal or a satisfactory one is found. There are generally two kinds of optimization algorithms: the deterministic algorithms and the heuristic algorithms. The former, such as greedy algorithm, gradient-based algorithms, etc., follows a precise sequence of actions. The latter always involves randomness, such as random walk, random forest, evolutionary algorithm (EA), etc. In this thesis, I focus on greedy algorithm and Genetic Algorithm that is a branch of EA.

Greedy Algorithm The greedy algorithm makes the optimal choice at each step in order to solve the entire problem. This strategy is simple and successful in some problems. However, as it only makes the decision at each step based on the information of that step, it may not get the optimal solution for the entire problem.

Some researchers used greedy algorithm to search satisfactory image perturbations for the image classification task. [Paulin et al. \(2014\)](#) proposed Image Transformation Pursuit (ITP) approach to select a set of transformations from several families of transformations. The families included cropping, homograph, scaling, colorimetry, JPEG compression and rotation. By assigning different transformation parameters, the seven families were extended to 40 transformations in total. ITP was designed to select transformation or combination of them from the 40 transformations. It greedily and iteratively choose one perturbation at a time. [Molchanov et al. \(2020\)](#) introduced a similar method, the Greedy Policy Search (GPS) method, to learn a policy for data augmentation in the test time. It starts with an empty policy set, then iteratively searches for the sub-policy that can boost the performance the most when adding it to the current policy set. The pool of candidate sub-policy has approximately 1000 sub-policies. The max iteration is controlled by the desired length of the policy set.

Genetic Algorithm Genetic Algorithm (GA) is a metaheuristic optimization algorithm. It is designed to simulate a biological reproduction process. The common components of GA are: (1) a population of chromosomes; (2) a selection of individuals whose chromosomes will be reproduced; (3) a fitness function for optimization; (4) crossover to reproduce the next generation; and (5) the random mutation of chromosomes in the next generation.

Using GA to augment data is emerging in recent years. [Correia et al. \(2019\)](#) employed GA to augment training data in the task of face detection. They presented an approach that automatically construct new face instances by GA by recombining facial elementary parts from different faces. Compared with other standard data augmentation techniques, the GA approach can enhance the face detection performance. [Terauchi and Mori \(2021\)](#) proposed Thermodynamic Genetic Algorithm based AutoAugment (TDGA AutoAugment). Their motivation is to extend the simple genetic algorithm (SimpleGA), as SimpleGA suffers from the problem of premature convergence of population, leading to the lack of population diversity in an early stage. By applying thermodynamic selection rule to SimpleGA, population diversity can be improved and fitness in evaluation space is maintained. Their experiment result shows the method selects various data

augmentation operations and the resulted performance is competitive. In addition, this method is faster than Fast AutoAugment ([Lim et al., 2019](#)).

Dataset

Several face databases are often used for face recognition, such as CelebA (Liu et al., 2015), CASIA-Webface (Yi et al., 2014), MS-Celeb-1M (Guo et al., 2016), Megaface (Kemelmacher et al., 2016), VGGFace (Parkhi et al., 2015), VGGFace2 (Cao et al., 2018), LFW (Huang et al., 2007), IJB-A/B/C (Klare et al., 2015; Whitelam et al., 2017; Maze et al., 2018), PaSC (Beveridge et al., 2013), etc. Among these databases, LFW, IJB-A/B/C and PaSC are commonly used for model testing, while the others are used during model training.

In this thesis, I use VGGFace2 database to train face recognition models and LFW to test the verification performance of the obtained models.

3.1 VGGFace2

VGGFace2 is a large-scale face database which contains 9131 identities and 3.31 million images in total. Each identity has about 362 images on average. Images have large variations in age, ethnicity, profession, pose, resolution and illumination. Each image is labeled with coordinates of bounding box and landmarks. The dataset contains the name of each images and is split to a training set and a test set. The test set has 500 identities and 170,000 images, all other images belonging to the other 8631 identities are in the training set.

In this thesis, I use the VGGFace2 training set to train face recognition models by classifying the 8631 classes. As I would like to use the obtained model to test the face verification performance, I randomly generate 5000 positive pairs and 5000 negative pairs, that is 10k balanced pairs in total, from the VGGFace2 test set as the validation set. Here, the positive pair refers to the two images belonging to the same identity, and the negative pair refers to the two images belonging to different identities. All identities in the VGGFace2 test set are included when generating the positive and negative pairs.

3.2 LFW

In order to test the performance of the obtained face recognition models and to experiment image perturbations in the test stage, I use another image database, the LFW funneled, in which all face images are aligned with funneling. The database contains 13233 images that belong to 5749 people. The datasets of LFW have two forms: view 1 and view2. View 1 has a training set and a test set, which can be used in the development for training and validation. View 2 is a dataset generated for 10-fold cross validation and it is suggested to be used for the final report of performance.

The training set of view 1 has 2200 pairs with half positive and half negative. The test set of view 1 has 1000 pairs with half positive and half negative. The view 2 dataset has 6000 pairs, which are divided into 10 folds, and each fold contains 300 positive pairs and 300 negative pairs.

Since the positive and negative pairs are balanced, I directly use the metric accuracy to check the model's performance.

Model Training

Günther et al. (2017) proposed Alignment-Free Facial Attribute Classification Technique (AFFACT) to train the model for facial attribute classification. By using the AFFACT perturbation technique to preprocess images during model training, the obtained model is impressively robust to facial misalignment.

In this chapter, I apply the AFFACT technique to train a target model for face recognition, mean while I also train a baseline model on aligned face images for comparison.

4.1 Architecture of the Neural Network

The backbone architecture of the convolutional neural network I use is ResNet50, but with the last fully connected layer removed. So when using the pretrained ResNet50, the pretrained weights of this layer will not be loaded. In addition, I add 2 fully connected layers as the embedding layer and the classification layer. The size of last fully connected layer is 8631, equal to the number of classes in the training set of VGGFace2. The second to last layer is the embedding layer with 512 neurons, which output the features of the input image (Figure 4.1).

As the number of layers of the neural network is 51, the target model trained with AFFACT technique will be referred as ResNet51, and the baseline model trained on aligned images will be referred as ResNet51_align later in this Thesis.

4.2 Procedure of AFFACT

AFFACT is a data augmentation technique proposed by Günther et al. (2017), and the approach can be divided to several steps:

Step 1: get the bounding boxes of faces. The hand-labeled coordinates of landmarks, which include two eyes and two mouth corners, are used to compute the bounding boxes of faces. The coordinates of left eye is denoted as \vec{t}_{el} , the right eye \vec{t}_{er} , the left mouth corner \vec{t}_{ml} , and the right mouth corner \vec{t}_{mr} . The center of eyes and mouth can be calculated respectively.

$$\vec{t}_e = \frac{(\vec{t}_{el} + \vec{t}_{er})}{2}, \quad \vec{t}_m = \frac{(\vec{t}_{ml} + \vec{t}_{mr})}{2} \quad (4.1)$$

The vectors \vec{t}_{el} , \vec{t}_{er} and \vec{t}_e can also be noted as (x_{el}, y_{el}) , (x_{er}, y_{er}) and (x_e, y_e) , which will be used in the computation later. Then, the distance of eye center and mouth center d is obtained:

$$d = \|\vec{t}_e - \vec{t}_m\| \quad (4.2)$$

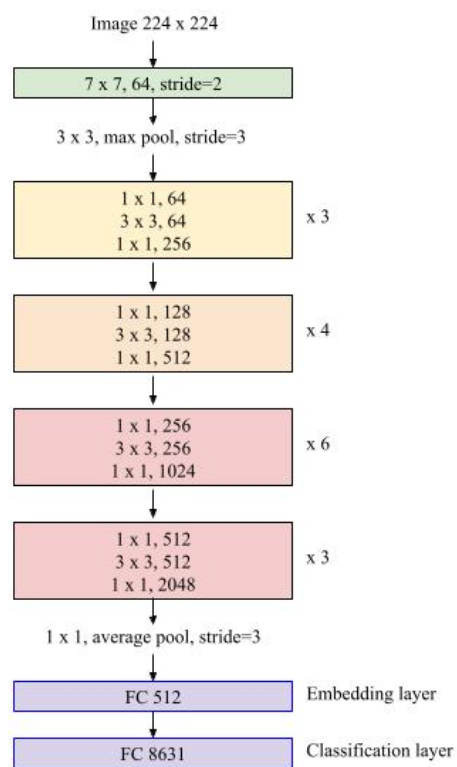


Figure 4.1: Architecture of ResNet51.

Based on the distance d , the width and height of the bounding box are estimated as $w = h = d \cdot 5.5$. And finally, the left, right, top and bottom coordinates of the bounding box (x_l, x_r, y_t, y_b) are calculated.

$$x_l = x_e - 0.5 \cdot w, \quad x_r = x_e + 0.5 \cdot w \quad (4.3)$$

$$y_t = y_e - 0.45 \cdot h, \quad y_b = y_e + 0.55 \cdot h \quad (4.4)$$

Step 2: calculate the original face angle and scale factor. The original face angle α_0 is calculated from the coordinates of left eye (x_{el}, y_{el}) and right eye (x_{er}, y_{er}) .

$$\alpha_0 = \arctan\left(\frac{y_{er} - y_{el}}{x_{er} - x_{el}}\right) \quad (4.5)$$

Suppose the crop size of the image is with width W and height H . Generally, the chosen width W and height H are equal. In the AFFACT paper, $W = H = 224$. The scale factor s_0 is estimated from the crop size and size of the bounding box:

$$s_0 = \frac{W}{w} = \frac{H}{h} \quad (4.6)$$

Step 3: apply perturbation of scaling, rotation, shifting, horizontal flipping and Gaussian blurring. The scaling parameter γ_s , the rotation parameter γ_a and the shifting parameter γ_x, γ_y are randomly drawn from Gaussian distributions with certain means and standard deviations. In the AFFACT paper, $\tilde{\gamma}_s \sim N(1, 0.1)$, $\tilde{\gamma}_a \sim N(0, 20)$, and $\tilde{\gamma}_x, \tilde{\gamma}_y \sim N(0, 0.05)$.

The rotation angle α , the scale s and the coordinates $(\tilde{x}_l, \tilde{x}_r, \tilde{y}_t, \tilde{y}_b)$ of the perturbed bounding box are calculated.

$$\alpha = \alpha_0 + \gamma_a, \quad s = s_0 \cdot \gamma_s \quad (4.7)$$

$$\tilde{x}_l = x_l + \gamma_x \cdot w, \quad \tilde{x}_r = x_r + \gamma_x \cdot w \quad (4.8)$$

$$\tilde{y}_t = y_t + \gamma_y \cdot w, \quad \tilde{y}_b = y_b + \gamma_y \cdot w \quad (4.9)$$

Meanwhile, the probability of horizontally flipping is 0.5, and a Gaussian filter with mean of zero and random standard deviation $\sigma \sim N(0, 3)$ is used to blur the image. By applying the AFFACT technique with these perturbation parameters, an image is randomly scaled, shifted, rotated, flipped, blurred and cropped with the crop size. In addition, interpolation (eg. Figure 2.4(c)) is applied whenever it is possible.

I'd like to add one more comment on the rotation angle. When the rotation parameter $\gamma_a = 0$ and the rotation angle is equal to the original face angle $\alpha = \alpha_0$, the face is aligned. When the rotation angle $\alpha = 0$, the image is not rotated, I denote it as non-aligned.

4.3 Image Preprocessing

4.3.1 Image Perturbation

To train the target model ResNet51 and the baseline model ResNet51_align, different perturbations are applied to preprocess the input images for training and validation. The detailed perturbation parameters are listed in Table 4.1.

To train ResNet51, the AFFACT perturbation is adopted with a slight modification on the training image data. The original AFFACT draws the scale parameter γ_s from a normal distribution

Model	Training Transformer	Validation Transformer	Best EER
ResNet51	$\tilde{\gamma}_s \sim N(1.2, 0.1)$ $\tilde{\gamma}_a \sim N(0, 20)$ $\tilde{\gamma}_x, \tilde{\gamma}_y \sim N(0, 0.05)$ <i>Horizontal Flip</i> $P = 0.5$ <i>Gaussian Filter</i> $\sim N(0, 3)$	$\gamma_s = 1.2$ $\alpha = 0 \rightarrow \text{not align}$ $(\gamma_x, \gamma_y) = (0, 0)$	0.0576
ResNet51_align	$\gamma_s = 1.2$ $\gamma_a = 0 \rightarrow \text{align}$ $(\gamma_x, \gamma_y) = (0, 0)$	$\gamma_s = 1.2$ $\gamma_a = 0 \rightarrow \text{align}$ $(\gamma_x, \gamma_y) = (0, 0)$	0.0562

Table 4.1: Training and Validation Transformers for ResNet51_align and ResNet51. Best EER is the minimal validation EER reached during the training.

with $mean = 1.0$ and $std = 0.1$. I modify the mean to 1.2, while the other perturbation parameters are the same as those in the AFFACT paper. The samples of training images are shown in Figure 4.2. The validation images are only scaled with $\gamma_s = 1.2$, and without rotation, shifting, flipping or blurring. The sample images of validation are shown in Figure 4.3.

To train ResNet51_align, the training and validation images are all aligned and scaled with $\gamma_s = 1.2$, and without other transformations. The sample images are shown in Figure 4.4.

Following the AFFACT paper, bob library (Anjos et al., 2017) is used to implement the perturbations. I adopted the **AFFACT transformer**¹ implemented by Yves Rutishauser and Noah Chavannes to perform the perturbation. By setting the perturbation parameters, different perturbation listed in the Table 4.1 can be implemented.

4.3.2 Normalization

After the images are perturbed, the three channels of the perturbed RGB images data are normalized with $mean = [0.485, 0.456, 0.406]$, $std = [0.229, 0.224, 0.225]$.

Here, the mean and standard deviation are the ones of the ImageNet instead of the ones of the VGGFace2 training data. Since I load the weights of the pretrained ResNet50 to initialize the model in the training and the ResNet50 was trained on ImageNet, the training data needs to be normalized with those means and standard deviations when the pretrained model is fine tuned on VGGFace2 dataset.

4.4 Hyperparameters

Stochastic Gradient Descent (SGD) optimizer is used to train the models, the starting learning rate is 0.001, the momentum is 0.9 and the weight decay is 0.001. Batch size of the training and validation datasets is 64. I also use the early stopping to prevent model from overfitting. When the validation metrics doesn't decrease for 8 epochs, the training stops.

