2022

Face Recognition with Attention Mechanisms

Qiangchang Wang
qw0007@mix.wvu.edu

Face Recognition with Attention Mechanisms

by

Qiangchang Wang

Dissertation submitted to the
Statler College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Computer Science

Guodong Guo, Ph.D., Chair
Donald Adjeroh, Ph.D.
Matthew Valenti, Ph.D.
Yuxin Liu, Ph.D.
Hong-jian Lai, Ph.D.

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2022

Keywords: Face Recognition; Attention Diversity; Attention Sparsity; Attention Erasing; Benchmark; Spatial Attentions; Attention Center Loss; Pyramid Attention; Multi-scale Features; Graph Convolutional Networks.

Abstract

Face Recognition with Attention Mechanisms

by

Qiangchang Wang
Doctor of Philosophy in Computer Science

West Virginia University

Guodong Guo, Ph.D., Chair

Face recognition has been widely used in people's daily lives due to its contactless process and high accuracy. Existing works can be divided into two categories: global and local approaches. The mainstream global approaches usually extract features on whole faces. However, global faces tend to suffer from dramatic appearance changes under the scenarios of large pose variations, heavy occlusions, and so on. On the other hand, since some local patches may remain similar, they can play an important role in such scenarios. Existing local approaches mainly rely on cropping local patches around facial landmarks and then extracting corresponding local representations. However, facial landmark detection may be inaccurate or even fail, which would limit their applications. To address this issue, attention mechanisms are applied to automatically locate discriminative facial parts, while suppressing noisy parts. Following this motivation, several models are proposed, including: the Local multi-Scale Convolutional Neural Networks (LS-CNN), Hierarchical Pyramid Diverse Attention (HPDA) networks, Contrastive Quality-aware Attentions (CQA-Face), Diverse and Sparse Attentions (DSA-Face), and Attention Augmented Networks (AAN-Face). Firstly, a novel spatial attention (local aggregation networks, LANet) is proposed to adaptively locate useful facial parts. Meanwhile, different facial parts may appear at different scales due to pose variations and expression changes. In order to solve this issue, LS-CNN are proposed to extract discriminative local information at different scales. Secondly, it is observed that some important facial parts may be neglected, if without a proper guidance. Besides, hierarchical features from different layers are not fully exploited which can contain rich low-level and high-level information. To overcome these two issues, HPDA are proposed. Specifically, a diverse learning is proposed to enlarge the Euclidean distances between each two spatial attention maps, locating diverse facial parts. Besides, hierarchical bilinear pooling is adopted to effectively combine features from different layers. Thirdly, despite the decent performance of the HPDA, the Euclidean distance may not be flexible enough to control the distances between each two attention maps. Further, it is also important to assign different quality scores for various local patches because various facial parts contain information with various importance, especially for faces with heavy occlusions, large pose variations, or quality changes. The CQA-Face is proposed which mainly consists of the contrastive attention learning and quality-aware networks where the former proposes a better distance function to enlarge the distances between each two attention maps and the latter applies a graph convolutional network to effectively learn the relations among different facial parts,

assigning higher quality scores for important patches and smaller values for less useful ones. Fourthly, the attention subset problem may occur where some attention maps are subsets of other attention maps. Consequently, the learned facial parts are not diverse enough to cover every facial detail, leading to inferior results. In our DSA-Face model, a new pairwise self-constrastive attention is proposed which considers the complement of subset attention maps in the loss function to address the aforementioned attention subset problem. Moreover, a attention sparsity loss is proposed to suppress the responses around noisy image regions, especially for masked faces. Lastly, in existing popular face datasets, some characteristics of facial images (e.g. frontal faces) are over-represented, while some characteristics (e.g. profile faces) are under-represented. In AAN-Face model, attention erasing is proposed to simulate various occlusion levels. Besides, attention center loss is proposed to control the responses on each attention map, guiding it to focus on the similar facial part. Our works have greatly improved the performance of cross-pose, cross-quality, cross-age, cross-modality, and masked face matching tasks.

Contents

List of Figures

List of Tables

List of Abbreviations

FR	face recognition
CNN	convolutional neural network
DL	deep learning
EER	equal error rate
FAR	false accept rate
FRR	false reject rate
TAR	true acceptance rate
PCA	principal component analysis
LDA	linear discriminant analysis
2DPCA	two-dimensional principal component analysis
EBGM	elastic bunch graph matching
LBP	local binary patterns
ReLU	rectified linear uUnits
LRN	local response normalization
BN	batch normalization

Chapter 1

Introduction

This chapter provides an introduction in this dissertation. Specifically, a conceptual background is introduced to provide context of face recognition. Related works are reviewed in the following section.

Biometric enables to recognize and authenticate people via the use of people's unique biological characteristics. A biometric authentication system collects a user's information and check it with the stored templates to determine if this user possesses the access right [15]. There are two types of biometrics, which are physiological measurements and behavioral measurements. The former contains fingerprints, vein patterns, irises, and faces, which do not change over time. The latter contains biological traits, including DNA, urine, voice recognition, and signature dynamics. These measurements are subject to change over time. Although the accuracy of Face Recognition (FR) as the biometric technology is lower than fingerprint recognition and iris recognition, it has been the prominent biometric technique and has been widely used in different areas due to its contactless process, such as human-computer interaction, surveillance and security, digital libraries, and so on.

FR is a technology which is capable of matching a human face with a facial image against a gallery set. FR consists of two sub-problems: face identification and face verification. For the former, as shown in Fig. 1.1, given a face image, the aim is to find the subject of this image in a gallery set. As for the latter, given a pairs of facial image, the target is to check if these face images belong to the same subject or not.

In the past several decades, FR has been a hot research topic. As shown in Fig. 1.2,

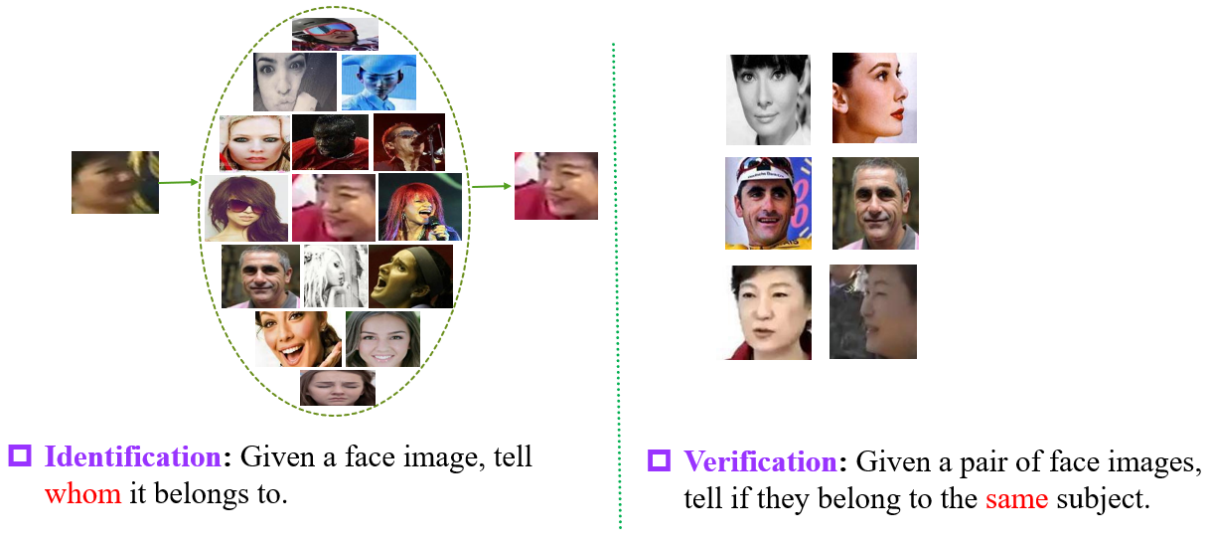

Figure 1.1: Face identification and face verification.

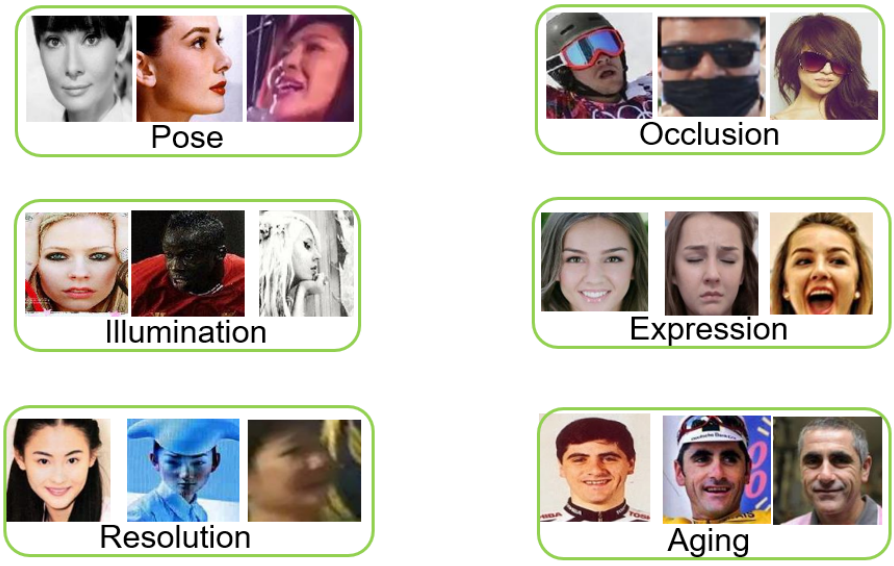

Figure 1.2: Many challenging factors that can affect the FR performance.

there are many factors that can affect the FR performance, including pose, illumination, occlusion, resolution, aging, and so on. is firstly studied by using traditional approaches where different types of features are extracted, followed by classifiers. However, these methods have inferior representational abilities, suffering from performance drops under some challenging scenarios. Deep learning has greatly improved the FR results, while most existing methods use global faces as the input. It is observed that global face appearances could change dramatically under large pose variations, heavy occlusions, and so on. In these cases, some local patches still remain similar, playing an important role in FR. Some works extract local patches by cropping discriminative facial parts around facial landmarks. However, facial landmark detection may be inaccurate or totally fail for some challenging faces with strong illumination changes, heavy occlusions. Recently, some attention mechanisms are designed to locate disciminative facial parts automatically. However, different attention maps tend to have limited responses around similar facial regions, missing some important ones. FR is challenging mainly because of two reasons: 1) High intra-class variations. Faces from the same subject may appear at different poses or occlusion levels. Consequently, if only few facial parts are located, the extracted features may not be sufficient when these parts are invisible under occlusions (e.g. face masks) or pose variations. 2) Small inter-class differences. Faces from different subjects may have similar local appearances, especially considering a large number of subjects. The performance would decline if the few emphasized facial parts across different subjects look similar. To alleviate the above issues, it is necessary to capture local representations that are as rich as possible. In such a way, more potentially useful facial parts can contribute to FR if the few emphasized parts are invisible or remain similar across different subject

The remaining is organized as follows: In Section 2, related works are reviewed comprehensively, including global , landmark-based, or attention-based approaches. In Section 3, current progresses are introduced. In Section 4, conclusions and future work are discussed.

1.1 Review of Related Research

In this section, we review the developments of FR, which include traditional approaches and deep learning based approaches.

1.1.1 Traditional Approaches

FR has been a popular research topic in the computer vision field for a long time. It was first pioneered in 1960s. Woody Bledsoe, Helen Chan Wolf, and Charles Bisson worked together to teach computers to recognize human faces. To find effective descriptors for FR, different holistic methods are proposed. EigenFace [16] was proposed 1991 where Principal Component Analysis (PCA) is used to efficiently reconstrcut human faces by a weighted sum of a small collection of images that define a facial basis. Fisherface was proposed in [17] where Linear Discriminant Analysis (LDA) is used to produce well-separated classes in low-dimensional subspace under the scenario of dramatic lighting variations and facial expressions. Better experimental results are obtained than EigenFaces. Two-Dimensional PCA (2DPCA) is developed in [18] for image feature extraction. Different from the conventional PCA, 2DPCA is calculated based on 2D image matrices rather than 1D vectors so that the image matrix does not need to transformed into a vector. Better experimental results are achieved in terms of accuracy and efficiency.

On the other hand, many local descriptors have gained attention due to their robustness to the uncontrolled facial changes, such as pose and illumination changes. Elastic Bunch Graph Matching (EBGM) [19] was proposed where Gabor wavelets responses around certain facial landmarks are used to represent faces and adapted graphs capture spatial relations of these landmarks, benefiting FR across large viewpoint changes. Local Binary Patterns (LBP) was proposed in [20] to extract efficient facial texture representations. A facial image is divided into several facial regions where LBP representations are extracted and concatenated into a feature vector. It is invariant to monotonic gray-level changes and has excellent computational efficiency, yielding outstanding results. Since kernels in Gabor wavelets are similar to the mammalian cortical cells with great characteristics of spatial locality and orientation selectivity, Gabor filter representations [21] are robust to illumination and ex-

pression variations. A combination of Gabor and LBP further boosts the FR performance. In [22], a Local Gabor Binary Pattern Histogram Sequence (LGBPHS) is proposed which has a great discriminative power and is robust to various image variations. This is achieved by combining local intensity distribution with the spatial information based on multi-scale and multi-orientation Gabor filters, making models robust to noise and lighting, occlusion, or pose variations.

Although current descriptor-based approaches have been proven to be effective to FR, they need to balance between the discriminative power and the robustness against various data variances. In order to tackle this issue, several learning-based methods are proposed to automatically encode useful representations. In [23], an unsupervised learning mechanism is designed to learn an encoder from training examples. In [24], a discriminant face descriptor is learned in a data-driven way where the differences between descriptors of the same person are minimized and those between different persons are maximized. However, these shallow representations tend to have a limited representational capacity on faces with complex non-linear appearance changes. Meanwhile, most of these approaches only take limited facial variations into considerations, such as one of the lighting, pose, occlusion, and age. Therefore, the FR was improving slowly, limiting the its application in the real-world scenario.

1.1.2 Deep Learning based Approaches

Starting from AlexNet [25] which won the ImageNet classification competition [26] by a significant margin, deep learning has dominated many computer vision fields. With the availability of large-scale datasets and the development of deep learning, FR has achieved decent performance [1, 27–31]. The Convolutional Neural Network (CNN) is one of the most popular deep learning methods which consists of multiple convolutional and pooling layers. CNNs can extract hierarchical information which corresponds to different levels of abstractions. Low layers output features similar to the Gabor feature defined by human experiences. As the layer goes deeper, learned features are becoming more complex which could be some facial parts, like eyes, noses.

Most of the progress depends on the large-scale training datasets [11, 32, 33], deep neural

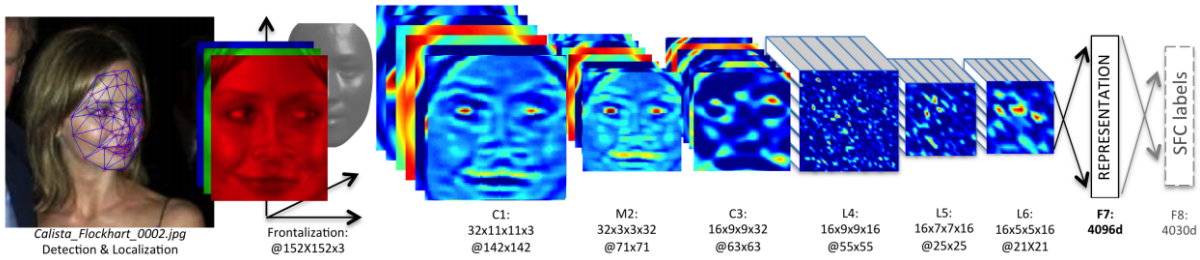

Figure 1.3: The overall framework of DeepFace [1].

networks [1, 34, 35], and effective loss functions [29–31, 36]. Most existing works use global faces as the input of CNNs, however, global face appearances tend to suffer from dramatic changes under some challenging scenarios, such large pose variations, heavy occlusion, and other quality changes.

On the other hand, some local patches still remain similar or the same, playing an important role in FR. Many methods crop facial parts around facial landmarks to benefit FR [37–42]. On the other hand, some attention mechanisms are designed to locate useful discriminative facial parts automatically [2, 43–45].

Global Approaches

In these approaches, global faces serve as the input of CNNs. We discuss the developments mainly in two aspects: 1) Neural network architectures; 2) Loss functions.

Neural network architectures Different models can be used for FR, including AlexNet [25] [46], VGG [47], GoogLeNet [48], Inception-v3 [49], ResNets [50], DenseNets [51], DeepFace [1], and LightCNN [35].

The AlexNet-v1 [25] model is the first model that successfully applies CNNs for large-scale image classification. The Dropout technique is employed to reduce overfitting; Rectified Linear Units (ReLU) can prevent neurons from saturating; Local response normalization (LRN) layers aid generalization. In AlexNet-v2 [46], LRN layers are removed without remarkable performance change. The VGG-16 [47] model is substantially deeper than the networks used before. 3×3 convolutional kernels are used to increase the network depth and reduce the number of parameters. In VGG-16-BN model, Batch Normalization (BN) layers [52] are added before ReLU, which can accelerate network training. The GoogLeNet [48] model has

much fewer parameters than the VGG-16, while still achieves a rather good performance. The Inception module is proposed to increase depth and width of deep networks. 1×1 convolutional kernels are applied to reduce the computational burden. The Inception-v3 [49] model is one of the Inception families. It has high computational efficiency and low parameter number. Factorizing large convolutional filters is implemented by smaller and asymmetric convolutions. Efficient grid size reduction is proposed to reduce computational cost. The ResNets [50] can be a depth of up to 152 layers but with a low complexity. A residual learning module is presented to ease the training of substantially deeper networks. ResNet-50 is a network with 50 layers. The DenseNets [51] also have much deeper architectures. It connects each layer to every other layer in a feed-forward fashion. It can alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

In [1], an effective neural network (called DeepFace) is developed to leverage a large-scale training dataset to extract face representations that generalize well to other test datasets. Specifically, the used dataset contains 4M facial images from 4000 identities. As shown in Fig. 1.3, a 3D model is firstly used to align faces to a canonical pose. Experimental results on the Labelled Faces in the Wild (LFW) [53] benchmark as well as the Youtube Faces in the wild (YTF) [54] benchmark demonstrate the effectiveness of this approach. This approach is trained on large-scale face identities and then use a bottleneck layer to represent faces. However, the representational size in the bottleneck layer is too large. LightCNN [35] introduces Max-Feature-Map (MFM) to learn a robust face representation, which not only separates between noisy features and informative signals but also plays an important role in feature selection.

Loss functions FR is challenging mainly because of two reasons: 1) High intra-class variations. Faces from the same subject may appear at different poses or occlusion levels. 2) Small inter-class differences. Faces from different subjects may have similar local appearances, especially considering a large number of subjects. However, softmax-based loss only encourages the separalibility between different classes.

In order to extract discriminative features, many loss functions are proposed for FR, which can be further divided into two categories: metric-based and softmax-based loss func-

tions. As for the former, contrastive loss [34] and triplet loss [1, 28] are used. Specifically, in [34], the classification signal pulls apart features from different subjects, while this signal has a weak constraint for faces from the same subject. To address this issue, the verification signal is used as additional information to suppress the intra-class variations. The formulation is defined as follows:

$$L = \begin{cases} \frac{1}{2}||f_i - f_j||_2^2 & \text{if } y_{ij}=1 \\ \frac{1}{2}max(0, m - ||f_i - f_j||_2)^2 & \text{if } y_{ij}=-1, \end{cases} \tag{1.1}$$

where f_i and f_j are feature vectors from two facial images, $y_{ij} = 1$ denotes that f_i and f_j are from the same subject, $y_{ij} = -1$ denotes that f_i and f_j are from different subjects, m is a margin to encourage the distances between facial images from different subjects.

In [1], it uses a siamense network where a pair of faces are compared using the Euclidean distance where positive pairs are minimized and negative pairs are maximized. In [28], a triplet-based loss function is used with a compact 128-D embedding, which separates the positive face pair with the negative by a distance margin. The specific formulation is defined as follows:

$$\sum_{i=1}^{N}[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha], \tag{1.2}$$

where x_i^a is an anchor of a specific person, x_i^p is the positive image from the same subject, x_i^n is a facial image from another person, N is the number of all possible triplets, and α is a margin between positive pairs and negative pairs. In such a way, models are forced to achieve much greater representational capacity, achieving decent performance on LFW and YTF benchmarks. However, the number of image pairs or triplets increases dramatically, which inevitably results in instability and slow convergence.

To address this issue, many softmax-based loss functions are proposed. The Softmax loss function is defined in the following:

$$L(I) = -\frac{1}{N}\sum_{i=1}^{N}\frac{e^{W_{y_i}^T x_i} + b_{y_i}}{\sum_{j=1}^{c} e^{W_j^T x_i + b_j}}, \tag{1.3}$$

where N and c denote the number of samples and classes, respectively. $x_i \in \mathrm{R}^d$ refers the i^{th} deep feature, belonging to the ${y_i}^{\text{th}}$ class. d is the feature dimension. $W_j \in R^d$ are the

weights in the last fully connected layer connecting to the j^{th} class. b_j is the bias term of the j^{th} class.

The main drawback of Softmax loss is that it aims to separate different subjects well but does not explicitly minimize the intra-subject variations. To this aim, several loss functions are used to enhance the discriminative power of the learned features. Center loss [29] is proposed to reduce intra-class variations. It can simultaneously learn a center for deep features of each class and penalize the distances between the deep features and their corresponding class centers. On one hand, center loss focuses on increasing intra-subject compactness. On the other hand, softmax loss aims at enlarging the inter-class distance. Intuitively, it is necessary to employ them jointly to supervise the training of CNNs. The formulation is in the following:

$$L(I) = -\sum_{i=1}^{N} \frac{e^{W_{y_i}^T x_i} + b_{y_i}}{\sum_{j=1}^{c} e^{W_j^T x_i + b_j}} + \frac{\lambda}{2} \sum_{i=1}^{N} \|x_i - c_{y_i}\|_2^2, \tag{1.4}$$

where the former part is the Softmax loss, and the later one is about Center loss. λ is used to balance the two loss functions. c_{y_i} is the $y_i{}^{\text{th}}$ class center of deep features.

There exits another trend in the research community which increases the margin between different subjects in training. SphereFace [36] presents A-Softmax loss to impose discriminative constraints on a hypersphere manifold, which intrinsically matches the prior that faces lie on a manifold. Formally it optimizes:

$$L(I) = -\frac{1}{N} \sum_{i=1}^{N} log \frac{e^{\|x_i\| \psi(\theta_{y_i,i})}}{e^{\|x_i\| \psi(\theta_{y_i,i})} + \sum_{j \neq y_i} e^{\|x_i\| cos\theta_{j,i}}}, \tag{1.5}$$

where $\psi(\theta_{y_i,i}) = (-1)^k cos(m\theta_{y_i,i}) - 2k$, $\theta_{y_i,i} \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right]$ and $k \in [0, m-1]$. $m \geq 1$ is an integer which controls the size of angular margin. $\theta_{j,i}$ is the angle between weight w_j and sample x_i.

In the CosFace loss [30], m is the cosine margin to maximize the decision margin in the angular space. The sample x_i is normalized and re-scaled to s.

$$L(I) = -\frac{1}{N} \sum_{i=1}^{N} log \frac{e^{s(\cos(\theta_{y_i,i}) - m)}}{e^{s(\cos(\theta_{y_i,i}) - m)} + \sum_{j \neq y_i} e^{s \cos \theta_{j,i}}}. \tag{1.6}$$

In the ArcFace loss [31], the additive angular margin penalty m is used to encourage the

intra-class compactness and inter-class discrepancy.

$$L(I) = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(\cos(\theta_{y_i,i}+m))}}{e^{s(\cos(\theta_{y_i,i}+m))} + \sum_{j\neq y_i} e^{s\cos\theta_{j,i}}}. \tag{1.7}$$

Local Approaches

Current local approaches can be grouped into several variants: 1) Landmark-based approaches [34, 38, 55, 56]: facial landmarks need to detected first, and cropping facial parts around landmarks with predefined sizes. Since facial landmark detection may fail for faces with strong illumination changes, heavy occlusions, and so on, this could limit the application scenarios. Besides, predefined crop size are not flexible to represent facial parts. For example, sunglasses may be extracted to worsen the representation. 2) Locally connected layers [1, 55]: Different facial parts exhibit different statistics. For example, eyebrows and mouth have very different appearances. It is unreasonable to apply the same convolution filters to different locations of facial images. To achieve this goal, locally connected layers are used in [1, 37] where every location in feature maps uses a different set of filters. However, using locally connected layers significantly increases the number of parameters. 3) Attention-based models [2, 43–45]: Rather than relying on landmarks, recent attention-based networks weighed varying local patches adaptively. Attention mechanisms are designed to model human behaviors where important information are selectively concentrated on and noisy ones are ignored. Attentions [57] have been widely used in many computer vision fields including FR. In this subsection, we mainly review related works on landmark-based and attention-based methods.

Landmark-based Approaches Most existing works employ global faces to train networks [29,31,36]. However, the performance is degraded if the global face appearance changes dramatically. To alleviate this issue, some prior works focus on subtle differences of facial parts by cropping patches around facial landmarks. In [37], a face image was first divided into several local regions centered around five facial landmarks (two eye centers, nose tip, and two mouth corners) at different scales and color channels, and these parts are used to extract local representations. As shown in Fig. 1.4, to make models robust to pose variations and occlusions, complementary information between global faces and different local patches

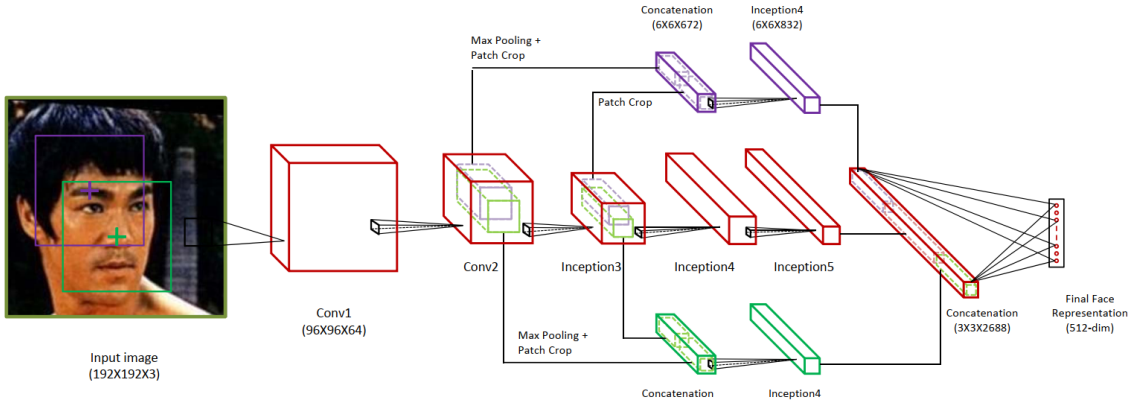

Figure 1.4: Model architecture of Trunk-Branch Ensemble CNN. The Trunk network learns holistic face representations and the trunk networks learn features for image patches cropped around facial landmarks. The trunk network and branch networks share low and middle feature maps, and they have independent high-level feature maps.

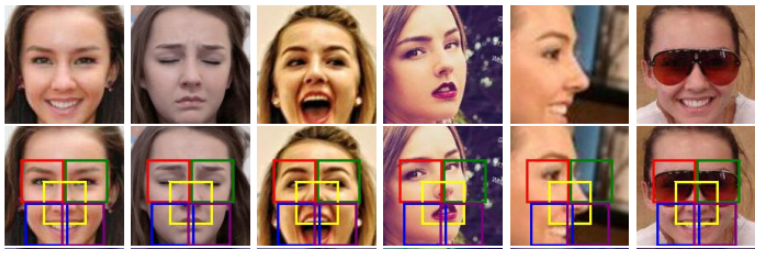

Figure 1.5: Row 1: some challenging faces. Row 2: even face landmarks are detected, predefined crops may not be flexible and robust under pose, expression and occlusion variations.

is explored in [38]. Different from previous works, this approach extracts local representations on feature maps, instead of images. In [39], seven different CNNs are used on seven overlapped image patches centered around facial landmarks, learning invariant features towards pose, occlusion, or expression changes. Some facial parts centered around landmarks (e.g. centers of eyes, tip of noses, and the corners of the mouths) are randomly removed with random-sized rectangles in [40]. This model can alleviate the overfitting issue and learn robust features. In [41], an attention mechanism is used on different sub-regions, and local descriptors between pairs of faces are compared. In [42], the pairwise relational network (PRN) was proposed which makes every possible pairs of local appearance features, then each pair of local appearance features is used to extract relational features.

However, face landmark detection is not reliable, which may fail for faces with large occlusion, pose, aging, and makeup variations. Besides, landmark-based methods crop face

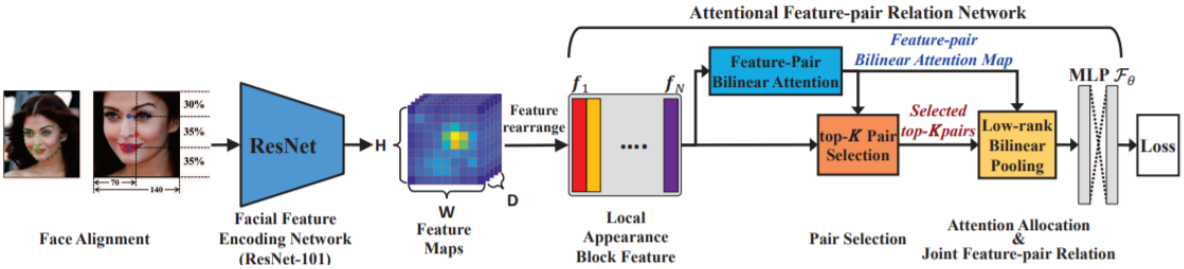

Figure 1.6: The overall framework of AFPN [2].

parts around landmarks, suffering from noise from adjacent parts or background. For example, as shown in Fig. 6.4, Columns 1, 2, 3, mouths have varying shapes or sizes (Row 1) due to expression changes. As a consequence, some crops contain noise from noses and background (Row 2). Meanwhile, because of pose variations (Row 1, Columns 4, 5) and occlusions (Row 1, Column 6), some parts are partly or completely invisible. In such cases, background (Row 2, Columns 4, 5) or sunglasses (Row 2, Column 6) are cropped, leading to noisy representations. Moreover, it is noticed that landmark detection may be not accurate, resulting in inaccurate crops (Row 2, Columns 4, 5). Furthermore, different facial parts should have different discriminative abilities. However, the importance of different facial parts is not taken into consideration.

Attention-based Approaches Rather than relying on landmarks, recent attention-based networks weighed varying local patches adaptively. As shown in Fig. 1.6, in [2], an attention feature-pair relation network (AFRN) was proposed to represent faces by relevant pairs of local appearance block features with attention scores, capturing unique and discriminative feature-pair relations to classify faces among different subjects. An attention-based convolutional neural network was proposed in [43], which reduces the information redundancy among channels and emphasizes important feature maps. Specifically, a channel attention and a spatial attention are used to adaptively aggregate feature maps in both channel and spatial dimensions to capture inter-channel and intra-channel attention maps. Extensive experiments on several challenging benchmarks demonstrate the usefulness of the proposed attention modules. To alleviate the problem of cross-pose face matching, a method was presented in [44] to progressively transform profile face representations to the frontal pose with an attentive pair-wise loss. Two-level attention fusion method was proposed in [45]

for RGB-D face recognition. Specifically, the first layer focuses on the fused feature maps generated by the feature extractor, exploiting the relationship between feature maps using LSTM recurrent learning. The second layer focuses on the spatial features of those maps using convolution.

In these approaches, since there is no explicit signal, different attention maps tend to have redundant responses around similar facial parts, missing some important facial parts. FR is challenging mainly because of two reasons: 1) High intra-class variations. Faces from the same subject may appear at different poses or occlusion levels. In such a case, if only few facial parts are located, the extracted features may not be sufficient when these parts are invisible under occlusions (e.g. face masks) or pose variations. 2) Small inter-class differences. Faces from different subjects may have similar local appearances, especially considering a large number of subjects. The performance would decline if the few emphasized facial parts across different subjects look similar. To alleviate the above issues, it is necessary to capture local representations that are as rich as possible. In such a way, more potentially useful facial parts can contribute to FR if the few emphasized parts are invisible or remain similar across different subjects.

Chapter 2

Datasets

The training and test datasets used in the dissertation are introduced.

2.1 Training Datasets

Several training datasets are used, including CASIA-WebFace [11], UMDFaces [58], VGGFace2 [32], and MS-Celeb-1M dataset [33] datasets.

CASIA-WebFace It contains 494,414 face images of 10,575 identities. The authors claimed that not all faces were detected and annotated correctly.

UMDFaces It has 367,888 faces from 8,277 identities. Humanly curated bounding boxes for faces are provided. The authors claim that it provides more pose variations than the popular WebFace dataset.

VGGFace2 It has 3,141,890 images of 8,631 identities. Human verified bounding boxes around faces are provided. It covers a large range of poses, ages, professions, and ethnicities.

MS-Celeb-1M The original MS-Celeb-1M dataset contains too much noise. To get a high-quality dataset, [31] refined the dataset and made it publicly available. There are about 85,000 subjects with 5.8 million aligned images. There is another version which is adopted in the benchmark: To get a high-quality dataset, DeepGlint (http://trillionpairs.deepglint.com/) refined the dataset and made it publicly available. There are 86,878 subjects with 3.92 million aligned images.

2.2 Test Datasets

Cross-Pose Face Matching There are 500 subjects in the CFP dataset [59]. Each subject has 10 frontal and 4 profile faces. Frontal-frontal (FF) and frontal-profile (FP) verification protocols are considered. Each protocol consists of 10 folds with 350 positive pairs and 350 negative pairs.