¹https://github.com/noahch/PyAffact/blob/master/preprocessing/affact_transformer.py

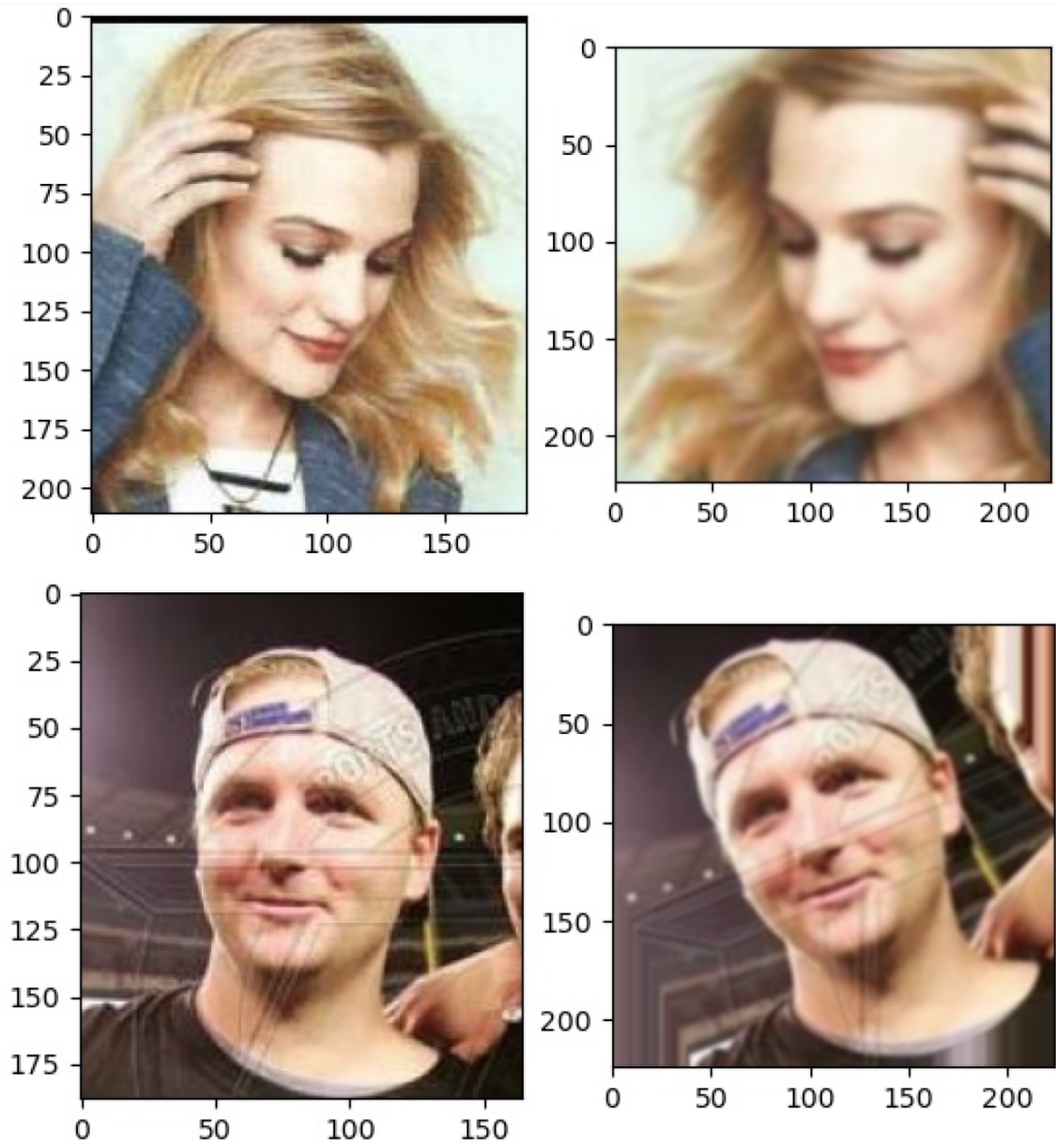


Figure 4.2: Original and Modified-AFFACT Perturbed Images. The left-side images are the original images from VGGFace2, and the right-side are their AFFACT transformed ones, which are used to train ResNet51.

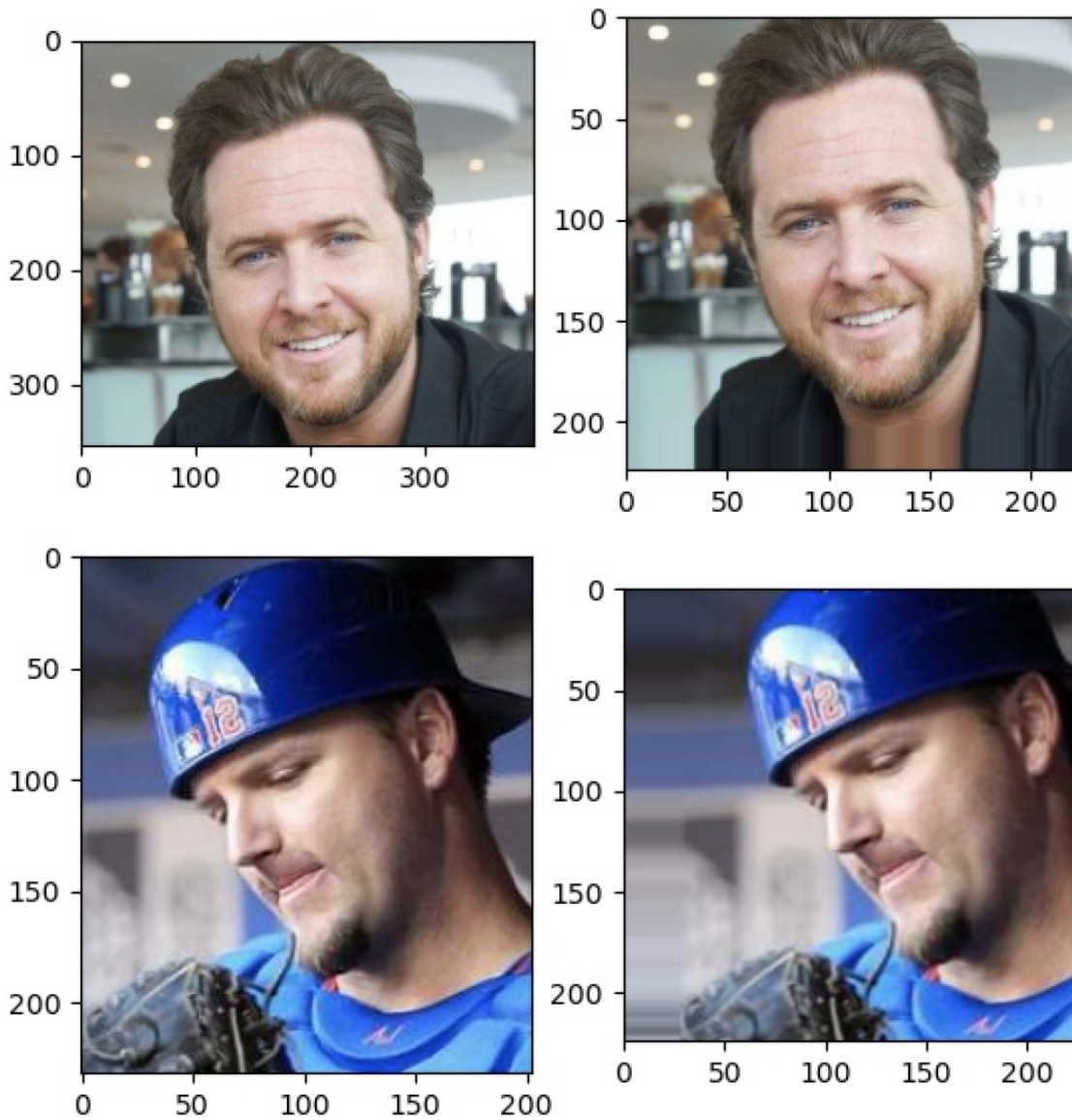


Figure 4.3: Original and Non-aligned Images. The left-side images are original ones from VGGFace2, and the right-side images are their non-aligned ones, which are used to validate ResNet51.

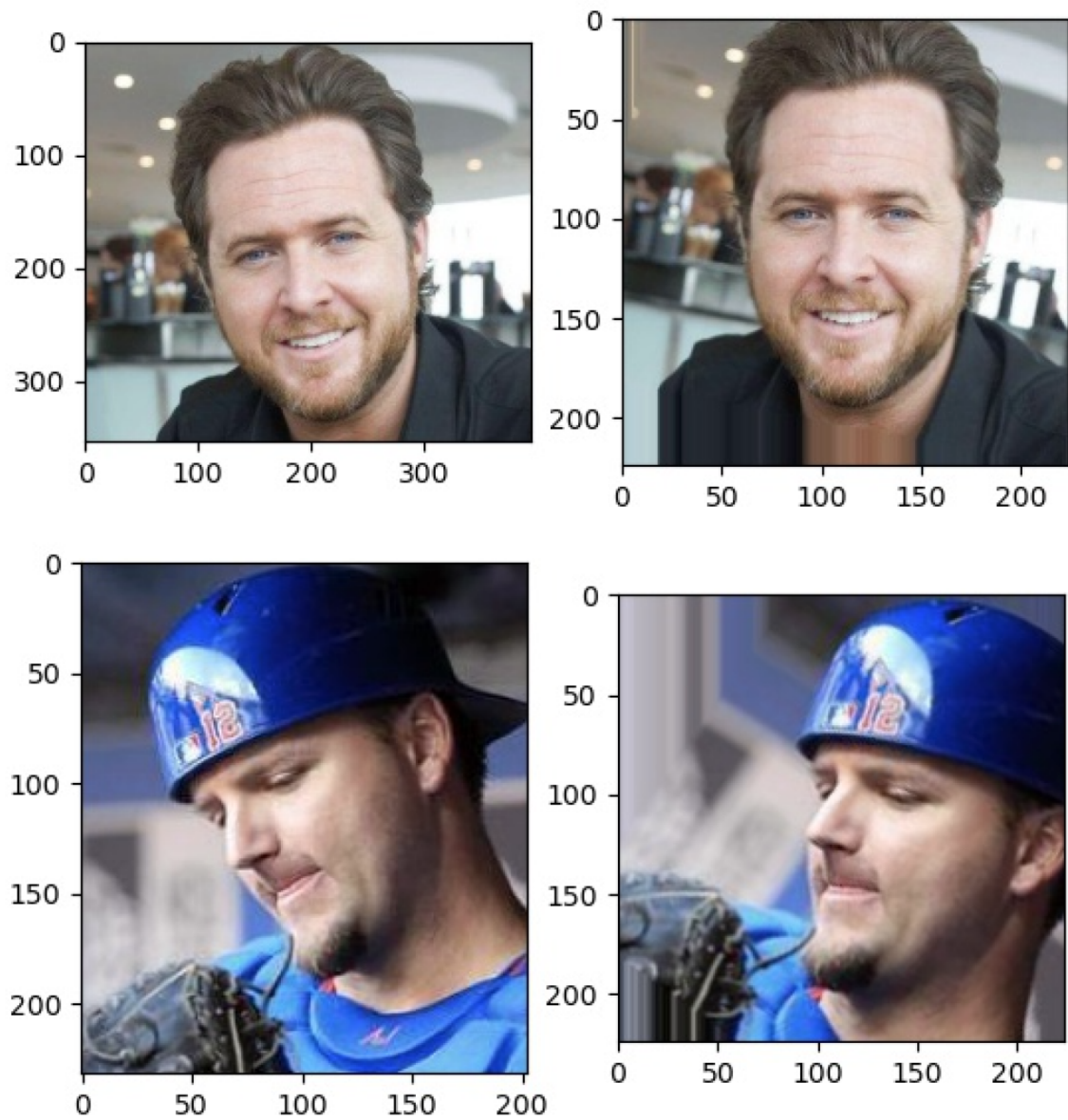


Figure 4.4: Original and Aligned Images. The left-side images are original ones from VGGFace2, and the right-side images are their aligned ones. Compared with the non-aligned images shown in Figure 4.3, the faces here are aligned. The aligned images are used to train and validate ResNet51_align.

4.5 Metrics

4.5.1 Metrics in Training

Categorical cross-entropy loss is used during the model training, as the face recognition problem is a multi-classes classification. The categorical cross entropy loss is defined as:

$$L_{CE} = - \sum_{i=1}^c t_i \log(p_i)$$

where t_i is the ground truth represented by one-hot vector of the i^{th} class, p_i is the probability of the i^{th} class. Classification accuracy is also calculated and tracked in the training.

4.5.2 Metrics in validation

As the model will be used to verify image pairs, I do not validate the model by classification accuracy of identities, but by the equal error rate (EER) of the verification of image pairs.

The validation dataset of image pairs are generated from VGGFace2 test set as I described in the section 3.1.

In the validation phase, the image pairs are firstly perturbed by the validation transformer shown in Table 4.1, and then normalized with the ImageNet means and standard deviations. Next, the normalized data are fed into the trained model, and the image features or image embedding vectors, which are the outputs of the embedding layer, are returned. After that, the cosine similarity of each image pairs is calculated with the features. Finally, the metric EER is computed based on the similarities and the true labels of the image pairs. The general way to compute EER is shown in the following pseudo code Function 1.

In stead of implementing the general computation of EER, I use the python API `bob.measure.eer`² provided by bob library to calculate EER easily.

In the model training, the reached smallest EER of ResNet51 is 0.0576, and the reached smallest EER of ResNet51_align is 0.0562.

The models with the smallest validation EER are supposed to be the best models for face verification. The obtained models will be used as extractors for further face recognition experiments in the next chapter.

²https://www.idiap.ch/software/bob/docs/bob/bob.measure/v4.2.0/py_api.html#bob.measure.eer

Function 1: Calculate EER

Input: *similarities, labels*; $FPRs \leftarrow \emptyset$; $FNRs \leftarrow \emptyset$;**for** *threshold* in the range $[\min(\text{similarities}), \max(\text{similarities})]$ **do** get the *predicted_labels* by the threshold; calculate False Positive Rate $FPR \leftarrow \text{number of false positive} / (\text{number of false positive} + \text{number of true negative})$; calculate False Negative Rate $FNR \leftarrow \text{number of false negative} / (\text{number of false negative} + \text{number of true positive})$; add FPR to $FPRs$; add FNR to $FNRs$;**end**get the index i where $|FPRs[i] - FNRs[i]|$ is the minimal ; $EER \leftarrow (FPRs[i] + FNR[i]) / 2$ **return** EER

Perturbation in Test Stage

In this chapter, I test verification performance of the obtained ResNet51_align and ResNet51 on different perturbed images of the LFW.

Firstly, I test different single perturbations for each extractor. Secondly, I use greedy algorithm to search the best combination of perturbations. Finally, I design an approach inspired by simple genetic algorithm to add weights on single perturbations and search the best weighted combination of perturbations.

The experiments are performed on the LFW view 1 training and test datasets, then the final results are compared and reported by testing on the 10-fold cross validation dataset of view 2.

5.1 Single Perturbations

Each single perturbation includes the operation of scaling, rotation, shifting, cropping and interpolation, the implementation of the perturbation is the same as the image perturbation in model training. The crop size is 224×224 . By setting different scale parameter γ_s , rotation parameter γ_a and shift parameters γ_x, γ_y , different single perturbation $[\gamma_s, \gamma_a, \gamma_x, \gamma_y]$ is applied on the images.

5.1.1 Perturbation Space

The parameters of scale, rotation and shift are selected in the sets:

$$\begin{aligned}\gamma_s &\in \{1.0, 1.1, 1.2, 1.3, 1.4\} \\ \gamma_a &\in \{-10, 0, 10\} \\ \gamma_x, \gamma_y &\in \{-10, 0, 10\}\end{aligned}$$

Therefore, there are 135 single perturbations in total. Here, the scale parameters γ_s is around 1.2, for it is used during model validation.

5.1.2 Experiment Process

For each single perturbation, I go through the pipeline of image preprocessing, feature extraction, feature storage and result calculation.

Image Preprocessing

The image preprocessing is similar as the preprocessing during model training. It consists of image perturbation and normalization.

The steps of image perturbation is the same as the AFFACT perturbation described in section 4.2:

Step 1: Calculate the bounding box. I use the landmarks detected by MTCNN (Zhang et al., 2016) to calculate the bounding boxes. There may be several faces on one image, I choose the landmarks corresponding to the biggest face. The detected landmarks are usually not precise, but by using the calculated bounding boxes, the effect of imprecision is mitigated.

Step 2: Calculate the original face angle and scale factor. As the images are aligned in LFW funneled, the original face angle is set to $\gamma_0 = 0$, scale factor $s_0 = \frac{W}{w} = \frac{H}{h}$, where the crop size $W = H = 224$, w and h are the width and height of the bounding box.

Step 3: Image perturbation. Each single perturbation is applied on the images of LFW funneled.

Finally, the data of perturbed images are normalized with $mean = [0.485, 0.456, 0.406]$, $std = [0.229, 0.224, 0.225]$, which are the mean and standard deviation of ImageNet.

Feature Extraction and Storage

After the images are perturbed and normalized, the data are passed into an extractor, and the features from the embedding layer of the extractor are outputted.

For each single perturbation, I extract the features of the perturbed LFW images by the obtained baseline model ResNet51_align and the target model ResNet51. All the features are stored in CPU hard disk for future use.

Result Calculation

Having the features in storage, I use the LFW training set of view 1 to get the best threshold and the corresponding training accuracy, then test on the test set of view 1 and get the test accuracy with this best threshold. The detailed calculation process is described in the pseudo code Function 2.

Function 2: Calculate Results

```

Input: dataset, perturbations, process = 'train', threshold = None;
similarities, labels ← Calculate_Similarities(dataset, perturbations);
if process = 'train' then
    get the threshold and EER;
    predicted_labels ← compute by the threshold and similarities;
    accuracy ← compute by comparing predicted_labels and labels;
    results ← threshold, accuracy
end
else if process = 'test' then
    predicted_labels ← compute by the threshold and similarities;
    accuracy ← compute by comparing predicted_labels and labels;
    results ← accuracy
end
return results

```

Before calculating the accuracy, cosine similarities of the image pairs are computed. In the pseudo code Function 3, for a single perturbation, there is only one perturbation in the parameter

Function 3: Calculate Similarities

```

Input: dataset, perturbations ;
similarities  $\leftarrow \emptyset$  ;
for each pair (name_1, name_2) in the dataset do
    features_1  $\leftarrow \emptyset$  ;
    features_2  $\leftarrow \emptyset$  ;
    for perturbation in perturbations do
        add the features of name_1 with the perturbation to features_1;
        add the features of name_2 with the perturbation to features_2;
    end
    /* average the features of the image with the perturbations */
    mean_features_1  $\leftarrow \text{mean}(\text{features}_1)$ ;
    mean_features_2  $\leftarrow \text{mean}(\text{features}_2)$ ;
    add Cosine_similarity(mean_features_1, mean_features_2) to similarities
end
get labels from dataset;
return similarities, labels

```

perturbations, which means the average features is still the features itself here.

5.1.3 Results

The results of ResNet51_align on perturbed images with the 135 single perturbations are shown in the Appendix Table A.1. The results of ResNet51 on perturbed images with the 135 single perturbations are shown in appendix Table A.3. The accuracy of each single perturbation of ResNet51_align and ResNet51 are compared, and the result shows that 102 out of 135 single perturbations' train accuracy of ResNet51 are larger than those of ResNet51_align, 122 out of 135 single perturbations' test accuracy of ResNet51 are larger than those of ResNet51_align. The mean and standard deviation of the difference of train accuracy between ResNet51 and ResNet51_align are 0.00375 and 0.00462. The mean and standard deviation of the difference of test accuracy between ResNet51 and ResNet51_align are 0.00894 and 0.00597.

To analyse the results, firstly I plot the histograms of the accuracy of ResNet51_align and ResNet51, then I make a Mann-Whitney U test to check the statistic difference. In addition, I will compare the top single perturbation that has the highest test accuracy with a baseline.

Histogram of Accuracy

According to the histograms of accuracy (Figure 5.1), the accuracy of ResNet51 shift to the right compared to that of ResNet51_align. The accuracy of ResNet51 lie in an range with greater values than those of ResNet51_align, especially for the test accuracy.

Mann-Whitney U Test

Mann-Whitney U test is implemented to check whether the difference of the accuracy of the two models are statistically significant or not. The null hypothesis and the alternative hypothesis are as follow:

- H_0 : accuracy of ResNet51 \leq accuracy of ResNet51_align.

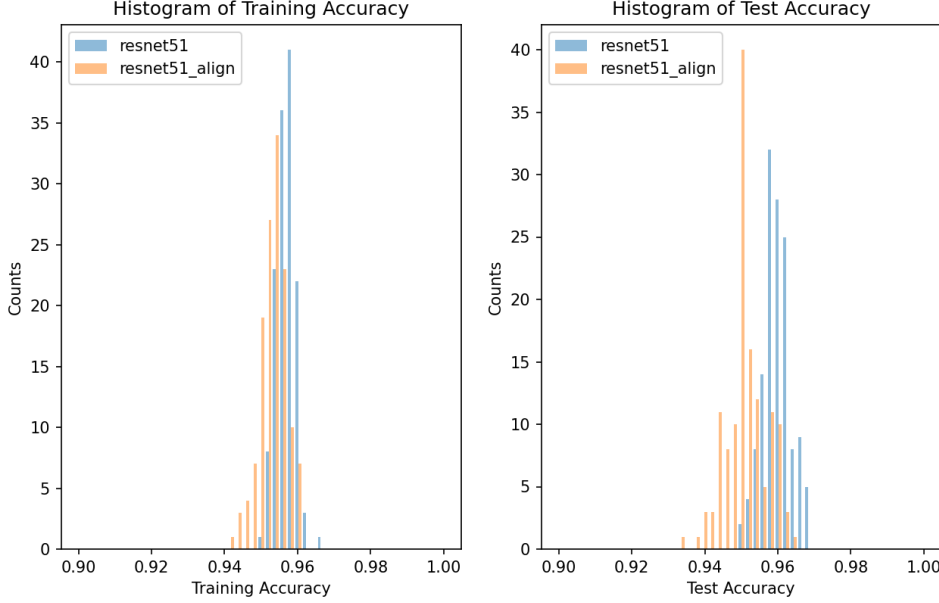


Figure 5.1: Histograms of Training and Test Accuracy of ResNet51 and ResNet51_align.

- H_a : accuracy of ResNet51 > accuracy of ResNet51_align.

The p_value of H_0 on training accuracy is $1.235e-17$, and the p_value of H_0 on the test accuracy is $1.088e-29$. Therefore, the null hypothesis H_0 is rejected, while the alternative hypothesis H_a is accepted, which means the training and the test accuracy of ResNet51 are statistically significantly greater than those of ResNet51_align.

The histograms and the Mann-Whitney U test indicate that compared to the baseline model which is trained on the aligned images, the model trained on AFFACT perturbed images can extract better features for face verification from images with large facial variance in terms of facial misalignment, image scale and position shift.

Baseline and Top Perturbations

As the obtained models are validated on images with $\gamma_s = 1.2$, the baseline should be the result of the perturbation with $[\gamma_s, \gamma_a, \gamma_x, \gamma_y] = [1.2, 0, 0, 0]$, which produces the scaled and aligned images. The baseline results and the results of the top single perturbations of the two models are listed in Table 5.1. The top single perturbation refers to the one that has the highest test accuracy. The preliminary results show:

- For both ResNet51_align and ResNet51, the top perturbations in the test stage improve the verification accuracy compared with the baseline aligned input. (0.965 vs. 0.952 and 0.969 vs. 0.956 for ResNet51_align and ResNet51 respectively.)
- When using ResNet51_align and ResNet51 to extract features from the baseline aligned images, the face verification test accuracy is 0.952 for ResNet51_align and 0.956 for ResNet51. ResNet51 performs a little better than ResNet51_align.

However, it's doubtful whether or not the result is statistically significant. I will further discuss the statistical conclusion in Section 5.4.

Extractor	Type	Perturbation $[\gamma_s, \gamma_a, \gamma_x, \gamma_y]$	Train Accuracy	Test Accuracy
ResNet51_align	Baseline	[1.2, 0, 0, 0]	0.95909	0.952
	Top	[1.1, -10, 0, -10]	0.96	0.965
ResNet51	Baseline	[1.2, 0, 0, 0]	0.95636	0.956
	Top	[1.1, -10, -10, 10]	0.95727	0.969

Table 5.1: Results of Baseline and Top Perturbation. Detected landmarks are used for perturbation. The accuracy is rounded to 5 decimal places, and zeros at the end are omitted.

5.2 Non-weighted Combination of Perturbations

As it is shown in the last section, the proper image perturbation in the test stage may improve the face verification accuracy, then how about combination of the perturbations? Can the accuracy increase further? In this section, I use greedy algorithm to search the non-weighted or equal-weighted combination of the single perturbations. Instead of majority voting or averaging predictions which are often seen in related literatures, I average the features that are extracted from the different perturbed images.

$$combined_features = \frac{\sum_{i=1}^n Extractor(Normalization(Perturbation_i(original_images)))}{n}$$

where n is the number of perturbations in the combination.

5.2.1 Greedy Algorithm

The process of greedy search is described in pseudo code Function 4. I go over the 135 single perturbations loop by loop, add one perturbation that can improve the accuracy the most in each loop, stop the iteration when the accuracy decreases by adding one more single perturbation. The search is conducted on the view 1 training set of LFW funneled, at the end of each loop, the combination of selected perturbations of this loop is tested on the view 1 test set. When the iteration stops, the best combination is chosen by the largest test accuracy.

The complete results of all loops in the greedy search listed in Appendix Table A.9 and Table A.10.

5.2.2 Results

Based on the test accuracy, the best combination for each extractor is finally selected (Table 5.2).

Function 4: Greedy Search

```

Input: train_set, test_set;
/* Initialize lists to store the best results of all loops.          */
selected_perturbations_record  $\leftarrow \emptyset$ ;
train_accuracies_record  $\leftarrow \emptyset$ ;
test_accuracies_record  $\leftarrow \emptyset$ ;
global_best_threshold  $\leftarrow -1$ ;
global_best_accuracy  $\leftarrow 0$ ;
loop_best_accuracy  $\leftarrow 0$ ;
/* Initialize a list to store selected perturbation in one loop.
   */
selected_perturbations  $\leftarrow \emptyset$ ;
while loop_best_accuracy  $\geq$  global_best_accuracy do
    Accs  $\leftarrow \emptyset$ ;
    Thresholds  $\leftarrow \emptyset$ ;
    for perturbation_i in the perturbation space do
        if perturbation_i in selected_perturbation then
            continue;
        else
            perturbations  $\leftarrow$  selected_perturbations  $\cup$  perturbation_i;
        end
        threshold, accuracy  $\leftarrow$  Calculate_Results
            (perturbations, train_set, process = 'train', threshold = None);
        add threshold to Thresholds;
        add accuracy to Accs;
    end
    idx  $\leftarrow$  index of max(Accs);
    loop_best_accuracy  $\leftarrow$  max(Accs);
    loop_best_threshold  $\leftarrow$  Thresholds[idx];
    if loop_best_accuracy  $\geq$  global_best_accuracy then
        global_best_accuracy  $\leftarrow$  loop_best_accuracy;
        global_best_threshold  $\leftarrow$  loop_best_threshold;
        add perturbation_idx to selected_perturbations;
        add loop_best_accuracy to train_accuracies_record;
        add selected_perturbation to selected_perturbation_record;
        /* Validate the selected perturbations on test set.          */
        test_accuracy  $\leftarrow$  Calculate_Results (selected_perturbations, test_set, threshold =
            global_best_threshold, process = 'test');
        /* Store the validation results.                              */
        add test_accuracy to test_accuracies_record;
    else
end
index  $\leftarrow$  index of the max(test_accuracies_record);
best_combination  $\leftarrow$  selected_perturbations_record[index];
return best_combination

```

For ResNet51_align, the best test accuracy of the found perturbation combination is 0.968, larger than that of the baseline input 0.952 and a bit larger than that of the top single perturbation 0.965.

For ResNet51, the test accuracy of the best combination is 0.964, better than the baseline 0.956, but not as good as that of the top single perturbation result 0.969. Whether the differences are statistically significant or not will be discussed in Section 5.4.

Extractor	Selected Perturbations	Train Accuracy	Test Accuracy
ResNet51_align	[1.3, -10, 0, -10] [1.1, -10, 0, -10] [1.0, 10, 10, -10] [1.2, -10, -10, -10] [1.0, -10, 0, 0]	0.96818	0.968
ResNet51	[1.0, -10, -10, -10] [1.0, -10, 0, 0] [1.0, -10, 10, -10]	0.96818	0.964

Table 5.2: Perturbation Combinations Found by Greedy Algorithm. Detected Landmarks are used in perturbations. Accuracy is rounded to 5 decimal places, zeros at the end are omitted.

5.3 Weighted Combination of Perturbations

Greedy algorithm can select a satisfactory combination of perturbations, however, perturbations in the set are without weight difference. Simple Genetic Algorithm (SimpleGA) has the potential to select perturbations with different weights due to its mutation process. In this section, I design an approach which is inspired by SimpleGA to assign different weights to single perturbations, and search the weighted combinations that may lead to better verification accuracy.

5.3.1 General SimpleGA

The general procedure of SimpleGA consists of population initialization, fitness score calculation, selection, crossover and mutation (Function 5). Firstly, a set of individual is initialized as the population. Each individual is represented by a set of variables or parameters. The set of parameters of the individual is called the chromosome of the individual, which is usually a solution to the problem that need to be solved. Secondly, the fitness score for each individual in the population is computed by fitness function. The fitness score determines how fit the individual solution is to the problem. Then, based on the fitness scores, some individuals are selected as parents for reproduction. For each pair in the parents pool, part of the chromosomes are exchanged (crossover) and mutated to reproduce two children, and then the fitness scores of the children are calculated to check how fit the children are. The processes of selection, crossover and mutation are repeated until it does not produce children that are significantly different from the parents.

Function 5: SimpleGA

```

population initialization;
calculate fitness score for each individual in the population;
REPEAT:
    Selection;
    Crossover;
    Mutation;
    Calculate fitness scores for each child;
UNTIL population has converged;
return

```

5.3.2 SimpleGA Variant Approach

In my work, I do not follow the standard procedure of SimpleGA, while I try a variant approach. The individual is defined as $[w, \gamma_s, \gamma_a, \gamma_x, \gamma_y]$, in which w is the weight parameter, $\gamma_s, \gamma_a, \gamma_x, \gamma_y$ are the perturbation parameters. The population is the 135 single perturbations with weights. The verification accuracy of each perturbation is already known. What I would like to do is to use the variation of SimpleGA to search the weighted combination of perturbations and check the verification accuracy when using the weighted combination of perturbations. My approach is illustrated by Figure 5.2 and pseudo code Function 6.

Individual and Population

As it is mentioned above, the individual is defined as $[w, \gamma_s, \gamma_a, \gamma_x, \gamma_y]$, where $\gamma_s, \gamma_a, \gamma_x, \gamma_y$ are the perturbation parameters, w is the weight assigned to this perturbation. The perturbation parameters are selected in the sets:

$$\begin{aligned}\gamma_s &\in \{1.0, 1.1, 1.2, 1.3, 1.4\} \\ \gamma_a &\in \{-10, 0, 10\} \\ \gamma_x, \gamma_y &\in \{-10, 0, 10\}\end{aligned}$$

Therefore, the population includes 135 individuals.

Weight Initialization

There are many ways to initialize weights of perturbations. I try to initialize the weights with three different methods.

- **Equal:** Initialize equal weights for each perturbation by assigning a number. In implementation, I set a positive integer to the equal weight.
- **Unequal:** Initialize unequal weights for perturbations by randomly selecting weights from a range. In implementation, positive integers that are randomly selected from a range are assigned to the weights.
- **Gaussian:** Initialize weights from a Gaussian distribution. In implementation the standard deviation is 1, by setting the mean, different Gaussian distribution can be generated. The weights are all rounded to three decimal places, and the negative weights are replaced by 0.

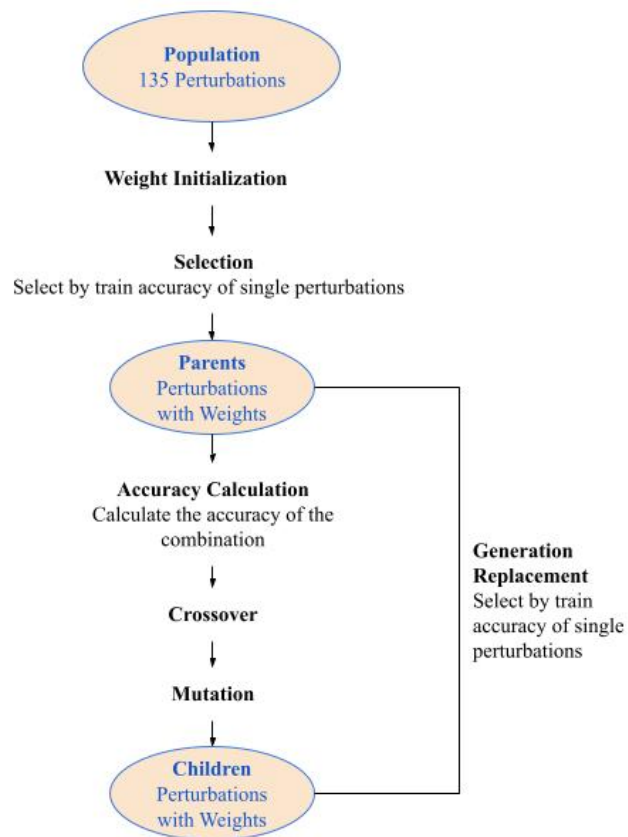


Figure 5.2: Illustration of SimpleGA Variant Approach.