CPLFW dataset [60] has 3K positive pairs and 3K negative pairs. Pose differences are added in positive pairs to increase intra-class variations. Negative pairs have limited attribute differences to reduce inter-class variations.

VGGFace2-FP dataset [32] has 300 subjects and matches face templates between frontal views and profile views. Each template has 5 face images. In which, the VGGFace2-test dataset has 169,396 images from 500 celebrities. The test has two scenarios. Pose template: a template consists of five faces from the same subject with a consistent pose which can be frontal, Three-quarter or profile view. Age template: a template consists of five faces from the same subject with either an apparent age below 34 (deemed young) or 34 and above (deemed mature). Each subject is represented by averaging the feature vector of all faces in each subject set. Then the similarity score is computed as the cosine similarity between feature vectors representing each subject.

On top of a subset of the UMDFaces dataset [58], they developed a large testing protocol that contains three tracks: small pose variations (easy), medium pose variations (moderate) and large pose variations (difficult). There are 5,000 positive pairs and 5,000 negative pairs within each track. A large number of image pairs make it possible to compare performance at a low FAR.

Cross-Age Face Matching The CACD-VS dataset [12] is used for age-invariant face recognition with varying illumination, pose variations and makeup. Following the configuration [12], we use the 10-fold cross-validation on 2,000 positive pairs and 2,000 negative pairs.

The CALFW dataset [61] consists of 4,025 subjects. Each subject has 2, 3 or 4 images. There are large age gaps between positive pairs to increase intra-subject variations. For negative pairs, only face pairs with the same gender and race are selected to reduce the

influence of attribute differences. It has 10 folds with 3,000 positive pairs and 3,000 negative pairs.

AgeDB dataset [62] has 12,240 images of 440 subjects. The age is in the range of [3, 101]. We focus on the group with 30 year gaps, which has 10 splits of face images. Each split has 300 positive pairs and 300 negative pairs.

Cross-Quality Face Matching[1] The IJB-A [63] and FaceScrub [64] datasets have images of different qualities. Follow the work in [65], we assess the face image quality and select low- and high-quality face images for cross-quality face matching. For IJB-A dataset, there are 1,543 high-quality images of 500 identities and 6,196 with low-quality images from 489 identities. For FaceScrub dataset, there are 10,089 high-quality images of 530 subjects and 362 low-quality images of 232 subjects. Each low-quality image is compared with every high-quality face image in this protocol, making the task very challenging. Verification performance is reported when FAR=0.01 and 0.001, respectively.

Cross-Modality Face Matching The CASIA NIR-VIS 2.0 [66] is a popular NIR-VIS database with a large intra-class variations, including lighting, expression, pose, and distance variations. It contains 725 subjects and each subject has 1-22 VIS and 5-50 NIR images. There are two views of evaluation protocols where view 1 is for hyper-parameter tuning and view 2 is used for training and testing. Without fine-tuning on training data in views 1 or 2, we directly evaluate our models using the test protocols in view 2. There are 10 folds where each fold is composed of 358 gallery images and around 6K probe images.

Masked Face Matching[1] Two masked face datasets in [67] are used: the real-world masked face recognition dataset (RMFRD) and masked LFW (MLFW) dataset. The former contains 403 overlapped subjects with 1,945 masked and 80,577 normal faces. Two tasks are designed based on the RMFRD: Masked2Normal (M2N) and Masked2Masked (M2M). First, **M2N** contains both verification and identification protocols. In verification, each masked face is matched with every normal face, generating 394K positive and 156M negative pairs. In identification, we compute the average of all normal faces in a subject as the corresponding gallery subject feature, and calculate the Rank-1 accuracy for all masked faces. Second, each

[1]These protocols will be released soon.
[1]These protocols will be released soon.

masked face is compared with every other masked face in **M2M**, creating 3.76M negative and 20K positive pairs.

As for the MLFW, a software is developed to wear masks for face images in LFW [67]. It has 5,749 subjects with 13,175 masked faces. Referring to the original protocol in LFW which has 6K pairs, there are 5,955 pairs in **MLFW** protocol because some faces are failed to wear masks.

Disguised Face Matching The DFW dataset [68] is the largest dataset of disguised faces and impostors. There are 1,000 subjects (400 in the training set and 600 in the test set) and 11,155 images. Each subject in the test set has one genuine image, some validation images, several disguised images and a few images of impostors. Each pair in the test set is assigned a ground-truth label ('positive', 'negative' or 'do not care'). The task is to recognize disguised faces as the subject that they belong to and identify impostors. The performance is evaluated when $FAR = 1\%, 0.1\%$, respectively.

LFW It [69] contains 13,233 images collected online from 5,749 identities. Following the 10-fold verification protocol, experimental results are reported accordingly.

IJB-C It [70] includes 31,334 still images and 117,542 frames from 11,779 full-motion videos.

Chapter 3

Benchmarking Deep Learning Techniques for Face Recognition

3.1 Introduction

During the past several decades, numerous face recognition approached are proposed [20] [71] [72] [73] where various hand-crafted features are used. Recently, Convolutional Neural Networks (CNNs) greatly advance the face recognition performance. For example, the accuracy on the Labeled Faces in the Wild (LFW) dataset [69] has increased to 99.83% [31].

However, training deep neural networks is a very complex process. There are several factors that can affect the training process and evaluation performance. Firstly, there are a number of deep learning frameworks available, including Caffe from UC Berkeley [74], TensorFlow from Google [75] and PyTorch from Facebook [76], which are widely used frameworks among deep learning researchers [77]. However, these frameworks may have different data layouts (e.g. NCWH, NWHC) and convolution implementations (e.g. GEMM, FFT, Winograd), resulting in different performance.

Secondly, recent progress in deep CNNs has substantially improved state-of-the-art performance in face recognition, and there are a number of models available, such as AlexNet [25], VGG [47], GoogLeNet [48], ResNet [50], DenseNet [51], LightCNN [78], Incepetion-DenseNet [79], DDML [80], Center-loss [29], SphereFace [36], CosFace [30], UniformFace [81] and ArcFace [31]. The inherent structures in these models are somehow different. For example,

GoogLeNet and DenseNet emphasize multi-scale feature learning; ResNet and DenseNet are deep networks to model complex data with a large spectrum of variations. Further, the developed deep models are usually reported on different training datasets, making it difficult to understand how different these models intrinsically, using the same training datasets.

Thirdly, training CNNs is a very time-consuming process. Graphical Processing Units (GPUs) are playing a key role in training networks. However, there are a number of GPU types, such as Titan Xp, GTX 1080Ti, Titan X(Pascal), Titan X(Maxwell) and Titan Z. It is interesting to know how differently these GPUs could perform in training deep models in practice.

Fourthly, there are different test datasets that may have different characteristics. In LFW dataset, almost all faces are frontal or close to frontal views; VGGFace2-test dataset [82] mainly focuses on pose and age variations; As for our processed IJB-A [63] quality dataset, it contains faces of different qualities; Disguise Faces in Wild (DFW) dataset [68] is the largest dataset of disguised faces and impostors; UMDFaces-test dataset [58] contains three tracks with different pose variations.

Fifthly, recent progress in face recognition is mainly being driven by advanced CNNs, fast GPUs, and large training datasets. Meanwhile, as CNNs are getting deeper, the requirement of large-scale training datasets becomes urgent. In recent years, several face datasets are made public with different scales, ranging from a few hundred thousand images, e.g., FaceScrub [64], CASIA-WebFace [11] and UMDFaces [58], to a few million images, e.g., VGGFace [27], MegaFace [83], MS-Celeb-1M [33] and VGGFace2 [82]. The deep CNNs may behave differently as the training datasets change. Many current face recognition models are trained on different training datasets, making it difficult to compare these models directly.

Given the variety of deep learning frameworks, face recognition models, GPU platforms, and training datasets, it is quite difficult for end users to select appropriate platforms to conduct their face recognition tasks. However, few works systematically investigate this issue. In [84], different types of neural networks are evaluated on a set of deep learning frameworks with different hardware platforms. However, there were no specific face recognition models or performance explored in [84]. Besides, more recent GPU types and newer versions of deep learning frameworks were not included in their study.

In this paper, based on several training datasets (i.e., UMDFaces, WebFace, MS-Celeb-1M and VGGFace2), we benchmark several face recognition models (i.e., AlexNet, VGG, GoogLeNet, ResNet, DenseNet, LightCNN, Center-loss, SphereFace, CosFace and ArcFace) under different deep learning frameworks (i.e., Caffe, TensorFlow and PyTorch) using different GPUs (i.e., Titan Xp, GTX 1080Ti, Titan X(Pascal), Titan X(Maxwell) and Titan Z) and test their performance on different datasets (i.e., LFW, VGGFace2-test, IJB-A quality, DFW challenge and UMDFaces-test).

The rest of paper is organized as follows. Section 2 introduces frameworks, deep models, and GPU platforms. Section 3 presents training and test datasets. Benchmarking results are presented in Section 4. Conclusion and future work are given in Section 5.

3.2 Frameworks, Deep Models, and GPU Platforms

We briefly review three main deep learning frameworks, some popular deep learning face recognition models, and five widely used GPU platforms.

3.2.1 Deep Learning Frameworks

With the growing success of deep learning, there are many popular open-source deep learning frameworks which aim to help researchers quickly develop deep learning models, including Caffe, TensorFlow and PyTorch.

Caffe [74] is developed by Berkeley AI Research (BAIR) and GitHub community contributors. It enables researchers and practitioners to train and deploy general-purpose CNNs and other deep models efficiently on commodity architectures based on Python and MATLAB bindings.

TensorFlow [75] is designed by Google to operate at large scale and in heterogeneous environments. It uses dataflow graphs to represent both the computation in an algorithm and the state on which the algorithm operates.

PyTorch [76] is a scientific computing framework with two high-level features: tensor computation with strong GPU acceleration; deep neural networks built on a tape-based autodiff system. It aims to provide users with maximum flexibility and speed.

3.2.2 Deep Models for Face Recognition

With the availability of large datasets, researchers have developed a number of CNN models. In this paper, we mainly focus on several publicly available models: AlexNet [25] [46], VGG [47], GoogLeNet [48], Inception-v3 [49], ResNets [50], DenseNets [51], LightCNN [78], Center-loss [29], SphereFace [36], CosFace [30], and ArcFace [31]. The last five ones are specifically designed for face recognition, while others are originally developed for ImageNet classification [85].

The AlexNet-v1 [25] model is the first model that successfully applies CNNs for large-scale image classification. The Dropout technique is employed to reduce overfitting; Rectified Linear Units (ReLU) can prevent neurons from saturating; Local response normalization (LRN) layers aid generalization. In AlexNet-v2 [46], LRN layers are removed without remarkable performance change.

The VGG-16 [47] model is substantially deeper than the networks used before. 3×3 convolutional kernels are used to increase the network depth and reduce the number of parameters. In VGG-16-BN model, Batch Normalization (BN) layers [52] are added before ReLU, which can accelerate network training.

The GoogLeNet [48] model has much fewer parameters than the VGG-16, while still achieves a rather good performance. The Inception module is proposed to increase depth and width of deep networks. 1×1 convolutional kernels are applied to reduce the computational burden.

The Inception-v3 [49] model is one of the Inception families. It has high computational efficiency and low parameter number. Factorizing large convolutional filters is implemented by smaller and asymmetric convolutions. Efficient grid size reduction is proposed to reduce computational cost.

The ResNets [50] can be a depth of up to 152 layers but with a low complexity. A residual learning module is presented to ease the training of substantially deeper networks. ResNet-50 is a network with 50 layers.

The DenseNets [51] also have much deeper architectures. It connects each layer to every other layer in a feed-forward fashion. It can alleviate the vanishing-gradient problem,

strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.

LightCNN [78] introduces Max-Feature-Map (MFM) to learn a robust face representation, which not only separates between noisy features and informative signals but also plays an important role in feature selection.

The Softmax loss function for LightCNN and ImageNet models is in the following:

$$L(I) = -\frac{1}{N}\sum_{i=1}^{N}\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{c} e^{W_j^T x_i + b_j}}, \tag{3.1}$$

where N and c denote the number of samples and classes, respectively. $x_i \in \mathrm{R}^d$ refers the i^{th} deep feature, belonging to the ${y_i}^{\text{th}}$ class. d is the feature dimension. $W_j \in R^d$ are the weights in the last fully connected layer connecting to the j^{th} class. b_j is the bias term of the j^{th} class.

The main drawback of Softmax loss is that it aims to separate different subjects well but does not explicitly minimize the intra-subject variations. To this aim, several loss functions are used to enhance the discriminative power of the learned features. Center loss [29] is proposed to reduce intra-class variations. It can simultaneously learn a center for deep features of each class and penalize the distances between the deep features and their corresponding class centers. On one hand, center loss focuses on increasing intra-subject compactness. On the other hand, softmax loss aims at enlarging the inter-class distance. Intuitively, it is necessary to employ them jointly to supervise the training of CNNs. The formulation is in the following:

$$L(I) = -\sum_{i=1}^{N}\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{c} e^{W_j^T x_i + b_j}} + \frac{\lambda}{2}\sum_{i=1}^{N}\|x_i - c_{y_i}\|_2^2, \tag{3.2}$$

where the former part is the Softmax loss, and the later one is about Center loss. λ is used to balance the two loss functions. c_{y_i} is the ${y_i}^{\text{th}}$ class center of deep features.

There exits another trend in the research community which increases the margin between different subjects in training. SphereFace [36] presents A-Softmax loss to impose discriminative constraints on a hypersphere manifold, which intrinsically matches the prior that faces

lie on a manifold. Formally it optimizes:

$$L(I) = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{\|x_i\|\psi(\theta_{y_i,i})}}{e^{\|x_i\|\psi(\theta_{y_i,i})} + \sum_{j\neq y_i} e^{\|x_i\| cos\theta_{j,i}}}, \tag{3.3}$$

where $\psi(\theta_{y_i,i}) = (-1)^k cos(m\theta_{y_i,i}) - 2k$, $\theta_{y_i,i} \in \left[\frac{k\pi}{m}, \frac{(k+1)\pi}{m}\right]$ and $k \in [0, m-1]$. $m \geq 1$ is an integer which controls the size of angular margin. $\theta_{j,i}$ is the angle between weight w_j and sample x_i.

In the CosFace loss, m is the cosine margin to maximize the decision margin in the angular space. The sample x_i is normalized and re-scaled to s.

$$L(I) = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j\neq y_i} e^{s\cos\theta_{j,i}}}. \tag{3.4}$$

In the ArcFace loss, the additive angular margin penalty m is used to encourage the intra-class compactness and inter-class discrepancy.

$$L(I) = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(\cos(\theta_{y_i,i}+m))}}{e^{s(\cos(\theta_{y_i,i}+m))} + \sum_{j\neq y_i} e^{s\cos\theta_{j,i}}}. \tag{3.5}$$

3.2.3 GPU Platforms

Deep learning is a field with intense computational requirements, and the choice of GPUs can make big differences. In this work, three major GPU architectures are investigated. Furthermore, it is necessary to use multi-GPUs to speed up the training process and improve the model performance.

Table 3.1: Some details about various GPUs that we tested.

	Titan Z	Titan X (Maxwell)	Titan X (Pascal)	GTX 1080Ti	Titan Xp
Boost Clock (MHz)	876	1075	1531	1582	1582
CUDA Cores	5760	3072	3584	3584	3840
Memory Speed (Gbps)	7	7	10	11	11.4
Standard Memory Config (GB)	12	12	12	11	12
Memory Interface width (bit)	768	384	384	352	384
Memory Bandwidth (GB/sec)	672	336.5	480	484	547.7
GPU Architecture	Kepler	Maxwell	Pascal	Pascal	Pascal

Table 3.1 gives some details of GPUs used in our experiments. Three different GPU architectures are benchmarked: Kepler (Titan Z), Maxwell (Titan X(Maxwell)) and Pascal (Titan X(Pascal), GTX 1080Ti and Titan Xp). Notice that we only use one of the two GK110 chips in Titan Z for the single GPU comparison.

3.3 Benchmarking Experiments

We first discuss the implementation details in Sec. 3.3.1. Different face recognition models are compared in Sec. 3.3.2 wwhere model comparisons within the same deep learning framework and between different frameworks are discussed. Running time comparison of different models on various GPU platforms and scalability of different deep learning frameworks on multi-GPUs are presented in Sec. 3.3.3. Finally, Sec. 3.3.4 shows performance comparison of different training datasets on several test datasets.

3.3.1 Implementation Details

For DFW and UMDFaces datasets, faces are cropped using the provided face coordinates and resized to the target image size. Other images are detected and cropped by the MTCNN [4] method. We use the Python implementation of the method MTCNN in [86]. Fig. 3.1 demonstrates the workflow on an example image. Referring to pre-processing operations in [87], we enlarge the bounding box size by 15% on each side and crop the image without the alignment procedure. If face detection is failed on one image, this image will be discarded if it is in the training dataset.

Deep models are trained on the WebFace dataset if not specified, so their relative performances can be compared. There are 453,413 detected faces in WebFace dataset. In order to reduce overfitting, all images are shuffled under PyTorch, Caffe and TensorFlow if the dataset is used for training. Let b, n refer to the batch size used in training and the image number in the dataset, respectively. There are $\lceil \frac{n}{b} \rceil$ batches in the dataset. Then a batch with b images is loaded sequentially from the beginning of the dataset, and then these images are pre-processed to feed the CNN model. Repeat these operations $\lceil \frac{n}{b} \rceil$ times until the end of the dataset. When the image number n is not divisible by the batch size b, the last

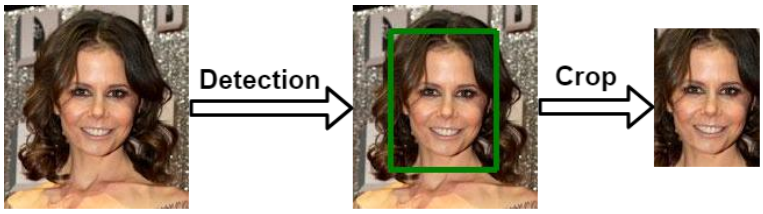

Figure 3.1: The pre-processing on an example image, which consists of two stages, i.e. detection and crop stages. The detection stage is used to detect faces with the pre-trained MTCNN model, and the crop stage is to crop the face based on the bounding box.

batch size will be smaller than b. Under such a scenario in training, different frameworks have different strategies: the image pointer will seek to the start of the dataset recursively to get enough b images when training in Caffe; the last insufficient batch is directly used in PyTorch; TensorFlow will discard the smaller batch. During testing, the last insufficient batch will be processed in PyTorch, Caffe and TensorFlow. We input the face image and extract features from the penultimate layer for later analysis.

3.3.2 Accuracy Comparison of Different Face Recognition Models

Table 3.2: The hyper-parameter settings and # of parameters in the models. The momentum is 0.9; iter means the global step during training; iterations per epoch (ipe) = # of images / (batch size).

Model	Batch size	Image size	Input size	Learning rate	Epochs or iters	Weight decay	# of params (million)		
							Caffe	TF	PyTorch
AlexNet-v1	256	256×256	227×227	$0.01 * 0.1^{(floor(iter/(21*ipe))}$	96	5e-4	100.17	-	-
AlexNet-v2	256	256×256	224×224	$0.01 * 0.1^{(floor(iter/(21*ipe)))}$	96	5e-4	-	89.53	100.33
DenseNet-121	40	256×256	224×224	$0.1 * 0.1^{(floor(iter/(10*ipe)))}$	30	1e-4	17.78	17.76	17.79
GoogLeNet	32	256×256	224×224	$0.01 * (1 - iter/(64 * ipe))^{0.5}$	64	2e-4	42.78	16.43	-
Inception-v3	25	299×299	299×299	$0.045 * 0.94^{(floor(iter/(2*ipe)))}$	100	4e-4	43.43	54.13	43.45
LightCNN-29	128	144×144	128×128	$0.1 * 0.457^{(floor(iter/(10*ipe)))}$	80	1e-4	8.19	-	-
ResNet-50	50	256×256	224×224	$0.1 * 0.1^{(floor(iter/(28*ipe)))}$	128	1e-4	45.16	45.18	45.18
VGG-16	128	256×256	224×224	$0.1 * 0.1^{(floor(iter/(17*ipe)))}$	74	5e-4	177.56	177.59	177.59
VGG-16-BN	128	256×256	224×224	$0.1 * 0.1^{(floor(iter/(17*ipe)))}$	74	5e-4	-	-	177.59
Softmax	512	112×112	112×112	$0.1 * 0.1^{(floor(iter/[16K,24K,28K]))}$	30K	5e-4	-	-	-
Center-loss	256	112×96	112×96	$0.1 * 0.1^{(floor(iter/[16K,24K]))}$	28K	5e-4	32.95	-	-
SphereFace	128	112×96	112×96	$0.1 * 0.1^{(floor(iter/[16K,24K]))}$	28K	5e-4	68.45	-	-
CosFace	512	112×112	112×112	$0.1 * 0.1^{(floor(iter/[16K,24K,28K]))}$	30K	5e-4	-	-	-
ArcFace	512	112×112	112×112	$0.1 * 0.1^{(floor(iter/[16K,24K,28K]))}$	32K	5e-4	-	-	-

Table 3.2 shows hyper-parameter settings and # of parameters of different face recognition models. After comparisons of different hyper-parameter settings on several models, there is little performance change compared with the original parameter settings in the

Table 3.3: Experimental results of various models in PyTorch on LFW and VGGFace2-test datasets.

PyTorch-based	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
AlexNet-v2	93.1	Front	0.625	0.595	0.393	Young	0.534	0.231
		Three-quarter	0.599	0.651	0.510	Mature	0.259	0.606
		Profile	0.390	0.508	0.625			
VGG-16	97.2		0.726	0.698	0.510		0.648	0.353
			0.705	0.739	0.603		0.382	0.711
			0.517	0.604	0.695			
VGG-16-BN	97.6		**0.778**	**0.756**	**0.599**		**0.714**	0.431
			0.760	**0.789**	0.669		0.452	**0.761**
			0.610	0.671	0.730			
Inception-v3	97.9		**0.778**	0.749	0.595		0.700	**0.454**
			0.756	**0.789**	**0.671**		**0.497**	0.758
			0.606	**0.680**	**0.743**			
ResNet-50	**98.2**		0.756	0.726	0.556		0.669	0.412
			0.727	0.755	0.629		0.447	0.710
			0.559	0.635	0.693			
DenseNet-121	97.8		0.755	0.731	0.576		0.682	0.419
			0.735	0.763	0.648		0.452	0.722
			0.582	0.654	0.709			

proposed papers. Therefore, we decided that parameter settings of every model are the same as the paper in which the model was initially proposed. The momentum is 0.9. The weight-decay of the DenseNet-121, LightCNN-29, ResNet-50 models is 1e-4, GoogLeNet is 2e-4, Inception-v3 is 4e-4, and for other models, it is 5e-4. 2 GPUs are used to train the DenseNet-121, ResNet-50, VGG-16, VGG-16-BN, Center-loss and SphereFace models. 4 GPUs are employed to train the Softmax, CosFace and ArcFace based models.

We also compare the number of parameters in different frameworks. One can observe that the same face recognition model has similar number of parameters on three deep learning frameworks except the AlexNet-v2 and GoogLeNet. The AlexNet-v2 models in TensorFlow and PyTorch apply a zero padding of size 0 and 2 in the first convolutional layer, resulting in the last convolutional layer with size 256×5×5 and 256×6×6, separately. Because this layer is connected with a fully connected layer with units 4096, the PyTorch has 4096×256×(6×6-5×5)=11.5 million more parameters compared with TensorFlow. As for the GoogLeNet model, the parameter number difference between Caffe and TensorFlow is that Caffe adds

Table 3.4: Experimental results of various models in PyTorch on IJB-A quality dataset.

PyTorch-based	IJB-A (%)					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
AlexNet-v2	29.1	11.9	5.1	55.3	34.9	23.0
VGG-16	47.3	23.7	7.5	77.6	54.4	35.5
VGG-16-BN	51.5	28.0	7.9	81.3	60.0	38.7
Inception-v3	51.3	28.2	7.9	81.4	60.2	38.8
ResNet-50	**62.3**	**38.9**	**12.7**	**85.9**	**68.0**	**48.9**
DenseNet-121	58.8	33.1	9.8	83.3	62.4	42.1

two auxiliary classifiers connected to intermediate layers, while TensorFlow does not.

We first compare model accuracy within the same deep learning framework and several loss functions in Sec. 3.3.2 and Sec. 3.3.2, respectively, and then illustrate model accuracy comparison between different frameworks in Sec. 3.3.2.

Model Accuracy Comparison within the Same Deep Learning Framework

Tables 3.3 and 3.4 are about experimental results of PyTorch based models. For LFW dataset, the AlexNet-v2 model performs the worst, while others have similar performances. This is because the LFW dataset is a relatively easy dataset. For VGGFace2-test dataset, the VGG-16-BN and Inception-v3 models rank the first. For IJB-A quality dataset, the ResNet-50 model gets the first place on cross-quality face matching. This proves the fact that the network depth is important. VGG-16-BN achieves better performance than VGG-16. This is because BN layers act as a regularizer to improve the performance for the VGG-16. However, BN layers may fail for some models, and an appropriate batch size is needed to avoid the possible out-of-memory problem [88].

Experimental results about Caffe based models are shown in Tables 3.5 and 3.6. For LFW dataset, GoogLeNet model can reach 97.8% accuracy. For VGGFace2-test and IJB-A datasets, LightCNN-29 model beats other models by a pronounced margin. It can be seen that LightCNN-29 can get better performance than other ImageNet based models. This

Table 3.5: Experimental results of various models in Caffe on LFW and VGGFace2-test datasets.

Caffe-based	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
AlexNet-v1	94.6	Front	0.595	0.543	0.301	Young	0.477	0.259
		Three-quarter	0.557	0.609	0.432	Mature	0.281	0.541
		Profile	0.310	0.430	0.583			
VGG-16	95.3		0.692	0.663	0.462		0.610	0.384
			0.675	0.709	0.564		0.395	0.656
			0.476	0.565	0.681			
GoogLeNet	**97.8**		0.698	0.652	0.440		0.605	0.372
			0.660	0.701	0.540		0.392	0.658
			0.450	0.541	0.644			
Inception-v3	96.1		0.712	0.692	0.560		0.626	0.415
			0.706	0.719	0.600		0.442	0.653
			0.572	0.602	0.596			
ResNet-50	97.5		0.744	0.710	0.505		0.667	0.425
			0.718	0.753	0.591		0.455	0.697
			0.513	0.594	0.677			
DenseNet-121	97.6		0.737	0.699	0.520		0.655	0.413
			0.707	0.746	0.604		0.442	0.698
			0.533	0.611	0.690			
LightCNN-29	97.6		**0.764**	**0.740**	**0.600**		**0.691**	**0.472**
			0.748	**0.774**	**0.666**		**0.501**	**0.729**
			0.607	**0.665**	**0.703**			

proves that the effectiveness of Max-Feature-Map designed for the face recognition task.

Tables 3.7 and 3.8 show the performance of TensorFlow based models. For LFW dataset, Inception-v3 gets 98.5% accuracy, slightly better than other models. For VGGFace2-test dataset, GoogLeNet, Inception-v3 and DenseNet-121 are obviously better than others. For IJB-A quality dataset, GoogLeNet dramatically improves the matching accuracy across diverse qualities.

Among these models, results show that the GoogLeNet model is the most discriminative which gets 3 highest accuracies though unavailable in PyTorch because the LRN regularization is not supported. Despite only supported in Caffe, the LightCNN-29 model obtains 2 best results. The ResNet-50 and Inception-v3 models get 2 and 2 best results, respectively. The GoogLeNet and Inception-v3 models have Inception modules which consist of parallel convolutional kernels (1×1, 3×3 and 5×5). Inception modules allow to extract both local

Table 3.6: Experimental results of various models in Caffe on IJB-A dataset.

Caffe-based	IJB-A (%), different FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
AlexNet-v1	12.1	4.2	1.7	35.8	21.3	13.2
VGG-16	29.6	12.3	4.5	61.0	38.5	24.2
GoogLeNet	45.4	21.2	5.6	75.8	55.4	36.3
Inception-v3	29.65	12.34	4.52	61.2	38.53	24.24
ResNet-50	39.2	1.4	0.2	74.5	46.3	5.8
DenseNet-121	38.8	6.8	0.4	73.5	49.5	22.6
LightCNN-29	**50.4**	**28.2**	**14.0**	**76.2**	**55.9**	**36.7**

features (small convolutional kernels) and abstract features (larger convolutional kernels). The extracted multi-scale features can characterize face images at various levels, improving the face recognition performance. The ResNet-50 model uses identity mappings as bypassing paths, easing the training of very deep networks and improving the capacity to describe faces. On the other side, identity mappings connect features from two layers, which have different receptive field sizes. The Max-Feature-Map design allows the LightCNN-29 model to extract informative signals from noisy features.

Model Accuracy Comparison of Several Loss Functions

Tables 3.9 and 3.10 compares the Softmax loss function with several loss functions: Center-loss, SphereFace, CosFace and ArcFace.

In these models, SphereFace and ArcFace based models achieve the best accuracy on LFW dataset. Center-loss based model has the best accuracy on VGGFace2-test task. SphereFace based model outperforms others on the IJB-A quality task overall. It is observed that due to the complex face distributions, some loss functions may tend to perform well on some tasks, while have poor results on some specific tasks. Comprehensive investigation is necessary to achieve good results on specific tasks.

Table 3.7: Experimental results of various models in TensorFlow on LFW and VGGFace2-test datasets.

TensorFlow-based	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
AlexNet-v2	91.1	Front	0.588	0.563	0.376	Young	0.497	0.255
		Three-quarter	0.567	0.608	0.470	Mature	0.262	0.563
		Profile	0.380	0.473	0.574			
VGG-16	95.3		0.714	0.710	0.538		0.657	0.372
			0.713	0.751	0.624		0.386	0.721
			0.545	0.621	0.708			
GoogLeNet	97.9		**0.829**	**0.811**	**0.683**		**0.783**	**0.543**
			0.816	**0.836**	**0.739**		**0.559**	**0.825**
			0.693	**0.746**	**0.791**			
Inception-v3	**98.5**		0.814	0.790	0.652		0.752	0.512
			0.791	0.815	0.709		0.532	0.776
			0.661	0.714	0.752			
ResNet-50	97.3		0.734	0.690	0.459		0.654	0.394
			0.695	0.726	0.557		0.423	0.687
			0.464	0.562	0.654			
DenseNet-121	97.2		0.805	0.787	0.632		0.728	0.468
			0.791	0.811	0.700		0.488	0.790
			0.646	0.707	0.766			

Model Accuracy Comparison between Different Frameworks

Tables 3.11, 3.12 3.13, 3.14 3.15, 3.16, 3.17, and 3.18 compare the VGG-16, Inception-v3, ResNet-50, and DenseNet-121 models across different deep learning frameworks. For the VGG-16 model, PyTorch gets the best accuracy on LFW and IJB-A quality datasets, and TensorFlow achieves the best result on VGGFace2-test dataset. For the Inception-v3 model, PyTorch gets the best accuracy on IJB-A quality dataset, and TensorFlow achieves the best result on LFW and VGGFace2-test datasets. For the ResNet-50 model, PyTorch can get the best results on LFW, VGGFace2-test and IJB-A quality datasets. For the DenseNet-121 model, PyTorch achieves the best performance on LFW and IJB-A quality datasets, and TensorFlow performs best on VGGFace2-test dataset.

Based on the above experiments, PyTorch based models tend to perform the best among these three frameworks, especially on LFW and IJB-A quality datasets. TensorFlow based models can get excellent results on VGGFace2-test dataset. Caffe based models have the worst performance. These frameworks have diverse default settings, including data pre-

Table 3.8: Experimental results of various models in TensorFlow on IJB-A dataset.

TensorFlow-based	IJB-A (%), differnt FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
AlexNet-v2	25.2	11.5	5.9	47.9	31.4	20.7
VGG-16	39.4	17.4	6.2	69.8	44.5	24.5
GoogLeNet	**58.3**	**34.1**	**15.6**	**84.1**	**62.8**	**37.9**
Inception-v3	47.7	25.6	9.7	76.6	53.4	31.6
ResNet-50	47.3	23.3	4.5	77.4	57.6	33.2
DenseNet-121	47.7	25.7	9.8	76.7	53.5	31.8

processing steps as shown in Table 3.19 and weight initialization methods as indicted in Table 3.20, resulting in different performance as well. Diverse kernel initialization methods may have 1% effect on accuracy [89]. It is suggested that the PyTorch framework should be used in order to have better accuracy.

3.3.3 Running Time Comparison

Running time is a main factor that a user should take into consideration when training deep networks. These deep learning frameworks are benchmarked by using different batch sizes in 4 CNN models (i.e. VGG-16, Inception-v3, ResNet-50 and DenseNet-121). These four models have their characteristics to test the performance of frameworks. The below batch sizes are tried in every model: 16, 32, 64, 128, 256 and 512. However, some of them may fail because a too large batch size may lead to the out of memory problem.

The system configuration is Ubuntu 16.04.3 LTS. Hardwares include Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz, 32GB RAM and 512GB SSD. Detailed information about several types of GPUs is shown in Table 3.1. Table 3.21 shows the information about different frameworks used in experiments. cuDNN [90] is a GPU-accelerated deep learning library, for neural network computing.

In subsequent experiments, running performance is evaluated by averaging 2,000 iterations. The timing method used is shown in the following:

Table 3.9: Experimental results of several loss functions on LFW and VGGFace2-test datasets.