Function 6: SimpleGA Variant Approach

```

Input: population;
population  $\leftarrow$  weight_initialization(population, ...);
parents  $\leftarrow$  Selection(population, ...);
generation  $\leftarrow$  0;
scores  $\leftarrow$   $\emptyset$ ;
combinations  $\leftarrow$   $\emptyset$ ;
while generation < max_iteration do
    generation += 1;
    score, selected_individuals  $\leftarrow$  Calculate_Accuracy_of_Combination(parents, ...);
    scores.append(score);
    combinations.append(selected_individuals);
    if number of parents <= 1 then
        | break;
    end
    children  $\leftarrow$   $\emptyset$ ;
    for each pair (parent1, parent2) in parents do
        for child in crossover(parent1, parent2) do
            | child  $\leftarrow$  mutation(child);
            | if the weight parameter of child > 0 then
            | | children.append(child);
            | end
        end
        if number of parents is odd then
            | add the parent who doesn't have partner to children;
        end
        parents  $\leftarrow$  generation_replacement(parents, children,...);
        if parents is  $\emptyset$  then
            | break;
        end
    end
end
return scores, combinations

```

Selection

Different methods can be used to select parents, such as (1) elite selection: select top individuals that have top fitness scores; (2) tournament selection: randomly select pairs, compare the fitness score between the two and put the better one of each pair into the parent pool.

Here, I use the elite selection method. Based on the 135 single perturbation results in section 5.1, I select n top individuals as parents for reproduction based on the training accuracy of the perturbations. I do the selection based on the training accuracy but not the test accuracy for the reason that the experiment should be conducted on the training dataset. The test dataset is used to test the found combinations of perturbations.

In the general procedure of SimpleGA, the selection process should be included in the iteration. However, in my approach, I do not repeat selection, but select the parents for one time. It leads to a limitation that the approach can not search the whole population space.

Accuracy of the Combination

Verification accuracy is calculated in the process shown in Function 2. However, when computing similarity, the combined features are not calculated by averaging the features from different perturbations, but by combining them with weights.

$$combined_features = \frac{\sum_{i=1}^n Weight_i \times Extractor(Normalization(Perturbation_i(original_images)))}{\sum_{i=1}^n Weight_i}$$

Where n is the number of perturbations in the combination.

In each generation, I select the combination of perturbations by two methods:

- **Simple:** simply use all weighted perturbations in the parent generation, and calculate the training and test accuracy. When using this method, all individuals in the parent generation are selected into the combination.
- **Greedy_search:** greedy search the weighted combination of perturbations in the parent generation, calculate the train and test accuracy and return the weighted combination with the greatest test accuracy. When using the method, part of the individuals in the parent generation will be selected to form the combination.

Crossover

In the process of crossover, pairs of individuals are selected from the parent generation, chromosomes of each pairs are crossed over to produce two children. Here the chromosomes refer to the weight and perturbation parameters, which is $[w, \gamma_s, \gamma_a, \gamma_x, \gamma_y]$. Figure 5.3 explains the cross over procedure. The crossover point is randomly selected from the range $[1, 3]$.

Mutation

The chromosomes of children are randomly mutated by mutation masks $[m_w, m_{\gamma_s}, m_{\gamma_a}, m_{\gamma_x}, m_{\gamma_y}]$ which are randomly generated with

$$\begin{aligned} m_w &\in \{-1, 1\} \\ m_{\gamma_s} &\in \{-0.1, 0, 0.1\} \\ m_{\gamma_a} &\in \{-10, 0, 10\} \\ m_{\gamma_x}, m_{\gamma_y} &\in \{-10, 0, 10\} \end{aligned}$$

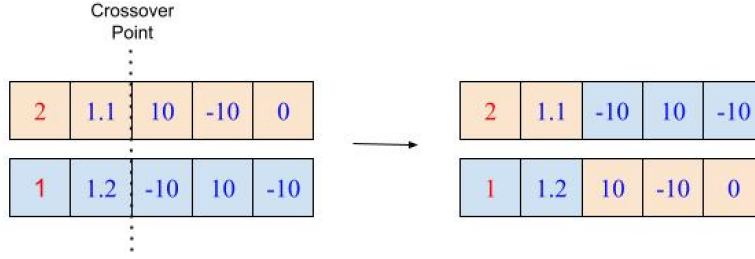


Figure 5.3: Illustration of Crossover. Weights are in red font, perturbation parameters are in blue font.

As the purpose is to use the genetic algorithm to get the weighted combination of the perturbations, the mutation will not produce new single perturbations that are outside of the 135 perturbations. If the mutated weight of a perturbation is 0, it is excluded from the children generation.

Pseudo code Function 7 explains the procedure I use to implement such mutation. Figure 5.4 is an example of the mutation.

Function 7: Mutation

```

Input: weighted perturbation  $[w, \gamma_s, \gamma_a, \gamma_x, \gamma_y]$  ;
 $m_w \leftarrow$  randomly select from  $\{-1, 1\}$  ;
 $m_{\gamma_s} \leftarrow$  randomly select from  $\{-0.1, 0, 0.1\}$  ;
 $m_{\gamma_a}, m_{\gamma_x}, m_{\gamma_y} \leftarrow$  randomly select from  $\{-10, 0, 10\}$  ;
 $w \leftarrow w + m_w$  ;
if  $\gamma_s + m_{\gamma_s}$  in a certain range then
  |  $\gamma_s \leftarrow \gamma_s + m_{\gamma_s}$ 
end
if  $\gamma_a * m_{\gamma_a} \leq 0$  then
  |  $\gamma_a \leftarrow \gamma_a + m_{\gamma_a}$ 
end
if  $\gamma_x * m_{\gamma_x} \leq 0$  then
  |  $\gamma_x \leftarrow \gamma_x + m_{\gamma_x}$ 
end
if  $\gamma_y * m_{\gamma_y} \leq 0$  then
  |  $\gamma_y \leftarrow \gamma_y + m_{\gamma_y}$ 
end
return  $[w, \gamma_s, \gamma_a, \gamma_x, \gamma_y]$ 

```

Generation Replacement

To update the generation, I try two ways:

- **Cover:** the parent generation are all replaced by the children.
- **Merge:** merge top percentage of individuals in the current parent generation with the top percentage of individuals in the children generation to get the next parent generation. The top individuals are selected based on the training accuracy of the individual perturbations.

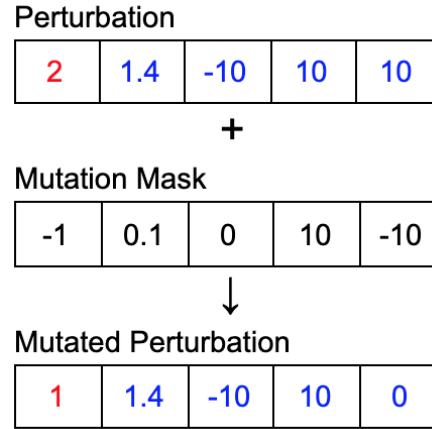


Figure 5.4: Illustration of Mutation. Conditions control the mutation to limit the mutated perturbation still in the 135 perturbations. The weights are marked in red, the perturbation parameters are in blue font.

Stop Condition

The iteration stops when (1) max_iteration is reached; (2) number of parents ≤ 1 .

5.3.3 Results

To run this SimpleGA variant approach, there are several hyperparameters that need to be set besides the hyperparameter "max iteration". I list them in Table 5.3.

Selection	Weight Initialization	Combination	Generation Replacement
Elite (Top n)	<ul style="list-style-type: none"> · Equal (Weight w) · unequal (Range of weight) · Gaussian (Dist. Mean μ) 	<ul style="list-style-type: none"> · simple · greedy_search 	<ul style="list-style-type: none"> · cover · merge (rate)

Table 5.3: Hyperparameters of the SimpleGA Variant Approach. The items in the table represent different methods in each process, and the texts inside the brackets behind the items denote the hyperparameters that should be set. For example, in the process of Generation Replacement, when the method is 'merge (0.5)', it means 50% top parents and 50% top children are merged together to form the next parent generation.

The SimpleGA variant approach result strongly depends on the hyperparameters. By experimenting different hyperparameters, the top weighted combinations of perturbations for the two extractors are returned. I present the top weighted combinations of perturbations and the hyperparameters to reach them in the Table 5.4. The test accuracy of the top combination for ResNet51_align is 0.966, larger than the baseline of ResNet51_align; the test accuracy of the top combination for ResNet51 is 0.968, which is greater than the baseline of ResNet51.

Here, the top one is probably not the optimal one, for this approach can not search the whole

population space and randomness is involved in the procedure. In addition, the search result also strongly depends on the selection method, however, some potential selection methods are not experimented.

Extractor	Conditions	Selected Perturbations $[\gamma_s, \gamma_a, \gamma_x, \gamma_y]$	Weights	Train Accuracy	Test Accuracy
ResNet51_align	Max iteration: 50	[1.1, 10, 10, 10]	12.5%	0.96182	0.966
	Elite selection: Top 12	[1.1, 0, 10, -10]	12.5%		
	Weight initialization: equal (5)	[1.1, -10, 10, 10]	12.5%		
	Combination method: simple	[1.0, 0, 0, 0]	7.5%		
	Generation replacement: cover	[1.1, 0, 10, 0]	17.5%		
		[1.2, -10, 10, -10]	12.5%		
		[1.2, -10, -10, 0]	12.5%		
		[1.1, -10, 10, 10]	2.5%		
		[1.3, -10, 10, -10]	10%		
ResNet51	Max iteration: 15	[1.0, -10, 10, 0]	25%	0.96	0.968
	Elite selection: Top 5	[1.2, 10, 0, 0]	25%		
	Weight initialization: unequal([1, 5])	[1.1, -10, -10, 0]	31.25%		
		[1.0, 10, 10, 0]	18.75%		
	Combination method: simple Generation replacement: cover				

Table 5.4: Weighted Perturbation Combinations Found by SimpleGA Variant Approach. "Elite selection: Top 5" means select the 5 single perturbations that have the top 5 greatest accuracy on the training set. "Weight initialization: unequal([1, 5])" means initializing each weight by randomly selecting an integer in the range [1,5]. "Combination method: simple" means to choose the combination of perturbations by the 'simple' method. "Generation replacement: cover" means the parent generation is replaced by the children generation totally. All accuracy are rounded to 5 decimal places, zeros at the end are omitted. Detected landmarks are used for perturbation. Accuracy is rounded to 5 decimal places, zeros at the end are omitted.

5.4 Results Comparison and Test on View 2 Dataset

To refresh the memory of all the results mentioned previously, I hereby put together the results of baseline, single top perturbation, non-weighted perturbation combination found by greedy algorithm and top weighted perturbation combination found by SimpleGA variant approach. Besides, these different situations of perturbation are tested on the LFW view 2 dataset (the 10-fold cross validation) for a final report. The results of ResNet51_align and ResNet are in Table 5.5.

Based on verification accuracy of the 10 folds cross validation, top single perturbation result, greedy searched result and the result of SimpleGA variant approach are all greater than the baselines for both models. Among them, the greedy search results boost the performance of models the most.

Compared with the view 2 mean test accuracy of the baselines, greedy searched combinations

	ResNet51_align				ResNet51			
	Selected Perturbations	Weights	View 1 Test Accuracy	View 2 Test Mean Acc \pm Std. (%)	Selected Perturbations	Weights	View 1 Test Accuracy	View 2 Test Mean Acc \pm Std. (%)
Baseline	[1.2, 0, 0, 0]	None	0.952	0.96 \pm 0.00741 (0%)	[1.2, 0, 0, 0]	None	0.956	0.95983 \pm 0.00547 (0%)
Single	[1.1, -10, 0, -10]	None	0.965	0.96033 \pm 0.0103 (0.825%)	[1.1, -10, -10, 10]	None	0.969	0.96283 \pm 0.00733 (7.468%)
Greedy	[1.3, -10, 0, -10]	Equal	0.968	0.96667 \pm 0.00913 (16.675%)	[1.0, -10, -10, -10]	Equal	0.964	0.96767 \pm 0.00589 (19.517%)
	[1.1, -10, 0, -10]				[1.0, -10, 0, 0]			
	[1.0, 10, 10, -10]				[1.0, -10, 10, -10]			
	[1.2, -10, -10, -10]							
	[1.0, -10, 0, 0]							
SimpleGA	[1.1, 10, 10, 10]	12.5%	0.966	0.96367 \pm 0.00834 (9.175%)	[1.0, -10, 10, 0]	25%	0.968	0.9645 \pm 0.0069 (11.626%)
	[1.1, 0, 10, -10]	12.5%			[1.2, 10, 0, 0]	25%		
	[1.1, -10, 10, 10]	12.5%			[1.1, -10, -10, 0]	31.25%		
	[1.0, 0, 0, 0]	7.5%			[1.0, 10, 10, 0]	18.75%		
	[1.1, 0, 10, 0]	17.5%						
	[1.2, -10, 10, -10]	12.5%						
	[1.2, -10, -10, 0]	12.5%						
	[1.1, -10, 10, 10]	2.5%						
	[1.3, -10, 10, -10]	10%						

Table 5.5: Result Comparison. The top single perturbation and the best perturbation combination are chosen according to the test accuracy on the view 1 test dataset, then they are tested on the view 2 dataset to get the view 2 test accuracy as the final report. Percentage in the parentheses is the reduction of error rate compared with the baseline. Detected landmarks are used for perturbations. Accuracy is rounded to 5 decimal places, zeros at the end are omitted.

reduce the error rate, which is defined as $1 - \text{accuracy}$, by 16.675% and 19.517% for ResNet51_align and ResNet51 respectively. SimpleGA variant approach reduces the error rate by 9.175% and 11.626% for ResNet51_align and ResNet51 respectively. Top single perturbation improves the performance very little for ResNet51_align, while it reduces the error rate by 7.468% for ResNet51.

T-tests are run to check whether the mean difference of different situations of perturbations are significant or not. Table 5.6 shows the p-value of each test.

Firstly, different perturbation results are compared with the baseline for the two models. It shows the mean accuracy difference between greedy search result and baseline result is statistically significant at 90% confidence level for ResNet51_align and at 99% confidence level for ResNet51. The improvement made by greedy search is statistically significant. The mean accuracy difference between SimpleGA variant approach and baseline is statistically significant at 88% confidence level for ResNet51, while the difference is not statistically significant for ResNet51_align. In addition, the difference between the top single perturbation result and baseline result is not statistically significant for both models.

Secondly, different situations of perturbation are compared pairwise. The mean accuracy difference between Greedy search results and top single perturbations are statistically significant at 83% confidence level for both models, while results of SimpleGA variant approach and top single perturbation has no significant difference. The difference between greedy search and SimpleGA variant approach is significant at 70% confidence level for ResNet51, while the two has no significant difference for ResNet51_align.

Furthermore, the difference between the two models at each situation of perturbation is compared. The p-values of T-tests are shown in Table 5.7.

The p-value of T-test for the two baseline is 0.955, which informs that the baseline result of the

	ResNet51_align	ResNet51
Baseline vs. Single	0.934	0.313
Baseline vs. Greedy	0.09 *	0.006 **
Baseline vs. SimpleGA	0.313	0.111
Single vs. Greedy	0.163	0.121
Single vs. SimpleGA	0.437	0.607
Greedy vs. SimpleGA	0.453	0.284

Table 5.6: P-Values Of T-Tests. ** 99% confidence level, * 90% confidence level. Detected landmarks are used in perturbations.

ResNet51_align vs. ResNet51	
Baseline	0.955
Single	0.54
Greedy	0.774
SimpleGA	0.81

Table 5.7: T Tests to Compare Two Models at the Same Situations of Perturbation. P-value of each test is listed in the table. Detected landmarks are used in perturbations.

two models have no significant difference. The p-values for the top single perturbation, greedy, SimpleGA of the two models are 0.540, 0.774, 0.810 respectively, which shows the two models have no significant difference on their top single perturbations, combinations found by greedy and combinations found by SimpleGA.

The comparisons reveal several findings:

- Model trained on the AFFACT-perturbed images (ResNet51) can extract features of aligned images (baseline) as good as model trained on aligned images (ResNet51_align), for their baselines have no significant difference.
- Proper single perturbation in the test stage may improve face verification accuracy for both models, but T-test shows the effect is not significant.
- Non-weighted combination of perturbations found by greedy algorithm in the test stage boost the performance of both models. The results are statistically larger than the results of baselines.
- Weighted combination of perturbations found by SimpleGA variant approach are not as good as the non-weighted combinations found by greedy algorithm. The result of this approach strongly depends on the large amount of experiments, the hyperparameters, and the methods of selection and generation replacement. More satisfactory results could be found if experimenting more and changing to some other selection methods.
- The test time perturbation can boost the performance of the two models to the same level, as the result difference of the same boosting method is not statistically significant for ResNet51_align and ResNet51 when using the LFW funneled dataset.

In conclusion, model trained with AFFACT technique (ResNet51) has better performance on images with large variance than model trained on aligned images (ResNet51_align), as the result of the 135 single perturbations reveals in the [histogram of accuracy](#) and the [Mann-Whitney U Test](#) in the Section 5.1. ResNet51 works as good as ResNet51_align on the baseline aligned faces of LFW funneled. In addition, proper combination of perturbations in the test time can boost the model performance.

Discussion

1. Why slightly modify AFFACT?

The scale parameter is Gaussian distributed with mean 1.0 and standard deviation 0.1 in the original research (Günther et al., 2017). The original work was conducted on CeleA dataset for facial attribute classification. As the training database and model task are changed, the scale parameter may not be suitable for my work. Therefore, I trained models with scale parameter of different distributions. ResNet51_align and ResNet51 are trained with both $\tilde{\gamma}_s \sim N(1.0, 0.1)$ and $\tilde{\gamma}_s \sim N(1.2, 0.1)$. The best validation EER for different scale parameters is shown in Table 6.1. N(1.2, 0.1) comes to the lower validation EER for both models, so I modify the mean of the distribution to 1.2.

	ResNet51_align	ResNet51
N(1.0, 0.1)	0.0594	0.0616
N(1.2, 0.1)	0.0562	0.0576

Table 6.1: Best Validation EERs. The best validation EERs when using different normal distribution of scale parameter to preprocess training images.

2. How about directly use detected bounding boxes for test time perturbations?

With AFFACT model, the landmarks are not necessarily needed. Detected bounding boxes can be directly used for perturbation in the test stage. I use the MTCNN detected bounding boxes for single perturbations and compared the results of the two models. Histogram of Accuracy is plotted (Figure 6.1) and Mann-Whitney U test is conducted. The p_value of H_0 on training accuracy is 6.40e-08, and the p_value of H_0 on the test accuracy is 4.38e-19. The result reveals the same fact that ResNet51 is more robust to facial variance than ResNet51_align.

Combinations of perturbations are also searched by greedy and SimpleGA variant approach, then all situations of perturbations are tested on view 2 dataset of LFW funneled (Table 6.2). T tests are conducted as well (Table 6.3). The conclusion is similar as in Section 5.4. The combination of perturbations found by greedy algorithm boost the model performance the most, reducing the error rate by 18.441% for ResNet51_align and 16.453% ResNet51, and the mean accuracy differences between greedy search result and baseline result is statistically significant at 95% confidence level. The combinations found by SimpleGA reduce the error rate by 13.941% for ResNet51_align and 5.786% for ResNet51, but at a low confidence level.

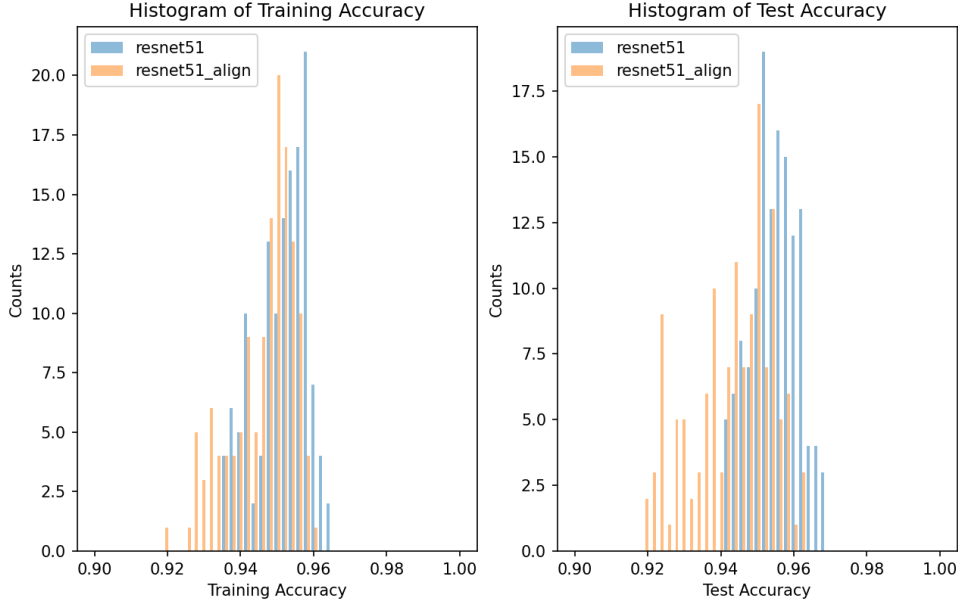


Figure 6.1: Histograms of Accuracy of ResNet51_align and ResNet51. Detected bounding boxes are used in perturbation.

However, the scale parameters of the selected perturbations vary a lot from those perturbations performed with detected landmarks. For perturbation with detected landmarks, the scale parameters of the selected perturbations are around 1.1 - 1.3, however for the detected bounding boxes, the scale parameters are around 0.5 - 0.6.

3. Why the scale parameters of the found perturbations differ a lot from those of perturbations by using detected landmarks?

The reason is that the heights and widths of the detected bounding boxes are smaller than the bounding boxes calculated by detected landmarks. As it's shown in Figure 6.2, the red rectangle on the original image is the detected bounding box, scale is calculated by $s = \min(\frac{W}{w}, \frac{H}{h}) \cdot \gamma_s$, where W and H are the crop size, w and h are the width and height of the bounding box. The blue dots on the original image are the detected landmarks. To calculate bounding box by the landmarks, the distance d between eye center and mouth center is computed at first, then the width and height of bounding box are set as $w = h = d \cdot 5.5$. The width and height of the calculated bounding box are quite high, greater than the detected ones. That is why by detected landmarks, the scale parameter γ_s is larger to realize the same scale s .

For this specific image sample in Figure 6.2, the width and height of bounding box calculated by the landmarks extend the full size of the image.

Figure 6.2 also visually compares different scale parameters for the two cases. The scaling of face with $\gamma_s = 0.5$ by detect bounding boxes is close to those with $\gamma_s = 1.1$ or 1.2 by detect landmarks.

The visual comparison also shows that when adding the scale parameter by 0.1, the image upscales more by using detected bounding box than by using detect landmarks. That is why the scale parameters of the selected perturbations conducted with detected landmarks scatter

	ResNet51_align				ResNet51			
	Selected Perturbations	Weights	View 1 Test Accuracy	View 2 Test Accuracy	Selected Perturbations	Weights	View 1 Test Accuracy	View 2 Test Accuracy
Baseline	[0.5, 0, 0, 0]	None	0.962	0.95933 \pm 0.00794 (0%)	[0.5, 0, 0, 0]	None	0.962	0.9625 \pm 0.00498 (0%)
Single	[0.5, 0, 0, 10]	None	0.963	0.95683 \pm 0.01357 (-6.147%)	[0.5, -10, 0, 0]	None	0.969	0.96533 \pm 0.00745 (7.547%)
Greedy	[0.5, -10, 0, 10]	Equal	0.97	0.96683 \pm 0.00687 (18.441%)	[0.5, 0, 10, -10]	Equal	0.97	0.96867 \pm 0.00618 (16.453%)
	[0.5, -10, 10, -10]				[0.5, -10, -10, 10]			
	[0.5, -10, 0, 0]				[0.5, -10, -10, -10]			
	[0.5, 0, 0, -10]				[0.5, 10, 10, 0]			
	[0.5, 0, 10, 10]				[0.5, -10, -10, 0]			
	[0.5, -10, -10, -10]				[0.5, -10, 0, -10]			
	[0.5, 0, 0, 10]				[0.5, 10, 0, -10]			
	[0.5, -10, 0, -10]				[0.5, -10, 0, 10]			
	[0.5, 0, 0, 0]				[0.5, -10, 10, 0]			
	[0.5, 10, 10, -10]				[0.5, 10, 0, 10]			
	[0.5, -10, 10, 0]				[0.5, 0, 10, 10]			
	[0.6, -10, 0, 10]				[0.5, -10, 10, -10]			
	[0.5, 0, 10, -10]							
	[0.5, 10, 0, 10]							
SimpleGA	[0.5, 0, 10, -10]	33.333%	0.969	0.965 \pm 0.00885	[0.5, 10, 0, 0]	66.667%	0.97	0.96467 \pm 0.00637
	[0.5, -10, 0, 10]	66.667%		(13.941%)	[0.5, 10, 10, 0]	33.333%		(5.786%)

Table 6.2: Result Comparison (Bounding Box). Detected bounding boxes are used in perturbations. The top single perturbation and the best perturbation combination are chosen according to the test accuracy on the view 1 test dataset, then they are tested on the view 2 dataset to get the view 2 test accuracy as the final report. Percentage in the parentheses is the reduction of error rate compared with the baseline. Accuracy is rounded to 5 decimal places, zeros at the end are omitted.

among 1.0 - 1.3, while those with detected bounding boxes dense at 0.5. It also explains why the histogram scatters in a wider range of accuracy here.

4. Why the baseline is chosen as [0.5, 0, 0, 0] when using detected bounding boxes to perform perturbations?

As the detected bounding box varies a lot from the bounding box calculated by the landmarks, the values of scale parameter γ_s are quite different to achieve the same scale s for the two methods. Therefore, the $\gamma_s = 1.2$ can not be set as the baseline here. As Figure 6.2 shows, the scale with $\gamma_s = 0.5$ when using detected bounding box for perturbation is similar to the scale with $\gamma_s = 1.2$ when using detected landmarks for perturbation, so I choose $[\gamma_s, \gamma_a, \gamma_x, \gamma_y] = [0.5, 0, 0, 0]$ as the baseline here.

5. Same verification accuracy for different perturbations

The number of pairs is not large in LFW dataset, so verification accuracy of many different single perturbations are the same or differ slightly. Other larger datasets may be chosen to perform the model test. One point needs to be mentioned is that the dataset biases can affect the generalization across datasets. Phillips (2017) compared different test dataset for VGGFace model for face recognition. He found that the model can achieve high accuracy on LFW and YTF, yet obtained

	ResNet51_align	ResNet51
Baseline vs. Single	0.621	0.331
Baseline vs. Greedy	0.037 **	0.024 **
Baseline vs. SimpleGA	0.149	0.408
Single vs. Greedy	0.052 *	0.290
Single vs. SimpleGA	0.128	0.832
Greedy vs. SimpleGA	0.611	0.171

Table 6.3: P_Values of T-Tests (Bounding Box). ** 95% confidence level, * 90% confidence level. Detected bounding boxes are used in perturbations.

very low accuracy on Ugly and Bad datasets ([Phillips et al., 2011](#)). Therefore, when selecting other test dataset, the dataset bias should be considered.

6. Other approach to use SimpleGA

The SimpleGA variant approach I use in this thesis is not a standard SimpleGA approach, and it has a severe limitation, that is the pool of individuals selected for reproduction is limited. I select some top individuals from the population, and then the reproduction is based on these individuals. Although mutation can introduce some other individual perturbations into the reproduction process, it cannot search the whole space of the population effectively. Other approaches could be designed to use the SimpleGA, for instance, an individual could be defined as a vector of 135 weights, each element of the vector corresponds to the weight of a perturbation, and the population can be initialized as several 135-dimension vectors. In this way, a standard SimpleGA approach can be implemented, and the result could be more promising.

7. Loss function

In my work, I use the categorical cross entropy loss to train the face recognition model. However, the loss is not sufficiently effective for the intra-person variations could be greater than the inter-person differences. Other loss function may be used to optimize the model, such as CosFace, ArcFace loss.



Figure 6.2: LFW Image Samples with Different Scale Parameters. The red rectangle on the original image is the detected bounding box, and the blue dots on the original image are the detected landmarks. The first row of perturbed images are with scale factor s_0 calculated by detected bounding boxes; the second row of perturbed images are with scale factor s_0 computed by detected landmarks.

Conclusion

In this thesis, a target model (ResNet51) is trained with AFFACT technique and a baseline model (ResNet51_align) is trained on aligned images for comparison. 135 perturbations are experimented in the test time for both models. The results of 135 perturbations reveal that model trained with AFFACT technique extracts better features for face verification from LFW images with large variance. I also use optimization algorithms (greedy and SimpleGA variant approach) to search for the satisfactory combinations of perturbations. It shows that combination of perturbations in the test time can boost the model performance. Compared with the baselines in the test time, greedy searched combinations reduce the error rate of face verification by 16.675% and 19.517% for ResNet51_align and ResNet51 respectively, and combinations found by SimpleGA variant approach reduce the error rate by 9.175% and 11.626% for ResNet51_align and ResNet51 respectively.

Using Evolutionary algorithm for data augmentation is an emerging research field in recent years. In this thesis, I use an approach inspired by SimpleGA to search the perturbation combinations, in which the chromosome of an individual is the weight and perturbation parameters. Based on the results, this approach is not effective enough, other SimpleGA methods could be experimented to improve the performance. Besides, other large test dataset could be adopted for more precise results, and the model could be optimized by using other loss functions.

Appendix A

Attachments

Table A.1: Single Perturbation Results of ResNet51_align Part 1 (Landmarks). Detected landmarks are used in perturbations.