Losses	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
Softmax	99.0	Front	0.512	0.492	0.392	Young	0.477	0.419
		Three-quarter	0.510	0.517	0.412	Mature	0.417	0.404
		Profile	0.406	0.411	0.397			
Center-loss	98.2		**0.857**	**0.846**	**0.751**		**0.816**	**0.606**
			0.847	**0.863**	**0.788**		**0.631**	**0.842**
			0.752	**0.791**	**0.812**			
SphereFace	**99.2**		0.549	0.396	0.201		0.401	0.253
			0.398	0.498	0.334		0.249	0.428
			0.217	0.332	0.487			
CosFace	99.1		0.552	0.539	0.427		0.531	0.468
			0.562	0.570	0.445		0.450	0.437
			0.450	0.446	0.407			
ArcFace	**99.2**		0.580	0.571	0.462		0.559	0.501
			0.593	0.600	0.478		0.488	0.472
			0.481	0.478	0.436			

- Caffe: "Caffe time" function is used to calculate the average running time between two consecutive iterations. It reads original images from hard disk provided by a file list.
- PyTorch: the timing function in Python is used to calculate average iteration time. It directly reads original images from hard disks.
- TensorFlow: the internal timing function in the TensorFlow-Slim library [91] outputs time details in a specified number of iterations. It uses a processed file format called TFRecord.

It should be noted that these frameworks have very flexible programming APIs. It is possible that there exist other timing methods. Therefore, we should make it clear that our implementations are not necessarily the best approach for training.

Running Time Comparison of Different Frameworks and GPU Platforms

The performance of the VGG-16 model is shown in the top left of Fig. 3.2. When the batch size is 128, the VGG-16 model fails to run on GTX 1080Ti. This is because GTX 1080Ti has a 11GB memory space, compare to 12GB in Titan Xp, Titan X(Pascal) and

Table 3.10: Experimental results of several loss functions on IJB-A quality dataset.

Losses	IJB-A (%), differnt FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
Softmax	55.3	**39.6**	20.4	81.3	72.7	53.6
Center-loss	57.9	33.3	16.1	88.1	68.8	46.4
SphereFace	54.7	38.8	**22.2**	**90.3**	**80.0**	**64.3**
CosFace	**59.6**	31.8	0.4	81.9	78.4	20.9
ArcFace	**59.6**	0.7	0	82.9	77.6	0.3

Table 3.11: Experimental results of the VGG-16 model on Caffe, PyTorch, and TensorFlow on LFW and VGGFace2-test datasets.

VGG-16	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
Caffe-based	**97.6**	Front	**0.737**	0.699	0.520	Young	0.655	**0.413**
		Three-quarter	0.707	0.746	0.604	Mature	**0.442**	0.698
		Profile	0.533	0.611	0.690			
PyTorch-based	97.2		0.726	0.698	0.510		0.648	0.353
			0.705	0.739	0.603		0.382	0.711
			0.517	0.604	0.695			
TensorFlow-based	95.3		0.714	**0.710**	**0.538**		**0.657**	0.372
			0.713	**0.751**	**0.624**		0.386	**0.721**
			0.545	**0.621**	**0.708**			

Table 3.12: Experimental results of the VGG-16 model on Caffe, PyTorch, and TensorFlow on IJB-A quality dataset.

VGG-16	IJB-A (%), different FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
Caffe-based	38.8	6.8	0.4	73.5	49.5	22.6
PyTorch-based	**47.3**	**23.7**	**7.5**	**77.6**	**54.4**	**35.5**
TensorFlow-based	39.4	17.4	6.2	69.8	44.5	24.5

Titan X(Maxwell). Since only one of the two GK110 chips in Titan Z is used, there is 6GB memory available, which is the reason why batch sizes 64 and 128 fail on Titan Z. On the other side, across different frameworks, PyTorch obtains the best performance among these

Table 3.13: Experimental results of the Inception-v3 model on Caffe, PyTorch, and TensorFlow on LFW and VGGFace2-test datasets.

Inception-v3	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
Caffe-based	96.1	Front	0.712	0.692	0.559	Young	0.626	0.415
		Three-quarter	0.706	0.719	0.597	Mature	0.442	0.653
		Profile	0.572	0.602	0.596			
PyTorch-based	97.9		0.778	0.749	0.595		0.700	0.454
			0.756	0.789	0.671		0.497	0.758
			0.606	0.680	0.743			
TensorFlow-based	**98.5**		**0.814**	**0.790**	**0.652**		**0.752**	**0.512**
			0.791	**0.815**	**0.709**		**0.532**	**0.776**
			0.661	**0.714**	**0.752**			

Table 3.14: Experimental results of the Inception-v3 model on Caffe, PyTorch, and TensorFlow on IJB-A quality dataset.

Inception-v3	IJB-A (%), different FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
Caffe-based	29.7	12.3	4.5	61.2	38.5	24.2
PyTorch-based	**51.3**	**28.2**	7.9	**81.4**	**60.2**	**38.8**
TensorFlow-based	47.7	25.6	**9.7**	76.6	53.4	31.6

Table 3.15: Experimental results of the ResNet-50 model on Caffe, PyTorch and TensorFlow on LFW and VGGFace2-test datasets.

ResNet-50	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
Caffe-based	97.5	Front	0.744	0.710	0.505	Young	0.667	**0.425**
		Three-quarter	0.718	0.753	0.591	Mature	0.455	0.697
		Profile	0.513	0.594	0.677			
PyTorch-based	**98.2**		**0.756**	**0.726**	**0.556**		**0.669**	0.412
			0.727	**0.755**	**0.629**		**0.447**	**0.710**
			0.559	**0.635**	**0.693**			
TensorFlow-based	97.3		0.734	0.690	0.459		0.654	0.394
			0.695	0.726	0.557		0.423	0.687
			0.464	0.562	0.654			

three frameworks, followed by Caffe and then TensorFlow. For Titan Xp, the improvement of PyTorch compared with Caffe and TensorFlow is 26.5% and 45.1% when the batch size

Table 3.16: Experimental results of the ResNet-50 model on Caffe, PyTorch and TensorFlow on IJB-A quality dataset.

ResNet-50	IJB-A (%), different FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
Caffe-based	39.2	1.4	0.2	74.5	46.3	5.8
PyTorch-based	**62.3**	**38.9**	**12.7**	**85.9**	**68**	**48.9**
TensorFlow-based	47.3	23.3	4.5	77.4	57.6	33.2

Table 3.17: Experimental results of the DenseNet-121 model on Caffe, PyTorch, and TensorFlow on LFW and VGGFace2-test datasets.

DenseNet-121	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
Caffe-based	97.6	Front	0.737	0.699	0.520	Young	0.655	0.413
		Three-quarter	0.707	0.746	0.604	Mature	0.442	0.698
		Profile	0.533	0.611	0.690			
PyTorch-based	**97.8**		0.755	0.731	0.576		0.682	0.419
			0.735	0.763	0.648		0.452	0.722
			0.582	0.654	0.709			
TensorFlow-based	97.2		**0.805**	**0.787**	**0.632**		**0.728**	**0.468**
			0.791	**0.811**	**0.700**		**0.488**	**0.790**
			0.646	**0.707**	**0.767**			

Table 3.18: Experimental results of the DenseNet-121 model on Caffe, PyTorch, and TensorFlow on IJB-A quality dataset.

DenseNet-121	IJB-A (%), different FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
Caffe-based	38.8	6.8	0.4	73.5	49.5	22.6
PyTorch-based	**58.8**	**33.1**	**9.8**	**83.3**	**62.4**	**42.1**
TensorFlow-based	47.7	25.7	9.7	76.7	53.5	31.6

is 16, 31.5% and 40% when the batch size is 32, and 34.1% and 38.3% when batch size is 64. Besides, PyTorch is more memory-efficient since Caffe and TensorFlow have the out of memory problem under the batch size 128, 256 or 512.

The performance comparison of the Inception-v3 is shown in the top right of Fig. 3.2.

Table 3.19: The data pre-processing of different CNN models in PyTorch/TensorFlow/Caffe.

Model name (Framework)	During training	During evaluation
AlexNet-v2, VGG-16, VGG-16-BN, ResNet-50, Inception-v3, DenseNet-121 (PyTorch)	1) Crop the image to random size and aspect ratio, followed by the resizing operation. 2) Randomly flip the image horizontally. 3) Convert range of the image to [0, 1]. 4) Normalize the image with mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225].	1) Center cropping and resize the image. 2) Convert the image pixel range to [0, 1]. 3) Normalize the image with mean [0.485, 0.456, 0.406] and standard deviation [0.229, 0.224, 0.225].
AlexNet-v2,VGG-16, ResNet-50, DenseNet-121 (TensorFlow)	1) Get the smaller side of the image, followed by aspect-preserving resizing. 2) Randomly flip the image horizontally. 3) Randomly crop the image to 224×224. 4) Subtract means [123.68, 116.779, 103.939] from each image channel.	1) Get the smaller side of the image, followed by aspect-preserving resizing. 2) Center cropping image to 224×224. 3) Subtract means [123.68, 116.779, 103.939] from each image channel.
GoogLeNet, Inception-v3 (TensorFlow)	1) Convert the image pixel range to [0, 1]. 2) Generate cropped images using a randomly distorted bounding box and resize it. 3) Randomly distort the colors (brightness, hue, saturation or contrast). 4) Transform the image range to [-1, 1].	1) Convert the image pixel range to [0, 1]. 2) Center cropping and resize the image. 3) Transform the image range to [-1, 1].
LightCNN-29 (Caffe)	1) Randomly crop the image to the input size. 2) Randomly flip the image horizontally. 3) Transform dimensions. 4) Reorder image channels to BGR. 5) Subtract mean [127.5, 127.5, 127.5] and scale the image to range [-1, 1].	1) Center cropping and resize the image. 2) Transform dimensions. 3) Reorder image channels to BGR. 4) Subtract mean [87.1, 102.7, 134.6] and scale the image to range [0, 1].
AlexNet-v1, VGG-16, GoogLeNet, ResNet-50, Inception-v3, DenseNet-121 (Caffe)	1) Randomly crop the image to the input size. 2) Randomly flip the image horizontally. 3) Transform dimensions. 4) Reorder image channels to BGR. 5) Subtract mean [87.1, 102.7, 134.6] and scale the image to range [0, 1].	

Batch sizes 128, 256 and 512 fail, due to the out of memory issue. Across different GPU platforms, the Inception-v3 model displays relatively similar performance on 5 types of GPUs like the VGG-16 model. Across different frameworks, PyTorch obtains better performance than TensorFlow and Caffe under nearly each batch size. However, the speedup is different between PyTorch and TensorFlow when batch size changes. As for Titan Xp, when batch size is 16, PyTorch achieves about 1.16% speedup. However, when batch sizes are 32, 64 separately, the improvements are 20%, 27%. This shows that the larger the batch size is, the more obvious the speedup is. In addition, Caffe is less memory-efficient compared with TensorFlow and PyTorch because Caffe runs out of memory if the batch size is 32 or 64.

The performance of the ResNet-50 model is shown in the bottom left of Fig. 3.2. There is out of memory problem if the batch size is 256 or 512. Across different frameworks, PyTorch achieves the best performance compared with TensorFlow and Caffe, except that the batch size is 64 on GTX 1080Ti. For the Titan Xp, speedup between PyTorch and TensorFlow is 25.4%, 28.2%, 31.5% and 27% when the batch size is 16, 32, 64 or 128 respectively. Titan Z and GTX 1080Ti cannot afford the batch size 128 under these three frameworks. Caffe

Table 3.20: The default weight initialization of diverse CNN models in Caffe/TensorFlow/PyTorch.
Suppose parameters of one layer have shape $(output\_channel\_num, input\_channel\_num, kernel\_size, kernel\_size)$, $fan_{\text{in}} = input\_channel\_num * kernel\_size * kernel\_size$ and $fan_{\text{out}} = output\_channel\_num * kernel\_size * kernel\_size$.

Model name (framework)	Convolutional layers	Fully connected layers
VGG-16 (PyTorch)	$W \in N\left(0, \sqrt{\frac{2}{fan_{\text{out}}}}\right)$	$W \in \left[-\sqrt{\frac{1}{fan_{\text{in}}}}, \sqrt{\frac{1}{fan_{\text{in}}}}\right]$
VGG-16 (Caffe)	$W \in \left[-\sqrt{\frac{3}{fan_{\text{in}}}}, \sqrt{\frac{3}{fan_{\text{in}}}}\right]$	$W \in \left[-\sqrt{\frac{3}{fan_{\text{in}}}}, \sqrt{\frac{3}{fan_{\text{in}}}}\right]$
VGG-16 (TensorFlow)	$W \in \left[-\sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}, \sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}\right]$	-
Inception-v3 (PyTorch)	$W \in N(0, stddev), -2 \leq W \leq 2$	$W \in \left[-\sqrt{\frac{1}{fan_{\text{in}}}}, \sqrt{\frac{1}{fan_{\text{in}}}}\right]$
Inception-v3 (Caffe)	$W_{input} \in N\left(0, \sqrt{\frac{2}{fan_{\text{in}}+fan_{\text{out}}}}\right)$	$W_{input} \in N\left(0, \sqrt{\frac{2}{fan_{\text{in}}+fan_{\text{out}}}}\right)$
Inception-v3 (TensorFlow)	$W \in \left[-\sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}, \sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}\right]$	-
ResNet-50 (PyTorch)	$W \in N\left(0, \sqrt{\frac{2}{fan_{\text{out}}}}\right)$	$W \in \left[-\sqrt{\frac{1}{fan_{\text{in}}}}, \sqrt{\frac{1}{fan_{\text{in}}}}\right]$
ResNet-50 (Caffe)	$W_{input} \in N\left(0, \sqrt{\frac{2}{fan_{\text{out}}}}\right)$, $W_{others} \in N\left(0, \sqrt{\frac{2}{fan_{\text{in}}}}\right)$	$W \in N(0, 0.01)$
ResNet-50 (TensorFlow)	$W \in \left[-\sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}, \sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}\right]$	-
DenseNet-121 (PyTorch)	$W \in N\left(0, \sqrt{\frac{2}{fan_{\text{in}}}}\right)$	$W \in \left[-\sqrt{\frac{1}{fan_{\text{in}}}}, \sqrt{\frac{1}{fan_{\text{in}}}}\right]$
DenseNet-121 (Caffe)	$W \in \left[-\sqrt{\frac{3}{fan_{\text{in}}}}, \sqrt{\frac{3}{fan_{\text{in}}}}\right]$	-
DenseNet-121 (TensorFlow)	$W \in \left[-\sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}, \sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}\right]$	$W \in \left[-\sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}, \sqrt{\frac{6}{fan_{\text{in}}+fan_{\text{out}}}}\right]$

based ResNet-50 model can run with a batch size 16 on Titan X(Maxwell), Titan X(Pascal), GTX 1080Ti and Titan Xp. Since one of two chips on Titan Z has less memory, Caffe based ResNet-50 model fails when the batch size is 16.

The performance of the DenseNet-121 model is shown in the bottom right of Fig. 3.2. It fails when the batch size is 128, 256 or 512. PyToch is better than other frameworks. For Titan Z, batch sizes of PyTorch and TensorFlow are 16 and 32; Caffe fails even when the batch size is 16. For other GPU platforms, PyTorch and TensorFlow can run with the batch size 64. For Titan Xp, PyTorch obtains 36.5%, 41.7% and 43.4% speedup than TensorFlow when batch sizes are 16, 32 and 64, respectively.

Across different GPU platforms for the VGG-16, Inception-v3, ResNet-50 and DenseNet-

Table 3.21: Versions of deep learning frameworks used in experiments.

Frameworks	Major version	cuDNN	CUDA
Caffe	1.0.0	V7.0	V9.0
TensorFlow	1.6.0	V7.0	V9.0
PyTorch	0.3.1	V7.0	V9.0

121 models, Titan Xp obtains the best performance, which is slightly better than GTX 1080Ti, Titan X(Pascal) and Titan X(Maxwell). Other four GPUs are at least 2 times faster than Titan Z. As shown in Table 3.1, architectures of Titan Z, Titan X(Maxwell) are Kepler, Maxwell, and the architecture of Titan X(Pascal), GTX 1080Ti and Titan Xp is Pascal. This shows that the more recent GPU architecture is, the faster the running performance is. Although Titan X(Pascal), GTX 1080Ti and Titan Xp have the same GPU architecture, their specifications are more or less different. GTX 1080Ti has better boost clock, memory speed and memory bandwidth than Titan X(Pascal), and Titan Xp has more CUDA cores, higher memory speed, wider memory interface width and memory bandwidth than GTX 1080Ti. This explains why Titan Xp is a little faster than GTX 1080Ti, and GTX 1080Ti is faster than Titan X(Pascal).

Across different frameworks, there are mainly two factors which may result in different computation time. First, in training CNNs, convolutional layers which are the most time-consuming layers are usually invoked by the high performance library cuDNN [90]. However, there are several types of convolution implementations, such as GEMM, FFT and Winograd. TensorFlow prefers to use the Winograd algorithm, and PyTorch could autotune to find the most efficient convolutional algorithm. While Caffe uses GEMM-based convolution. The FFT- and Winograd-based convolutions are faster than GEMM-based convolution in general [63]. The sub-optimal convolution makes Caffe slower than TensorFlow and PyTorch in most cases. Second, PyTorch uses NCHW layout naively which is implicitly supported in cuDNN, while TensorFlow uses NHWC layout. As shown in [92], changing data layout in TensorFlow from NHWC from NCHW leads to 15% speedup. This explains why PyTorch is faster. It is worth noting that PyTorch is a dynamic framework to maximize flexibility, and TensorFlow is a static framework with less computation cost. However, PyTorch minimizes

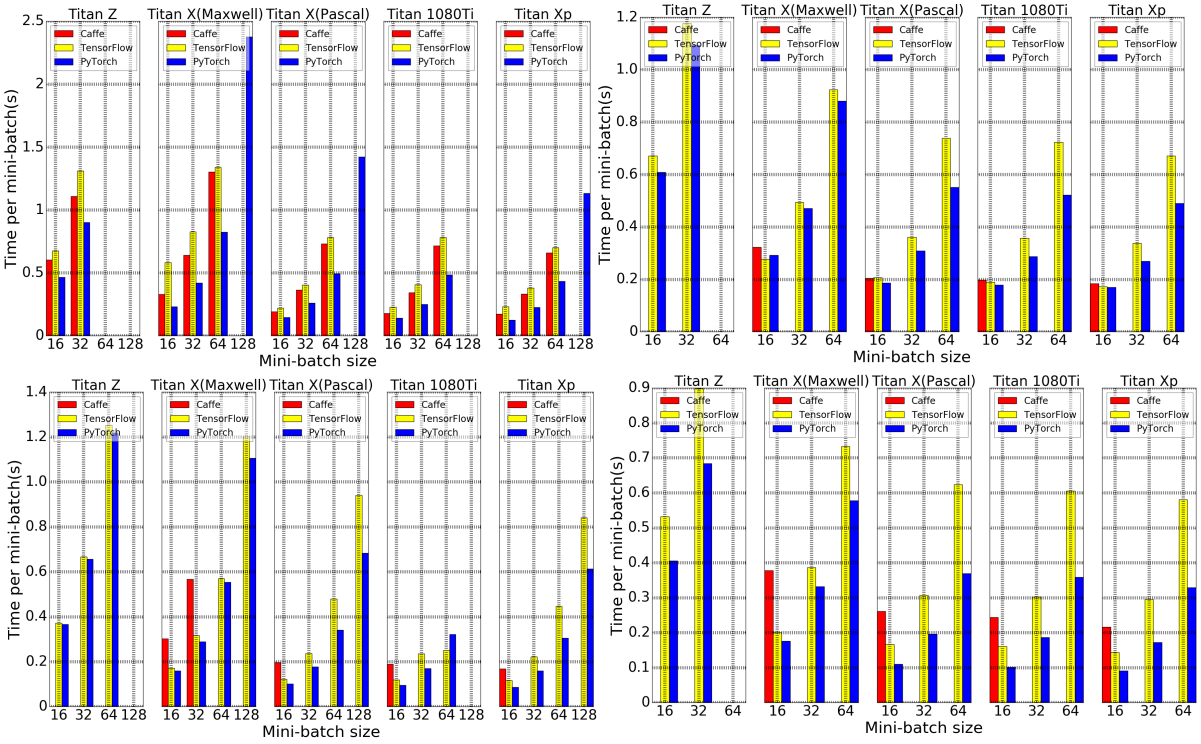

Figure 3.2: Running time comparison of VGG-16 (top left), Inception-v3 (top right), ResNet-50 (bottom left), and DenseNet-121 (bottom right) models on GPU platforms.

the computational cost of graph construction in every iteration and implements faster GPU kernels for frequent workloads, allowing for more efficient dynamic computation.

Scalability of Different Deep Learning Frameworks on Multi-GPUs

Since the AlexNet model [25] won the ILSVRC 2012 challenge [85], CNNs have become ubiquitous in various computer vision tasks [93] [94] [95] [96] [97] [98]. Making deeper and more complicated networks has been the general trend to improve the performance [47] [50] [51] [49]. However, a single GPU only has a limited memory space, which seriously limits the network capacity. On the other side, training CNNs is time-consuming, especially for deeper networks and large training datasets. Therefore, scalability is very critical for deep learning frameworks, allowing for accelerating the training process by spliting the data across multiple GPUs or machines, and training extremely large models which would not fit into the limited memory of one GPU. Support of multiple GPUs and machines becomes necessary for frameworks. The distributed synchronous stochastic gradient descent (SGD)

method [99] [100] is widely used to achieve a better scaling performance on multiple GPUs or machines.

In this Section, running time of the VGG-16, Inception-v3, ResNet-50 and DenseNet-121 model is evaluated on 1, 2 and 4 Titan X(Pascal) GPUs. A proper batch size for each model is determined to make the model run on every framework and better utilize the GPU resources. We use the metric (i.e. # of samples processed per second) to measure the throughput. To show the scalability, the speedup is calculated to indicate how much the throughput with GPUs could be increased:

$$speedup = \frac{throughput_{2\times(\#\text{ of gpus})} - throughput_{\#\text{ of gpus}}}{throughput_{\#\text{ of gpus}}} \tag{3.6}$$

Ideally, the value of speedup should be 100% when the GPU number is doubled.

Results of the VGG-16 model with a batch size 64 per GPU are shown in the top left of Fig. 3.3. PyTorch has the highest throughput, which can process 127, 203 and 377 images per second as the GPU number increases from 1, 2 to 4, compared with 79, 156 and 316 images in Caffe. TensorFlow has the lowest processing speed on the VGG-16 model, which can process 76, 117 and 196 images per second, respectively. It can be seen that on a single GPU, PyTorch is the best, and Caffe is slightly better than TensorFlow. With the number of GPUs doubls, scalability of Caffe is the most remarkable (97%), while PyTorch and TensorFlow have similar speedup performances, over 50%. When the GPU number is 4, speedup of Caffe is up to 103% compared with 68% in TensorFlow and 86% in PyTorch.

The performance of the Inception-v3 model with a batch size 25 per GPU is shown in the top right of Fig. 3.3. As for the throughput across deep learning frameworks, PyTorch (114, 192 and 357 images per second on 1, 2 and 4 GPUs) is faster than Caffe (74, 146 and 294 images) and TensorFlow (106, 160 and 279 images). On a single GPU, PyTorch displays a better performance than TensorFlow and Caffe. When the GPU number doubles, speedup of Caffe, TensorFlow and PyTorch is 97%, 51% and 68%. When the number of GPUs increases to 4, Caffe, TensorFlow and PyTorch have 101%, 68% and 86% improvement than the performance on 2 GPUs.

The performance of the ResNet-50 model with a batch size 25 per GPU is shown in the bottom left of Fig. 3.3. PyTorch can process 464, 268 and 171 images per second on 4, 2

and 1 GPUs, compared with 300, 149 and 77 images in Caffe and 252, 222 and 160 images in TensorFlow. On a single GPU, PyTorch is the fastest compared with TensorFlow and Caffe. When the number of GPUs doubles, speedup of Caffe, TensorFlow and PyTorch is 94%, 39% and 57%. If the number of GPUs changes to 4, Caffe has the best speedup (101%), while PyTorch is much better than TensorFlow (73% compared with 13.5%).

The batch size per GPU of the DenseNet-121 model is set to 20 in our experiments, and results are shown in the bottom right of Fig. 3.3. TensorFlow (368, 212 and 119 images per second) has the second-best throughput behind PyTorch (388, 237 and 155), with Caffe lagging relatively far behind (238, 115 and 59). The throughput of PyTorch is slightly better than TensorFlow, and PyTorch and TensorFlow are almost two times faster than Caffe on a single GPU. When the number of GPUs is doubled, scalability of Caffe (95%) is the best, followed by TensorFlow (78%) and PyTorch (53%). With 4 GPUs, Caffe, TensorFlow and PyTorch further achieve 107%, 74% and 64% speedup, respectively.

As for the efficiency of frameworks, TensorFlow implicitly uses NHWC tensor layout. On the contrary, PyTorch adopts NCHW layout naively in which cuDNN is better optimized. This can explain why PyTorch is able to process more images per second than TensorFlow. Meanwhile, PyTorch tends to use FFT-based convolutions which are faster than GEMM-based convolutions used in Caffe [92]. Because the convolutional operation is the computation bottleneck in CNN models, PyTorch is faster than Caffe. Due to its excellent throughput in above experiments, PyTorch is selected to conduct the remaining experiments.

But as for the scalability of different frameworks, it should be noted that the speedup of Caffe is around 100%. We attribute this to the reason that loading images from hard disks instead of more efficient LMDB data format becomes a bottleneck in Caffe. To train CNN models with SGD, the model is iteratively updated with the feeding data. There are four steps generally in one iteration. 1) Load a mini-batch of data from hard disks to CPU memory and transfer the data to GPU memory. The time in this step is represented by t_{data}. 2) Each GPU launches kernels to finish the forward- and backward- operations. 3) The gradients from all GPUs are aggregated. t_{agg} refers to the time of the gradient aggregation. 4) The model in each GPU is updated based on the aggregated gradients. Let t_{gpu} represent the total time in step 2) and 3). It is obvious to represent the time t_{iter} in one iteration with

the following equation:

$$t_{iter} = t_{data} + t_{gpu} + t_{agg} \tag{3.7}$$

Since loading images for the next iteration can be overlapped with GPU computation for the current iteration, and gradient aggregation on the previous iteration can be parallelized with backward operation on the current iteration. So the average iteration time in Caffe is:

$$t_{iter} = max\{t_{data}, t_{gpu}, t_{agg}\} \tag{3.8}$$

Because we receive info "waiting for data" during training in Caffe, prefetching data is too slow for the next iteration, and GPU computation and gradient aggregation needs to wait for the data loading in every iteration. In other words, t_{data} is larger than t_{gpu} and t_{agg}. When more GPUs are used, GPU computation and gradient aggregation are still hidden behind the data loading, which leads to a bottleneck in the scaling performance. Therefore a linear speedup is achieved in Caffe.

Efficient data communication between GPUs is important to ease the overhead of gradient synchronization: TensorFlow and PyTorch use the Google's Remote Procedure Call (gRPC) [101] Library and NVIDIA Collective Communications Library (NCCL) [102], respectively. Since NCCL has a higher efficiency and lower latency in collective gradient communications [103], TensorFlow has a sub-optimal scaling performance compared with PyTorch.

3.3.4 Comparison of Different Training Datasets

In this section, we investigate the effect of different training datasets (i.e. UMDFaces, WebFace, VGGFace2 and MS-Celeb-1M) on five test datasets (i.e. LFW, VGGFace2-test, IJB-A quality, DFW and UMDFaces). Experimental results are shown in Tables **??** and 3.24.

Based on comparative results of different frameworks, PyTorch is selected to run experiments in this section because of its high throughput and relatively good accuracies. Due to its good performance, the ResNet-50 is used as the baseline model. The model is trained with 14 epochs. The learning rate is set to 0.1 and divided by 10 every 6 epochs. The batch size is 210 on 3 GPUs. In order to compare training datasets fairly, overlapped subjects

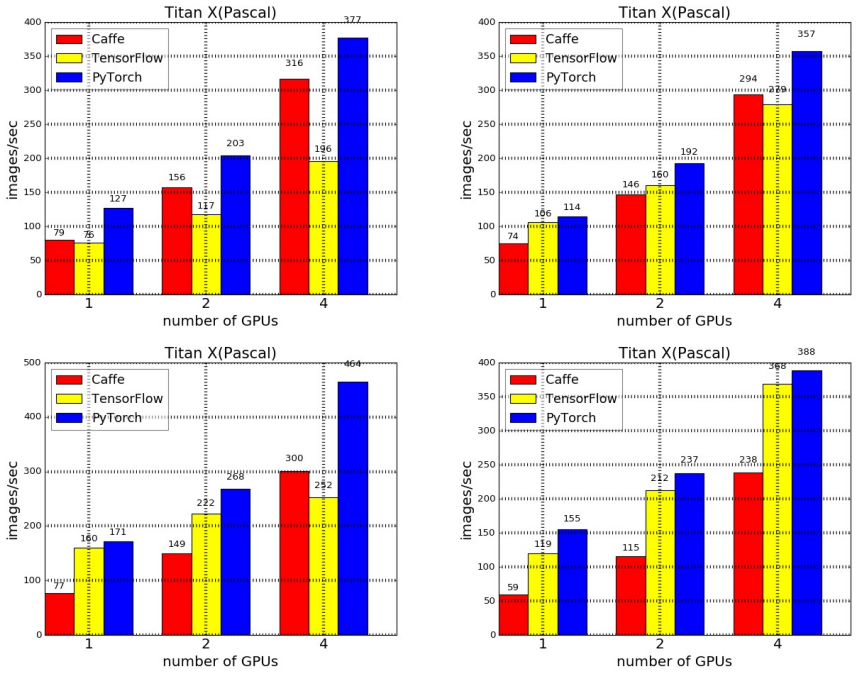

Figure 3.3: Scalability comparison of the VGG-16 model with a batch size 64 (top left), the Inception-v3 model with a batch size 25 (top right), the ResNet-50 model with a batch size 25 (bottom left) and the DenseNet-121 model with a batch size 20 (bottom right) w.r.t. multiple GPUs.

Table 3.22: Experimental results of the ResNet-50 trained on the UMDFaces, WebFace and VGGFace2 datasets and tested on the LFW and VGGFace2-test datasets.

Dataset	LFW (%)	VGGFace2 (Similarity Score)						
		Pose				Age		
			Front	Three-quarter	Profile		Young	Mature
UMDFaces	95.5	Front	0.783	0.755	0.555	Young	0.707	0.436
		Three-quarter	0.760	0.793	0.645	Mature	0.452	0.743
		Profile	0.558	0.642	0.749			
WebFace	98.2		0.783	0.761	0.601		0.707	0.438
			0.767	0.793	0.677		0.464	0.762
			0.611	0.683	0.739			
VGGFace2	98.8		**0.820**	**0.792**	**0.673**		**0.748**	**0.497**
			0.798	**0.826**	**0.736**		**0.522**	**0.790**
			0.678	**0.740**	**0.786**			
MS-Celeb-1M	**98.9**		0.776	0.734	0.610		0.683	0.412
			0.739	0.777	0.684		0.431	0.739
			0.615	0.685	0.741			

Table 3.23: Experimental results of the ResNet-50 trained on the UMDFaces, WebFace, and VGGFace2 datasets and tested on the IJB-A quality dataset.

Dataset	IJB-A (%), different FARs					
	Low2High			Middle2High		
	1%	0.1%	0.01%	1%	0.1%	0.01%
UMDFaces	40.8	19.5	7.7	69.4	45.5	29.3
WebFace	54.5	29.8	6.9	82.3	62.2	41.8
VGGFace2	**81.8**	**62.8**	**38.1**	94.8	85.4	69.0
MS-Celeb-1M	76.2	59.5	37.6	**95.0**	**89.1**	**77.2**

Table 3.24: Experimental results of the ResNet-50 trained on the UMDFaces, WebFace and VGGFace2 datasets and tested on the DFW and UMDFaces-test datasets.

Dataset	DFW (%), different FARs		UMDFaces (%), different FARs								
	1%	0.1%	Easy			Moderate			Difficult		
			1%	0.1%	0.01%	1%	0.1%	0.01%	1%	0.1%	0.01%
UMDFaces	21.4	7.6	76.5	59.3	44.9	65.6	43.0	23.8	56.7	32.6	19.4
WebFace	**50.6**	**26.6**	69.8	54.3	42.7	57.7	36.8	22.6	48.0	24.4	10.4
VGGFace2	49.0	25.3	**80.3**	**66.6**	**54.5**	**72.2**	**55.9**	**43.5**	**66.0**	**43.4**	**24.7**
MS-Celeb-1M	26.8	8.4	66.9	49.4	37.1	55.1	30.2	14.5	47.1	22.9	8.3

between training and test datasets are removed. After removing, an overview of MS-Celeb-1M, UMDFaces, VGGFace2, and WebFace datasets is presented in Tab. 3.25. In terms of

Table 3.25: An overview of MS-Celeb-1M, UMDFaces, VGGFace2, and WebFace datasets, after removing overlapped subjects with test datasets.

Dataset	# of subjects	# of images	# of images per subject (average)
MS-Celeb-1M	84,936	3,838,654	45
UMDFaces	5,222	236,856	45
VGGFace2	8,410	3,141,890	374
WebFace	10,182	393,700	39

breadth (number of subjects), MS-Celeb-1M dataset (84,936) gets the first place, followed by WebFace dataset (10,182), VGGFace2 dataset (8,410), and then UMDFaces dataset (5,222). As for depth (number of images per subject), VGGFace2 dataset (374) is significantly superior than both UMDFaces (45), MS-Celeb-1M (45) and WebFace (39) datasets. The depth of UMDFaces, MS-Celeb-1M and WebFace datasets are very similar.

For the LFW verification task, MS-Celeb-1M dataset achieves the best performance, 0.1% improvement compared with the VGGFace2 dataset. The performance of WebFace dataset slightly lags behind, while UMDFaces dataset ranks the last by a large margin. This indicates that MS-Celeb-1M, VGGFace2 and WebFace datasets are more suitable for this task. This may be because these three datasets are broader compared to UMDFace dataset, allowing the model to learn more discriminative features to increase inter-class distances and verify if one pair of images belong to the same subject or not. It is worth noting that the LFW dataset was an early benchmark for face verification. Its performance is almost saturated [30] [31] because most faces in LFW dataset are close to frontal. MS-Celeb-1M and VGGFace2 datasets may have good potentials for more challenging tasks.

For the cross-pose face matching task on VGGFace2-pose and UMDFaces-test, the VGGFace2 dataset outperforms the UMDFaces for training by a significant margin. WebFace and MS-Celeb-1M datasets lag behind. Since modern deep learning is heavily data-driven, the generalization performance depends on the distribution of the training dataset [104]. It is known that the data generation pipeline in VGGFace2 dataset aims to encourage age and pose diversity for each subject [82]. Pose information is provided in UMDFaces dataset which has more pose variations compared to the WebFace [58]. Although both UMDFaces

and VGGFace2 datasets have pose information, their large gaps on depth and breadth explains why VGGFace2 dataset achieves much better performances than UMDFaces dataset for cross-pose matching tasks. In contrast, despite a large number of images in the MS-Celeb-1M dataset, most of them are frontal faces because of without specific data collection rules in the data collection stage.

For the cross-quality face matching task on IJB-A, VGGFace2 dataset ranks the first on the Low2High task, and MS-Celeb-1M gets the first place on the Middle2High task, respectively, followed by WebFace dataset and then UMDFaces dataset. Because images in these four datasets are crawled from the Internet, they contain various quality levels. If there are more images in the training dataset, there are more images with different image qualities, which guide the model to learn robust features to various image qualities. This explains why VGGFace2 and MS-Celeb-1M datasets are substantially superior to WebFace and UMDFaces. On the other side, MS-Celeb-1M dataset (cleaned by DeepGlint) tends to have high ratios about middle- and high- level image qualities than VGGFace2 dataset, resulting in a better performance in the Middle2High task.

For the DFW challenge, WebFace dataset performs better than other two datasets, although it is just marginally better than VGGFace2 dataset. The reason may be that images from WebFace dataset are not manually identity labeled and appropriate noise level makes the model more robust to recognize the disguised faces and reject impersonators. VGGFace2 dataset contains much more information and has closer gap with DFW dataset than UMDFaces dataset in terms of depth and breadth, resulting in the remarkable performance improvement. On the other side, MS-Celeb-1M dataset cleaned by DeepGlint (http://trillionpairs.deepglint.com/) may have fewer disguised faces in the dataset, leading to an inferior performance

For the cross-age face matching task on VGGFace2-age dataset, VGGFace2 dataset obviously obtains a better performance, achieving around 3% improvement because it contains more age variations within the same subject during the image crawling stage [82]. As shown in Table **??**, for these three training datasets, the performance improvement is more significant when face matching across young and mature faces, which is more difficult than young to young and mature to mature face matching. Besides, young to young matching has more

remarkable improvement compared with mature to mature matching. These results can be attributed to the reason that more young faces in VGGFace2 dataset enable the model to learn more age-invariant features, especially for young to mature face matching. Although WebFace dataset has more images than UMDFaces dataset, these images are more likely to deem as mature faces. This explains why mature to mature face matching has better accuracy improvement, followed by young to mature matching. Young to young face matching performance remains almost the same for WebFace and UMDFaces datasets. The ratio of mature faces in the MS-Celeb-1M dataset may be much higher than young faces. Therefore, MS-Celeb-1M dataset performs poorly in the cross-age task.

Because VGGFace2 dataset takes depth and breadth into consideration, guaranteeing rich intra-subject variations and inter-subject diversity. Moreover, it collects face images with a wide range of poses and ages. This explains why it obtains the best accuracy on cross-pose, cross-age and cross-quality face matching tasks. Since a large number of high-quality faces in the revised MS-Celeb-1M dataset, it achieves the best result on the LFW verification task. For the DFW challenge, some outliers in WebFace dataset make CNN more robust to disguised faces and impersonators, so WebFace provides slightly better performance than VGGFace2. Although UMDFaces dataset has a smaller number of images than WebFace dataset, it has more pose variations and performs better on the cross-pose face matching task.

3.4 Summary

We have performed a benchmark of various models (i.e. AlexNet-v1, AlexNet-v2, VGG-16, VGG-16-BN, GoogLeNet, Inception-v3, ResNet-50, DenseNet-121 and LightCNN-29, Center-loss, SphereFace, CosFace and ArcFace) for face recognition under different frameworks (i.e. PyTorch, TensorFlow and Caffe) in two folds. First, we compare the accuracy of different CNN models within the same deep learning framework. Experimental results show that models, such as the Center-loss, SphereFace, CosFace, ArcFace, GoogLeNet, Inception-v3 and ResNet-50 models achieve better performances. Second, we compare the accuracy of the same CNN model under different frameworks. Results show that the PyTorch based

models tend to obtain the best performance among these three frameworks because of its better weight initialization methods and data pre-processing steps, followed by TensorFlow and then Caffe.

The running time performance of frameworks PyTorch, TensorFlow and Caffe, and GPU platforms are compared as well. PyTorch has the highest throughput and TensorFlow obtains slightly better throughput than Caffe in most cases because of the difference of data layout and convolution implementations among these frameworks. For five GPU platforms, five different designs (e.g. architecture, CUDA core number, memory speed and bandwidth) are tested and we found that the Titan Xp displays slightly faster speed than GTX 1080Ti, Titan X(Pascal) and Titan X(Maxwell) and Titan Z is nearly 2 times slower than the other four GPUs. We also compare the scalability of different frameworks. Experiments show that Caffe has almost linear speedup because of the data loading bottleneck. Because the communication library used in PyTorch (i.e. NCCL) is better than TensorFlow (i.e. gRPC), PyTorch has a slightly better scaling performance.

Four training datasets (i.e. UMDFaces, WebFace, VGGFace2, and MS-Celeb-1M) are investigated for training the CNN models, and five test datasets (i.e. LFW, VGGFace2-test, IJB-A, DFW, and UMDFaces-test) are used for recognition accuracy measures. Because of its rich intra-subject variations and inter-class diversity, age and pose information, VGGFace2 dataset is preferable to train CNN models; MS-Celeb-1M dataset (http://trillionpairs.deepglint.com/) has more high-quality frontal faces, which makes it suitable for the controlled face verification task, like access control; Due to the larger pose variations, UMDFaces dataset achieves a good performance on pose related tasks than the WebFace dataset; WebFace allows CNNs to learn more robust features, distinguishing disguised faces and impersonators and surpassing UMDFaces and VGGFace2 datasets on the DFW challenge.

Further, a set of deep face models trained on the WebFace dataset under three frameworks will be made publicly available. Users can take them as baselines to further fine-tune them for their tasks. This will save the training time significantly required for face recognition models, and helpful for beginners.

In future, we plan to evaluate the performance of distributed frameworks over multi-machine environments. And also, newer state-of-the-art face recognition models will be

added continuously with more and more newer GPU platforms.

Chapter 4

LS-CNN: Characterizing Local Patches at Multiple Scales for Face Recognition

4.1 Motivations

Although great progresses have been made in face recognition, few existing works could incorporate multi-scale representations and characterize local regions together to describe faces.

Learning multi-scale information is necessary to boost the face recognition performance. Discriminative face regions may occur at multiple scales. For example, as shown in Fig. 4.1 Columns 1, 2 and 3, even though faces have dramatic changes, some local regions remain to be similar but have different sizes. Thus, perceiving information from multiple scales is important for understanding local facial regions. Different from the prior work [55] that concatenates multi-scale features from the last two layers, we propose the Harmonious multi-Scale Networks (HSNet) which covers a wide range of receptive fields. It learns multi-scale features from two harmonious perspectives. On one hand, Inception [48] [49] extracts multi-scale representations in a single layer by kernels of different sizes. On the other hand, DenseNets [51] form multi-scale information from different layers because each layer is directly connected to each preceding layer. Besides, due to good information and gradient

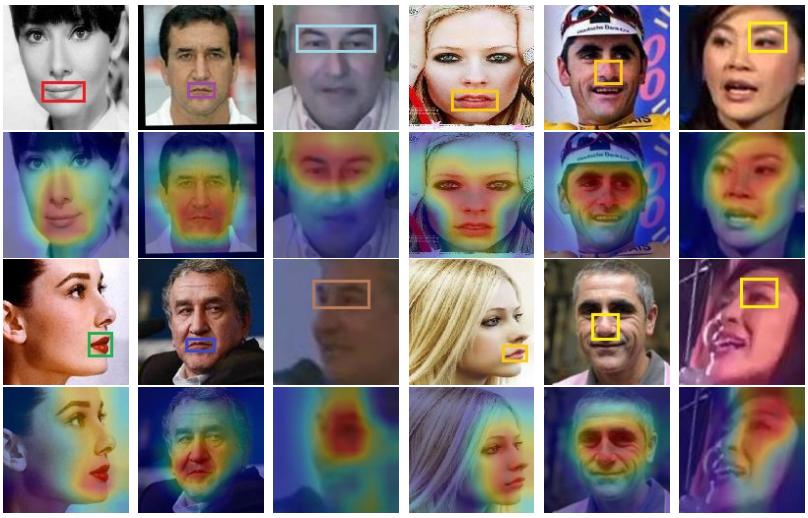

Figure 4.1: Some positive pairs from CFP, CALFW and IJB-A quality datasets and their corresponding class activation maps (CAMs) [3] learned by the proposed LS-CNN model. Faces are affected by several challenging factors, such as pose, aging, occlusion, resolution, blur, expression and illumination. Column 1: Similar mouths with different sizes. Column 2: Similar mouths with different sizes. Column 3: Similar eyes with different sizes. Column 4: Similar mouth parts. Column 5: Similar pointy noses. Column 6: Similar eyes.

flow, the HSNet model can scale naturally hundreds of layers. Because very deep models have a better representational ability than shallower ones [105], the HSNet has a good representational power without optimization difficulties, modeling complex faces like Fig. 4.1.

Spatial attention is introduced to characterize informative regions automatically. Global face geometry and appearances may be significantly different. As a consequence, identifying similar facial regions is of vital importance. As shown in Fig. 4.1, some local regions remain similar despite pose variations (mouths in Column 4), aging (noses in Column 5) and face quality changes (eyes in Column 6). To learn local representations, several works train CNNs on cropped patches around face landmarks [34, 38, 39, 55]. However, face landmark detection may fail in some cases, as illustrated in Fig. 4.2. Illumination changes make detailed face texture missing (Row 1, Columns 1, 2); occlusions cause some face organs to be invisible, such as microphone on the mouth (Row 1, Columns 3, 4); poses are self-occlusion and can lead to some face regions completely missing (Row 1, Columns 5, 6). Besides, we notice two observations. First, different face regions exhibit different discriminative abilities. As

Figure 4.2: Some challenging faces and their corresponding class activation maps (CAMs) [3] learned by LS-CNN model. Faces are influenced by illumination changes (Columns 1, 2), occlusions (Columns 3, 4), and pose variations (Columns 5, 6). MTCNN [4] fails to detect landmarks, while our LS-CNN model can locate discriminative face regions (Row 2).

Figure 4.3: Four channels are visualized for a positive pair with similar eyes, where purple areas correspond to essential areas, and green areas mean less important ones. Different weights are assigned through the channel attention.

presented in [1], areas between eyes and eyebrows are more discriminative than those between the nose and mouth in frontal faces. Second, a convolution kernel is considered as a feature detector [106]. It can detect specific features, while may have noisy responses on distraction parts, such as an uncontrolled background in Fig. 4.9 Column 5. Based on the discussion above, we propose an attention mechanism, i.e. local aggregation network (LANet), to localize the most discriminative face regions. Moreover, background information is filtered flexibly to reduce distraction. As illustrated in Fig. 4.2, our model can not only locate faces, but also focus on useful face regions and filter out distraction regions.

Further, channel attention is incorporated into the Harmonious multi-Scale CNN (HSNet) which highlights important channels and suppresses less informative ones. When employing a CNN to extract features, channels that contain information with various importance are extracted. This observation comes from the fact that different convolution kernels detect different features. As observed in Fig. 4.3, four channels correspond to different face parts for

each face image. Besides, hierarchical channels from multiple layers in the HSNet should have different discriminative abilities. This is mainly based on the following two reasons. First, channels from low layers may contain local face details, and high layers tend to have high-level representations. Second, local discriminative face regions from different face images may have different sizes (e.g. Fig. 4.1 Columns 1, 2 and 3), which may appear at different layers. To overcome these problems, the SENet module [107] is incorporated into our HSNet model to assign weights for each channel, where discriminative channels are enhanced while irrelevant channels are suppressed. For example, the similar eyes of faces in Fig. 4.3 are useful information to verify that these two faces belong to the same subject. Through the SENet module, these less informative channels are assigned with smaller weights (Columns 2, 3) and important channels that capture eye information have larger weights (Columns 4, 5).

Since SENet model works locally and LANet model applies globally on channels, it is intuitive to combine them together to form the Dual Face Attentions (DFA). Attention mechanisms have been widely used in various tasks [107–109]. However, to the best of our knowledge, this is the first time to apply the attention module for the general face recognition task except for video face recognition which aggregates multiple frames into one representation. Thus, Local and multi-Scale Convolutional Neural Networks (LS-CNN) model is developed by integrating the DFA into HSNet model. As demonstrated in Fig. 4.1, the LS-CNN model can locate discrminative local regions despite their various sizes.

The proposed Local and multi-Scale Convolutional Neural Networks (LS-CNN) model is studied for the unconstrained face recognition task. Our major contributions include:

1. We propose the Harmonious multi-Scale Network (HSNet), which allows us to learn multi-scale features from different perspectives: utilization of different kernel sizes in a single layer; combination of multi-scale feature maps from different layers. It outperforms many backbone networks in terms of accuracy, model capacity and parameter efficiency.

2. Channel and spatial attentions are incorporated to form the Dual Face Attentions (DFA). As far as we know, this is the first time to use attention modules for the

general face recognition task. The SENet module is integrated to learn what features to emphasize: more informative channels are highlighted and less important ones are inhibited. The LANet module is proposed to decide where to focus: discriminative local patches are assigned with larger weights, and less useful ones have smaller weights. Combining the DFA with the HSNet results in the proposed Local and multi-Scale Convolutional Neural Networks (LS-CNN) model.

3. Trained on publicly available CASIA-WebFace or VGGFace2 dataset, the proposed LS-CNN model yields better results than state-of-the-art methods on cross-age, cross-pose and cross-quality face matching. It also achieves a comparable performance on the LFW dataset.

4.2 A New Deep Network

Our proposed deep network, called Local and multi-Scale Convolutional Neural Networks or simply LS-CNN mainly contains four modules: Inception, DenseNet, SENet, and LANet. We will describe them in the following.

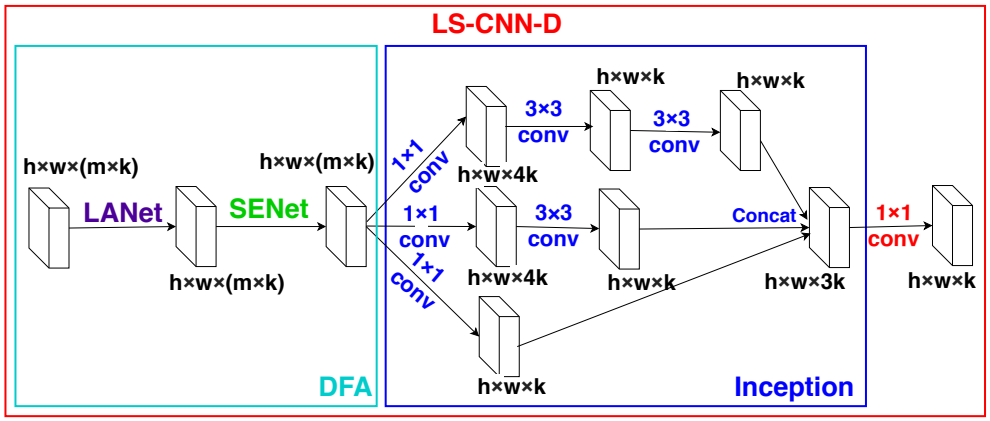

Figure 4.4: LS-CNN-Dense block (LS-CNN-D): the composite operation of the DFA-Inception module in dense blocks of DenseNets, where h and w refer to height and width of channels, respectively. k and m mean the growth rate and m_{th} layer within a dense block, respectively. DFA consists of LANet and SENet.

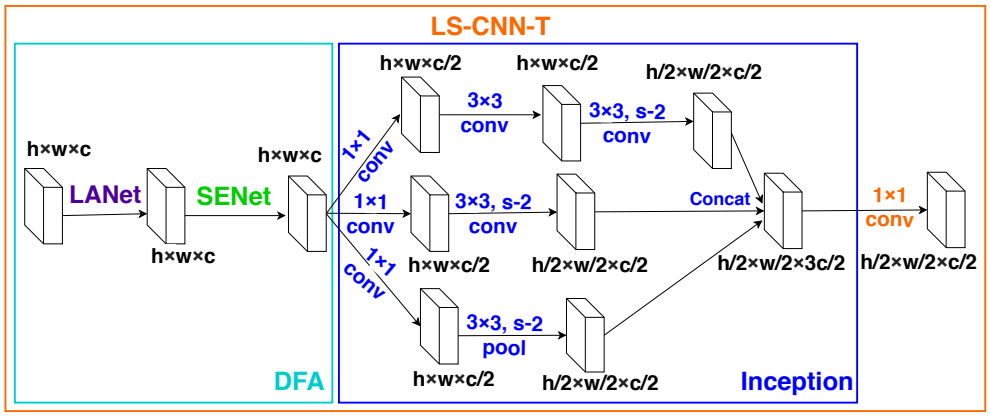

Figure 4.5: LS-CNN-Transitional (LS-CNN-T): the implementation of the DFA-Inception module in the transitional layer of DenseNets, where h, w and c refer to height, width and number of channels, respectively. S-2 means stride 2. DFA consists of LANet and SENet.

4.2.1 Inception Module

The Inception [48] maps cross-channel and spatial correlations simultaneously by using different convolution kernels. Following the Inception-v3 [49], two consecutive 3×3 filters are used to replace 5×5 filters. This can reduce about 28% parameters as well as computation time without loss of the representation ability. As the Inception shown in Fig. 4.4, we have three branches: 1×1 convolution, 3×3 convolution and two 3×3 convolutions. Meanwhile, the bottleneck layer (i.e. 1×1 convolution) is used in the branch wherever computational requirements would dramatically increase otherwise.

To reduce the channel size in a multi-scale way, a max-pooling branch and two convolution branches with stride 2 for each are used. The bottleneck layer is first used to reduce the number of channels. See the Inception in Fig. 4.5 for an illustration.

4.2.2 DenseNet Module

In order to improve the information and gradient flow, dense connections are proposed in [51]. Each layer is directly connected to each preceding layer. Namely, the output of the m_{th} layer can be stated as $x_m = H_m([x_0, x_1, ..., x_{m-1}])$, where $[x_0, x_1, ..., x_{m-1}]$ represents channel concatenation from preceding layers (i.e. $0, 1, ..., m-1$). H_m is the composite LS-CNN-D operation shown in Fig. 4.4, which outputs k channels. As a result, the m_{th} layer

has $(m-1)+k_0$ input channels and outputs k channels, where k_0 is the number of input channels for the dense block. Let k refer to the growth rate, which controls the width of the network. Like [49] [110], the bottleneck layer is used before the 3×3 convolution to reduce the number of input channels to $4k$, improving parameter efficiency.

On the other hand, as an essential part of CNNs, the pooling operation reduces the channel size to produce more robust features. The pooling is used between two dense blocks, which is referred as the transitional layer. To improve the computational efficiency, the transitional layer outputs $c/2$ channels if the previous dense block produces c channels.

Because of dense connections, rich hierarchical features from different layers contain multi-scale representations. Meanwhile, intermediate layers contain middle-level visual features about object parts, and high layers detect high-level representations about objects [106], so different levels of visual semantic features (e.g. local details) may be combined to benefit face recognition.

4.2.3 Harmonious Multi-Scale Networks

On one hand, the Inception module characterizes faces at various scales in a single layer. On the other hand, the DenseNet module concatenates hierarchical features from different layers with various receptive fields. Therefore, a new backbone network, i.e. Harmonious multi-Scale Networks (HSNet), which integrates the Inception with the DenseNet is proposed. It extracts multi-scale features from two complementary perspectives. Besides, the HSNet model has identity mapping and deep supervision, enabling it to have a good generalization capacity for complex faces.

As shown in Fig. 4.4, the Inception module is used in dense blocks of the DenseNet. Inception module contains three branches: 1×1, 3×3, and two 3×3 convolutional kernels. The 1×1 branch outputs k channels, where k refers to the growth rate. The other two branches first output $4k$ channels by a bottleneck layer (i.e. 1×1 kernel) to improve parameter efficiency. Then different convolutional kernels are applied in each branch to output k channels, respectively. Next, a concatenation operation is conducted among three branches to output $3k$ channels. Finally, a bottleneck layer is used to output k channels.

The Inception module is also applied in transitional layers of DenseNets. As shown in Fig. 4.5, the grid size reduction method in [49] is used. Let c represent the number of input channels of the transitional layer. A bottleneck layer is first employed to output $c/2$ channels. After that, a max-pooling and two different convolutional operations with stride 2 are used in every branch to reduce the channel size. In the following, channels from three branches are concatenated together to output $3c/2$. Last, another bottleneck layer is applied to reduce the number of channels from $3c/2$ to $c/2$.

4.2.4 LANet Module

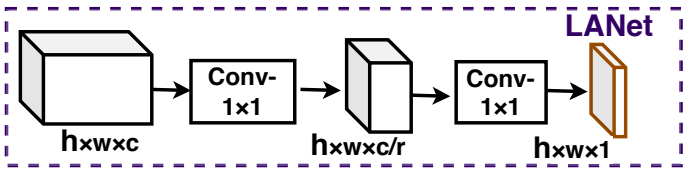

Figure 4.6: The LANet module, where h, w and c refer to height, width and number of channels, respectively.

To characterize local regions automatically, the local aggregation network (LANet), shown in Fig. 4.6, is proposed. It has two consecutive 1×1 convolutional layers to aggregate spatial information across channels to one channel. The first convolutional layer outputs c/r channels, where c and r refer to the number of input channels and reduction ratio, respectively, followed by a ReLU function [25]. Then, another 1×1 convolutional layer outputs 1 channel with a sigmoid function, namely spatial attention. Finally, every channel is scaled by the spatial attention.

Since every unit in the spatial attention corresponds to a local patch of the input image, more informative local regions are expected to have higher weights and less important ones are pushed to have smaller values. Since the input and output channels have the same size, the LANet can be easily plugged into any existing CNNs.

4.2.5 SENet Module

The SENet [107] module is used to select informative channels and suppress less discriminative ones on demand. For example, when we want to verify a positive face pair with similar

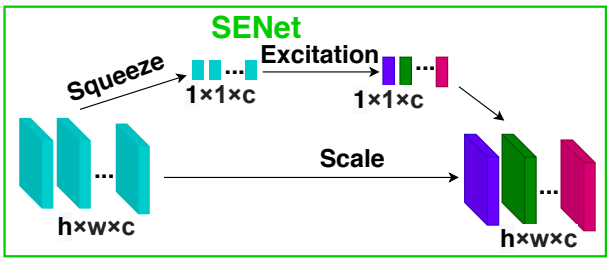

Figure 4.7: The SENet module, where h, w and c refer to height, width and number of channels, respectively.

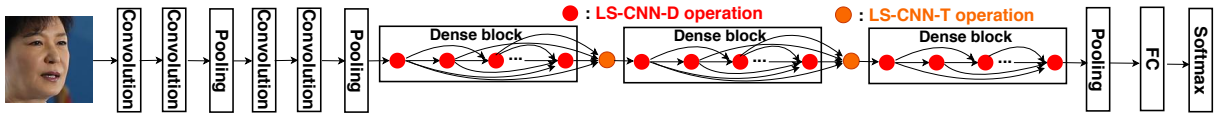

Figure 4.8: The overall framework of the Local and multi-Scale Convolutional Neural Networks (LS-CNN) model.

eyes, SENet module assigns higher weights on channels which have effective eye information, illustrated in Fig. 4.3.

It mainly consists of two operations, as shown in Fig. 4.7: squeeze and excitation. The squeeze operation is used to squeeze global channel information into a one channel descriptor, which is achieved by a global average pooling operation. Formally, the signal z of channel t is generated by averaging across spatial locations w as following: $z_t = \frac{1}{w}\sum_{i=1}^{w}\sum_{j=1}^{h} u_t(i,j)$, where $u_t(i,j)$ is an element of channel t at position (i,j). The excitation operation is followed, which aims at modelling the channel-wise dependencies flexibly. Two fully-connected (FC) layers are employed: $s = \sigma(w_2 g(w_1 z))$, where σ refers to the sigmoid function and g is a ReLU function, $w_1 \in R^{\frac{c}{r}}$ and $w_2 \in R^{c\times\frac{c}{r}}$ with the number of channels c and reduction ratio r. In order to avoid overfitting and aid generalization, w_1 is a dimension-reduction layer and w_2 is a dimension-increasing layer. Finally, the scale operation is used to rescale every channel by the transformation with learned activations, which is dynamically conditioned on inputs. $x_i = s_{ii}$, where s_i represents a scalar about the i_{th} channel and $u_i \in R^w$ means the i_{th} channel.

Table 4.1: The architecture of the Local and multi-Scale Convolutional Neural Networks (LS-CNN) model, where k refers to the growth rate of Harmonious multi-Scale Networks model. (N_1, N_2, N_3) refers to the repeated times in the first, second and third dense blocks, respectively. LS-CNN-D and LS-CNN-T refer to the architecture in Fig. 4.4 and Fig. 4.5, respectively.

Layer Type	Size, Stride, Pad	Output Size
Convolution	3, 1, 1	$128 \times 128 \times k$
Convolution	3, 1, 1	$128 \times 128 \times k$
Max Pooling	3, 2, 0	$63 \times 63 \times k$
Convolution	3, 1, 1	$63 \times 63 \times 2k$
Convolution	3, 1, 1	$63 \times 63 \times 2k$
Max Pooling	3, 2, 0	$31 \times 31 \times 2k$
N_1×LS-CNN-D	-	$31 \times 31 \times (2 + N_1)k$
LS-CNN-T	-	$15 \times 15 \times (2 + N_1) \times 0.5k$
N_2×LS-CNN-D	-	$15 \times 15 \times ((2 + N_1) \times 0.5 + N_2)k$
LS-CNN-T	-	$7 \times 7 \times ((2 + N_1) \times 0.5 + N_2) \times 0.5k$
N_3×LS-CNN-D	-	$7 \times 7 \times (((2 + N_1) \times 0.5 + N_2) \times 0.5 + N_3)k$
Average Pooling	7, 1, 0	$1 \times 1 \times (((2 + N_1) \times 0.5 + N_2) \times 0.5 + N_3)k$
Fully Connected	-	512
Softmax	-	# of subjects

4.2.6 Local and Multi-Scale Convolutional Neural Networks

We use both channel and spatial attentions in the Harmonious multi-Scale Networks (HSNet), as shown in Figs. 4.4 and 4.5. First, channel attention (i.e. SENet module) is applied to learn what features to emphasize. More useful channels are assigned with higher weights, and less important ones have smaller weights, as illustrated in Fig. 4.3. Second, spatial attention (i.e. LANet module) is proposed to decide where to focus. Informative regions are emphasized and less important ones are suppressed. As demonstrated in Figs. 4.9, 4.10 and 4.11, different weights are assigned to areas with different discriminative abilities. The LANet and SENet modules are combined to form the Dual Face Attentions (DFA) where the LANet module is used before the SENet module, refining local face details fist before weighing the channel-wise representation. As a result, a new model is created for face recognition, called Local and multi-Scale Convolutional Neural Networks (LS-CNN). It can learn rich multi-scale and local representations by integrating DFA with HSNet models.

The overall framework of LS-CNN model is shown in Fig. 4.8. Its details are presented in Table 4.1. We start testing the DFA-Inception modules at higher layers for better memory efficiency, keeping lower layers in the traditional convolutional fashion. In earlier layers, two 3×3 convolutional layers are used, followed by a max-pooling layer as suggested by VGG [47]. This can reduce the number of parameters without loss of representational ability. We repeat this procedure twice before the first dense block. After that, the proposed LS-CNN-D module, as shown in Fig. 4.4, is repeated N_1 times, which learns multi-scale representations, characterize local patches and model channels-wise importance from multiple layers. The LS-CNN-T module, shown in Fig. 4.5, is used to reduce the channel size and output more robust features. Repeating LS-CNN-D and LS-CNN-T several times until a global average pooling layer, which minimizes overfitting by reducing the number of parameters. There is one fully connected (FC) layer with 512 units before the softmax layer. It is clear that (N_1, N_2, N_3) in Table 4.1 controls the depth of network. Softmax loss L is used to extract discriminative features and minimize the classification error with the following formulation:

$$L = -\sum_{i=1}^{N} y_{\mathrm{i}} log p_{\mathrm{i}}, \tag{4.1}$$

where N is the number of subjects, y_{i} represents whether the face belongs to subject i, and p_{i} means the probability of belonging to subject i.

4.3 Experiments

Experimental results of our proposed Local and multi-Scale Convolutional Neural Networks (LS-CNN) are presented. We first introduce preprocessing methods and then perform ablation studies. Next, we compare the proposed Harmonious multi-Scale Network (HSNet) with other popular networks. Then, we compare the Dual Face Attentions (DFA) with recent attention mechanisms. Finally, the proposed model is compared with state-of-the-art methods for face recognition.

4.3.1 Preprocessing

The face detection method is the MTCNN [4]. In the training process, face images are resized to 144×144 and then randomly cropped to 128×128.

4.3.2 Implementation Details

The proposed Local and multi-Scale Convolutional Neural Networks (LS-CNN) model is implemented in PyTorch [76]. In training on WebFace dataset, the learning rate begins at 0.1 and is divided by 10 every 10 epochs. The training process stops at the 25_{th} epoch. In training on VGGFace2 dataset, the learning rate begins at 0.1 and is divided by 10 every 4 epochs. The training process stops at the 10_{th} epoch. The weight-decay and momentum are $1e-4$ and 0.9, respectively. Two Harmonious multi-Scale Network (HSNet) backbone networks are mainly used: HSNet-61 and HSNet-97 models. Their growth rates k and depths (N_1, N_2, N_3) in Table 4.1 are 48, (3, 3, 5) and 80, (6, 6, 8), respectively.

4.3.3 Ablation Study

In this subsection, we first show the importance of four contributing modules: Inception, DenseNet, SENet, and LANet. Then we show the effect of different model widths and depths.

Finally, we compare different ways to combine SENet and LANet modules.

Table 4.2: Trained on VGGFace2 dataset, performance comparison (%) of different module combinations on LFW, CALFW, IJB-A quality and CFP datasets. The growth rate k is 48. (N_1, N_2, N_3) in Table 4.1 is (3,3,5).

Model	LFW	CALFW	IJB-A		CFP	
			FAR=0.01	FAR=0.001	FF	FP
Inception	94.7	70.5	46.6	16.7	90.5	86.3
DenseNet	98.1	86.1	70.2	46.1	96.9	92.0
HSNet	98.8	90.3	81.4	65.3	98.6	95.0
SENet-HSNet	99.1	89.9	84.3	68.8	98.8	95.9
LANet-HSNet	99.0	90.0	82.4	66.3	98.6	95.2
DFA-HSNet (LS-CNN)	99.3	90.5	85.2	70.3	99.0	96.0

Importance of four Contributing Modules

In order to obtain a deep insight into our proposed Local and multi-Scale Convolutional Neural Networks (LS-CNN) model, four contributing modules are analyzed: Inception, DenseNet, SENet, and LANet. We compare different module combinations and show results in Table 4.2. To compare fairly, these models have the same number of layers and channels.

The performance of the Inception model lags behind the DenseNet significantly. For example, the DenseNet obviously boosts the performance on the relatively easy LFW task (about 3.4%). This is because, although the Inception model concatenates channels generated by multi-scale kernels, it has few layers and channels to have a strong representational capacity. In contrast, dense connections in the DenseNet encourage multi-scale feature propagation and reuses, thus resulting in a better performance.

When the Inception module is incorporated into DenseNet module, namely Harmonious multi-Scale Network (HSNet), it outperforms the individual Inception and DenseNet by a large margin. This proves the necessity of incorporating the Inception with DenseNet. On one hand, the Inception module learns features with parallel kernels with different sizes

in a single layer. On the other hand, dense connections enable the DenseNet module to combine features from multiple layers. Therefore, two complementary multi-scale learning modules explains why the HSNet model has a better representation ability compared with the individual Inception or DenseNet model.

The SENet further improves performance by weighing the importance of different channels. It is plugged into dense blocks (Fig. 4.4) and transitional layers (Fig. 4.5) before the multi-branch operation in Inception module. Important channels are emphasized and less informative ones are suppressed, as shown in Fig. 4.3. This is the reason why the SENet-HSNet performs slightly better than the HSNet, as indicated in Table 4.2.

A global feature vector tends to pay attention to overall appearances rather than local discriminative regions, which may ignore discriminative local facial details. The LANet module is introduced to characterize local patches automatically. To show the effectiveness of the LANet module, it is first integrated into the HSNet model, i.e. LANet-HSNet. In most cases, the LANet-HSNet has a better performance than the HSNet model.

Since channel attention (i.e. SENet) applies globally and spatial attention (i.e. LANet) module works locally on channels, it is intuitive to combine them together, taking advantage of each other. We apply LANet module first, followed by SENet module as shown in Figs. 4.4 and 4.5. As indicated in Table 4.2, the LS-CNN outperforms the LANet-HSNet and SENet-HSNet, verifying the complementarity of SENet and LANet modules.

Sensitivity to Model Capacity

There are generally two ways to increase model capacity: width and depth.

As for the width, we change the growth rate k in DenseNets module in Table 4.1 to have different widths. The growth rate k means the number of channels that each layer outputs in dense blocks. Since each previous layer is concatenated together in a dense block, we can see the growth rate k controls the width of the network. The effect of k is investigated in Table 4.3. The table indicates that the performance tends to be better under four test datasets as the k increases.