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
1	[1.0, -10, -10, -10]	0.95455	0.95	46	[1.1, 10, -10, -10]	0.95	0.944
2	[1.0, -10, -10, 0]	0.95091	0.953	47	[1.1, 10, -10, 0]	0.94818	0.952
3	[1.0, -10, -10, 10]	0.95182	0.953	48	[1.1, 10, -10, 10]	0.95091	0.95
4	[1.0, -10, 0, -10]	0.95455	0.961	49	[1.1, 10, 0, -10]	0.95364	0.959
5	[1.0, -10, 0, 0]	0.95636	0.96	50	[1.1, 10, 0, 0]	0.95455	0.953
6	[1.0, -10, 0, 10]	0.95455	0.962	51	[1.1, 10, 0, 10]	0.95545	0.958
7	[1.0, -10, 10, -10]	0.95091	0.949	52	[1.1, 10, 10, -10]	0.94818	0.949
8	[1.0, -10, 10, 0]	0.95182	0.949	53	[1.1, 10, 10, 0]	0.94818	0.955
9	[1.0, -10, 10, 10]	0.94636	0.95	54	[1.1, 10, 10, 10]	0.95091	0.948
10	[1.0, 0, -10, -10]	0.95182	0.951	55	[1.2, -10, -10, -10]	0.95909	0.955
11	[1.0, 0, -10, 0]	0.95273	0.961	56	[1.2, -10, -10, 0]	0.95273	0.953
12	[1.0, 0, -10, 10]	0.95455	0.953	57	[1.2, -10, -10, 10]	0.95455	0.947
13	[1.0, 0, 0, -10]	0.95455	0.955	58	[1.2, -10, 0, -10]	0.95727	0.958
14	[1.0, 0, 0, 0]	0.95545	0.963	59	[1.2, -10, 0, 0]	0.95636	0.954
15	[1.0, 0, 0, 10]	0.94909	0.953	60	[1.2, -10, 0, 10]	0.95909	0.963
16	[1.0, 0, 10, -10]	0.95455	0.951	61	[1.2, -10, 10, -10]	0.96	0.957
17	[1.0, 0, 10, 0]	0.94909	0.959	62	[1.2, -10, 10, 0]	0.95727	0.951
18	[1.0, 0, 10, 10]	0.95273	0.948	63	[1.2, -10, 10, 10]	0.95455	0.95
19	[1.0, 10, -10, -10]	0.94818	0.95	64	[1.2, 0, -10, -10]	0.95364	0.948
20	[1.0, 10, -10, 0]	0.94545	0.951	65	[1.2, 0, -10, 0]	0.95455	0.954
21	[1.0, 10, -10, 10]	0.94909	0.953	66	[1.2, 0, -10, 10]	0.95636	0.951
22	[1.0, 10, 0, -10]	0.94909	0.951	67	[1.2, 0, 0, -10]	0.95727	0.954
23	[1.0, 10, 0, 0]	0.94364	0.959	68	[1.2, 0, 0, 0]	0.95909	0.952
24	[1.0, 10, 0, 10]	0.95091	0.955	69	[1.2, 0, 0, 10]	0.95636	0.949
25	[1.0, 10, 10, -10]	0.94364	0.95	70	[1.2, 0, 10, -10]	0.95636	0.958
26	[1.0, 10, 10, 0]	0.94545	0.959	71	[1.2, 0, 10, 0]	0.95727	0.96
27	[1.0, 10, 10, 10]	0.94273	0.951	72	[1.2, 0, 10, 10]	0.95545	0.961
28	[1.1, -10, -10, -10]	0.95364	0.952	73	[1.2, 10, -10, -10]	0.95364	0.95
29	[1.1, -10, -10, 0]	0.95636	0.956	74	[1.2, 10, -10, 0]	0.95182	0.944
30	[1.1, -10, -10, 10]	0.95455	0.957	75	[1.2, 10, -10, 10]	0.95273	0.951
31	[1.1, -10, 0, -10]	0.96	0.965	76	[1.2, 10, 0, -10]	0.95545	0.96
32	[1.1, -10, 0, 0]	0.95909	0.961	77	[1.2, 10, 0, 0]	0.95364	0.95
33	[1.1, -10, 0, 10]	0.96	0.96	78	[1.2, 10, 0, 10]	0.95273	0.958
34	[1.1, -10, 10, -10]	0.95455	0.951	79	[1.2, 10, 10, -10]	0.95545	0.951
35	[1.1, -10, 10, 0]	0.95455	0.949	80	[1.2, 10, 10, 0]	0.94818	0.954
36	[1.1, -10, 10, 10]	0.95273	0.954	81	[1.2, 10, 10, 10]	0.95091	0.951
37	[1.1, 0, -10, -10]	0.95182	0.958	82	[1.3, -10, -10, -10]	0.95818	0.951
38	[1.1, 0, -10, 0]	0.95182	0.959	83	[1.3, -10, -10, 0]	0.95455	0.948
39	[1.1, 0, -10, 10]	0.95364	0.953	84	[1.3, -10, -10, 10]	0.95273	0.944
40	[1.1, 0, 0, -10]	0.96	0.96	85	[1.3, -10, 0, -10]	0.96091	0.946
41	[1.1, 0, 0, 0]	0.96	0.96	86	[1.3, -10, 0, 0]	0.95545	0.951
42	[1.1, 0, 0, 10]	0.95455	0.957	87	[1.3, -10, 0, 10]	0.95455	0.954
43	[1.1, 0, 10, -10]	0.95636	0.952	88	[1.3, -10, 10, -10]	0.95364	0.95
44	[1.1, 0, 10, 0]	0.95	0.958	89	[1.3, -10, 10, 0]	0.95545	0.946
45	[1.1, 0, 10, 10]	0.95727	0.95	90	[1.3, -10, 10, 10]	0.95182	0.946

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
91	[1.3, 0, -10, -10]	0.95455	0.953	114	[1.4, -10, 0, 10]	0.95182	0.944
92	[1.3, 0, -10, 0]	0.95545	0.952	115	[1.4, -10, 10, -10]	0.95182	0.94
93	[1.3, 0, -10, 10]	0.95273	0.943	116	[1.4, -10, 10, 0]	0.95	0.949
94	[1.3, 0, 0, -10]	0.95636	0.947	117	[1.4, -10, 10, 10]	0.95273	0.94
95	[1.3, 0, 0, 0]	0.96	0.944	118	[1.4, 0, -10, -10]	0.95273	0.949
96	[1.3, 0, 0, 10]	0.95545	0.943	119	[1.4, 0, -10, 0]	0.94909	0.949
97	[1.3, 0, 10, -10]	0.95545	0.952	120	[1.4, 0, -10, 10]	0.95182	0.934
98	[1.3, 0, 10, 0]	0.95455	0.955	121	[1.4, 0, 0, -10]	0.95545	0.943
99	[1.3, 0, 10, 10]	0.95364	0.949	122	[1.4, 0, 0, 0]	0.95636	0.942
100	[1.3, 10, -10, -10]	0.95273	0.943	123	[1.4, 0, 0, 10]	0.95455	0.937
101	[1.3, 10, -10, 0]	0.95455	0.948	124	[1.4, 0, 10, -10]	0.95455	0.95
102	[1.3, 10, -10, 10]	0.95182	0.948	125	[1.4, 0, 10, 0]	0.95636	0.95
103	[1.3, 10, 0, -10]	0.95364	0.95	126	[1.4, 0, 10, 10]	0.95364	0.945
104	[1.3, 10, 0, 0]	0.95455	0.948	127	[1.4, 10, -10, -10]	0.94818	0.949
105	[1.3, 10, 0, 10]	0.95545	0.948	128	[1.4, 10, -10, 0]	0.94455	0.949
106	[1.3, 10, 10, -10]	0.95364	0.952	129	[1.4, 10, -10, 10]	0.94909	0.954
107	[1.3, 10, 10, 0]	0.95182	0.957	130	[1.4, 10, 0, -10]	0.95091	0.941
108	[1.3, 10, 10, 10]	0.95091	0.942	131	[1.4, 10, 0, 0]	0.95273	0.945
109	[1.4, -10, -10, -10]	0.95182	0.946	132	[1.4, 10, 0, 10]	0.95	0.939
110	[1.4, -10, -10, 0]	0.95545	0.949	133	[1.4, 10, 10, -10]	0.94636	0.951
111	[1.4, -10, -10, 10]	0.95273	0.944	134	[1.4, 10, 10, 0]	0.94818	0.95
112	[1.4, -10, 0, -10]	0.95273	0.945	135	[1.4, 10, 10, 10]	0.95	0.946
113	[1.4, -10, 0, 0]	0.95364	0.944				

Table A.2: Single Perturbation Results of ResNet51_align Part 2 (Landmarks). Detected landmarks are used in perturbations.

Table A.3: Single Perturbation Results of ResNet51 Part 1 (Landmarks) . Detected landmarks are used in perturbations.

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
1	[1.0, -10, -10, -10]	0.96727	0.96	46	[1.1, 10, -10, -10]	0.95818	0.967
2	[1.0, -10, -10, 0]	0.96091	0.962	47	[1.1, 10, -10, 0]	0.96	0.968
3	[1.0, -10, -10, 10]	0.96	0.961	48	[1.1, 10, -10, 10]	0.95636	0.964
4	[1.0, -10, 0, -10]	0.96	0.961	49	[1.1, 10, 0, -10]	0.95818	0.962
5	[1.0, -10, 0, 0]	0.95909	0.964	50	[1.1, 10, 0, 0]	0.95909	0.964
6	[1.0, -10, 0, 10]	0.95909	0.962	51	[1.1, 10, 0, 10]	0.95818	0.962
7	[1.0, -10, 10, -10]	0.96182	0.955	52	[1.1, 10, 10, -10]	0.95909	0.968
8	[1.0, -10, 10, 0]	0.96	0.959	53	[1.1, 10, 10, 0]	0.96	0.961
9	[1.0, -10, 10, 10]	0.96	0.963	54	[1.1, 10, 10, 10]	0.96091	0.957
10	[1.0, 0, -10, -10]	0.95818	0.958	55	[1.2, -10, -10, -10]	0.95545	0.964
11	[1.0, 0, -10, 0]	0.95455	0.966	56	[1.2, -10, -10, 0]	0.95818	0.962
12	[1.0, 0, -10, 10]	0.95364	0.964	57	[1.2, -10, -10, 10]	0.95636	0.963
13	[1.0, 0, 0, -10]	0.95636	0.961	58	[1.2, -10, 0, -10]	0.95727	0.963
14	[1.0, 0, 0, 0]	0.95545	0.962	59	[1.2, -10, 0, 0]	0.95636	0.961
15	[1.0, 0, 0, 10]	0.95636	0.963	60	[1.2, -10, 0, 10]	0.95636	0.961
16	[1.0, 0, 10, -10]	0.96091	0.967	61	[1.2, -10, 10, -10]	0.95909	0.955
17	[1.0, 0, 10, 0]	0.95545	0.959	62	[1.2, -10, 10, 0]	0.96	0.96
18	[1.0, 0, 10, 10]	0.95909	0.962	63	[1.2, -10, 10, 10]	0.95545	0.961
19	[1.0, 10, -10, -10]	0.95727	0.968	64	[1.2, 0, -10, -10]	0.95727	0.961
20	[1.0, 10, -10, 0]	0.96091	0.959	65	[1.2, 0, -10, 0]	0.95727	0.961
21	[1.0, 10, -10, 10]	0.95818	0.965	66	[1.2, 0, -10, 10]	0.95727	0.963
22	[1.0, 10, 0, -10]	0.95455	0.968	67	[1.2, 0, 0, -10]	0.96	0.957
23	[1.0, 10, 0, 0]	0.96091	0.96	68	[1.2, 0, 0, 0]	0.95636	0.956
24	[1.0, 10, 0, 10]	0.96273	0.966	69	[1.2, 0, 0, 10]	0.95364	0.959
25	[1.0, 10, 10, -10]	0.96091	0.962	70	[1.2, 0, 10, -10]	0.95909	0.959
26	[1.0, 10, 10, 0]	0.96091	0.961	71	[1.2, 0, 10, 0]	0.96	0.96
27	[1.0, 10, 10, 10]	0.95727	0.96	72	[1.2, 0, 10, 10]	0.95364	0.957
28	[1.1, -10, -10, -10]	0.96	0.966	73	[1.2, 10, -10, -10]	0.95727	0.963
29	[1.1, -10, -10, 0]	0.95909	0.964	74	[1.2, 10, -10, 0]	0.95818	0.966
30	[1.1, -10, -10, 10]	0.95727	0.969	75	[1.2, 10, -10, 10]	0.95636	0.961
31	[1.1, -10, 0, -10]	0.96091	0.962	76	[1.2, 10, 0, -10]	0.95909	0.962
32	[1.1, -10, 0, 0]	0.95636	0.956	77	[1.2, 10, 0, 0]	0.95545	0.963
33	[1.1, -10, 0, 10]	0.95909	0.963	78	[1.2, 10, 0, 10]	0.95636	0.962
34	[1.1, -10, 10, -10]	0.96091	0.958	79	[1.2, 10, 10, -10]	0.95727	0.959
35	[1.1, -10, 10, 0]	0.96182	0.957	80	[1.2, 10, 10, 0]	0.95727	0.96
36	[1.1, -10, 10, 10]	0.95909	0.966	81	[1.2, 10, 10, 10]	0.95818	0.955
37	[1.1, 0, -10, -10]	0.95909	0.958	82	[1.3, -10, -10, -10]	0.96	0.959
38	[1.1, 0, -10, 0]	0.95636	0.958	83	[1.3, -10, -10, 0]	0.95727	0.962
39	[1.1, 0, -10, 10]	0.95455	0.959	84	[1.3, -10, -10, 10]	0.95545	0.96
40	[1.1, 0, 0, -10]	0.96	0.959	85	[1.3, -10, 0, -10]	0.95364	0.961
41	[1.1, 0, 0, 0]	0.95636	0.961	86	[1.3, -10, 0, 0]	0.95364	0.96
42	[1.1, 0, 0, 10]	0.95545	0.963	87	[1.3, -10, 0, 10]	0.95273	0.965
43	[1.1, 0, 10, -10]	0.96091	0.962	88	[1.3, -10, 10, -10]	0.95909	0.958
44	[1.1, 0, 10, 0]	0.95909	0.961	89	[1.3, -10, 10, 0]	0.95727	0.959
45	[1.1, 0, 10, 10]	0.95818	0.961	90	[1.3, -10, 10, 10]	0.95545	0.957

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
91	[1.3, 0, -10, -10]	0.95727	0.957	114	[1.4, -10, 0, 10]	0.95182	0.962
92	[1.3, 0, -10, 0]	0.95909	0.957	115	[1.4, -10, 10, -10]	0.95364	0.959
93	[1.3, 0, -10, 10]	0.95	0.962	116	[1.4, -10, 10, 0]	0.95636	0.958
94	[1.3, 0, 0, -10]	0.95364	0.951	117	[1.4, -10, 10, 10]	0.95455	0.958
95	[1.3, 0, 0, 0]	0.95545	0.958	118	[1.4, 0, -10, -10]	0.95455	0.958
96	[1.3, 0, 0, 10]	0.95364	0.957	119	[1.4, 0, -10, 0]	0.95636	0.958
97	[1.3, 0, 10, -10]	0.95545	0.959	120	[1.4, 0, -10, 10]	0.95545	0.959
98	[1.3, 0, 10, 0]	0.95636	0.957	121	[1.4, 0, 0, -10]	0.95455	0.953
99	[1.3, 0, 10, 10]	0.95364	0.955	122	[1.4, 0, 0, 0]	0.95545	0.951
100	[1.3, 10, -10, -10]	0.95818	0.961	123	[1.4, 0, 0, 10]	0.95455	0.952
101	[1.3, 10, -10, 0]	0.95909	0.96	124	[1.4, 0, 10, -10]	0.95273	0.954
102	[1.3, 10, -10, 10]	0.95727	0.963	125	[1.4, 0, 10, 0]	0.95545	0.952
103	[1.3, 10, 0, -10]	0.95364	0.958	126	[1.4, 0, 10, 10]	0.95455	0.955
104	[1.3, 10, 0, 0]	0.95545	0.959	127	[1.4, 10, -10, -10]	0.95545	0.96
105	[1.3, 10, 0, 10]	0.95364	0.959	128	[1.4, 10, -10, 0]	0.95273	0.956
106	[1.3, 10, 10, -10]	0.95727	0.956	129	[1.4, 10, -10, 10]	0.95455	0.959
107	[1.3, 10, 10, 0]	0.95636	0.96	130	[1.4, 10, 0, -10]	0.95545	0.955
108	[1.3, 10, 10, 10]	0.95636	0.959	131	[1.4, 10, 0, 0]	0.95182	0.958
109	[1.4, -10, -10, -10]	0.95545	0.961	132	[1.4, 10, 0, 10]	0.95273	0.953
110	[1.4, -10, -10, 0]	0.95636	0.966	133	[1.4, 10, 10, -10]	0.95636	0.955
111	[1.4, -10, -10, 10]	0.95455	0.958	134	[1.4, 10, 10, 0]	0.95273	0.956
112	[1.4, -10, 0, -10]	0.95364	0.966	135	[1.4, 10, 10, 10]	0.95364	0.958
113	[1.4, -10, 0, 0]	0.95273	0.958				

Table A.4: Single Perturbation Results of ResNet51 Part 2 (Landmarks) . Detected landmarks are used in perturbations.

Table A.5: Single Perturbation Results of ResNet51_align Part 1 (Bounding Box). Detected bounding boxes are used in perturbations.