We have models with different depths by changing (N_1, N_2, N_3) in Table 4.1. (N_1, N_2, N_3) means how many times the LS-CNN-D in Fig. 4.4 repeats in Table 4.1. Experimental results

Table 4.3: Trained on WebFace dataset, performance comparison (%) of the HSNet model with different growth rates k on LFW, CALFW, IJB-A quality, and CFP datasets. (N_1, N_2, N_3) in Table 4.1 is (3,3,5).

Model	LFW	CALFW	IJB-A		CFP	
			FAR=0.01	FAR=0.001	FF	FP
32	97.5	82.5	63.6	40.0	97.4	92.7
48	98.1	84.8	67.0	44.8	98.0	93.0
64	98.0	84.1	67.9	46.0	98.0	93.6
80	98.1	85.3	69.2	47.9	98.3	93.6
96	98.2	86.0	70.3	48.0	98.3	94.0
112	98.4	86.4	71.0	50.5	98.6	94.3

Table 4.4: Trained on WebFace dataset, performance comparison (%) of the HSNet model with different depths on LFW, CALFW, IJB-A quality and CFP datasets. The growth rate k in HSNet model is 48. (N_1, N_2, N_3) refers to the configuration in Table 4.1.

Depth	(N_1, N_2, N_3)	LFW	CALFW	IJB-A		CFP	
				FAR=0.01	FAR=0.001	FF	FP
33	(1,1,2)	97.0	81.8	59.3	35.0	96.7	90.9
61	(3,3,5)	98.1	84.8	67.0	44.8	98.0	93.0
97	(6,6,8)	98.3	86.0	70.3	49.1	98.3	93.5
177	(12,12,16)	98.4	86.8	73.0	52.3	98.6	94.7

are shown in Table 4.4. The table shows that performance benefits from deeper models.

These experiments indicate that our model can utilize the increased model capacity of wider and deeper models. On one hand, deeper CNNs can extract richer and more descriptive features for complex face distributions. Meanwhile, wider CNNs are able to capture more local features [111], characterizing fine-grained face details. On the other hand, they do not suffer from overfitting or optimization problems. The explanation for this observation is that implicit supervision signals with shorter connections to loss functions can benefit individual layers, guiding early and intermediate layers to learn more discriminative features.

Table 4.5: Trained on VGGFace2 dataset, performance comparison (%) of different ways to combine SENet and LANet modules on LFW, CALFW, IJB-A quality and CFP datasets. The HSNet-61 model is used as the backbone network.

Model	LFW	CALFW	IJB-A		CFP	
			FAR=0.01	FAR=0.001	FF	FP
SENet (Before Inception)	99.1	89.9	84.3	68.8	98.8	95.9
LANet (Before Inception)	99.0	90.0	82.4	66.3	98.6	95.2
SENet+LANet (Before Inception)	99.3	90.7	85.1	70.1	98.8	96.0
SENet&LANet (Before Inception)	99.3	90.4	84.4	69.1	99.0	96.0
LANet+SENet (After Inception)	99.1	90.4	85.1	69.9	98.9	95.4
LANet+SENet (Before Inception)	99.3	90.5	85.2	70.3	99.0	96.0

Different Ways to Form Dual Face Attentions (DFA)

We study different ways to combine LANet and SENet modules into DFA, which could differ in two aspects: 1) order; 2) location. For the first aspect, there are three options: first LANet then SENet ('LANet+SENet'), first SENet then LANet ('SENet+LANet') and parallel use of SENet and LANet ('SENet&LANet'). As for the second aspect, we compare two locations of applying LANet and SENet modules: 'Before Inception' (used in Figs. 4.4 and 4.5) and 'After Inception'. 'After Inception' refers to applying SENet and LANet modules after the Inception module in Figs. 4.4 and 4.5.

Table 4.5 summarizes the experimental results. The SENet learns what features to emphasize, and LANet focuses on which facial parts to focus, which complement each other. This explains why all ways to combine SENet and LANet modules achieve better performances than using LANet or SENet independently. Besides, we observe that the 'LANet+SENet' performs slightly better than 'SENet+LANet' and 'SENet&LANet'. We attribute this observation to the reason that more benefits are learned by refining local facial representations before learning global channel-wise inter-dependencies. We find the best location to apply LANet and SENet modules is 'Before Inception', where rich hierarchical

Table 4.6: Parameter settings of different models. The *iter* means the global step during training. The momentum is 0.9. The m refers to million. The iterations per epoch (ipe) = # of images / (batch size). The $(k, (N_1, N_2, N_3))$ in the HSNet-61 and HSNet-97 models are (48, (3, 3, 5)) and (80, (6, 6, 8)), respectively, where (N_1, N_2, N_3) is from Table 4.1, and k means the growth rate.

Model	Batch size	Input size	Learning rate	Epochs	Weight decay
AlexNet-v2	256	224	$0.01*0.1^{(floor(iter/(21*ipe)))}$	96	5e-4
DenseNet-121	40	224	$0.1*0.1^{(floor(iter/(10*ipe)))}$	30	1e-4
DPN-92	48	224	$0.1*0.1^{(floor(iter/(30*ipe)))}$	90	1e-4
Inception-v3	25	299	$0.045*0.94^{(floor(iter/(2*ipe)))}$	100	4e-4
ResNet-50	50	224	$0.1*0.1^{(floor(iter/(28*ipe)))}$	128	1e-4
ResNet-101	50	224	$0.1*0.1^{(floor(iter/(28*ipe)))}$	128	1e-4
VGG-16	128	224	$0.1*0.1^{(floor(iter/(17*ipe)))}$	74	5e-4
HSNet-61	128	128	$0.1*0.1^{(floor(iter/(10*ipe)))}$	25	1e-4
HSNet-97	128	128	$0.1*0.1^{(floor(iter/(10*ipe)))}$	25	1e-4

channel information in DenseNets needs to be recalibrated. In contrast, less channel information needs to be weighed in 'After Inception'. More specifically, channels with size $h \times w \times (m \times k)$ in 'Before Inception' are weighed, compared with $h \times w \times 3k$ channels in 'After Inception' in LS-CNN-D, as shown in Fig. 4.4; channels with size $h \times w \times c$ in 'Before Inception' are readjusted, compared with $h/2 \times w/2 \times 3c/2$ channels in 'After Inception' in LS-CNN-T, as shown in Fig. 4.5.

Comparison with Different Backbone Networks

In this work, we study the integration of the DenseNet [51] and the recent Inception module [49] into the Harmonious multi-Scale Network (HSNet). Several popular CNNs are compared, including AlexNet-v2 [46], VGG-16 [47], Inception-v3 [49], ResNet-50, ResNet-101 [50], DenseNet-121 [51] and DPN-92 [112]. Detailed hyper-parameter settings and experimental results are shown in Tables 4.6 and 4.7. The growth rates k and depths (N_1, N_2, N_3) in Table 4.1 of the HSNet-61 and HSNet-97 models are 48, (3, 3, 5) and 80, (6, 6, 8), respec-

Table 4.7: Performance comparison (%) of different models. Trained on WebFace dataset and tested on LFW, CALFW, IJB-A quality and CFP datasets. The LS-CNN model uses HSNet-97 model as the backbone.

Model	LFW	CALFW	IJB-A		CFP		Params	Speed (ms)
			FAR= 0.01	FAR= 0.001	FF	FP		
AlexNet-v2	97.6	69.6	29.1	11.9	90.4	80.8	100.4m	46
DenseNet-121	97.8	80.4	58.8	33.1	97.6	93.5	17.8m	196
DPN-92	97.5	75.0	60.2	35.3	97.8	93.4	63.4m	467
Inception-v3	97.9	81.8	51.3	28.2	98.1	94.4	43.5m	309
ResNet-50	98.2	76.8	62.3	38.9	98.1	94.0	45.2m	178
ResNet-101	98.3	78.4	65.8	42.0	98.6	94.4	64.2m	276
VGG-16	97.6	79.2	47.3	23.7	96.5	89.8	177.6m	261
HSNet-61	98.3	86.0	70.3	49.1	98.3	93.5	18.5m	234
HSNet-97	98.5	87.3	72.9	52.1	98.6	94.7	41.2m	441
LS-CNN	-	-	-	-	-	-	-	643

tively.

The HSNet-61 model has the second fewest parameters. However, it achieves the highest accuracy on LFW, CALFW and IJB-A quality datasets except CFP dataset. More specifically, both HSNet-61 and ResNet-101 models achieve the second best accuracy (98.3%) on LFW dataset, lagging behind the HSNet-97 model, while the ResNet-101 model has almost 3.5× parameters. This indicates that HSNet model has better parameter efficiency. There are several factors which can explain this observation: the bottleneck layer used in the DenseNet module; factorizing larger convolutions (5×5 convolutions) into smaller convolutions (two 3×3 convolutions) without loss of expressiveness adopted in the recent Inception module; dimension reduction (i.e. the bottleneck layer) used in the Inception module. Besides, the HSNet model obviously boosts the accuracy on CALFW dataset (5.5%) and IJB-A quality dataset (4.5%, 7.1% when FAR=0.01, 0.001) than ResNet-101 model. IJB-A quality dataset contains faces influenced by many challenging factors, like poses, expressions, resolution and occlusions. In such cases, discriminative facial information may exist at various scales, mak-

ing it necessary to learn multi-scale features. Further, although HSNet-61 only has 61 layers, dense connections in HSNet model incorporate multi-scale representations with local details which may appear at various scales.

However, HSNet-61 model has a slightly worse performance (98.3%) than ResNet-101 model (98.6%) under the CFP-FF test scenario and performs worse than Inception-v3, ResNet-50 and ResNet-101 models under the CFP-FP test scenario. One possible explanation is that the growth rate k and depth (N_1, N_2, N_3) is too small, which is insufficient to obtain comprehensive fine-grained features to describe profile faces. Therefore, the growth rate k and depth (N_1, N_2, N_3) are increased from 48 to 80 and from (3, 3, 5) to (6, 6, 8), respectively, namely HSNet-91. It has the third fewest parameters, followed by the DenseNet-121 and HSNet-61 models. HSNet-91 model improves the accuracy on all datasets, which shows its powerful generalization ability.

We explain why HSNet models achieve better performances than other models. First, AlexNet and VGG models only have a small range of receptive fields, which is insufficient to capture local face patches with various sizes. Second, ResNet and DenseNet models concatenate features from different layers similarly by the short connection and dense connections, respectively. DPN model combines ResNet and DenseNets, sharing a similar way to extract multi-scale features. Third, Inception model learns multi-scale features in a single layer using parallel kernels of different sizes. However, our HSNet model extracts multi-scale features from two harmonious perspectives: parallel multi-scale kernels in a single layer (e.g. Inception-v3); multi-scale feature maps from different layers (e.g. ResNet, DenseNet, DPN).

We also measure the speed of the HSNet model. The system configuration is Ubuntu 16.04.3 LTS. Other hardware information includes: Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz, 32GB RAM, 512GB SSD, and Titan X (Pascal). The PyTorch version is 0.3.1. We average the running time of 2,000 iterations with a batch-size 32. In the HSNet model, due to dense connections in DenseNets and multiple branches in Inception, HSNet-61 ranks the fourth and HSNet-91 has the penultimate running time among different backbone networks. We also show that the LS-CNN model which uses HSNet-97 as the backbone has a higher running time.

To sum up, the HSNet model has a good representation capacity to perform well under

Table 4.8: Trained on VGGFace2 dataset, performance comparison (%) of different attention modules on LFW, CALFW, IJB-A quality, and CFP datasets. The HSNet-61 model is used as the backbone network.

Model	LFW	CALFW	IJB-A		CFP	
			FAR=0.01	FAR=0.001	FF	FP
SENet [107]	99.1	89.9	84.3	68.8	98.8	95.9
CBAM [108]	99.0	88.5	81.4	62.8	98.5	94.9
DFA (Max&avg pool)	98.8	86.4	79.9	59.6	98.0	94.0
BAM [109]	99.1	90.3	83.9	67.9	98.9	95.8
DFA (Dilated conv)	99.3	90.2	85.0	70.3	98.8	95.9
DFA	99.3	90.5	85.2	70.3	99.0	96.0

complex data distributions and utilizes parameters effectively to be less prone to overfitting. However, because of complex operations, its running speed is slightly slower. Thus, the HSNet and LS-CNN models are suitable for some tasks where accuracy is more important than speed, like the highly confidential access control and anti-terrorism video surveillance.

Comparison with Different Attention Modules

We compare our proposed Dual Face Attentions (DFA) with several recent attention modules, i.e. SENet [107], CBAM [108] and BAM [109], which are proposed for general classification tasks. We integrate these attention modules into the same HSNet model and report experimental results in Table 4.8.

The SENet only uses channel attention, while ignores spatial attention. Therefore, local facial details may be failed to be captured. In contrast, our DFA module aims at learning channel and spatial attentions simultaneously, demonstrating a better performance.

The CBAM has both channel and spatial attentions to learn what and where to emphasize or suppress in channels, in which max- and average-pooling are used. As demonstrated in Table 4.8, our proposed DFA module outperforms the CBAM. To explain this result, we replace the average pooling with the max & average pooling used in the CBAM and remain the same for other parts in the DFA, i.e. DFA (Max&avg pool). However, its performance is worse than the DFA, demonstrating that the max-pooling is not appropriate. The max-

pooling encodes the most salient parts, which may be influenced by the background noise, such as the background person in Fig. 4.3 Row 1 Column 1.

The BAM also uses channel and spatial attentions. The dilated convolution [113] is employed in spatial attention to enlarge the receptive field. We remain unchanged in the SENet and use the dilated convolution in the LANet, namely the DFA (dilated conv). One explanation for its slightly worse performance is that as the network goes deeper, the receptive field size is enlarged exponentially by the dilated convolution, corresponding to more coarse face information. This inevitably leads to loss of local details and be more sensitive to pose and age variations.

4.3.4 Experiments on Cross-Pose Face Matching

Table 4.9: Trained on VGGFace2 dataset, performance comparison (%) of the LS-CNN model with state-of-the-art methods on CFP dataset. The HSNet-97 model is used as the backbone network.

Methods	CFP (FP)
Deep features [59]	84.91
TDE [114]	89.17
FV-DCNN [115]	91.97
PIM [116]	93.10
DR-GAN [117]	93.41
DR-GAN$_{\text{AM}}$ [118]	93.89
DA-GAN [119]	95.96
p-CNN [120]	94.39
NoiseFace [121]	96.40
ArcFace [122]	97.15
Human	94.57
LS-CNN	97.17

We compare the performance of the Local and multi-Scale Convolutional Neural Networks (LS-CNN) model with the state-of-the-art on CFP dataset in Table 4.9. Fig. 4.9 shows class

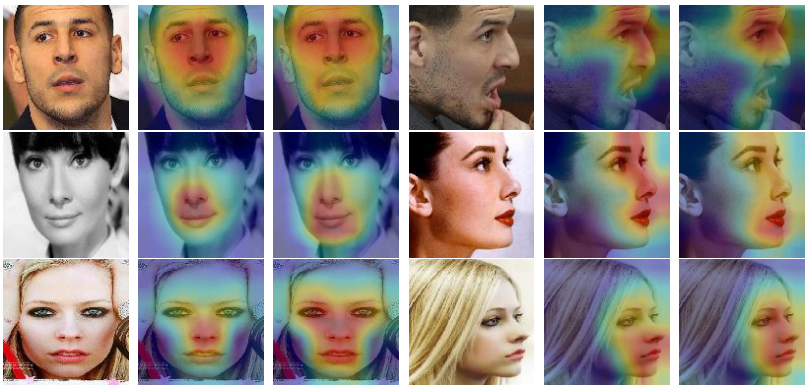

Figure 4.9: Class activation maps (CAMs) [3] with/without LANet module on three positive pairs from CFP dataset. Column 1 & 4: frontal and profile faces of the same subject with various challenging factors (e.g. pose, makeup and expression). Column 2 & 5: CAMs generated by SENet-HSNet model. Column 3 & 6: CAMs generated by Local and multi-Scale Convolutional Neural Networks (LS-CNN) model. Note that the fully connected layer before the final layer is removed in both models.

activation maps (CAMs) [3] with/without LANet module on three positive pairs from CFP dataset.

Global learning methods (Deep features [59], TDE [114] and FV-DCNN [115]) accept whole channels as the input without filtering out the background information of profile faces and emphasizing important regions. More specifically, deep features [59] extract CNN features directly, which has 84.91% accuracy. TDE [114] learns low-dimensional embeddings using triplet probability constraints, which improves to 89.17%. FV-DCNN [115] combines Fisher vector and CNN for unconstrained face verification, achieving 91.97%.

There are several methods that extract pose-invariant representations and perform face frontalization simultaneously. PIM [116] trains a frontalization network that perceives global structures and local details and a discriminative network that learns discriminative representations jointly, achieving 93.10% accuracy. DR-GAN [117] proposes an encoder-decoder structure for the generator and disentangles face representation from pose variations, which improves the performance to 93.41%. DR-GAN$_{\text{AM}}$ [118] extends DR-GAN [117] to improve the model generalization during training, reaching 93.89%. DA-GAN [119] combines a prior data distribution and domain knowledge to synthesize photorealistic and identity-preserving profile faces. The accuracy is 95.96%. p-CNN [120] introduces the stochastic routing scheme

to different paths for faces with various poses, which obtains 94.39%. NoiseFace [121] weighs training samples using angular margin based loss to train CNNs on large-scale noisy data. Its accuracy is 96.40%. ArcFace [122] model maximizes the decision boundary in angular space based on normalized weights and features, achieving 97.15% when trained on VGGFace2 dataset.

Is is noticed that due to self-occlusion in profile faces, discriminative local face parts have smaller sizes than those in frontal faces. The powerful HSNet backbone network can capture rich multi-scale information. However, local face regions in lower channels may fail to propagate as the network goes deeper. To alleviate this problem, SENet-HSNet model emphasizes important channels in lower layers by using SENets, as illustrated in Fig. 4.9, Columns 2, 5. Further, the LANet module is introduced to alleviate the effect of background inconsistency, especially for profile faces. As shown in Fig. 4.9, compared to the SENet-HSNet model, class activation maps (CAMs) generated by the LS-CNN model tend to locate more discriminative parts in frontal faces (Column 3 to 2) and suppress less informative regions in profile faces (Column 6 to 5). Finally, compared to the state-of-the-art which either requires complex data augmentation (DR-GAN, DR-GAN$_{\mathrm{AM}}$, PIM, DA-GAN) or multi-task training (p-CNN) or a noise-tolerate paradigm (NoiseFace) or a more advanced loss function (ArcFace), our approach is simple and effective.

4.3.5 Experiments on Cross-Age Face Matching

It is well known that age-invariant face recognition (AIFR) is very meaningful for various applications, such as looking for missing children after years. However, large age variations make the AIFR problem challenging. We compare the performance of the proposed Local and multi-Scale Convolutional Neural Networks (LS-CNN) model with the state-of-the-art on CACD-VS and CALFW datasets in Table 4.10.

There are several approaches that propose advanced loss functions. VGGFace [27] learns a face embedding using a triplet loss. Center loss [29] learns a center for deep features of each subject to increase the intra-subject compactness. Marginal loss [123] minimizes the intra-class distances and maximizes the inter-class variances based on marginal samples. Noisy

Table 4.10: Trained on VGGFace2 dataset, performance comparison (%) of the LS-CNN model with state-of-the-art methods on CALFW and CACD-VS datasets. The HSNet-97 model is used as the backbone network.

Methods	CACD-VS	CALFW
VGGFace [27]	96.00	86.50
Center loss [29]	97.48	-
Marginal loss [123]	98.95	-
Noisy Softmax [124]	-	82.52
CCL [125]	99.23	91.15
DeepVisage [105]	99.13	-
LF-CNN [126]	98.50	-
AFJT-CNN [127]	99.00	85.20
OE-CNN [128]	99.20	-
DAL [129]	99.40	-
Human, Average [130]	85.70	-
Human, Voting [130]	94.20	-
LS-CNN	99.50	92.00

Softmax [124] injects annealed noise in softmax to mitigate the early saturation behavior of the softmax. CCL [125] encourages face samples to distribute dispersedly across the coordinate space and pushes classification vectors to lie on a hypersphere. DeepVisage [105] introduces a feature normalization before the softmax loss, ensuring that features have an equal distribution.

There are several approaches proposed to solve the AIFR problem. More specifically, LF-CNN [126] couples learning of the CNN and latent identity analysis parameters to extract age-invariant features. AFJT-CNN [127] trains the identity discrimination model and the age discrimination model jointly by sharing the same feature layers to extract cross-age identity features. OE-CNN [128] decomposes face features into age-related and identity-related components using A-Softmax loss [36]. DAL [129] introduces a linear feature factorization based algorithm to regularize decomposed feature learning.

Figure 4.10: Class activation maps (CAMs) [3] with/without LANet module on three positive pairs from CALFW dataset. Column 1 & 4: Input faces of the same subject with various challenging factors (e.g. pose, occlusion, aging and expression). Column 2 & 5: CAMs generated by SENet-HSNet model. Column 3 & 6: CAMs generated by Local and multi-Scale Convolutional Neural Networks (LS-CNN) model. Note that the fully connected layer before the final layer is removed in both models.

We can see that our LS-CNN model achieves better performances than the state-of-the-art on CACD-VS and CALFW datasets. Besides, the LS-CNN model surpasses human performance on CACD-VS dataset significantly. Unlike some models (LF-CNN, AFJT-CNN, OE-CNN, and DAL), LS-CNN model is not specifically designed for the AIFR problem, demonstrating its good generalization ability. In addition, the LS-CNN model only uses the softmax loss without feature normalization. Therefore, performance improvement is expected by adopting more advanced loss functions or feature normalization.

It is observed that humans tend to pay more attention to more salient parts instead of the whole scene when localizing objects [131]. Intuitively, this is applicable to the AIFR problem, because some local regions remain the same, despite aging affect the global face appearances. As demonstrated in Fig. 4.10, although the age gap is large among these face pairs, some facial regions still look similar, such as pointy noses (Row 1, Columns 1, 4), intraocular regions (Row 2, Columns 1, 4) and mouths (Row 3, Columns 1, 4). We show class activation maps (CAMs) [3] generated by SENet-HSNet and LS-CNN models in Fig. 4.10. We can observe that LS-CNN model tends to emphasize more discriminative face regions than SENet-HSNet model. Like the SENet-HSNet model, these methods under comparisons are likely to model less informative facial patches, which inevitably leads to a sub-optimal

Table 4.11: Trained on VGGFace2 dataset, performance comparison (%) of the LS-CNN model with state-of-the-art methods on IJB-A quality and FaceScrub quality datasets. The HSNet-97 model is used as the backbone network.

Methods	IJB-A		FaceScrub	
	FAR=0.01	FAR=0.001	FAR=0.01	FAR=0.001
VGGFace [27]	60.5	36.7	59.5	38.9
LightCNN [35]	56.6	40.2	50.3	33.0
Center loss [29]	52.1	31.3	49.3	34.1
SphereFace [36]	54.8	39.6	45.8	34.3
LS-CNN	87.5	75.5	80.5	70.4

performance.

4.3.6 Experiments on Cross-Quality Face Matching

In unconstrained scenarios like video surveillance and access control, face matching may be conducted between low-quality faces captured in real-world environments and high-quality mugshots. We show three positive pairs in IJB-A quality dataset in Fig. 4.11 with various challenging factors (e.g. pose, blur, resolution and expression).

We compare the performance of the Local and multi-Scale Convolutional Neural Networks (LS-CNN) model with several methods on IJB-A quality and FaceScrub quality datasets in Table 4.11. Results of VGGFace [27], LightCNN [35] and Center loss [29] models are from [65]. We use the publicly available SphereFace [36] model to conduct the same experiments.

As shown in Table 4.11, we obtain significantly better accuracies on both the IJB-A and FaceScrub datasets at different false accept rate (FAR) measures. The performance is greatly improved, improving 21% at least and 35.3% accuracy at most. This proves the benefits of learning multi-scale representations by two complementary ways to characterize a complex data distribution under different image qualities. Besides, the SENet module enhances useful channels and suppresses noisy channels. Furthermore, the LANet module is especially useful to characterize faces of different qualities. We visualize the class activation maps (CAMs) [3] generated by SENet-HSNet and LS-CNN models in Fig. 4.11. Note that

Figure 4.11: Class activation maps (CAMs) [3] with/without LANet module on three positive pairs from IJB-A quality dataset. Column 1 & 4: high- and low-quality faces of the same subject with various challenging factors (e.g. pose, illumination, blur, resolution and expression). Column 2 & 5: CAMs generated by SENet-HSNet model. Column 3 & 6: CAMs generated by Local and multi-Scale Convolutional Neural Networks (LS-CNN) model. Note that the fully connected layer before the final layer is removed in both models.

CAMs of LS-CNN model emphasize representational patches (Column 3 to 2) and suppress less informative parts (Column 6 to 5) than SENet-HSNet model. The compared methods in the Table are similar to SENet-HSNet model without characterizing discriminative facial regions and filtering out less informative parts.

4.3.7 Experiments on the LFW Dataset

To show the generalization ability, we compare with other approaches on the LFW dataset. Table 4.12 demonstrates the experimental results.

Compared with LF-CNN [126], OE-CNN [128] and DAL [129] models which are proposed to learn age-invariant deep face representations, the proposed LS-CNN model achieves a better performance.

The p-CNN [120] model is proposed for face images with different poses. However, almost all faces in the LFW dataset are frontal or close to frontal views. This explains why its performance is worse than our proposed LS-CNN model.

The Marginal loss [123], VGGFace [27], Center loss [29], LightCNN [35], Feature transfer [132], SphereFace [36], Noisy Softmax [124], CCL [125], DeepVisage [105], CosFace [30] and ArcFace [122] models are generic models which aim at the generic unconstrained face

Table 4.12: Trained on VGGFace2 dataset, performance comparison (%) of the LS-CNN model with several state-of-the-art methods on LFW dataset. The HSNet-97 model is used as the backbone network.

Methods	LFW
LF-CNN [126]	99.10
OE-CNN [128]	99.47
DAL [129]	99.47
p-CNN [120]	98.27
Marginal loss [123]	98.95
VGGFace [27]	99.13
Center loss [29]	99.28
LightCNN [35]	99.33
Feature transfer [132]	99.37
SphereFace [36]	99.42
Noisy Softmax [124]	99.48
CCL [125]	99.58
DeepVisage [105]	99.62
CosFace [30]	99.73
ArcFace [122]	99.78
LS-CNN	99.52

recognition problem. As we can observe, our proposed LS-CNN model has a better performance than these models except the CCL, DeepVisage, CosFace and ArcFace models, demonstrating its excellent generalization ability. The performance of ArcFace model (99.78%) is based on the VGGFace2 dataset. Since our LS-CNN model is trained only using the softmax loss, we expect that the performance can be improved by using some advanced loss functions in CCL, CosFace and ArcFace models.

4.4 Summary

We have developed a new network structure for face recognition, based on the integration of rich multi-scale feature learning, correlating and weighing different channels, and automatic characterizing local face regions. The proposed model, called Local and multi-Scale Convolutional Neural Networks, or simply LS-CNN, has the capability of characterizing complex face images to reduce intra-class variations. It generalizes well across multiple datasets. Experimental results on several databases have shown that the LS-CNN model can achieve a better performance than the state-of-the-art methods for cross-quality, cross-age and cross-pose face matching and obtain a competitive performance on LFW dataset. In future, several more advanced loss functions will be validated to further improve the recognition performance.

Chapter 5

Hierarchical Pyramid Diverse Attention Networks for Face Recognition

5.1 Introduction

CNN representations achieve the state-of-the-art in face recognition (FR). However, most existing models learn global representations where whole faces are regarded as CNN inputs [27, 29, 31, 36]. Few works take hierarchical multi-scale local representations into account.

It is observed that global face geometry and appearances may change dramatically under pose, age, or large quality variations. In contrast, some facial parts remain similar in these cases, which would play important roles in FR. For instance, as shown in Fig. 5.1, it is difficult to consider the global face representation to match the frontal face (Row 1) with the profile (Row 2) which is influenced by blur, pose and background distraction. However, we notice that similar eyes in these two faces can contribute to verification. Similar observations about pointy noses (Rows 3, 4) and big lips (Rows 5, 6) can be made. Thus, representing similar facial parts become especially important. Previous works mainly depend on face landmarks to incorporate local information [2, 38, 39, 41, 42, 55, 133]. However, landmark detection may be inaccurate or even fail due to occlusions, large head poses, extreme illuminations, or dramatic expression changes. As shown in Fig. 6.4, Row 1, MTCNN [4] fails

to detect landmarks for faces which are influenced by occlusions (Columns 1, 2), expressions (Columns 3, 4), illumination (Columns 5, 6), and poses (Columns 7, 8).

Figure 5.1: **Illustration on the effects of local CNNs, the diverse learning and global CNN**. Local CNNs learn diverse local representations using multiple local branches at various scales from different hierarchical layers. For good illustration, only one scale and the last convolutional layer are used. The number of local branches is 3. The diverse learning guides multiple local branches to locate diverse local patches. Global CNN extracts holistic representations. **Column 1**: faces with varying challenging factors (e.g. resolution, pose, occlusion, aging and expression). **Column 2**: a model which has a single local branch. **Columns 3, 4 and 5**: 1^{st}, 2^{nd} and 3^{rd} local branch of local CNNs, which are guided by the diverse learning. **Column 6**: local CNNs. **Column 7**: global CNN. **Column 8**: fused global and local CNNs.

Without relying on face landmarks, discriminative local patches are located automatically in [2, 5]. As shown in Fig. 5.1, compared to the model without an attention module (Column 7), important facial parts are enhanced and some useless ones are suppressed when the attention is applied (Column 2). However, it is observed that only specific facial regions are located, while neglecting some potential discriminative regions. For instance, the

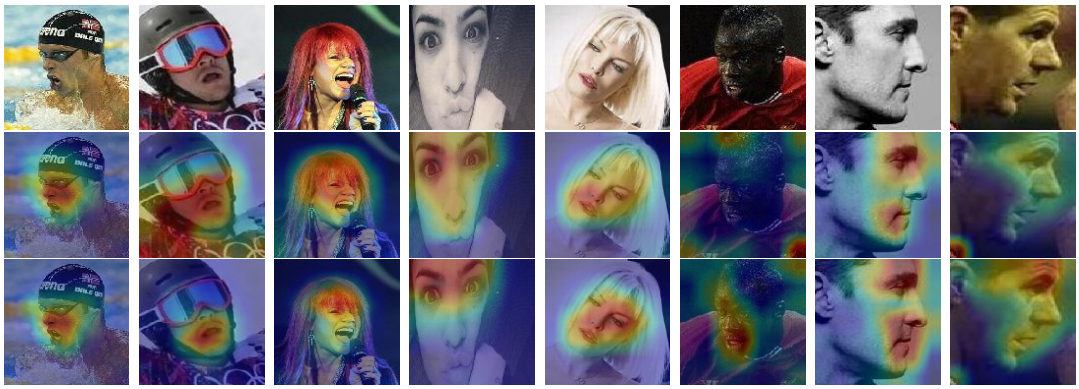

Figure 5.2: **Qualitative comparison between prior local methods and the proposed HPDA model**. Landmark-based methods suffer from landmark detection failure (**Row 1**) and attention-based methods locate uninformative or even noisy regions (**Row 2**). The proposed HPDA can emphasize discriminative information and inhibit less important (**Row 3**).

attention map has strong responses around eyes (Column 2), but ignores some discriminative regions, such as similar big lips (Column 2, Rows 5, 6). From the above analysis, we expect that emphasized facial parts should be well distributed over face images to extract more useful features. To achieve this goal, a diverse learning is developed to guide multiple attention modules to accurately locate diverse discriminative facial parts as well as reduce the background distraction.

Learning multi-scale representations is beneficial to various tasks [48, 96, 134, 135]. As for FR, local patches may have various sizes or shapes under pose or expression changes, making it necessary to learn multi-scale features. Take faces in Fig. 6.4 as examples, eyes with different expressions in Columns 3, 4 and 5 appear at varying sizes and shapes; due to pose variations, mouths have different sizes in Columns 6, 7 and 8. [55] fuses multi-scale features from the last two layers. [5] extracts rich multi-scale features from two harmonious perspectives: different convolutional sizes in a single layer and hierarchical concatenation of feature maps from varying layers. However, both approaches ignore the fact that features in a layer may cover a large range of scales. This is especially important for layers which concatenate feature maps from prior layers and generate feature maps by multi-scale convolution kernels. To address this challenge, we propose a pyramid attention which scales feature maps within a layer under different scales, exploring multi-grained information.

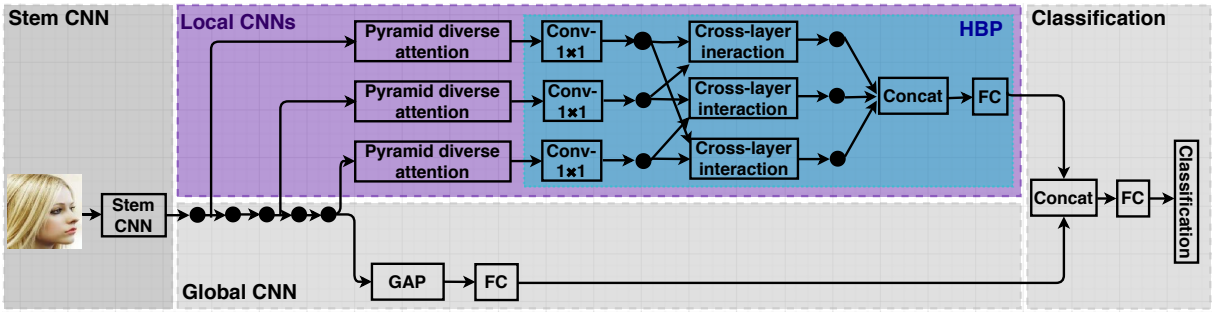

Figure 5.3: **Framework of the proposed hierarchical pyramid diverse attention (HPDA) model**. GAP and FC layers mean global average pooling and fully connected layers. Local CNNs learn rich local representations which mainly consist of a pyramid diverse attention (PDA) and a hierarchical bilinear pooling (HBP). The PDA aims at learning multi-scale diverse local features. The HBP fuses complementary local information from hierarchical layers.