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
1	[0.5, -10, -10, -10]	0.95273	0.951	46	[0.6, 10, -10, -10]	0.94818	0.949
2	[0.5, -10, -10, 0]	0.94818	0.958	47	[0.6, 10, -10, 0]	0.95182	0.95
3	[0.5, -10, -10, 10]	0.95364	0.957	48	[0.6, 10, -10, 10]	0.95273	0.953
4	[0.5, -10, 0, -10]	0.95636	0.958	49	[0.6, 10, 0, -10]	0.95091	0.954
5	[0.5, -10, 0, 0]	0.95545	0.959	50	[0.6, 10, 0, 0]	0.95182	0.946
6	[0.5, -10, 0, 10]	0.96	0.956	51	[0.6, 10, 0, 10]	0.95182	0.948
7	[0.5, -10, 10, -10]	0.95455	0.949	52	[0.6, 10, 10, -10]	0.95091	0.948
8	[0.5, -10, 10, 0]	0.95273	0.958	53	[0.6, 10, 10, 0]	0.95	0.947
9	[0.5, -10, 10, 10]	0.94545	0.954	54	[0.6, 10, 10, 10]	0.95273	0.95
10	[0.5, 0, -10, -10]	0.95182	0.962	55	[0.7, -10, -10, -10]	0.94818	0.948
11	[0.5, 0, -10, 0]	0.95273	0.951	56	[0.7, -10, -10, 0]	0.94727	0.951
12	[0.5, 0, -10, 10]	0.94636	0.955	57	[0.7, -10, -10, 10]	0.94909	0.941
13	[0.5, 0, 0, -10]	0.95636	0.958	58	[0.7, -10, 0, -10]	0.95273	0.942
14	[0.5, 0, 0, 0]	0.95818	0.962	59	[0.7, -10, 0, 0]	0.95364	0.944
15	[0.5, 0, 0, 10]	0.95545	0.963	60	[0.7, -10, 0, 10]	0.94909	0.94
16	[0.5, 0, 10, -10]	0.95545	0.961	61	[0.7, -10, 10, -10]	0.95364	0.945
17	[0.5, 0, 10, 0]	0.95091	0.955	62	[0.7, -10, 10, 0]	0.95636	0.952
18	[0.5, 0, 10, 10]	0.94909	0.956	63	[0.7, -10, 10, 10]	0.95545	0.944
19	[0.5, 10, -10, -10]	0.95	0.951	64	[0.7, 0, -10, -10]	0.95182	0.942
20	[0.5, 10, -10, 0]	0.94636	0.954	65	[0.7, 0, -10, 0]	0.95091	0.946
21	[0.5, 10, -10, 10]	0.94	0.956	66	[0.7, 0, -10, 10]	0.95091	0.948
22	[0.5, 10, 0, -10]	0.95	0.953	67	[0.7, 0, 0, -10]	0.95091	0.947
23	[0.5, 10, 0, 0]	0.95364	0.955	68	[0.7, 0, 0, 0]	0.95182	0.94
24	[0.5, 10, 0, 10]	0.95455	0.953	69	[0.7, 0, 0, 10]	0.95091	0.943
25	[0.5, 10, 10, -10]	0.94727	0.958	70	[0.7, 0, 10, -10]	0.95273	0.949
26	[0.5, 10, 10, 0]	0.94636	0.955	71	[0.7, 0, 10, 0]	0.95364	0.947
27	[0.5, 10, 10, 10]	0.95	0.955	72	[0.7, 0, 10, 10]	0.95091	0.953
28	[0.6, -10, -10, -10]	0.95455	0.955	73	[0.7, 10, -10, -10]	0.94091	0.949
29	[0.6, -10, -10, 0]	0.95	0.944	74	[0.7, 10, -10, 0]	0.94545	0.95
30	[0.6, -10, -10, 10]	0.95091	0.946	75	[0.7, 10, -10, 10]	0.94727	0.954
31	[0.6, -10, 0, -10]	0.95636	0.948	76	[0.7, 10, 0, -10]	0.94818	0.941
32	[0.6, -10, 0, 0]	0.95727	0.951	77	[0.7, 10, 0, 0]	0.94727	0.943
33	[0.6, -10, 0, 10]	0.95545	0.951	78	[0.7, 10, 0, 10]	0.95	0.949
34	[0.6, -10, 10, -10]	0.95273	0.954	79	[0.7, 10, 10, -10]	0.94727	0.944
35	[0.6, -10, 10, 0]	0.95818	0.955	80	[0.7, 10, 10, 0]	0.94636	0.944
36	[0.6, -10, 10, 10]	0.95273	0.952	81	[0.7, 10, 10, 10]	0.94909	0.944
37	[0.6, 0, -10, -10]	0.95182	0.948	82	[0.8, -10, -10, -10]	0.94182	0.941
38	[0.6, 0, -10, 0]	0.95364	0.957	83	[0.8, -10, -10, 0]	0.94727	0.932
39	[0.6, 0, -10, 10]	0.95636	0.946	84	[0.8, -10, -10, 10]	0.94455	0.938
40	[0.6, 0, 0, -10]	0.95909	0.949	85	[0.8, -10, 0, -10]	0.94364	0.935
41	[0.6, 0, 0, 0]	0.95364	0.943	86	[0.8, -10, 0, 0]	0.94455	0.93
42	[0.6, 0, 0, 10]	0.95455	0.946	87	[0.8, -10, 0, 10]	0.94455	0.934
43	[0.6, 0, 10, -10]	0.95273	0.955	88	[0.8, -10, 10, -10]	0.94182	0.937
44	[0.6, 0, 10, 0]	0.95364	0.95	89	[0.8, -10, 10, 0]	0.94273	0.938
45	[0.6, 0, 10, 10]	0.95455	0.952	90	[0.8, -10, 10, 10]	0.94273	0.934

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
91	[0.8, 0, -10, -10]	0.94727	0.941	114	[0.9, -10, 0, 10]	0.92727	0.923
92	[0.8, 0, -10, 0]	0.94818	0.943	115	[0.9, -10, 10, -10]	0.93455	0.929
93	[0.8, 0, -10, 10]	0.94727	0.938	116	[0.9, -10, 10, 0]	0.93273	0.922
94	[0.8, 0, 0, -10]	0.95	0.945	117	[0.9, -10, 10, 10]	0.93182	0.922
95	[0.8, 0, 0, 0]	0.94636	0.937	118	[0.9, 0, -10, -10]	0.93727	0.924
96	[0.8, 0, 0, 10]	0.94636	0.934	119	[0.9, 0, -10, 0]	0.92818	0.929
97	[0.8, 0, 10, -10]	0.94727	0.935	120	[0.9, 0, -10, 10]	0.93364	0.928
98	[0.8, 0, 10, 0]	0.94273	0.935	121	[0.9, 0, 0, -10]	0.94	0.936
99	[0.8, 0, 10, 10]	0.94364	0.935	122	[0.9, 0, 0, 0]	0.93182	0.923
100	[0.8, 10, -10, -10]	0.93818	0.938	123	[0.9, 0, 0, 10]	0.93182	0.927
101	[0.8, 10, -10, 0]	0.94	0.94	124	[0.9, 0, 10, -10]	0.93727	0.937
102	[0.8, 10, -10, 10]	0.94091	0.938	125	[0.9, 0, 10, 0]	0.93636	0.93
103	[0.8, 10, 0, -10]	0.94091	0.936	126	[0.9, 0, 10, 10]	0.93364	0.924
104	[0.8, 10, 0, 0]	0.94273	0.949	127	[0.9, 10, -10, -10]	0.92	0.919
105	[0.8, 10, 0, 10]	0.94545	0.937	128	[0.9, 10, -10, 0]	0.92727	0.926
106	[0.8, 10, 10, -10]	0.93909	0.942	129	[0.9, 10, -10, 10]	0.93	0.92
107	[0.8, 10, 10, 0]	0.94	0.943	130	[0.9, 10, 0, -10]	0.93	0.923
108	[0.8, 10, 10, 10]	0.93818	0.938	131	[0.9, 10, 0, 0]	0.92818	0.924
109	[0.9, -10, -10, -10]	0.93636	0.927	132	[0.9, 10, 0, 10]	0.93182	0.927
110	[0.9, -10, -10, 0]	0.93636	0.929	133	[0.9, 10, 10, -10]	0.92909	0.928
111	[0.9, -10, -10, 10]	0.93182	0.931	134	[0.9, 10, 10, 0]	0.92545	0.923
112	[0.9, -10, 0, -10]	0.93091	0.923	135	[0.9, 10, 10, 10]	0.92818	0.921
113	[0.9, -10, 0, 0]	0.93545	0.923				

Table A.6: Single Perturbation Results of ResNet51_align Part 2 (Bounding Box). Detected bounding boxes are used in perturbations.

Table A.7: Single Perturbation Results of ResNet51 Part 1 (Bounding Box). Detected bounding boxes are used in perturbations.

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
1	[0.5, -10, -10, -10]	0.96091	0.962	46	[0.6, 10, -10, -10]	0.95818	0.962
2	[0.5, -10, -10, 0]	0.96273	0.967	47	[0.6, 10, -10, 0]	0.95545	0.954
3	[0.5, -10, -10, 10]	0.95909	0.968	48	[0.6, 10, -10, 10]	0.95545	0.953
4	[0.5, -10, 0, -10]	0.96091	0.966	49	[0.6, 10, 0, -10]	0.95818	0.962
5	[0.5, -10, 0, 0]	0.95818	0.969	50	[0.6, 10, 0, 0]	0.95455	0.962
6	[0.5, -10, 0, 10]	0.95727	0.968	51	[0.6, 10, 0, 10]	0.95636	0.957
7	[0.5, -10, 10, -10]	0.96182	0.961	52	[0.6, 10, 10, -10]	0.96091	0.961
8	[0.5, -10, 10, 0]	0.96273	0.961	53	[0.6, 10, 10, 0]	0.95818	0.957
9	[0.5, -10, 10, 10]	0.95909	0.963	54	[0.6, 10, 10, 10]	0.95818	0.957
10	[0.5, 0, -10, -10]	0.95909	0.963	55	[0.7, -10, -10, -10]	0.95364	0.958
11	[0.5, 0, -10, 0]	0.95909	0.959	56	[0.7, -10, -10, 0]	0.95455	0.961
12	[0.5, 0, -10, 10]	0.95727	0.959	57	[0.7, -10, -10, 10]	0.95182	0.962
13	[0.5, 0, 0, -10]	0.95545	0.96	58	[0.7, -10, 0, -10]	0.95273	0.953
14	[0.5, 0, 0, 0]	0.95545	0.962	59	[0.7, -10, 0, 0]	0.95364	0.957
15	[0.5, 0, 0, 10]	0.95	0.961	60	[0.7, -10, 0, 10]	0.95182	0.955
16	[0.5, 0, 10, -10]	0.96364	0.957	61	[0.7, -10, 10, -10]	0.95455	0.955
17	[0.5, 0, 10, 0]	0.95909	0.959	62	[0.7, -10, 10, 0]	0.95364	0.958
18	[0.5, 0, 10, 10]	0.96091	0.962	63	[0.7, -10, 10, 10]	0.95273	0.959
19	[0.5, 10, -10, -10]	0.95727	0.964	64	[0.7, 0, -10, -10]	0.95182	0.955
20	[0.5, 10, -10, 0]	0.96182	0.963	65	[0.7, 0, -10, 0]	0.95273	0.957
21	[0.5, 10, -10, 10]	0.95636	0.965	66	[0.7, 0, -10, 10]	0.95182	0.953
22	[0.5, 10, 0, -10]	0.95727	0.961	67	[0.7, 0, 0, -10]	0.95364	0.957
23	[0.5, 10, 0, 0]	0.96364	0.967	68	[0.7, 0, 0, 0]	0.95364	0.952
24	[0.5, 10, 0, 10]	0.96	0.964	69	[0.7, 0, 0, 10]	0.95545	0.952
25	[0.5, 10, 10, -10]	0.96	0.966	70	[0.7, 0, 10, -10]	0.95455	0.955
26	[0.5, 10, 10, 0]	0.96	0.96	71	[0.7, 0, 10, 0]	0.95182	0.952
27	[0.5, 10, 10, 10]	0.95818	0.962	72	[0.7, 0, 10, 10]	0.95273	0.954
28	[0.6, -10, -10, -10]	0.95636	0.962	73	[0.7, 10, -10, -10]	0.95273	0.957
29	[0.6, -10, -10, 0]	0.95636	0.963	74	[0.7, 10, -10, 0]	0.95364	0.958
30	[0.6, -10, -10, 10]	0.95636	0.956	75	[0.7, 10, -10, 10]	0.95273	0.956
31	[0.6, -10, 0, -10]	0.95727	0.964	76	[0.7, 10, 0, -10]	0.95636	0.956
32	[0.6, -10, 0, 0]	0.95727	0.958	77	[0.7, 10, 0, 0]	0.95273	0.952
33	[0.6, -10, 0, 10]	0.95545	0.961	78	[0.7, 10, 0, 10]	0.95273	0.955
34	[0.6, -10, 10, -10]	0.95909	0.958	79	[0.7, 10, 10, -10]	0.95364	0.955
35	[0.6, -10, 10, 0]	0.95818	0.959	80	[0.7, 10, 10, 0]	0.95455	0.956
36	[0.6, -10, 10, 10]	0.95818	0.957	81	[0.7, 10, 10, 10]	0.95455	0.954
37	[0.6, 0, -10, -10]	0.95455	0.956	82	[0.8, -10, -10, -10]	0.94727	0.953
38	[0.6, 0, -10, 0]	0.95545	0.954	83	[0.8, -10, -10, 0]	0.94545	0.949
39	[0.6, 0, -10, 10]	0.95364	0.96	84	[0.8, -10, -10, 10]	0.94545	0.954
40	[0.6, 0, 0, -10]	0.95636	0.956	85	[0.8, -10, 0, -10]	0.94727	0.953
41	[0.6, 0, 0, 0]	0.95636	0.958	86	[0.8, -10, 0, 0]	0.94909	0.953
42	[0.6, 0, 0, 10]	0.95455	0.952	87	[0.8, -10, 0, 10]	0.94909	0.958
43	[0.6, 0, 10, -10]	0.95909	0.96	88	[0.8, -10, 10, -10]	0.95	0.959
44	[0.6, 0, 10, 0]	0.95545	0.954	89	[0.8, -10, 10, 0]	0.94909	0.956
45	[0.6, 0, 10, 10]	0.95636	0.953	90	[0.8, -10, 10, 10]	0.95091	0.958

No.	Perturbation	Train Acc	Test Acc	No.	Perturbation	Train Acc	Test Acc
91	[0.8, 0, -10, -10]	0.94727	0.952	114	[0.9, -10, 0, 10]	0.93727	0.948
92	[0.8, 0, -10, 0]	0.94727	0.948	115	[0.9, -10, 10, -10]	0.94273	0.946
93	[0.8, 0, -10, 10]	0.94818	0.948	116	[0.9, -10, 10, 0]	0.94273	0.948
94	[0.8, 0, 0, -10]	0.94727	0.953	117	[0.9, -10, 10, 10]	0.94182	0.95
95	[0.8, 0, 0, 0]	0.94364	0.952	118	[0.9, 0, -10, -10]	0.93636	0.948
96	[0.8, 0, 0, 10]	0.94727	0.945	119	[0.9, 0, -10, 0]	0.93818	0.946
97	[0.8, 0, 10, -10]	0.95273	0.955	120	[0.9, 0, -10, 10]	0.93545	0.944
98	[0.8, 0, 10, 0]	0.94727	0.946	121	[0.9, 0, 0, -10]	0.94	0.944
99	[0.8, 0, 10, 10]	0.95	0.952	122	[0.9, 0, 0, 0]	0.93909	0.943
100	[0.8, 10, -10, -10]	0.94727	0.953	123	[0.9, 0, 0, 10]	0.93818	0.942
101	[0.8, 10, -10, 0]	0.94545	0.949	124	[0.9, 0, 10, -10]	0.94091	0.949
102	[0.8, 10, -10, 10]	0.94818	0.95	125	[0.9, 0, 10, 0]	0.94273	0.944
103	[0.8, 10, 0, -10]	0.95091	0.958	126	[0.9, 0, 10, 10]	0.94091	0.941
104	[0.8, 10, 0, 0]	0.95	0.953	127	[0.9, 10, -10, -10]	0.93636	0.947
105	[0.8, 10, 0, 10]	0.94818	0.951	128	[0.9, 10, -10, 0]	0.94091	0.945
106	[0.8, 10, 10, -10]	0.94818	0.96	129	[0.9, 10, -10, 10]	0.93909	0.945
107	[0.8, 10, 10, 0]	0.94636	0.949	130	[0.9, 10, 0, -10]	0.94	0.946
108	[0.8, 10, 10, 10]	0.94909	0.953	131	[0.9, 10, 0, 0]	0.93818	0.95
109	[0.9, -10, -10, -10]	0.93818	0.942	132	[0.9, 10, 0, 10]	0.94091	0.95
110	[0.9, -10, -10, 0]	0.94	0.948	133	[0.9, 10, 10, -10]	0.94091	0.943
111	[0.9, -10, -10, 10]	0.93545	0.95	134	[0.9, 10, 10, 0]	0.94455	0.942
112	[0.9, -10, 0, -10]	0.94091	0.944	135	[0.9, 10, 10, 10]	0.94727	0.941
113	[0.9, -10, 0, 0]	0.93818	0.946				

Table A.8: Single Perturbation Results of ResNet51 Part 2 (Bounding Box). Detected bounding boxes are used in perturbations.