Most previous works only use the last convolutional layer, lacking of low-level information. This is because units in high layers have large receptive fields, and hence respond around large-scale facial parts and represent high-level semantic information, but inevitably lack of locally detailed information or small-scale face parts in low layers. [5] can alleviate the above problem by combining hierarchical information from different layers within a block. [133] incorporates low-level features with high-level features to capture discriminative representations. Both methods combine hierarchical information simply by concatenation, which leads to sub-optimal cross-layer information fusion.

In this work, we propose a hierarchical pyramid diverse attention (HPDA) network which can describe diverse local patches at various scales adaptively and automatically from varying hierarchical layers. Fig. 5.3 illustrates the framework. First, we propose a pyramid diverse attention, as shown in Fig. 5.4. Specifically, feature maps are pooled to various scales, allowing for exploiting features at different scales. Meanwhile, since attention modules tend to have redundant responses around some similar face regions, a diverse learning is proposed to guide multiple local branches in each scale to focus on diverse facial regions automatically, instead of relying on face landmarks. As shown in Fig. 5.1, the diverse learning leads to localization of different discriminative local patches in Columns 3, 4 and 5. Second, a hierarchical bilinear pooling is proposed to combine complementary information from different

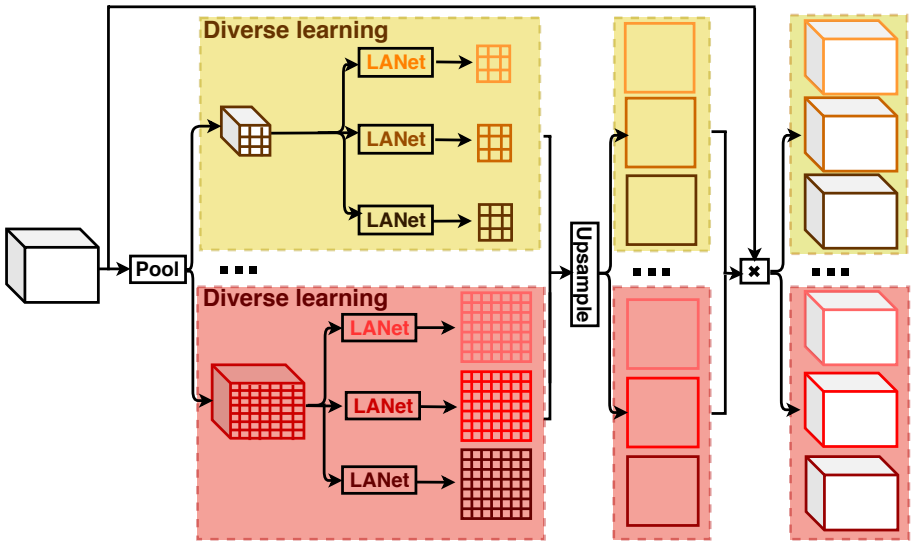

Figure 5.4: **Framework of the proposed pyramid diverse attention (PDA)**. We set the number of local branches to 3 as an example. It consists of a pyramid attention and a diverse learning. The former allows the network to weigh face parts at various scales automatically. The latter guides multiple local branches to extract diverse local representations.

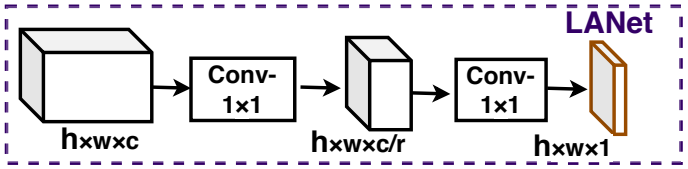

Figure 5.5: **Framework of the LANet**, where h, w and c represent height, width and number of feature maps, respectively. r is the reduction ratio.

hierarchical layers. Exactly, it uses different cross-layer bilinear modules to integrate both the high-level abstraction and the low-level detained information. The major contributions of our work are three-fold:

1. The proposed pyramid diverse attention introduces multiple attention-based local branches at different scales to emphasize different discriminative facial regions at various scales automatically, avoiding the need of face landmark detection. To our knowledge, this is the first attempt of automatic locating multiple complementary facial regions in general face recognition.

2. Instead of simple concatenation or addition, a hierarchical bilinear pooling is presented to combine features from different hierarchical layers, covering both local details or small-

scale face regions in low layers to high-level abstraction and large-scale parts in high layers.

3. The proposed model achieves the state-of-the-art performance on several challenging face recognition tasks.

5.2 Proposed Method

5.2.1 Overall Framework

As shown in Fig. 5.3, our framework consists of four parts: stem CNN, local CNNs, global CNN and classification.

Due to its good performance, HSNet-61 model [5] is used. SENet [107] can enhance important feature maps and inhibit less informative. Since layers in HSNet model contain feature maps captured at various sizes, we fuse SENet with HSNet model to improve the model capacity, namely SENet-HSNet. There are three blocks. The first two blocks are used as the stem CNN, as illustrated in Fig. 5.3. In the third block, every two layers of five layers are used as inputs of local CNNs which are designed to extract hierarchical multi-scale diverse local features. More details are discussed in Sec. 5.2.2. SENet-HSNet model is the default model if not specified. We also consider LS-CNN model [5] as the stem CNN due to its powerful generalization ability. In theory, our proposed hierarchical pyramid diverse attention module can be applied to any networks. Here, we investigate the above two networks as representatives.

Finally, the learned global and local information is combined to further boost the accuracy. Global CNN extracts the holistic face representation by consecutive global average pooling (GAP) and fully connected (FC) layers with 512 units. Local CNNs output 512 units. We can obtain joint local and global representations with 1024 units, followed by a FC layer with 512 units and a classification layer.

5.2.2 Local CNNs

Local CNNs are developed to extract multi-scale diverse local features hierarchically, which consist of a pyramid attention and a diverse learning. Specifically, due to pose vari-

ations or large expression changes, facial parts may have varying sizes. Therefore, it is necessary to represent local patches at various sizes. To this aim, a pyramid attention is proposed to locate multi-scale discriminative facial regions adaptively. However, it is possible that multiple local branches have redundant responses around the similar regions, like similar eye responses in Fig. 5.1, Column 2. To address this issue, a diverse learning is proposed to guide multiple branches to locate diverse facial regions automatically. The pyramid diverse attention (PDA) is formed by combining the pyramid attention with the diverse learning.

Besides, few works consider hierarchical features from different layers in face recognition, resulting in loss of discriminative features from low layers. To alleviate this issue, the hierarchical bilinear pooling is used to explore good inter-layer interactions.

5.2.3 Pyramid Attention

The framework is shown in Fig. 5.4. There are multi-scale local representations encoded in a single layer, making it necessary to calibrate features at different scales. Pyramid scales refer to different scales. The structure is very like a pyramid, which is also similar to [134].

Let $X \in R^{\mathrm{h} \times \mathrm{w} \times \mathrm{c}}$ denote the input of the pyramid attention, where h, w and c represent height, width and number of feature maps, respectively. First, feature maps are splited into outputs with different scales $\left[X_1, X_2, ..., X_S\right]$, where S is the number of scales. $X_i \in R^{\mathrm{h_i} \times \mathrm{w}_i \times \mathrm{c}}$ is the output of the i^{th} level, where $h_i \times \mathrm{w}_i$ represent the spatial size. The finest level has the same size as input X. The other levels split feature maps into different sub-regions, and then pool features in corresponding sub-regions.

Second, for the i^{th} scale, the target is to output diverse attention masks $\left[M_i^1, M_i^2, ..., M_i^B\right]$, where B is the number of local branches. The j^{th} output attention mask in the i^{th} scale has the same spatial size as the input X_i, i.e. $M_i^j \in R^{\mathrm{h_i} \times \mathrm{w_i}}$. To model the feature map spatially, the Local Attention Network (LANet) from [5] is used, as shown in Fig. 5.5. Weights of different spatial locations are regressed by two consecutive convolutional layers. The first layer has $\frac{c}{r}$ feature maps, followed by a ReLU layer to increase the non-linearity. r means the channel reduction ratio. The second one generates a feature map (i.e. M_i^j) with a sigmoid function.

Third, we upsample different attention masks M_i^j across multiple local branches ($j \in [1, 2, ..., B]$) in different scales ($i \in [1, 2, ..., S]$) to have the same size as input $X \in R^{\,\mathrm{h} \times \mathrm{w} \times \mathrm{c}}$ using a bilinear interpolation.

Then, refined feature maps R_i^j for the j^{th} local branch in the i^{th} scale is aggregated by the product of the attention mask M_i^j and input X:

$$R_i^j = X \circ M_i^j, \tag{5.1}$$

where $\circ$ denotes Hadamard product.

Finally, output of i^{th} scale is obtained by first concatenating B local branches and followed by a 1×1 convolutional layer to output c feature maps. Then, feature maps across different scales are concatenated, regarding as outputs of the pyramid attention.

However, it may be difficult for multiple local branches in the same scale to find different discriminative regions simultaneously. To address this problem, a diverse learning is proposed to guide the learning of complementary information across different local branches.

5.2.4 Diverse Learning

A divergence loss L_{D} is proposed to guide multiple local branches to learn diverse attention masks (i.e. $M_i^1, M_i^2, ..., M_i^B$), locating diverse face regions and achieving a robust recognition from diversified parts (e.g. the left eye and mouth if the right eye is occluded in Fig. 5.1, Row 6, Column 8). The formulation is defined as following:

$$L_{\mathrm{D}} = \frac{2}{SB(B-1)} \sum_{i=1}^{S} \sum_{j=1}^{B} \sum_{\substack{\mathrm{k}=1, \\ k \neq j}}^{B} max(0, t - (M_i^j - M_i^k)^2), \tag{5.2}$$

where t is a hyper-parameter margin. M_i^j and M_i^k represent attention masks learned by local branches j and k in the i^{th} scale, respectively.

The diverse learning encourages each local branch to learn different attention masks by increasing their distances. Since each attention mask is applied on the same feature maps, diverse attention masks can locate different local patches. Consequently, the diverse learning can alleviate the problem of redundant responses in Fig. 5.1, Column 2, and extract diverse local patches, as shown in Fig. 5.1, Columns 3, 4 and 5. It is possible that some

less important facial parts and even noisy background noise are located, resulting in sub-optimal facial representations. To overcome this issue, the classification loss is used with the divergence loss to select discriminative local patches. Only discriminative local patches are emphasized. This explains why attention masks may have overlaps among different local branches where some attention masks emphasize small regions while some focus on larger facial parts.

5.2.5 Hierarchical Bilinear Pooling

Most existing works learn features from the last convolutional layer. However, representations from the individual layer are not comprehensive. We consider to fuse features from multiple hierarchical layers.

There are five layers in the last block of HSNet-61 model, shown in Fig. 5.3. The pyramid diverse attention is applied in every two layers, extracting complementary local information across different layers. Each single layer contains three parallel paths and the deepest path has three convolutional layers. Therefore, different features can be extracted hierarchically in every two layers. Instead of simple concatenation or addition, we adopt the approach in [136] to aggregate information from different layers.

Suppose $X, Y \in R^{\,\mathrm{h}\times\ \mathrm{w}\times\mathrm{c}}$ are outputs of two different layers, where h, w and c represent height, width and number of feature maps, respectively. We denote a c dimensional feature at a spatial location i on X as $x_i = \left[X_1, X_2, ..., X_{\mathrm{c}}\right]^\top$. Similarly, a c dimensional feature from Y is $y_i = \left[Y_1, Y_2, ..., Y_{\mathrm{c}}\right]^\top$.

To capture more comprehensive local features, the cross-layer interaction is used, which is defined as following:

$$z_{\mathrm{i}} = x_i^\top w_{\mathrm{i}} y_i = x_i^\top U_{\mathrm{i}} {V_{\mathrm{i}}}^\top y_i = {U_{\mathrm{i}}}^\top x_i \circ {V_{\mathrm{i}}}^\top y_i, \tag{5.3}$$

where $w_{\mathrm{i}} \in R^{\mathrm{c}\ \times\ \mathrm{c}}$ is the projection matrix and $\mathrm{z_i}$ is the projected output. $\circ$ means Hadamard product. In [136], the projection matrix is factorized into two one-rank vectors $U_{\mathrm{i}}, V_{\mathrm{i}} \in R^{\mathrm{c}}$.

To encode local information, features should be expanded to a high dimensional space by linear mappings. Therefore, a weight matrix $w = \left[w_1, w_2, ..., w_{\mathrm{d}}\right]$ is defined to obtain a d

dimensional feature z.

$$z = U^{\top} x_{\mathrm{i}} \circ V^{\top} y_{\mathrm{i}}, \tag{5.4}$$

where $U, V \in R^{\mathrm{c} \times \mathrm{d}}$ and d is the dimension of the projected feature.

We consider to aggregate features from more layers, capturing more discriminative local features. Let X^1, X^2, X^3 represent outputs from three different layers. Eq. (5.4) is extended to concatenate multiple cross-layer representations:

$$z = Concat(U^{\top} x^1 \circ V^{\top} x^2, U^{\top} x^1 \circ S^{\top} x^3, V^{\top} x^2 \circ S^{\top} x^3). \tag{5.5}$$

Finally, a FC layer is used to reduce dimension to 512.

5.3 Experiments

In this section, we conduct experimental validation of the proposed HPDA model on several datasets.

5.3.1 Implementation Details

CPLFW and CALFW datasets provide cropped faces. For other datasets, faces are cropped by MTCNN [4].

The t in Eq. (5.2) is set to 1 empirically. After comparative experiments, the number of local branches is 3 (i.e. $B = 3$) and the pyramid diverse attention in Fig. 5.4 has three different scales (i.e. $S = 3$).

5.3.2 Ablation Analysis

We conduct several experiments to analyze the proposed model on LFW, CPLFW and CALFW datasets. Tables 5.1 and 5.2 give detailed investigations.

Importance of Diverse Learning

To locate diverse facial regions automatically in multiple local branches, a diverse learning is proposed. Competitive results demonstrate its superiority in Table 5.1.

Table 5.1: Ablation analysis (%) of the HPDA. Global CNN or local CNNs refer to the framework in Fig. 5.3.

Model	LFW	CPLFW	CALFW
W/o diverse learning	99.15	85.82	90.52
W/o hierarchical	99.07	85.63	90.35
Global CNN	99.10	79.30	89.90
Local CNNs	99.22	85.73	90.20
HPDA	99.33	86.07	90.93

Table 5.2: Comparison of varying number of **optimal scale levels** S and **local branches** B, values of **hyper-parameter** t and **methods to fuse channels from different layers**.

S	B	t	Channel fusion	LFW	CPLFW	CALFW
1	3	1.00	HBP	99.30	85.93	90.57
2	3	1.00	HBP	99.17	85.77	90.58
3	3	1.00	HBP	99.33	86.07	90.93
4	3	1.00	HBP	99.15	85.53	91.10
3	1	1.00	HBP	99.08	85.63	90.47
3	3	1.00	HBP	99.33	86.07	90.93
3	5	1.00	HBP	99.07	86.14	90.87
3	3	0.25	HBP	99.20	85.98	90.30
3	3	0.50	HBP	99.15	85.73	90.75
3	3	1.00	HBP	99.33	86.07	90.93
3	3	2.00	HBP	98.97	85.18	90.65
3	3	4.00	HBP	99.20	85.97	90.43
3	3	1.00	Concat	99.17	85.82	90.88
3	3	1.00	Add	99.18	85.67	90.55
3	3	1.00	HBP	99.33	86.07	90.93

Besides, this is a face landmark free approach. For example, MTCNN [4] fails to detect landmarks for faces in Fig. 6.4, Row 1. For such challenging cases, our model can still learn discriminative local patches regardless of loss of face organs and noisy background (Row 3). Besides, it is also observed that in Fig. 5.1, Columns 3, 4 and 5, diverse attentions are generated by three local branches when the diverse learning is used, compared with the model without the diverse learning in Fig. 5.1, Column 2. This is because the diverse learning can emphasize informative face regions and suppress less important ones or background distraction. On one hand, discriminative local patches are enlarged. For example, as demonstrated in Fig. 6.4, Columns 6, 8, without diverse learning, discriminative face regions are lost in Row 2 under dramatic illumination changes or pose variations. However, our method locates rich discriminative face regions in Row 3. This benefits from the diverse learning which pushes models to learn more discriminative regions. On the other hand, our method suppresses noisy regions, like goggles in Fig. 6.4, Columns 1, 2, Row 3, compared with the model without the diverse learning in Row 2.

Importance of Hierarchical Features

Compared with the last convolutional layer which captures high-level face abstractions, lower layers preserve more local details or small-scale face parts, exhibiting complementary components.

Our model fuses features from three different layers hierarchically, as shown in Fig. 5.3. To investigate the necessity, we compare the HPDA model with the model without hierarchical information (i.e. w/o hierarchical) which only extracts features based on the last convolutional layer. As demonstrated in Table 5.1, our HPDA has slightly better overall performances. This shows the complementary information contained in low layers.

Effects of Hierarchical Bilinear Pooling

It is intuitive to directly concatenate or add feature maps from different layers. However, simply concatenation or addition fails to capture rich inter-layer feature relations. To overcome this issue, we use hierarchical bilinear pooling (HBP) to incorporate multiple cross-layer interaction modules to learn complementary information from different layers.

Experiments are conducted to compare HBP with concatenation and addition. For fair comparison, channels in each layer is transformed to a high dimension by 1×1 convolution as the HBP, encoding rich local information. Table 5.2 shows that HBP achieves slightly better performances than concatenation or addition, demonstrating that its superiority in capturing inter-layer feature relations.

Effects of Number of Local Branches B

We set B to 1, 3 and 5. The performance increases when B is changed from 1 to 3. This is because more discriminative facial regions are located by varying local branches which are guided by the diverse learning, boosting the performance. However, the performance drops slightly when B is changed from 3 to 5. This can be explained by the fact that the diverse learning encourages diverse information extracted by different branches, resulting in useless or noisy information in some branches when B is too large.

Importance of Local CNNs and Global CNN

First, as shown in Fig. 5.1, the global CNN tends to characterize some less informative regions (e.g. cheek in Column 7, Rows 3, 4) and model more background information, compared with the local CNNs (Column 7 to Column 6, Rows 3, 4). On the other hand, the local CNNs focus on informative areas (Column 6, Rows 3, 4). This explains why the local CNNs offer performance improvement compared with the global CNN, as shown in Table 5.1.

Second, global features describe general information of whole faces (Fig. 5.1, Column 7). Differently, the local CNNs encode more detailed characteristics within different local patches (Fig. 5.1, Column 6). Take faces in Fig. 5.1, Rows 3, 4 as examples, the noses and the shape of whole faces remain similar in these two faces despite changes of the global face appearances. The global CNN can represent holistic face shapes (Column 7) and the local CNNs can describe noses (Column 6). It is intuitive to combine the global CNN with the local CNNs to boost the performance. It is observed that the combined global and local representation (Column 8) is more descriptive compared with the global CNN (Column 7) and the local CNNs (Column 6).

Effects of the Scale Level Number S

We set the scale number S in the pyramid attention to 1, 2, 3 and 4, respectively, where scale 7 is used for $S = 1$ and scale 7, 5, 3 and 1 are used for $S = 4$. However, setting proper scale levels is necessary to boost the performance. The HPDA achieves the best overall performance when $S = 3$ in Table 5.2.

Discriminative face regions have varying sizes due to expression changes (e.g. eyes in Fig. 6.4, Row 1, Columns 3, 4, 5) and pose changes (e.g. mouths in Fig. 6.4, Row 1, Columns 6, 7, 8). Although HSNet model extracts rich multi-scale features in a single layer by utilization of multi-scale convolutional kernels and concatenation of feature maps from previous layers, it can just weigh parts with fixed scales, while lacking the capacity to emphasize regions with flexible scales. In contrast, when the pyramid attention is used, attention masks are able to capture information at various scales.

Effects of Hyper-parameter t

Values of t (0.25, 0.5, 1, 2 and 4) are compared. It can be seen that performance increases when t increases from 0.25 to 1, but drops when t increases from 1 to 4.

Table 5.3: Performance comparison (%) of the HPDA model with state-of-the-art methods. LS-CNN model and MS-Celeb-1M dataset are used as the stem CNN and training dataset, respectively. As the baseline, LS-CNN model is run on both VGGFace2 and MS-Celeb-1M datasets.

Methods	IJB-A		CALFW	CACD-VS	CPLFW	VGGFace2-FP	LFW
	FAR=0.01	FAR=0.001					
VGGFace [27]	60.5	36.7	86.50	96.00	-	-	99.13
Center loss [29]	52.1	31.3	85.48	97.48	77.48	75.10	99.28
SphereFace [36]	54.8	39.6	90.30	-	81.40	20.10	99.42
VGGFace2 [32]	-	-	90.57	-	84.00	62.22	99.43
LS-CNN [5] (VGGFace2)	87.5	75.5	92.00	99.50	88.03	69.92	99.52
LS-CNN [5] (MS-Celeb-1M)	87.2	77.5	94.40	99.10	-	-	-
ArcFace [31]	68.6	65.7	95.45	-	92.08	46.20	99.83
Co-mining [137]	-	-	93.28	-	87.31	-	-
MV-Softmax [138]	-	-	95.63	-	89.69	-	99.79
HPDA	87.6	80.3	95.90	99.55	92.35	95.32	99.80

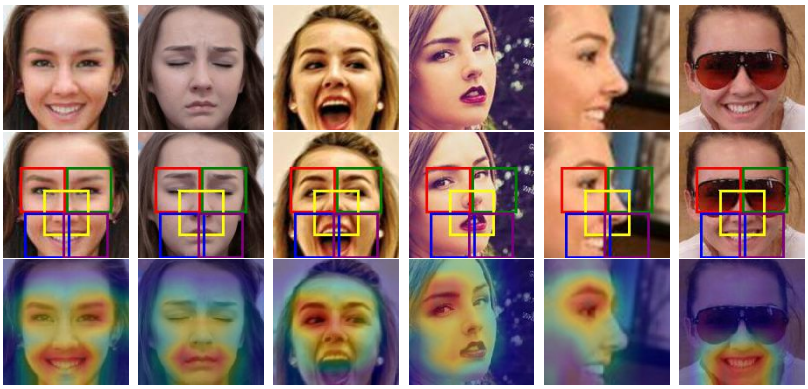

Figure 5.6: **Qualitative comparison between landmark-based methods and the proposed HPDA model**. **Row 1**: some challenging faces. **Row 2**: even face landmarks are detected, predefined crops may not be flexible and robust under pose, expression and occlusion variations. **Row 3**: the HPDA model can locate discriminative facial parts flexibly.

5.3.3 Experiments on Cross-Quality Face Matching

Notice that this task is very close to real-world scenarios where the match is between faces from access control, video surveillance, or public safety (low-quality faces) and enrolled photos (high-quality faces). Following the work [65], several public models are compared. As shown in Table 5.3, our method HPDA increases the accuracies.

There are many factors that can influence the quality of face images, such as resolution, pose, expression, aging, and illumination. A positive pair of high-quality and low-quality faces are shown in Fig. 5.1, Rows 1, 2. In such cases, the local CNNs can learn discriminative and diverse local regions. As illustrated in Column 6, it is observed that the local CNNs filter out useless background information and enhance important regions. In contrast, without the diverse learning, these compared models cannot filter out background, which would inevitably decrease the performance.

Previous landmark-based and attention-based local methods are qualitatively compared with the proposed HPDA model. Landmark-based methods crop face parts around landmarks, suffering from noise from adjacent parts or background. For example, as shown in Fig. 5.6, Columns 1, 2, 3, mouths have varying shapes or sizes (Row 1) due to expression changes. As a consequence, some crops contain noise from noses and background (Row 2). Meanwhile, because of pose variations (Row 1, Columns 4, 5) and occlusions (Row 1, Col-

umn 6), some parts are partly or completely invisible. In such cases, background (Row 2, Columns 4, 5) or sunglasses (Row 2, Column 6) are cropped, leading to noisy representations. Moreover, it is noticed that landmark detection may be not accurate, resulting in inaccurate crops (Row 2, Columns 4, 5). For the attention-based models without diverse learning, many unimportant or noisy facial regions are emphasized in Fig. 6.4, Row 2. In contrast, our proposed HPDA model can locate discriminative parts and suppress less informative, as shown in Row 3, demonstrating its superiority.

5.3.4 Experiments on Cross-Age Face Matching

We compare performances on CALFW and CACD-VS datasets in Table 5.3. It can be seen that HPDA model can slightly outperform others. For this task, as shown in Fig. 5.1, Rows 3, 4, even if these faces have undergone dramatic appearance changes, some local facial regions (e.g. the pointy nose) remain to be very characteristic and the face shape (i.e. the oval face) is still similar. Accordingly, the global CNN (Column 7) and local CNNs (Column 6) focus on facial shapes and pointy noses, respectively.

5.3.5 Experiments on Cross-Pose Face Matching

As shown in Table 5.3, our HPDA model outperforms the state-of-the-art on CPLFW and VGGFace2-FP datasets. Results on VGGFace2-FP dataset of Center loss, SphereFace and ArcFace models are from [139].

There are several reasons that can explain the results. First, local patches appear at different sizes due to pose changes. Especially, profile faces are self-occlusion where only partial organs are visible, like the small-scale mouths in Fig. 6.4, Columns 7, 8. The problem above motivates us to develop the pyramid attention to capture multi-scale features. Second, background distraction is a common problem for faces with large pose variations. On one hand, there are some transition regions between discriminative local patches (e.g. noses, eyes, mouths) and background for frontal faces. On the other hand, discriminative local patches are near to noisy background for profile faces. As a result, noisy background tends to be cropped with discriminative local patches, leading to inferior representations. This

is especially serious for landmark-based local methods because face landmarks may lie in the background, as shown in Fig. 5.6, Row 2, Columns 4, 5. Meanwhile, because previous attention-based local methods do not consider the attention diversity, some useful regions are missed, as shown Fig. 6.4, Columns 7, 8, Row 2. The proposed diverse learning can alleviate this issue by emphasizing discriminative face regions and suppressing less important and background distraction, as illustrated in Row 3.

Besides, the local CNNs extract complementary information to the global CNN. One positive pair with pose changes are shown in Fig. 5.1, Rows 5, 6, which have similar eyes and big lips. The global CNN describes eyes in Column 7. Meanwhile, the local CNNs focus on big lips in Column 6. The combined local and global CNNs (Column 8) show that the complementarity between the local CNNs and the global CNN is learned.

5.3.6 Experiments on LFW Dataset

We also report the accuracy on the popular LFW dataset where most faces are frontal or near-frontal. As shown in Table 5.3, our model is only slightly worse than ArcFace, but outperforms all other models. Although our proposed HPDA model is designed to solve challenging face matching tasks, it still has an excellent generalization ability on LFW dataset.

5.4 Summary

We have proposed the HPDA model for face recognition, which adaptively extracts hierarchical multi-scale local representations. A pyramid attention has been applied to locate multi-scale discriminative face regions automatically and weigh adaptively for multiple facial parts. To capture diverse local facial representations, a diverse learning has been introduced to guide multiple attentions to locate diverse facial parts. The hierarchical bilinear pooling has been used to fuse complementary features from different layers. Experimental results on several very challenging face recognition tasks have validated the effectiveness and importance of our proposed HPDA model.

Chapter 6

DSA-Face: Diverse and Sparse Attentions for Face Recognition Robust to Pose Variation and Occlusion

6.1 Motivations

Face recognition (FR) relying on deep convolutional neural networks has achieved promising results [31, 138, 140, 141]. Most existing works extracted features on global face images. However, as illustrated in Fig. 6.1, the global face images may change dramatically caused by face masks in (1), quality changes in (2), age gaps in (3), and pose variations in (4). In contrast, local patches may change less, such as the similar noses in (3) and eyes in (1), (2), and (4), which would play an important role for face matching.

Some works extracted local features on cropped parts centered around facial landmarks [37, 38, 42]. However, landmark detection may fail under extreme illuminations, strong pose variations, severe occlusions, or heavy makeup. Without relying on landmarks, recent attention-based networks applied attention modules to automatically locate important parts [2, 5]. However, as illustrated in Fig. 6.2 (c), these networks may have the attention redundancy problem: Multiple attention maps tend to have redundant responses around the

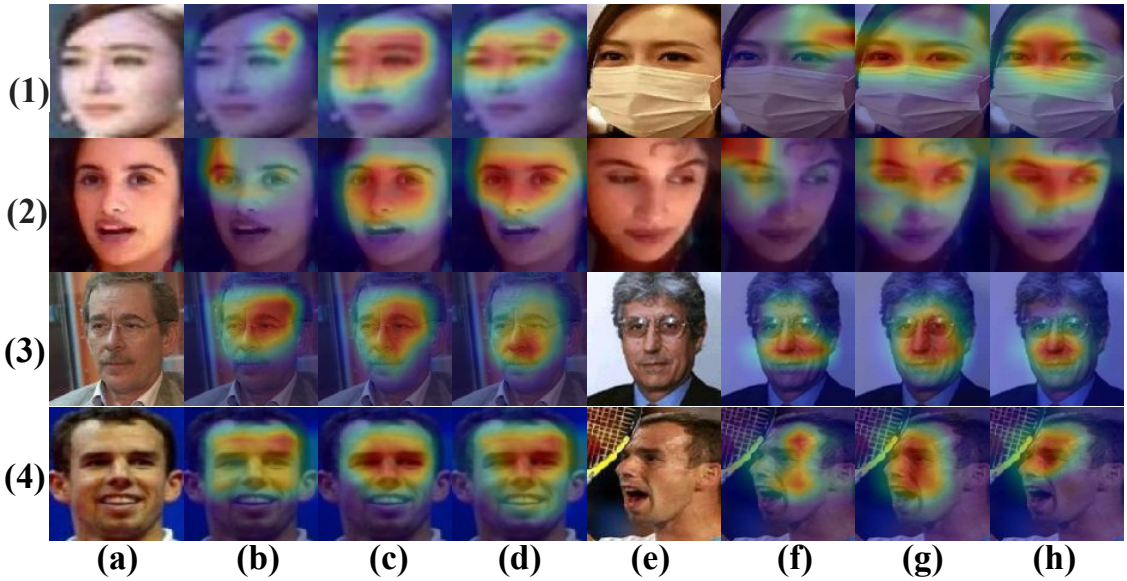

Figure 6.1: Effects of the DSA-Face model on four positive pairs. **(a)** & **(e):** Input faces which are affected by face masks **(1)**, quality changes **(2)**, aging **(3)**, and pose variations **(4)**. **(b)** & **(f):** Without a proper guidance, multiple attention maps focus on few local patches, while ignoring some potentially important regions. **(c)** & **(g):** The PSCA can locate rich local patches. However, because every facial part is mined thoroughly, it is possible that some unimportant patches may be taken into consideration, e.g. cheeks or foreheads. **(d)** & **(h):** The ASL is able to filter out useless facial parts by suppressing responses in attention maps.

same facial parts, while missing other potentially useful regions. This is serious for faces where the emphasized parts may be invisible due to occlusions or pose variations, or being hard to understand caused by other face image quality changes.

To alleviate the above issues, it is critical to capture local representations that are as diverse as possible. Our previous work [14] achieved this partially by maximizing pairwise attention distances which are Euclidean distances between every pair of attention maps. As a result, varying attention maps are encouraged to be different from each other, and thus extracting different facial parts. However, the attention subset problem may occur. As illustrated in Fig. 6.2 (d), centered on the same part (i.e. the nose), some attention maps emphasize larger areas, while some others highlight smaller areas which are subsets of the larger ones. In such a case, the pairwise distances may still satisfy the criteria in [14].

Consequently, the performance is deteriorated due to lacking of some other discriminative facial parts (e.g. eyes). Besides, some unimportant facial parts may also be emphasized in large response areas, like cheeks or foreheads, limiting the representational ability of features. This problem would become especially serious in case of the masked face matching where face masks are extracted as well.

The aforementioned problems motivate us to develop diverse and sparse attentions, called DSA-Face, which mainly consist of pairwise self-contrastive attentions (PSCA) and attention sparsity loss (ASL). First, the PSCA enlarges pairwise attention distances in a self-contrastive way. In such a manner, different attention maps are encouraged to repulse each other, and thus locate different facial parts. Second, since every image detail is mined thoroughly under the guidance of the PSCA, some noisy information like background or face masks may be extracted if there is not an explicit signal. To address this issue, the ASL is proposed to encourage sparse responses in attention maps where important regions tend to be emphasized and noisy parts are likely to be suppressed. Besides, we take masked face matching to promote and validate the development in this field. Due to the ongoing pandemic of the COVID-19, people are wearing face masks to prevent its spread in many scenarios, such as access control and security checks [142]. Most existing FR models are designed based on deep learning which require a large number of training images. However, most existing training datasets have limited face images wearing masks. As a consequence, some open questions are raised: Could existing FR models identify masked faces? How about our DSA-Face model? Experimental results show that our DSA-Face model outperforms other models dramatically. The main contributions of our work are three-fold:

1. Pairwise self-contrastive attentions are proposed to enforce models to extract diverse local features;

2. An attention sparsity loss is devised to encourage sparse responses in attention maps, discouraging the emphasis on noisy regions while encouraging focus on discriminative facial parts;

3. The masked face matching is presented to benefit from our algorithm. Experimental results on this and several other challenging tasks validate the importance and usefulness of our proposed method.

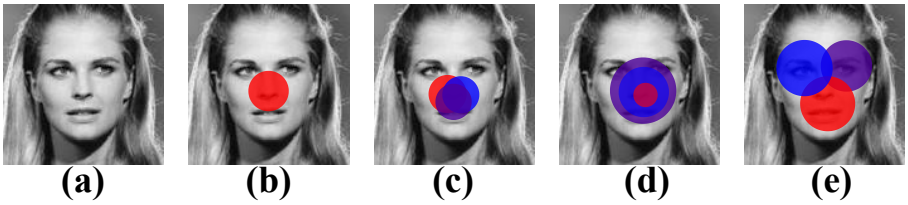

Figure 6.2: Comparison of different attention-based networks. **(a):** An example face. **(b):** The response of an attention map. **(c): Attention redundancy problem**: Multiple attention maps just focus on the same facial part. **(d): Attention subset problem**: Centered on the same part, some attention maps emphasize larger areas, while some highlight smaller areas which are subsets of the larger ones. **(e): Diverse attention maps** generated by our proposed DSA-Face model.

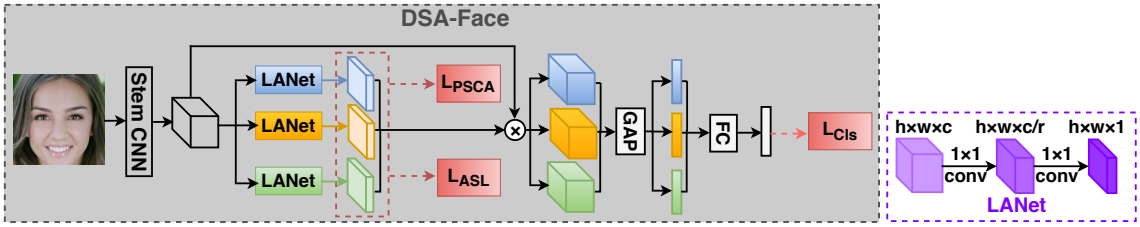

Figure 6.3: Overall framework. L_{PSCA} encourages models to locate diverse local patches by mining every facial image detail. L_{ASL} guides models to have sparse responses in attention maps, suppressing emphasis on noisy regions. L_{Cls} is the classification loss to learn discriminative features. LANet is a spatial attention in [5], where h, w, and c is the height, width, and number of features maps, respectively, and r is the reduction ratio. GAP and FC mean global average pooling and fully-connected layer, respectively. Notice that only three local branches are used for good illustration.

6.2 Proposed Method

The overall framework is shown in Fig. 6.3. The stem CNN extracts high-level feature maps for extracting diverse local features. Then multiple local branches are used to extract diverse local features, which are guided by the PSCA and ASL. Local features are extracted and used to predict the identity.

Due to its effectiveness, HSNet-61 model [5] is selected as the default stem CNN if unspecified. Besides, we also consider ResNet-100 [50], LS-CNN and LS-CNN-177 [5] models. There are three stages in LS-CNN. Compared with LS-CNN which has 6, 6, and 8 repeated

operations in three stages, LS-CNN-177 sets the three stages to 12, 12, and 16, respectively, further boosting the representational capacity.

6.2.1 Pairwise Self-Contrastive Attentions

Since the same architecture (i.e. LANet in Fig. 6.3) is used in different local branches, highly similar parameters would be learned if there is not a proper guidance. This would result in the attention redundancy problem. Therefore, a diversification prior should be designed, which aims at diversifying multiple local branches.

The loss in [14] achieved this goal partially by maximizing pairwise attention distances, which is defined as follows:

$$L_{\mathrm{PCA}} = \frac{2}{B(B-1)} \sum_{i=1}^{B} \sum_{j=i+1}^{B} max(0, t - dist(M_i, M_j)), \tag{6.1}$$

where M_{i} and M_{j} denote the attention maps which are generated by the i^{th} and j^{th} local branches, respectively. B is the total number of local branches, t refers to a hyper-parameter margin, $dist(M_{\mathrm{i}}, M_{\mathrm{j}})$ is the Euclidean distance between M_{i} and M_{j}. As a result, varying attention maps are encouraged to be different from each other, and thus focus on different facial parts. Here, we call them pairwise contrastive attentions (PCA). Although it boosts the performance, the PCA could result in the attention subset problem (Fig. 6.2 (d)).

To alleviate this problem, we propose the pairwise self-contrastive attention (PSCA), which is guided by a divergence loss L_{PSCA}. The loss is defined as follows:

$$\begin{aligned} L_{\mathrm{PSCA}} = \frac{2}{B(B-1)} \sum_{i=1}^{B} \sum_{j=i+1}^{B} max(0, t - dist(M_i, M_j) + \\ \alpha * dist(M_i, \overline{M_j})). \end{aligned} \tag{6.2}$$

It maximizes the distance between M_{i} and M_{j} and minimizes the distance between M_{i} and $\overline{M_{\mathrm{j}}}$. Notice that $M_j + \overline{M_{\mathrm{j}}} = E$ where E is a matrix with all-ones elements, meaning that areas with high responses in M_{j} would have low responses in $\overline{M_{\mathrm{j}}}$, and vice versa.

For local representations in FR, only few important local patches (e.g. the nose, mouth, both eyes, or mouth) can play a crucial role. Therefore, it is expected that M_{i} and M_{j} have sparse responses on few parts. On the other hand, $\overline{M_{\mathrm{j}}}$ has high responses around large

unimportant areas. Consequently, $dist(M_i, \overline{M_j})$ tends to be much larger than $dist(M_i, M_j)$. Therefore, we use a coefficient (i.e. α) to balance these two values. Notice that Eqn. (8.3) can be seen as a special case of Eqn. (6.2) when $\alpha = 0$. Compared with PCA, it is more difficult to meet the constraint of the attention subset problem in PSCA, which would be proved in the following.

Proof. We use set operations to explain why the proposed PSCA can alleviate the attention subset problem.

Since $dist(M_\mathrm{i}, M_\mathrm{j})$ is about the Euclidean distance between M_i and M_j, the square operation is used. If we want to use the set operation to explain the above issue, $dist(M_\mathrm{i}, M_\mathrm{j})$ can be converted to $M_\mathrm{i} \oplus M_\mathrm{j}$ as follows:

$$M_\mathrm{i} \oplus M_\mathrm{j} = (M_\mathrm{i} - M_\mathrm{j}) \cup (M_\mathrm{j} - M_\mathrm{i}), \tag{6.3}$$

where $\oplus$, $-$, and $\cup$ mean the disjunctive union, difference set, and union set operations, respectively.

For the PCA [14], it is expected that the following condition should be satisfied for Eqn. (8.3):

$$dist(M_\mathrm{i}, M_\mathrm{j}) \geq t. \tag{6.4}$$

If the set operation is used, the above formula can be converted as follows:

$$\begin{aligned} M_\mathrm{i} \oplus M_\mathrm{j} &\geq t. \\ (M_\mathrm{i} - M_\mathrm{j}) \cup (M_\mathrm{j} - M_\mathrm{i}) &\geq t. \end{aligned} \tag{6.5}$$

If the attention subset problem occurs (assume $M_\mathrm{j} \subset M_\mathrm{i}$), the following inequality should be met as follows:

$$(M_\mathrm{i} - M_\mathrm{j}) \geq t. \tag{6.6}$$

For our PSCA, we expect that the following formula should be satisfied for Eqn. (6.2):

$$dist(M_\mathrm{i}, M_\mathrm{j}) \geq t + \alpha * dist(M_\mathrm{i}, \overline{M_\mathrm{j}}), \tag{6.7}$$

where $\overline{M_{\mathrm{i}}}$ is produced by using the complement set operation on M_{i}. If the set operation is used, the above formula can be converted as follows:

$$\begin{aligned}(M_{\mathrm{i}} \oplus M_{\mathrm{j}}) \geq t + \alpha * (M_{\mathrm{i}} \oplus \overline{M_{\mathrm{j}}}).\\ (M_{\mathrm{i}} - M_{\mathrm{j}}) \cup (M_{\mathrm{j}} - M_{\mathrm{i}}) \geq t + \alpha *\\ ((M_{\mathrm{i}} - \overline{M_{\mathrm{j}}}) \cup (\overline{M_{\mathrm{j}}} - M_{\mathrm{i}})).\end{aligned} \tag{6.8}$$

If the attention subset problem occurs (assume $M_{\mathrm{j}} \subset M_{\mathrm{i}}$), the following equation should be met as follows:

$$(M_{\mathrm{i}} - M_{\mathrm{j}}) \geq t + \alpha * (M_{\mathrm{j}} \cup \overline{M_{\mathrm{i}}}). \tag{6.9}$$

Compared with the constraint (Eqn. (6.6)) for the PCA, the constraint (Eqn. (6.9)) for the PSCA is more difficult to be satisfied. In other words, the attention subset problem is less likely to occur in the PSCA. In such a manner, different local branches are encouraged to repulse from each other and extract more diverse local representations.

6.2.2 Attention Sparsity Loss

For local patches in face image, there are several characteristics: 1) Few important local patches (e.g. the nose, mouth, both eyes, or mouth) can contribute to FR; 2) Some regions are less discriminative, like the cheek or forehead; 3) Background or occlusions are noisy. This issue is especially important for robust FR where background information or face masks may be represented because discriminative parts are close to important parts (e.g. eyes) in profile or masked faces, respectively. The attention module automatically determines which regions should be highlighted. It is highly desired that few discriminative parts should be emphasized and unimportant or noisy regions should be suppressed. Therefore, the attention responses should be sparse.

However, few existing approaches have designed a mechanism to control the attention responses. When some noisy regions are extracted, such as noisy background or face masks, they may deteriorate the recognition performance. This issue is especially serious when the PSCA is used, because every face detail is explored to extract comprehensive features. For example, for masked face matching, because face masks are very close to discriminative

parts, noisy information tends to be represented. To avoid this issue, we propose an attention sparsity loss L_{ASL} to augment the objective function, which is defined as follows:

$$L_{\text{ASL}} = \frac{1}{B \cdot h \cdot w} \sum_{i=1}^{B} \sum_{j=1}^{h} \sum_{k=1}^{w} (M_i^{j,k})^2, \tag{6.10}$$

where B is the total number of local branches. h and w is the height and width of the attention map M_i. The main incentive behind this idea is to impose sparsity penalization loss by encouraging response sparsity in attention maps. By penalizing highly responded areas, the L_{ASL} constrains the values in attention maps towards zero. In such a manner, only useful facial parts tend to respond, while noisy regions are suppressed.

6.2.3 Overall Loss

In order to supervise the network to learn discriminative features, the classification loss L_{Cls} should be used, including CosFace loss [30] ($L_{\text{Cls-cosface}}$) and ArcFace loss [31] ($L_{\text{Cls-arcface}}$), which can generate discriminative features.

CosFace loss is formulated as follows:

$$L_{\text{Cls-cosface}} = -\frac{1}{N} \sum_{i=1}^{N} log \frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j \neq y_i} e^{s \cos \theta_{j,i}}}, \tag{6.11}$$

where N represents the number of samples. m is the cosine margin to maximize the decision margin in the angular space. The sample x_i is normalized and re-scaled to s, belonging to the y_i class.

The additive angular margin penalty m is used in ArcFace loss to encourage the intra-class compactness and inter-class discrepancy.

$$L_{\text{Cls-arcface}} = -\frac{1}{N} \sum_{i=1}^{N} log \frac{e^{s(\cos(\theta_{y_i,i}+m))}}{e^{s(\cos(\theta_{y_i,i}+m))} + \sum_{j \neq y_i} e^{s \cos \theta_{j,i}}}. \tag{6.12}$$

In our approach, the overall loss is formulated as follows:

$$L = L_{\text{Cls}} + \lambda L_{\text{PSCA}} + \beta L_{\text{ASL}}, \tag{6.13}$$

where L_{Cls} is the classification loss, supervising the network to learn discrimination between subjects. Theoretically, our proposed DSA-Face model can work together with any loss, like

CosFace loss [30] and ArcFace [31]. L_{PSCA} encourages the localization of diverse facial parts. Since every detail is explored across an image, L_{Cls} may fail to filter out some noisy regions, like face masks or other occluded parts. To address this problem, L_{ASL} is able to enhance the ability of suppressing noisy responses. λ and β control the balance in these three losses.

6.3 Experiments

6.3.1 Implementation Details

During training on VGGFace2, the batch-size is set 256. ArcFace loss [31] in Eqn. (7.8) is used where $s = 32$ and $m = 0.35$. Learning rate is 0.05 and is divided by 10 at the 7^{th} and 10^{th} epochs, respectively. It is set to $1e-4$ at the 12^{th} epoch. Training stops at the 12^{th} epoch. During training on MS1MV2 database, there are different settings in order to fairly compare existing models: First, **DSA-Face[a]** and **DSA-Face[c]** adopt ResNet-100 [50] and LS-CNN [5] as the stem CNN where the learning rate is initially set to 0.1 and 0.05. The learning rate is multiplied by 0.1 at the 13^{th} and 19^{th} epochs. It is set to $1e-4$ at the 23^{nd} epoch. Training stops at the 24^{th} epoch. The batch-size is 512. Both are supervised by ArcFace loss [31] in Eqn. (6.12). Second, **DSA-Face[b]** employees LS-CNN [5] as the stem CNN. The batch-size is set to 256. The number of warm-up epochs is 2. The learning rate starts at 0.03 and multiplied by 0.1 at the 7^{th} and 10^{th} epochs, respectively. It is set to $1e-4$ at the 12^{th} epoch. Totally there are 12 training epochs. Third, **DSA-Face[d]** uses the LS-CNN-177 as the backbone. The batch-size is set to 256. The number of warm-up epochs is 2. The learning rate starts at 0.03 and is multiplied by 0.1 at the 13^{th} and 19^{th} epochs, respectively. It is set to $1e-4$ at the 23^{rd} epoch. Totally there are 24 epochs. **DSA-Face[b]** and **DSA-Face[d]** are supervised by CosFace loss [30] in Eqn. (6.11) where scale s and margin m are set to 32 and 0.5, respectively.

For data pre-processing, faces are cropped and aligned to the size of 112×112 by the MTCNN [4]. After comparative experiments, t and α in Eqn. (6.2) are set to 1 and 0.05, respectively. The number of local branches (B) is 6. λ and β in Eqn. (6.13) are assigned with 0.1 and 0.05, respectively.

6.3.2 Ablation Study

In the ablation study, results are reported on CFP-FP, CPLFW, MLFW, and AFDB datasets, as shown in Tables 6.1 and 6.2. To show the results efficiently, a tuple (a, b, c, d) is used where a, b, and c demonstrate the accuracy on CFP-FP, CPLFW, and MLFW datasets and d is the Rank-1 accuracy on M2N protocol of AFDB dataset.

Effects of the proposed components

The proposed DSA-Face model mainly consists of two components: the PSCA and ASL. Their performances are shown in Table 6.1.

Without an explicit guidance, the attention redundancy problem may occur in multiple local branches, as shown in Fig. 6.2 (c). To alleviate this problem, the PSCA is proposed to guide different attention maps to locate diverse local patches. Some local patches are occluded due to pose variations or face masks. It is possible that some local patches remain the same for face pairs in such cases, consequently, diverse discriminative local patches are mined thoroughly by the PSCA to boost the performance from (96.87%, 91.10%, 78.93%, 18.92%) to (96.97%, 91.25%, 81.66%, 19.97%). As illustrated in Fig. 6.4, due to strong occlusions in (1) & (3), poor lighting in (2), and pose variations in (4), MTCNN method [4] fails to detect landmarks for these challenging faces. However, guided by the PSCA, diverse local patches are located in three local branches, as shown in (d), (e), & (f). This greatly improves the representation ability. Besides, as illustrated in Fig. 6.1, supervised by the PSCA, rich discriminative local patches are located in (c) & (g), compared with the model without a proper guidance in (b) & (f).

Since every image detail is explored, noisy information may be extracted if there is not an explicit control. To address this issue, the ASL is proposed to suppress responses in attention maps towards zero, discouraging responses around noisy face masks. Consequently, the ASL boosts the performance from 81.66% to 81.97% on MLFW and from 19.97% to 20.26% on AFDB. Similarly, the ASL constrains responses on background, which results in the accuracy improvement from 96.97% to 97.27% on CFP-FP and from 91.25% to 91.57% on CPLFW. As shown in Fig. 6.1, responses on attention maps are narrowed to focus on discriminative

local patches (e.g. noses in (3), eyes in (1), (2) & (4)), while neglecting less important parts (e.g. foreheads or cheeks). Therefore, the discriminative ability of features are boosted to achieve a better performance.

Comparison between PSCA and PCA

Diverse local representations can contribute to FR, which can be implemented in two ways: the PCA and PSCA. Performance comparison is shown in Table 6.1. Since the PSCA is less likely to have the attention subset problem, it has a better ability to generate diverse attention maps, and thus outperforms the PCA. Specifically, the overall accuracy is boosted from 97.17% to 97.27% on CFP-FP, from 91.43% to 91.57% on CPLFW, from 81.89% to 81.97% on MLFW, and from 19.33% to 20.26% on AFDB, respectively. We also qualitatively compare the PCA and PSCA on some challenging faces in Fig. 6.4. Compared with the PCA which emphasizes noisy background or noisy parts in (b), the PSCA is able to filter out noisy background and focus on important facial parts in (c).

Varying values of B

Different values of B, such as 2, 4, 6, and 8, are compared in Table 6.2. If B is too small, it is very likely that comprehensive facial parts are not well-explored, lacking of some discriminative local representations. This explains why the overall performance is boosted from (96.63%, 90.82%, 82.60%, 14.29%) to (96.81%, 91.25%, 77.48%, 16.56%) when B is increased from 2 to 4. Similarly, when B is changed to 6, improving the performances to (97.27%, 91.57%, 81.97%, 20.26%), respectively. However, if B is too large, some unnecessary information may not be removed. Consequently, the overall results are slightly dropped to (97.11%, 91.48%, 82.52%, 17.17%) if $B = 8$.

Different values of λ

Different values of λ in Eqn. (6.13) are compared in Table 6.2: 1.0, 0.5, 0.1, 0.05, and 0.01. When λ is reduced from 1.0, 0.5 to 0.1, performances are improved to (97.27%, 91.57%, 81.97%, 20.26%). This suggests that the signal of the PSCA is too strong, leading to the localization of diverse but noisy facial regions. When λ is reduced from 0.1 to 0.05 and 0.01,

Table 6.1: Effectiveness of different components and their combinations.

PSCA	ASL	PCA	CFP-FP	CPLFW	MLFW	AFDB
			96.87	91.10	78.93	18.92
√			96.97	91.25	81.66	19.97
	√	√	97.17	91.43	81.89	19.33
√	√		97.27	91.57	81.97	20.26

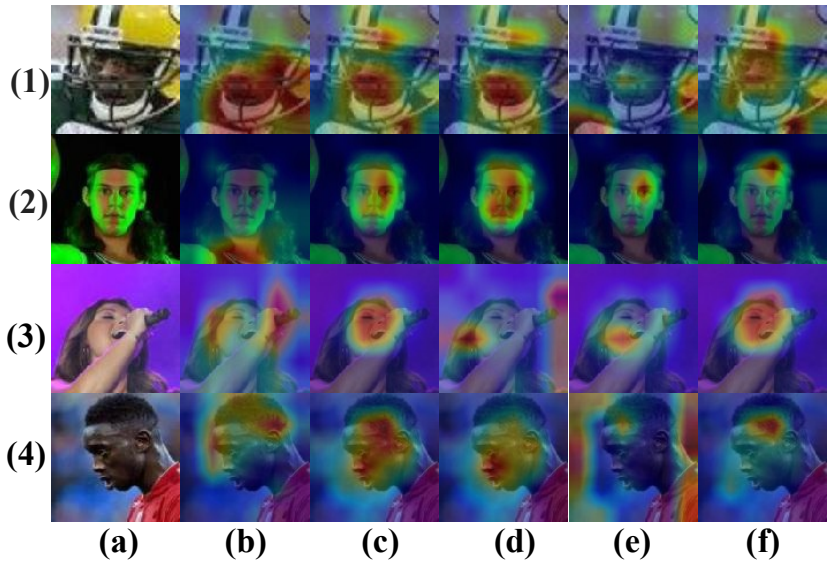

Figure 6.4: Qualitative comparison between the PCA and PSCA on some challenging faces. **(a)**: The MTCNN [4] fails to detect landmarks for some faces under heavy occlusions, poor lighting, low quality factors, and profile poses. **(b)**: Visualization of the PCA. **(c)**: Visualization of the PSCA. **(d)** & **(e)**, and **(f)**: 1^{st}, 2^{nd}, and 3^{rd} local branches where different local patches are located in each local branch. For a good illustration, only three local branches are used to show class activation maps (CAMs) [3].

overall performances drop. This is because the attention redundancy problem occurs if the signal of the PSCA is weak, resulting in missing of some other important features. Therefore, λ is set 0.1 to achieve a good trade-off between L_{Cls} and L_{PSCA}.

Different values of β

Several values of β in Eqn. (6.13) are compared in Table 6.2: 0.5, 0.1, 0.05, 0.01, and 0.005. When the β is decreased from 0.5 to 0.1, the overall performance is increased from (96.93%, 91.65%, 78.82%, 16.35%) to (96.96%, 91.57%, 82.60%, 17.22%). Likewise, if β is decreased from 0.1 to 0.05, the accuracy achieves (97.27%, 91.57%, 81.97%, 20.26%). This is because the L_{ASL} suppresses some important regions, leading to incomplete representations, if the value of β is too large. In contrast, if $\beta = 0.01$, or 0.005, the L_{ASL} is too weak, leading to features with noisy information. This is the main reason why the performances are dropped.

Varying values of t

We compare different values of the hyper-parameter t: 0, 0.2, 0.4, 0.6, 0.8, and 1. As illustrated in Table 6.2, the best overall results are achieved by setting $t = 1$.

6.3.3 Experiments on Masked Face Matching

We show results of our DSA-Face models in Tables 6.3 and 6.4 on M2N, M2M, and MLFW protocols.

M2N refers to matching between masked faces and normal faces. Notice that either masked or normal faces are captured in real-world scenarios. As shown in Fig. 6.5, faces have dramatically different appearances under quality changes in (a) & (b), pose variations in (c) & (d), large age gaps in (e) & (f), and heavy occlusion in (g) & (h). Multiple challenging factors make this task very difficult. For a clear presentation, a tuple (a, b, c, d) is used where a, b, and c represent True Acceptance Rate (TAR) values on False Acceptance Rate (FAR) = 0.1, 0.01, and 0.001, respectively, and d denotes Rank-1 accuracy. Trained on global faces, ArcFace [31], CurricularFace [143], and Subcenter ArcFace [144] achieve (31.53%, 8.54%,

Table 6.2: Performance comparison (%) of different hyper-parameters: Varying number of local branches (B) and values of λ and β in Eqn. (6.13) and t in Eqn. (6.2).

B	λ	β	t	CFP-FP	CPLFW	MLFW	AFDB
2	0.10	0.05	1.0	96.63	90.82	82.60	14.29
4	0.10	0.05	1.0	96.81	91.25	77.48	16.56
6	0.10	0.05	1.0	97.27	91.57	81.97	20.26
8	0.10	0.05	1.0	97.11	91.48	82.52	17.17
6	**1.00**	0.05	1.0	96.53	91.10	76.71	16.50
6	**0.50**	0.05	1.0	97.01	91.22	82.05	17.33
6	**0.10**	0.05	1.0	97.27	91.57	81.97	20.26
6	**0.05**	0.05	1.0	97.34	91.50	76.73	18.35
6	**0.01**	0.05	1.0	97.19	91.75	80.79	18.41
6	0.10	**0.50**	1.0	96.93	91.65	78.82	16.35
6	0.10	**0.10**	1.0	96.96	91.57	82.60	17.22
6	0.10	**0.05**	1.0	97.27	91.57	81.97	20.26
6	0.10	**0.01**	1.0	96.99	91.07	76.94	17.48
6	0.10	**0.005**	1.0	96.57	91.07	79.16	15.37
6	0.10	0.05	**0.0**	97.19	91.28	78.51	17.22
6	0.10	0.05	**0.2**	96.77	91.37	80.22	18.87
6	0.10	0.05	**0.4**	96.99	91.15	81.85	17.94
6	0.10	0.05	**0.6**	96.76	91.35	77.62	17.33
6	0.10	0.05	**0.8**	96.63	91.33	78.57	17.79
6	0.10	0.05	**1.0**	97.27	91.57	81.97	20.26

Table 6.3: Performance comparison (%) of our DSA-Face models with several models on M2N and M2M protocols of masked face matching.

Method	M2N				M2M		
	FAR= 0.1	FAR= 0.01	FAR= 0.001	Rank-1	FAR= 0.1	FAR= 0.01	FAR= 0.001
ArcFace [31]	31.53	8.54	1.45	15.73	29.22	7.25	2.06
CurricularFace [143]	33.43	13.21	4.42	17.38	31.55	9.79	2.95
Subcenter Arcface [144]	24.60	4.51	0.56	12.49	28.95	8.57	3.15
HPDA [14]	57.41	30.94	14.98	34.76	58.99	37.68	**22.01**
DSA-Face[a]	48.04	21.53	8.09	22.67	47.96	21.47	7.93
DSA-Face[b]	62.09	36.59	18.55	43.08	**60.81**	**38.92**	19.58
DSA-Face[c]	48.28	21.60	7.42	23.91	51.05	25.54	11.16
DSA-Face[d]	**62.83**	**39.21**	**21.14**	**43.50**	59.28	38.14	17.08

1.45%, 15.73%), (33.43%, 13.21%, 4.42%, 17.38%), and (24.60%, 4.51%, 0.56%, 12.49%), respectively. As shown in Fig. 6.5, wearing masks makes some face clues invisible (i.e. noses and mouths) and dramatically changes global face appearances. However, since global faces are directly employed to train ArcFace, CurricularFace, and Subcenter ArcFace, noisy masks tend to be represented in features, resulting in sub-optimal representations. In contrast, some local patches may remain to be similar, suggesting their importance. Trained on the same backbone (i.e. ResNet-100) as ArcFace, CurricularFace, and Subcenter ArcFace, DSA-Face[a] boosts results to (48.04%, 21.53%, 8.09%, 22.67%), which shows the importance of our proposed modules. HPDA [14] has the capacity of locating diverse local patches, and thus greatly improves performances to (57.41%, 30.94%, 14.98%, 34.76%). However, it may miss some important regions due to the attention subset problem. Our DSA-Face model can alleviate this issue, learning diverse and comprehensive local features effectively. Specifically, compared with HPDA, DSA-Face[b] has the same training setting, but increases results to (62.09%, 36.59%, 18.55%, 43.08%). Besides, using LS-CNN as the stem CNN, DSA-Face[c] achieves better overall result of (48.28%, 21.60%, 7.42%, 23.91%) than DSA-Face[a]. Furthermore, we employ a more powerful LS-CNN-177 as the stem CNN which increases the depth of LS-CNN greatly, namely DSA-Face[d]. It achieves a result of (62.83%, 39.21%, 21.14%, 43.50%), which is even better.

Table 6.4: Performance comparison (%) of our DSA-Face models with several models on MLFW protocol of masked face matching.

Method	MLFW			
	FAR=0.1	FAR=0.01	FAR=0.001	Acc.
ArcFace [31]	44.92	12.85	1.81	69.25
CurricularFace [143]	59.24	23.45	3.62	74.61
Subcenter Arcface [144]	50.05	30.32	18.82	69.89
HPDA [14]	89.06	85.51	76.65	92.34
DSA-Face[a]	82.81	69.33	36.77	87.47
DSA-Face[b]	89.20	85.84	79.77	92.46
DSA-Face[c]	85.68	74.91	51.12	88.63
DSA-Face[d]	**89.74**	**86.72**	**82.25**	**92.91**

M2M refers to matching between masked and masked faces. A tuple (a, b, c) is used where a, b, and c represent TAR values on FAR= 0.1, 0.01, and 0.001, respectively. As shown in Table 6.3, ArcFace, CurricularFace, and Subcenter ArcFace models have results of (29.22%, 7.25%, 2.06%), (31.55%, 9.79%, 2.95%), and (28.95%, 8.57%, 3.15%), respectively. These models are trained on global faces. However, some parts in both gallery and probe sets are occluded to have global appearance changes. Consequently, large appearance gaps in train and test data lead to the inferior results. Differently, HPDA [14] automatically locates different parts, boosting results to (58.99%, 37.68%, 22.01%). DSA-Face models achieve promising results. First, compared with ArcFace, CurricularFace, and Subcenter ArcFace models, DSA-Face[a] achieves (47.96%, 21.47%, 7.93%); Second, compared with HPDA, DSA-Face[b] obtains a slightly better performance of (60.81%, 38.92%, 19.58%); Thirdly, both DSA-Face[a] and DSA-Face[c] have similar training settings while the former uses ResNet-100 and the latter employs LS-CNN. Under the more challenging M2M protocol, DSA-Face[c] boosts the result to (51.05%, 25.54%, 11.16%) compared to DSA-Face[a] which demonstrates the excellent generalization ability of LS-CNN, especially on masked face recognition; Fourthly, because of over-fitting potentials on non-masked faces, the result of DSA-Face[d] is slightly decreased to (59.28%, 38.14%, 17.08%).

MLFW denotes the protocol in the masked LFW dataset. A tuple (a, b, c, d) is used

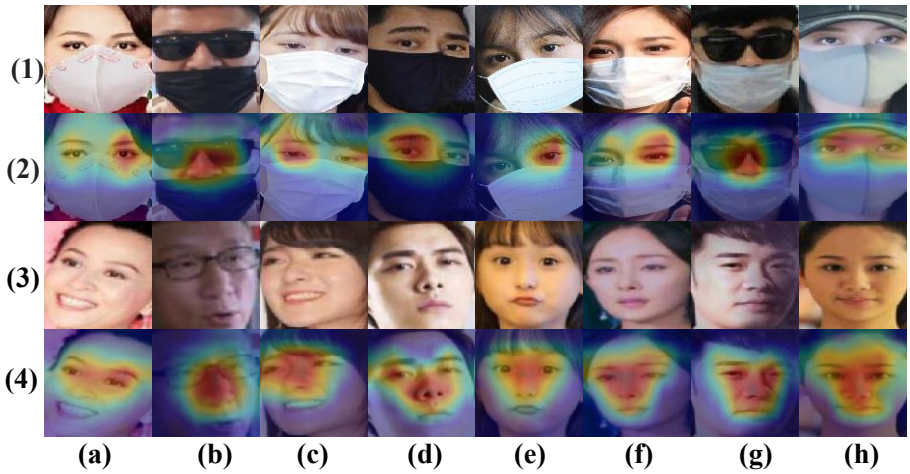

Figure 6.5: Qualitative illustration of the DSA-Face model [6] on masked face matching. **(1)** & **(3)**: Input faces with quality changes (**a** & **b**), pose variations (**c** & **d**), aging (**e** & **f**), and heavy occlusions (**g** & **h**). **(2)** & **(4)** show CAMs on these faces.

where a, b, and c represent TAR values on FAR= 0.1, 0.01, and 0.001, respectively, and d denotes accuracy. As illustrated in Table 6.4, although ArcFace[1] achieves 99.77% on original LFW, its performance is greatly reduced to (44.92%, 12.85%, 1.81%, 69.25%) on MLFW. Similarly, CurricularFace [143] obtain 99.80% accuracy on original LFW, its result is dropped to (59.24%, 23.45%, 3.62%, 74.61%) on MLFW. Subcenter ArcFace [144] achieves the performance of (50.05%, 30.32%, 18.82%, 69.89%). In contrast, HPDA [14] achieves 99.80% accuracy on the original LFW, while obtaining (89.06%, 85.51%, 76.65%, 92.34%) on MLFW. First, DSA-Face[a] achieves the performance of (82.81%, 69.33%, 36.77%, 87.47%), which significantly outperforms ArcFace, CurricularFace, and Subcenter ArcFace models. Second, DSA-Face[b] surpasses HPDA which is (89.20%, 85.84%, 79.77%, 92.46%). Third, compared with DSA-Face[a] which uses ResNet-100, DSA-Face[c] adopts LS-CNN as the stem CNN which increases the performance from (82.81%, 69.33%, 36.77%, 87.47%) to (85.68%, 74.91%, 51.12%, 88.63%). Last, DSA-Face[d] achieves the best performance of (89.74%, 86.72%, 82.25%, 92.91%).

There are two reasons to explain the above results. First, the PSCA ensures that diverse discriminative local patches are emphasized, as illustrated in Fig. 6.5 (4). This is especially useful for masked faces where other potentially discriminative parts can contribute to match-

[1]https://github.com/deepinsight/insightface/wiki/Model-Zoo

ing beyond occluded parts. As shown in Fig. 6.5 (2), eyes in (a), (h) and (c-f) are important if mouths and noses are invisible caused by masks, or noses in (b) & (g) can provide information when eyes and mouths are occluded due to sunglasses and masks. Second, the ASL suppresses responses around noisy regions, such as masks, hats, and sunglasses in Fig. 6.5 (2).

It can be observed that the attention mechanism usually concentrates on one single eye in Fig. 6.5 (2). There are 8 facial images in (2). Two eyes are occluded by sunglasses in (b) & (g); The illumination distribution on the face image in (e) is not balanced where the left eye tends to be more easily emphasized compared with the right eye. Differently, both eyes in (a), (c), and (h) are located where they are assigned with different weights due to their different importance in the recognition. It is also observed that the right eye in (d) and the left eye in (f) are not located. This may be attributed to the fact that these two eyes are very close to the background. Consequently, they are noisy by background information, not captured by the attention mechanism.

6.3.4 Experiments on Cross-Pose Face Matching

We conduct experiments on CFP-FP, CPLFW, and VGGFace2-FP datasets. A tuple (a, b, c, d) is used where a, b, and c represent accuracy on CFP-FP, CPLFW, and VGGFace2-FP datasets, respectively.