Iteration No.	Selected Perturbations No.	Train Accuracy	Test Accuracy
1	[16]	0.9636363636	0.957
2	[16, 3]	0.9654545455	0.968
3	[16, 3, 1]	0.9663636364	0.968
4	[16, 3, 1, 26]	0.9663636364	0.967
5	[16, 3, 1, 26, 2]	0.9663636364	0.968
6	[16, 3, 1, 26, 2, 4]	0.9663636364	0.965
7	[16, 3, 1, 26, 2, 4, 22]	0.9672727273	0.965
8	[16, 3, 1, 26, 2, 4, 22, 6]	0.9672727273	0.967
9	[16, 3, 1, 26, 2, 4, 22, 6, 8]	0.9672727273	0.967
10	[16, 3, 1, 26, 2, 4, 22, 6, 8, 24]	0.9672727273	0.967
11	[16, 3, 1, 26, 2, 4, 22, 6, 8, 24, 18]	0.9672727273	0.969
12	[16, 3, 1, 26, 2, 4, 22, 6, 8, 24, 18, 7]	0.9672727273	0.97
13	[16, 3, 1, 26, 2, 4, 22, 6, 8, 24, 18, 7, 23]	0.9672727273	0.968
14	[16, 3, 1, 26, 2, 4, 22, 6, 8, 24, 18, 7, 23, 5]	0.9672727273	0.969

Table A.9: Tracked Results of Greedy algorithm For ResNet51_align. The perturbation parameters $[\gamma_s, \gamma_a, \gamma_x, \gamma_y]$ of certain perturbation No. can be looked up in Appendix Table A.1-A.2. Detected Landmarks are used for perturbations.

Iteration No.	Selected Perturbations No.	Train Accuracy	Test Accuracy
1	[1]	0.9672727273	0.96
2	[1, 5]	0.9681818182	0.961
3	[1, 5, 7]	0.9681818182	0.964
4	[1, 5, 7, 4]	0.9681818182	0.963
5	[1, 5, 7, 4, 55]	0.9681818182	0.963
6	[1, 5, 7, 4, 55, 19]	0.9681818182	0.963
7	[1, 5, 7, 4, 55, 19, 56]	0.9681818182	0.963
8	[1, 5, 7, 4, 55, 19, 56, 28]	0.9681818182	0.962

Table A.10: Tracked Results of Greedy Algorithm for Resnet51. The perturbation parameters $[\gamma_s, \gamma_a, \gamma_x, \gamma_y]$ of certain perturbation No. can be looked up in Appendix Table A.3-A.4. Detected landmarks are used for perturbations.

Conditions	Selected Perturbations $[\gamma_s, \gamma_a, \gamma_x, \gamma_y]$	Weights	Training Accuracy	Test Accuracy
Elite selection: Top 12 Weight initialization: equal (5) Combination method: simple Generation replacement: cover	[1.1, 10, 10, 10] [1.1, 0, 10, -10] [1.1, -10, 10, 10] [1.0, 0, 0, 0] [1.1, 0, 10, 0] [1.2, -10, 10, -10] [1.2, -10, -10, 0] [1.1, -10, 10, 10] [1.3, -10, 10, -10]	12.5% 12.5% 12.5% 7.5% 17.5% 12.5% 12.5% 2.5% 10%	0.96182	0.966
Elite selection: Top 12 Weight initialization: equal (5) Combination method: simple Generation replacement: cover	[1.3, 10, -10, 10] [1.1, -10, 0, -10] [1.0, -10, 0, -10] [1.0, 0, -10, 10] [1.3, 10, 10, -10]	23.077% 15.385% 34.615% 15.385% 11.538%	0.96182	0.966
Elite selection: Top 12 Weight initialization: equal (5) Combination method: simple Generation replacement: cover	[1.2, 0, -10, -10] [1.3, 0, 10, -10] [1.1, 10, 0, -10] [1.0, 0, 0, 10] [1.3, 10, 10, -10]	25% 12.5% 33.333% 16.667% 12.5%	0.96182	0.965

Table A.11: Top 3 Weighted Perturbation Combinations Found by SimpleGA Variant Approach for Resnet51_align. Elite selection: select the 12 single perturbations that have the top 12 greatest accuracy on the training set into the parent pool. Weight initialization: equally initializing weights as 5. Combination method: choose the combination of perturbations and calculate the verification accuracy by the 'simple' method; Generation replacement: the parent generation is replaced by the children generation totally. Detected landmarks are used for perturbations. Accuracy is rounded to 5 decimal places, zeros at the end are omitted.

Conditions	Selected Perturbations $[\gamma_s, \gamma_a, \gamma_x, \gamma_y]$	Weights	Training Accuracy	Test Accuracy
Elite selection: Top 5	[1.0, -10, 10, 0]	25%	0.96	0.968
Weight initialization: unequal ([1, 5])	[1.2, 10, 0, 0]	25%		
Combination method: simple	[1.1, -10, -10, 0]	31.25%		
Generation replacement: cover	[1.0, 10, 10, 0]	18.75%		
Elite selection: Top 10	[1.2, 0, -10, 10]	8.333%	0.95818	0.967
Weight initialization: equal (5)	[1.0, 0, -10, 0]	19.444%		
Combination method: simple	[1.2, -10, 0, 0]	19.444%		
Generation replacement: cover	[1.2, 10, 0, 0]	13.889%		
	[1.1, 0, 0, 10]	19.444%		
	[1.3, 10, 10, -10]	8.333%		
	[1.1, -10, 10, 10]	8.333%		
	[1.3, -10, 0, 10]	2.778%		
Elite selection: Top 5	[1.2, 0, -10, -10]	36.364%	0.95636	0.966
Weight initialization: equal (5)	[1.3, -10, 0, 10]	36.364%		
Combination method: simple	[1.0, 0, -10, 0]	27.273%		
Generation replacement: cover				

Table A.12: Top 3 Weighted Perturbation Combinations Found by SimpleGA Variant Approach for Resnet51. "Elite selection: Top 5" means select the 5 single perturbations that have the top 5 greatest accuracy on the training set. "Weight initialization: unequal([1, 5])" means initializing each weight by randomly selecting an integer in the range [1,5]. "Combination method: simple" means to choose the combination of perturbation by the 'simple' method. "Generation replacement: cover" means the parent generation is replaced by the children generation totally. Detected landmarks are used for perturbations. Accuracy is rounded to 5 decimal places, zeros at the end are omitted.

List of Figures

2.1	Illustration of Flipping	4
2.2	Illustration of Cropping	4
2.3	Illustration of Scaling	5
2.4	Illustration of Rotation	5
2.5	Illustration of Shift	6
2.6	Illustration of Blurring	6
2.7	Process of Landmark Perturbation	7
2.8	Random Perturbations on CeleA	8
2.9	An Overview of Face Transformations	9
4.1	Architecture of ResNet51	16
4.2	Original and Modified-AFFACT Perturbed Images	19
4.3	Original and Non-aligned Images	20
4.4	Original and Aligned Images	21
5.1	Histograms of Training and Test Accuracy of ResNet51 and ResNet51_align	28
5.2	Illustration of SimpleGA Variant Approach	33
5.3	Illustration of Crossover	36
5.4	Illustration of Mutation	37
6.1	Histograms of Accuracy of ResNet51_align and ResNet51	44
6.2	LFW Image Samples with Different Scale Parameters	47

List of Tables

4.1	Training and Validation Transformers for ResNet51_align and ResNet51	18
5.1	Results of Baseline and Top Perturbation	29
5.2	Perturbation Combinations Found by Greedy Algorithm.	31
5.3	Hyperparameters of the SimpleGA Variant Approach	37
5.4	Weighted Perturbation Combinations Found by SimpleGA Variant Approach . . .	38
5.5	Result Comparison	39
5.6	P-Values Of T-Tests	40
5.7	T Tests to Compare Two Models at the Same Situations of Perturbation	40
6.1	Best Validation EERs	43
6.2	Result Comparison (Bounding Box)	45
6.3	P-Values of T-Tests (Bounding Box)	46
A.1	Single Perturbation Results of ResNet51_align Part 1 (Landmarks)	52
A.2	Single Perturbation Results of ResNet51_align Part 2 (Landmarks)	53
A.3	Single Perturbation Results of ResNet51 Part 1 (Landmarks)	54
A.4	Single Perturbation Results of ResNet51 Part 2 (Landmarks)	55
A.5	Single Perturbation Results of ResNet51_align Part 1 (Bounding Box)	56
A.6	Single Perturbation Results of ResNet51_align Part 2 (Bounding Box)	57
A.7	Single Perturbation Results of ResNet51 Part 1 (Bounding Box)	58
A.8	Single Perturbation Results of ResNet51 Part 2 (Bounding Box)	59
A.9	Tracked Results of Greedy algorithm For ResNet51_align	59
A.10	Tracked Results of Greedy Algorithm for Resnet51.	60
A.11	Top 3 Weighted Perturbation Combinations Found by SimpleGA Variant Approach for Resnet51_align	61
A.12	Top 3 Weighted Perturbation Combinations Found by SimpleGA Variant Approach for Resnet51	62

Bibliography

- Anjos, A., Günther, M., de Freitas Pereira, T., Korshunov, P., Mohammadi, A., and Marcel, S. (2017). Continuously reproducing toolchains in pattern recognition and machine learning experiments. In *International Conference on Machine Learning (ICML)*.
- Balas, B., Gable, J., and Pearson, H. (2018). The effects of blur and inversion on the recognition of ambient face images. *Perception*, 48:030100661881258.
- Beveridge, J., Phillips, P. J., Bolme, D., Draper, B., Given, G., Lui, Y., Teli, M., Zhang, H., Scruggs, W., Bowyer, K., Flynn, P., and Cheng, S. (2013). The challenge of face recognition from digital point-and-shoot cameras. pages 1–8.
- Cao, Q., Shen, L., Xie, W., Parkhi, O., and Zisserman, A. (2018). Vggface2: A dataset for recognising faces across pose and age. pages 67–74.
- Correia, J., Martins, T., and Machado, P. (2019). Evolutionary data augmentation in deep face detection. pages 163–164.
- Deng, J., Guo, J., Xue, N., and Zafeiriou, S. (2019a). Arcface: Additive angular margin loss for deep face recognition. pages 4685–4694.
- Deng, J., Trigeorgis, G., Zhou, Y., and Zafeiriou, S. (2019b). Joint multi-view face alignment in the wild. *IEEE Transactions on Image Processing*, PP:1–1.
- Fiche, C., Ladret, P., and Vu, N. S. (2010). Blurred face recognition algorithm guided by a no-reference blur metric. *Blurred face recognition algorithm guided by a no-reference blur metric*.
- Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. (2016). Ms-celeb-1m: A dataset and benchmark for large-scale face recognition.
- Günther, M., Rozsa, A., and Boulton, T. (2017). Affact: Alignment-free facial attribute classification technique. pages 90–99.
- Hasan, M. K. and Pal, C. (2011). Improving alignment of faces for recognition. pages 249 – 254.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. pages 770–778.
- Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. (2017). Deep learning scaling is predictable, empirically.
- Howard, A. (2013a). Some improvements on deep convolutional neural network based image classification.

- Howard, A. (2013b). Some improvements on deep convolutional neural network based image classification.
- Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-excitation networks. pages 7132–7141.
- Huang, G. B., Jain, V., and Learned-Miller, E. (2007). Unsupervised joint alignment of complex images. In *ICCV*.
- Jin, X. and Tan, X. (2016). Face alignment in-the-wild: A survey. *Computer Vision and Image Understanding*, 162.
- Kemelmacher, I., Seitz, S., Miller, D., and Brossard, E. (2016). The megaface benchmark: 1 million faces for recognition at scale. pages 4873–4882.
- Klare, B., Klein, B., Taborsky, E., Blanton, A., Cheney, J., Allen, K., Grother, P., Mah, A., and Jain, A. (2015). Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.
- Lim, S., Kim, I., Kim, T., Kim, C., and Kim, S. (2019). Fast autoaugment.
- 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)*.
- Lv, J.-J., Shao, X.-H., Huang, J.-S., Zhou, X.-D., and Zhou, X. (2016). Data augmentation for face recognition. *Neurocomputing*, 230.
- Mash, R., Borghetti, B., and Pecarina, J. (2016). Improved aircraft recognition for aerial refueling through data augmentation in convolutional neural networks. volume 10072.
- Masi, I., Trn, A., Hassner, T., Leksut, J., and Medioni, G. (2016). Do we really need to collect millions of faces for effective face recognition? volume 9909, pages 579–596.
- Maze, B., Adams, J., Duncan, J., Kalka, N., Miller, T., Otto, C., Jain, A., Niggel, W., Anderson, J., Cheney, J., and Grother, P. (2018). Iarpa janus benchmark - c: Face dataset and protocol. pages 158–165.
- Molchanov, D., Lyzhov, A., Molchanova, Y., Ashukha, A., and Vetrov, D. (2020). Greedy policy search: A simple baseline for learnable test-time augmentation.
- Nguyen, H. and Bai, L. (2010). Cosine similarity metric learning for face verification. volume 6493, pages 709–720.
- Parkhi, O. M., Vedaldi, A., and Zisserman, A. (2015). Deep face recognition. In *British Machine Vision Conference*.
- Paulin, M., Revaud, J., Harchaoui, Z., and Schmid, C. (2014). Transformation pursuit for image classification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.
- Phillips, P. (2017). A cross benchmark assessment of a deep convolutional neural network for face recognition. pages 705–710.

- Phillips, P. J., Beveridge, J., Draper, B., Givens, G., O'Toole, A., Bolme, D., Dunlop, J., Lui, Y., Sahibzada, H., and Weimer, S. (2011). An introduction to the good, the bad, the ugly face recognition challenge problem. pages 346–353.
- Qinghe, Z., Yang, M., Tian, X., Jiang, N., and Wang, D. (2020). A full stage data augmentation method in deep convolutional neural network for natural image classification. *Discrete Dynamics in Nature and Society*, 2020:1–11.
- Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *Proc. CVPR*.
- Shan, S., Chang, Y., Gao, W., Cao, B., and Yang, P. (2004). Curse of mis-alignment in face recognition: Problem and a novel mis-alignment learning solution. pages 314–320.
- Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.
- Sun, Y., Wang, X., and Tang, X. (2014). Deep learning face representation by joint identification-verification. *Proc. NIPS*, 27.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. pages 1–9.
- Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification.
- Taylor, L. and Nitschke, G. (2017). Improving deep learning using generic data augmentation.
- Terauchi, A. and Mori, N. (2021). pages 9851–9858.
- Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511.
- Wang, H., Wang, Y., Zhou, Z., Ji, X., Li, Z., Gong, D., Zhou, J., and Liu, W. (2018). Cosface: Large margin cosine loss for deep face recognition.
- Wang, N., Gao, X., Tao, D., and Liu, W. (2014). Facial feature point detection: A comprehensive survey. *Neurocomputing*, 275.
- Wang, X., Wang, K., and Lian, S. (2019). A survey on face data augmentation.
- Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. volume 9911, pages 499–515.
- Whitelam, C., Taborsky, E., Blanton, A., Maze, B., Adams, J., Miller, T., Kalka, N., Jain, A., Duncan, J., Allen, K., Cheney, J., and Grother, P. (2017). Iarpa janus benchmark-b face dataset. pages 592–600.
- Xie, S. and Tu, Z. (2017). Holistically-nested edge detection. *International Journal of Computer Vision*, 125:1–16.
- Xiong, X. and De la Torre, F. (2013). Supervised descent method and its applications to face alignment. pages 532–539.
- Xu, Y., Li, X., Yang, J., and Zhang, D. (2014). Integrate the original face image and its mirror image for face recognition. *Neurocomputing*, 131:191–199.

- Yang, H. and Patras, I. (2015). Mirror, mirror on the wall, tell me, is the error small? pages 4685–4693.
- Yi, D., Lei, Z., Liao, S., and Li, S. (2014). Learning face representation from scratch.
- Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multi-task cascaded convolutional networks. *IEEE Signal Processing Letters*, 23.
- Zhang, X., Fang, Z., Wen, Y., Li, Z., and Qiao, Y. (2017). Range loss for deep face recognition with long-tailed training data. pages 5419–5428.
- Zhou, E., Cao, Z., and Yin, q. (2015). Naive-deep face recognition: Touching the limit of lfw benchmark or not?