Most existing methods are trained on global faces. However, for profile faces or faces with large pose variations, they suffer from self-occlusions where only some facial parts are visible, like faces in Fig. 6.1 (4) (e). In such a case, if the model only focuses on few parts, it is very likely to predict the wrong identity when these parts are occluded. Therefore, it is necessary to extract diverse local representations from different facial parts. HPDA [14] achieved this by using the PCA, which achieves the accuracy of (-, 92.35%, 95.32%) where - denotes the result is not available. Compared with HPDA, our DSA-Face[b] has the same training setting, but increases the performance to (98.14%, 92.73%, 95.66%).

Besides, using the same backbone (i.e. ResNet-100) as those compared models, DSA-Face[a] obtains a higher overall accuracy of (98.53%, 93.70%, 95.96%). Using LS-CNN as

Table 6.5: Results (%) of our DSA-Face models and the state-of-the-art on CFP-FP, CPLFW, VGGFace2-FP, LFW, and IJB-C datasets.

Method	CFP-FP	CPLFW	VGGFace2-FP	LFW	IJB-C
MV-Softmax [138]	95.70	89.69	-	99.79	-
ArcFace [31]	98.37	92.08	-	99.83	95.60
MagFace [145]	98.46	92.87	79.28	99.83	95.97
HPDA [14]	-	92.35	95.32	99.80	-
DBM [140]	-	92.63	-	99.78	-
CurricularFace [143]	98.37	93.13	94.60	99.80	96.10
GroupFace [146]	98.63	93.17	-	99.85	**96.26**
BroadFace [147]	98.63	93.17	-	99.85	96.03
DSA-Face[a]	98.53	**93.70**	95.96	99.83	93.93
DSA-Face[b]	98.14	92.73	95.66	99.75	94.62
DSA-Face[c]	98.66	93.33	95.96	99.82	94.30
DSA-Face[d]	**98.69**	93.62	**96.24**	**99.85**	95.51

the backbone, DSA-Face[c] has similar results of (98.66%, 93.33%, 95.96%). Furthermore, DSA-Face[d] uses a more powerful network (i.e. LS-CNN-177) and achieves the highest accuracy of (98.69%, 93.62%, 96.24%). This is because our DSA-Face model is less likely to have the attention subset problem and effectively emphasizes diverse discriminative local patches. Specifically, the PSCA can focus on more diverse local patches and the ASL is able to suppress responses from useless facial parts or background.

As shown in Fig. 6.1 (4), diverse local patches are located for the frontal face in (d), including mouth, nose, and both eyes. However, due to dramatic pose variations in (e), only the right eye is visible, which may cause a wrong match since other parts are partly or totally invisible. If there is no supervision, the attention map misses the right eye in (f). In contrast, our PSCA successfully emphasizes the right eye and nose, as illustrated in (g). Furthermore, the ASL further constrains responses from useless regions, like cheeks as shown in (h).

Table 6.6: Performance comparison (%) of our DSA-Face models with several models on cross-quality face matching.

Method	FAR=0.01	FAR=0.001
Center loss [29]	52.5	31.3
SphereFace [36]	54.8	39.6
LS-CNN [5]	87.5	75.5
ArcFace [31]	68.6	65.7
HPDA [14]	87.6	80.3
DSA-Face[a]	87.3	80.4
DSA-Face[b]	89.5	83.2
DSA-Face[c]	**91.2**	**86.7**
DSA-Face[d]	90.5	85.3

6.3.5 Experiments on LFW and IJB-C Datasets

Notice that LFW dataset is a relatively easy dataset where most faces are frontal or nearly frontal. As shown in Table 6.5, in addition to its excellent performances on various challenging tasks, our DSA-Face[d] obtains the highest accuracy on LFW (99.85%) as well.

We also compare the TAR@FAR=0.0001 of different models on IJB-C dataset. In IJB-C dataset, there are 3,531 identities with 31.3K still images and 117.5K frames from 11,799 subjects, which contain 19,557 genuine pairs and 15,639K impostor matches. As demonstrated in Table 6.5, our models have slightly lower results on this challenging benchmark. Since video frames are used in this evaluation benchmark, blur variations are serious issues. Consequently, these blur variations tend to distract our DSA-Face models to locate useless or even noisy image regions. In future, we may further improve our models by incorporating different modules to handle the blur issue specifically.

6.3.6 Experiments on Cross-Quality Face Matching

Since pose variations and occlusions have important influence on the quality scores, we conduct experiments on cross-quality face matching task [65]. This task simulates some real-world applications where a low-quality face (Fig. 6.1 (2) (e)) is matched to a high-quality

face (Fig. 6.1 (2) (a)). Different methods are compared on IJB-A quality in Table 6.6. A tuple (a, b) is reported at FAR= 0.01 and 0.001.

Center loss [29], SphereFace [36], and ArcFace [31] designed loss functions, while they ignore discriminative local patches. In contrast, although it is only guided by Softmax loss, LS-CNN [5] emphasizes important local parts (Fig. 6.2 (b)), and thus obtains better results which are (87.5%, 75.5%). However, due to lacking of a proper guidance, the attention redundancy problem occurs (Fig. 6.2 (c)). HPDA [14] alleviates this problem by enlarging pairwise attention distances. Consequently, it locates diverse parts, and obtains the accuracy of (87.6%, 80.3%), which surpasses LS-CNN. However, the HPDA may have the attention subset problem (Fig. 6.2 (d)), missing some important local patches.

Our DSA-Face model can encourage the localization of diverse and comprehensive discriminative parts (Fig. 6.2 (e)). To compare performances fairly, we conduct experiments based on ResNet-100 and LS-CNN, respectively. First, using the same ResNet-100 as ArcFace [31], DSA-Face[a] significantly increases the performance to (87.3%, 80.4%). Second, compared with HPDA [14], DSA-Face[b] achieves a better accuracy, i.e. (89.5%, 83.2%) versus (87.6%, 80.3%). This is because, despite quality changes, as shown in Fig. 6.1 (2), discriminative parts are located in (c) and (g). Third, in cross-quality face matching, the blur factor plays an important role in dividing face images into low- or high-quality. Since there exists an imbalanced blur distribution in the training dataset, LS-CNN-177 tends to suffer from poor generalization ability on cross-quality face matching. This may explain why DSA-Face[c] and DSA-Face[d] use LS-CNN and LS-CNN-177 as the stem CNN. However, the former obtains the highest accuracy of (91.2%, 86.7%) and the latter has the result of (90.5%, 85.3%).

6.4 Summary

We have developed a DSA-Face model, which consists of novel pairwise self-contrastive attentions (PSCA) and an attention sparsity loss (ASL). First, the PSCA is capable of extracting diverse local representations by enlarging pairwise attention distances. Second, the ASL shrinks responses from noisy regions in attention maps towards zeros. Consequently,

only discriminative parts are emphasized, while noisy regions are discouraged. Besides, the masked face matching task has been executed to show the usefulness of the method, which could play an important role in FR under the spread of the COVID-19. Further, our DSA-Face model surpasses other methods on other challenging FR tasks as well, demonstrating the general FR performance of our model.

Chapter 7

AAN-Face: Attention Augmented Networks for Face Recognition

7.1 Motivations

Face recognition (FR) is a challenging task mainly because of two reasons: 1) Large intra-class variations. Faces in the same subject may have dramatically different appearances under different challenging factors, like occlusions, pose variations, quality changes, or aging. 2) Small inter-class differences. Faces from different subjects may be very similar, especially under the scenario with large-scale subjects. To alleviate these issues, many loss functions have been proposed to enhance features' discriminative ability [29] [36] [31]. Aside from delicatedly-designed loss functions, training data is one of the crucial factors that affects the performance. In general, training data should exhibit a balanced distribution among different classes. However, a common problem is that most existing training data inherently follows an imbalanced data distribution where a small number of classes are over-represented (e.g. frontal or non-occluded faces) and some of the remaining rarely appear (e.g. profile or heavily occluded faces). This is the reason why the models are biased toward the majority classes, while suffer from performance degradation in minority classes. For example, due to the outbreak of the COVID-19, the use of face masks has become ubiquitous. However, because training data exhibits limited masked face images, few FR models tend to be effective in many real-world applications, such as access control or security checks.

Figure 7.1: CAMs without/with our AAN-Face model on four positive face pairs. **Columns 1&4**: Positive face pairs which are affected by various challenging factors (e.g. occlusion, pose, makeup, and aging changes), leading to the landmark detection failure. **Columns 2&5**: CAMs without our AAN-Face. **Columns 3&6**: CAMs with our AAN-Face.

Some works design data augmentation methods via synthesizing faces [148] [149]. Different from these methods which aim at solving a specific variation problem (e.g. pose variations), our AAN-Face model can be applied to a wide range of variations. Most existing models extract holistic face representations, while failing to consider discriminative local representations. For example, as demonstrated in Fig. 7.1, although these faces have dramatic changes because of varying challenging factors (e.g. occlusion, pose, aging, or makeup variations), similar eyes (Row 1), noses (Rows 2&4), or lips (Rows 3&4) can still contribute to the verification of these face pairs.

To extract discriminative local features, existing methods can be split into two categories: landmark-based and attention-based. As for the former, different networks are trained on multiple facial parts which are cropped around facial landmarks [37] [38] [40]. However, landmark detection may fail in some cases. For instance, due to dramatic pose variations, extreme illuminations, large expression changes, or heavy occlusion levels, MTCNN [4] fails to detect landmarks for faces in Fig. 7.1. In contrast, without relying on facial landmarks, attention-based methods can locate discriminative facial parts automatically [5] [42] [2], as illustrated in Fig. 7.1, Columns 2&5. However, as pointed out in our previous work [14],

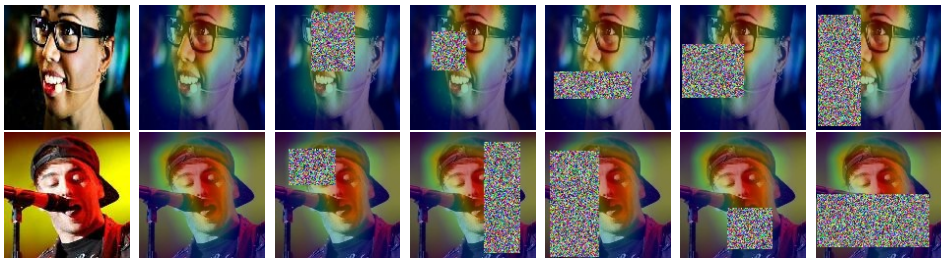

Figure 7.2: Examples generated by the AE. **Column 1**: Original faces. **Column 2**: Class Activation Maps (CAMs) [7]. **Columns 3-7**: CAMs with varying occlusion levels which are generated by the AE.

without a proper guidance, multiple attention modules tend to have redundant responses around the similar facial regions, while neglecting other discriminative facial parts. This may lead to performance degradation if these emphasized parts are occluded in some scenarios, such as large pose or occlusion changes. To address this issue, [14] proposes a diverse learning to locate diverse local patches in different attention maps. Consequently, other local patches can contribute besides the occluded ones. However, this approach can only alleviate this issue to some extent because training samples still exhibit limited occlusion variations. Therefore, it is unavoidable that results would drop for faces under large pose or occlusion changes, like masked faces.

To overcome this issue, we propose an augmented attention network, called AAN-Face. First, when some parts of a face are occluded, it is expected that models are still able to recognize the identity from non-occluded facial parts. To achieve this goal, an attention erasing (AE) scheme is proposed to randomly erase units in attention maps. As a result, augmented faces with various occlusion levels can be generated, which well prepares models towards pose or occlusion changes. Fig. 7.2 shows some examples which are generated by the AE where erased masked have different sizes and locations. As illustrated in Fig. 7.6, compared with the model without AE (Row 2), the model with AE is more robust to occlusions or large pose variations (Row 3). Second, an attention center loss (ACL) is proposed to learn a center for each attention map, and thus force the same attention map to locate the same facial part for each facial image. This is a strong signal to control each attention map to emphasize important facial regions and suppress noisy ones. As shown in Fig. 7.5 Row 1 Columns 3&6, more useful facial parts are emphasized when the

ACL is used, compared with the model without ACL in Row 1 Columns 2&5. Differently, distracted information are suppressed in Row 2 Columns 3&6, compared with Row 2 Columns 2&5. Third, the ACL is combined with the AE to form the AAN-Face. Because the AE removes some facial parts which are located by attention maps, models are encouraged to learn different attention centers. Consequently, diverse facial parts can be located, allowing models to focus on non-occluded facial parts beyond occluded ones. As illustrated in Fig. 7.1, some less useful information (e.g. cheeks) or even noisy features are located if without AAN-Face model, as demonstrated in Columns 2&5. In contrast, when AAN-Face model is used, discriminative facial regions are more accurately located and emphasized, while noisy facial parts are inhibited, as shown in Columns 3&6.

Our contributions can be summarized as follows:

1. An attention erasing (AE) scheme randomly erases units in attention maps, which well prepares models for occlusion or pose variations.

2. An attention center loss (ACL) is proposed to learn a center for each attention map, encouraging each attention map to emphasize informative facial regions and inhibit the distracted.

3. Incorporating the AE with the ACL, our AAN-Face model can generate attention maps to accurately locate diverse facial parts, and thus extract complementary local information.

4. The proposed model outperforms the state-of-the-art on different challenging face matching tasks, especially on masked face matching which is especially important due to the ongoing pandemic of the COVID-19.

7.2 Proposed Method

In this section, details of the proposed approach are discussed. The overview structure is illustrated in Fig. 7.3. It mainly consists of four parts: stem CNN, local CNNs, global CNN, and classification, which are discussed as follows.

There are several models used in the stem CNN. First, HSNet-61 model [5] is selected as the default stem CNN if unspecified which can extract multi-scale features from two harmonious perspectives: utilization of multi-scale convolutional kernels in a single layer

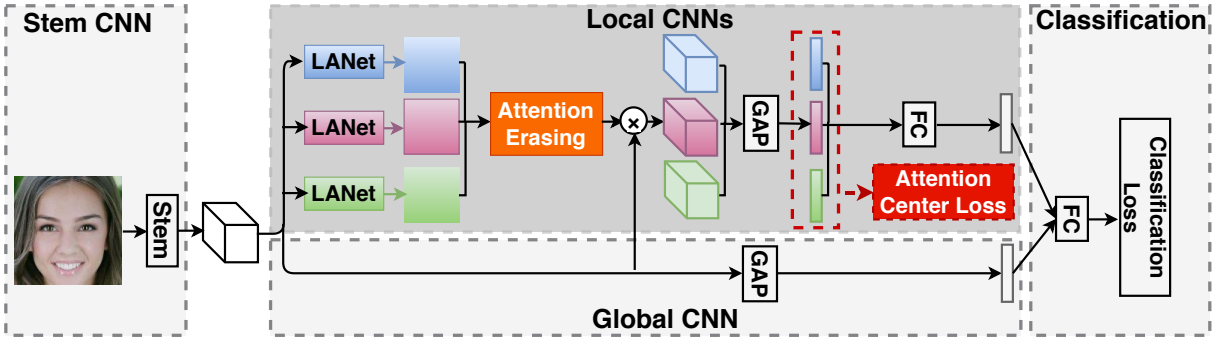

Figure 7.3: Overall framework of the proposed AAN-Face model. The attention erasing (AE) makes the model robust to pose and occlusion changes. The attention center loss (ACL) guides the same attention map to focus on the same facial part, emphasizing informative facial parts and suppressing distracted. GAP and FC mean global average pooling and fully-connected layers, respectively.

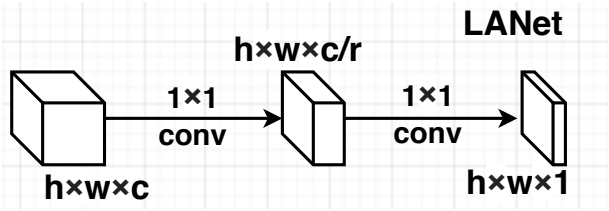

Figure 7.4: Framework of LANet [5]. h, w, and c refer to the height, width, and number of feature maps, respectively. r represents the reduction ratio.

and concatenation of feature maps from different layers. Second, to compare fairly with existing the state-of-the-art models, IR-100 [31] is used as the backbone. Third, to further boost the performance, LS-CNN and LS-CNN-177 models are employed in [5] where the latter is a more powerful network with 12, 12, and 16 repeated operations in three stages.

Local CNNs aim at learning local facial representations. The global CNN focuses on holistic face features. Both of them have 512 units. Details are discussed in the following subsections.

In the classification stage, a joint local and global feature representation is obtained by concatenating outputs of the local and global CNNs, followed by a fully connected (FC) layer with 512 units and a classification layer.

7.2.1 Local CNNs and Global CNN

Local CNNs

Instead of cropping discriminative parts around facial landmarks, we use an attention module (i.e. LANet [5] shown in Fig. 7.4) to automatically locate discriminative parts. It is trained in an end-to-end way, allowing for locating useful facial regions flexibly. It is expected that more important facial regions are assigned with higher weights, while less informative ones have smaller weights.

Let $X \in R^{\mathrm{h}\times\mathrm{w}\times\mathrm{c}}$ denote the input of local CNNs where h, w, and c refer to height, width, and number of feature maps, respectively. First, an attention map $M_{\mathrm{i}} \in R^{\mathrm{h}\times\mathrm{w}}$ is generated by the LANet in the i^{th} local branch. It represents informative facial parts, like eyes or noses. As illustrated in Fig. 7.4, the first layer has $\frac{c}{r}$ feature maps, followed by a ReLU layer to increase the non-linearity. The second layer generates M_{i} with a Sigmoid function. Suppose b refer to the number of local branches. Since each branch generates an attention map, there are b attention maps which would be used for data augmentation.

Second, the refined feature map R_{i} for the i^{th} branch is aggregated by product of the attention map M_{i} and the input X:

$$R_{\mathrm{i}} = X \circ M_{\mathrm{i}}, \tag{7.1}$$

where $\circ$ denotes the element-wise multiplication operation.

Third, the local facial representation P_{i} for the i^{th} local branch is computed by a Global Average Pooling (GAP) layer:

$$P_{\mathrm{i}} = GAP_{\mathrm{i}}(R_{\mathrm{i}}). \tag{7.2}$$

There are b local facial representations (i.e. $P_1, P_2, ..., P_{\mathrm{b}}$), followed by a FC layer with 512 units.

Global CNN

The global CNN is used to extract the holistic face representation G, which takes X as the input:

$$G = GAP_{\mathrm{g}}(X), \tag{7.3}$$

where $\mathrm{GAP_g}$ refers to the GAP layer in the global CNN. Following a FC layer, the holistic face representation (i.e. G) has 512 units.

Attention Erasing

Notice that the AE is only conducted in the training stage with a probability p. It erases units within a rectangle area of a size S which is selected from attention maps $M \in R^{\mathrm{h\times w\times b}}$.

The AE is conducted on feature maps with the following steps. First, a rectangle area of size S_e is selected within attention maps M, where size ratio $\frac{S_e}{h\times w} \in [s_l, s_h]$ and aspect ratio $a \in [a_l, a_h]$. Consequently, if height h_e is set to $\sqrt{S_e \times a}$, then width w_e is $\sqrt{\frac{S_e}{a}}$. Second, a point (x_e, y_e) is randomly initialized in M. If the point meets the criterion: $x_e + w_e \leq w$ and $y_e + h_e \leq h$, then units in the rectangle $(x_e, y_e, x_e + w_e, y_e + h_e)$ are replaced with random values in range $[0, 1]$. Otherwise, the above procedures are repeated until an appropriate area is selected.

Since some facial parts are removed, models would pay more attention to non-occluded parts and extract other potentially discriminative local representations. In this way, the AE forces models to represent the face image in a global perspective, rather than focusing on few facial parts which may be unavailable due to occlusions, pose, or viewpoint variations. This well prepares the model for various real-world occlusions.

Attention Center Loss

It is expected that the same local branch focuses on the same facial part for every facial image, making models to emphasize important facial parts and suppress useless or noisy ones. To this aim, an attention center loss (ACL) is proposed to effectively regularize the attention learning process.

Specifically, an attention center is learned for each attention map in local CNNs. In the training stage, attention center A_i from the i^{th} local branch is updated where the distance between A_i and the local representation P_i (See Eqn. (7.2)) is minimized. The following formulation can be used:

$$L_{\mathrm{ACL}} = \sum_{i=1}^{b} ||P_i - A_i||_2^2. \tag{7.4}$$

The A_i is initially set to 0, and updated at each iteration by moving average:

$$A_i^{t+1} = A_i^t + \beta(P_i^t - A_i^t), \tag{7.5}$$

where β controls the update from iteration t to iteration $t+1$.

By penalizing the variances between the attention center and the corresponding local representation, the L_{ACL} encourages each local branch to locate the same semantic facial part of the input face image. Consequently, only discriminative facial parts are emphasized and less informative or noisy ones are suppressed.

Overall Loss

In order to supervise the network to learn discriminative features, the classification loss L_{Cls} should be used, including Softmax loss ($L_{\text{Cls-softmax}}$), CosFace loss [30] ($L_{\text{Cls-cosface}}$), and ArcFace loss [31] ($L_{\text{Cls-arcface}}$) are used. Compared with Softmax loss, the CosFace and ArcFace loss can generate more discriminative features but need a longer training time.

Softmax loss is defined as follows:

$$L_{\text{Cls-softmax}} = -\frac{1}{N}\sum_{i=1}^{N}\frac{e^{W_{y_i}^T x_i} + b_{y_i}}{\sum_{j=1}^{K} e^{W_j^T x_i + b_j}}, \tag{7.6}$$

where N and K denote the number of samples and classes, respectively. $x_i \in R^d$ refers the i^{th} deep feature, belonging to the y_i class. d is the feature dimension. W_j and b_j are the weights and bias term in the last FC layer connecting to the j^{th} class.

CosFace loss is formulated as follows:

$$L_{\text{Cls-cosface}} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j\neq y_i} e^{s\cos\theta_{j,i}}}, \tag{7.7}$$

where m is the cosine margin to maximize the decision margin in the angular space. The sample x_i is normalized and re-scaled to s.

The additive angular margin penalty m is used in ArcFace loss to encourage the intra-class compactness and inter-class discrepancy.

$$L_{\text{Cls-arcface}} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(\cos(\theta_{y_i,i}+m))}}{e^{s(\cos(\theta_{y_i,i}+m))} + \sum_{j\neq y_i} e^{s\cos\theta_{j,i}}}. \tag{7.8}$$

The L_{Cls} is incorporated with L_{ACL} to form the overall loss as follows:

$$L = L_{\text{Cls}} + \lambda L_{\text{ACL}}, \tag{7.9}$$

where λ controls the trade-off in these two losses.

The AAN-Face Model

In local CNNs, ensemble concatenates representations from multiple local branches. The key to success is the excellent performance of the individual local branch as well as outstanding complementarity among multiple local branches. The classification loss L_{Cls} ensures the excellent performance of individual local branch. L_{ACL} controls the same attention map to focus on the same semantic facial part for every facial image. For example, the i^{th} attention map emphasizes eyes for every face image. However, if without a proper guidance, multiple local branches may have the similar attention centers, and thus focus on similar facial parts, while missing potentially discriminative ones. To enhance the diversity among local branches, the AE is used before the ACL, aiming to differentiating the feature embeddings among different branches. This is achieved by removing some facial parts in the training process, which results in the localization of non-removed facial parts. Consequently, diverse complementary facial parts are emphasized to boost the performance.

7.3 Experiments

7.3.1 Implementation Details

During training on VGGFace2 data, Softmax loss is employed. The batch-size is 330. Learning rate starts at 0.1 and is multiplied by 0.1 at the 5^{th} and 9^{th} epochs, respectively. Training stops at the 10^{th} epoch. During training on MS1MV2 data, there are several different settings: First, if IR-100 (**AAN-Face***) is adopted as the stem CNN, ArcFace loss in Eqn. (7.8) is adopted where $s = 32$ and $m = 0.35$. The batch-size is 512. The learning rate is 0.1 initially and is multiplied by 0.1 at the 13^{th}, 19^{th}, and 23^{nd} epochs, respectively. Training stops at the 24^{th} epoch; Second, **AAN-Face**† has the same epoch setting as HPDA [14]; Third, when LS-CNN (**AAN-Face**‡) or LS-CNN-177 (**AAN-Face**$^{\circ}$)

Table 7.1: Effectiveness of the proposed components and their combinations.

Global CNN	Local CNNs	ACL	AE	LFW	CPLFW	CALFW
√				97.85	76.06	84.73
√	√			98.02	77.63	85.92
√	√	√		98.13	78.33	86.18
√	√		√	98.06	77.73	86.87
√	√	√	√	**98.27**	**78.72**	**87.03**

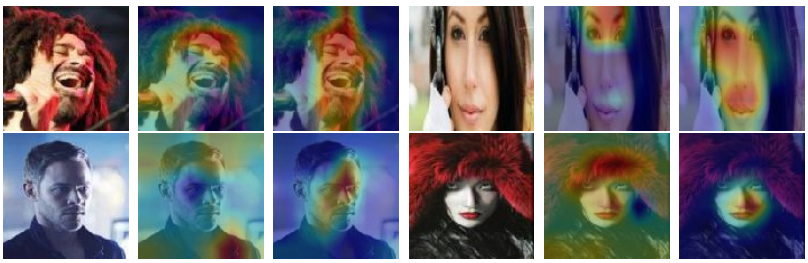

Figure 7.5: CAMs without/with the ACL. **Columns 1&4**: Four faces with various challenging factors. **Columns 2&5**: CAMs without the ACL. **Columns 3&6**: CAMs with the ACL.

are used as backbone, CosFace loss in Eqn. (8.8) is used where scale s and margin m are set to 32 and 0.5, respectively. The batch-size is set to 256. The number of warm-up epochs is 2. The learning rate starts at 0.03 and is multiplied by 0.1 at the 13^{th} and 19^{th} epochs, respectively. It is set to $1e-4$ at the 23^{rd} epoch. Totally there are 24 training epochs.

The number of local branches b is 6 after conducting comparative analysis. The AE is performed with a probability $p = 0.5$. The aspect ratio a and upper bound area ratio s_h in the AE is 0.3 and 0.4, respectively. β in Eqn. (7.5) is set to 0.05. λ in Eqn. (7.9) is experimentally set to 1.

7.3.2 Ablation Study

Several experiments are performed to investigate the proposed model on the LFW, CPLFW, and CALFW datasets. Tables 7.1 and 7.5 give detailed investigations.

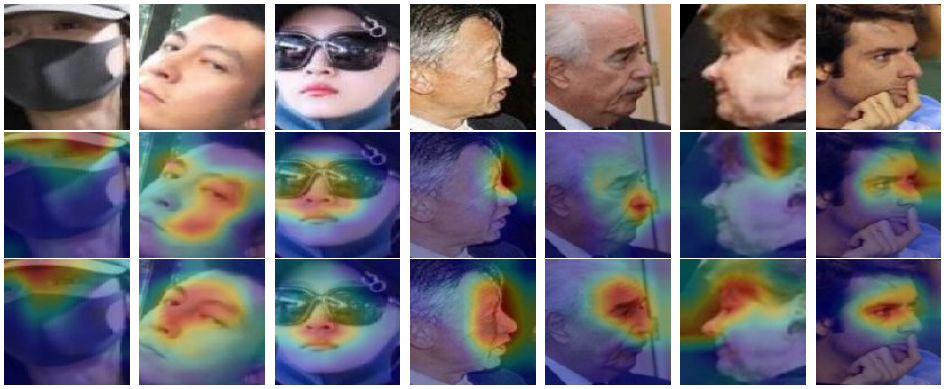

Figure 7.6: CAMs without/with the AE. **Row 1**: Faces with occlusions or large pose variations. **Row 2**: CAMs without the AE. **Row 3**: CAMs with the AE.

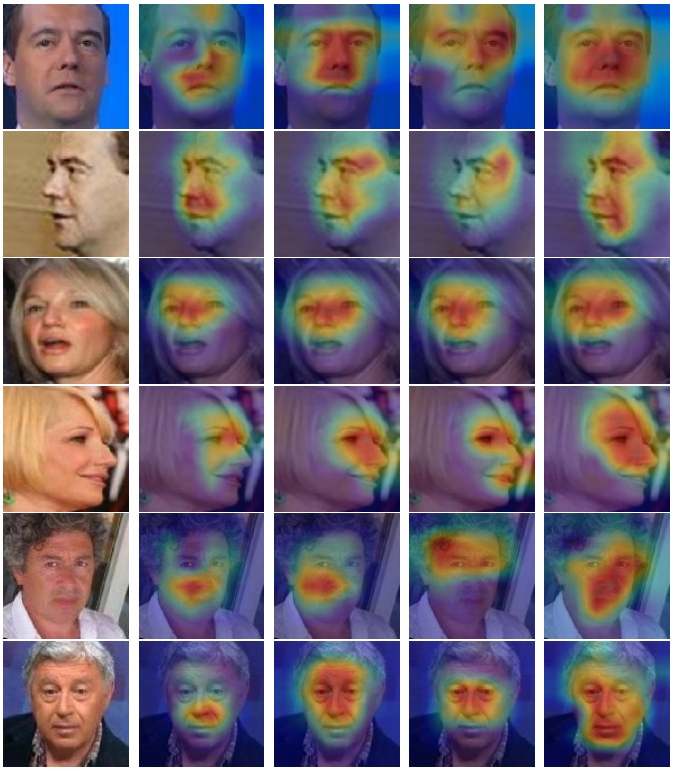

Figure 7.7: Class Activation Maps (CAMs) [7] on three positive pairs with pose variations (Rows 1, 2, 3&4) and age gaps (Rows 5&6). **Column 1**: Original faces. **Column 2**: CAMs with local CNNs. **Column 3**: CAMs with ACL. **Column 4**: CAMs with AE. **Column 5**: CAMs with ACL and AE.

Table 7.2: Performance comparison (%) on Pitch 0, different Yaw of M^{2}FPA dataset [10].

Method	Pitch 0, different Yaw					
	15	30	45	60	75	90
w/o AE	**99.92**	99.68	98.99	97.33	91.63	52.37
w/ AE	99.89	**99.79**	**99.31**	**97.68**	**92.66**	**53.92**

Table 7.3: Performance comparison (%) on Pitch +15, different Yaw of M^{2}FPA dataset [10].

Method	Pitch +15, different Yaw						
	0	15	30	45	60	75	90
w/o AE	**99.89**	99.87	99.89	99.31	96.22	85.66	45.12
w/ AE	99.79	**100.0**	**99.97**	**99.36**	**96.46**	**85.93**	**45.79**

Contributions of proposed components

As discussed above, our AAN-Face mainly consists of two components: the AE and the ACL. We perform comparative experiments in Table 7.1 to show their effectiveness.

First, the performance is slightly increased on test datasets by adding local CNNs, which rises from 97.85% to 98.02% on LFW, from 76.06% to 77.63% on CPLFW, and from 84.73% to 85.92% on CALFW, respectively. The outcome can be attributed to the fact that local CNNs automatically select informative regions which would play important roles in FR. Since local facial representations are complementary with the global face description, a better performance is achieved.

Second, it should be noticed that without a proper guidance, there exist two issues: 1) Attention maps may miss some important facial parts, as illustrated in Fig. 7.5 Row 1 Columns 2&5; 2) Attention maps may take useless or even noisy regions into consideration, as shown in Fig. 7.5 Row 2 Columns 2&5. To address this challenge, the ACL is proposed to control the i^{th} attention map in the i^{th} local branch to focus on the same facial part for every facial image. Because the ACL pulls the deep embeddings of the same local branch to the same attention centers, this is a very strong signal to control the responses of attention maps on facial images, which can address the aforementioned two issues. First, the randomness between different attention maps are reduced, encouraging models to locate

Table 7.4: Performance comparison (%) on Pitch +30, different Yaw of M^{2}FPA dataset [10].

Method	Pitch +30, different Yaw				
	0	22.5	45	67.5	90
w/o AE	99.63	**99.23**	96.46	81.05	37.95
w/ AE	**99.89**	99.17	**96.83**	**81.78**	**39.07**

discriminative facial parts. As illustrated in Fig. 7.5 Row 1, more rich important facial parts are emphasized in Columns 3&6, compared with Columns 2&5. Second, the same attention map tends to locate the same semantic facial parts. In such a way, only useful facial parts are emphasized, while noisy ones are suppressed, as illustrated in Fig. 7.5 Row 2 Columns 3&6, compared with Row 2 Columns 2&5. important facial parts tend to be highlighted, while less informative regions are inevitably suppressed. This explains why 0.11% increase on LFW, 0.70% increase on CPLFW, and 0.26% increase on CALFW are obtained, respectively. The ACL can guide each attention map to locate the same semantic facial parts.

Third, the AE can generate occlusions in feature maps. Consequently, models are encouraged to learn useful representations from non-occluded facial regions, resulting multiple diverse local representations. Therefore, accuracy improvements occur when the AE is applied, which are from 98.02% to 98.06% on LFW and from 85.92% to 86.87% on CALFW, respectively. Besides, training data would exhibit rich occlusion changes, which well prepares models to pose variations. Consequently, the accuracy on CPLFW is boosted from 77.63% to 77.73%. As show in Fig. 7.6, the model with AE (Row 3) is more robust to occlusions or large pose variations than the case of without AE (Row 2). Besides, M^{2}FPA [10] is also evaluated. It is the most comprehensive multi-view face database for facial pose analysis which has 397,544 images of 229 subjects with different yaw, pitch, attribute, illumination, and accessory variations. As shown in Tables 7.2, 7.3, and 7.4, the model with AE achieves the better performances than the model without AE. More specifically, it is observed that the larger the yaw angles, the lower the performances are obtained. The AE enables models to well prepare various pose variations. This explains why the performance improvement is larger if the AE is used when the yaw angle increases, compared with the model w/o the AE.

Fourth, the ACL encourages the same attention map to focus on the same facial part for every facial image. Although supervised by the ACL, the overlap of different attention maps is reduced. However, it is still possible that different attention maps focus on the same salient, discriminative features because there is no explicit mechanism to repel different attention centers scattered. Consequently, they tend to ignore other potentially discriminative regions.

To alleviate this challenge, the AE is conducted before performing the ACL, as demonstrated in Fig. 7.3, where each attention map is multiplied with the same erasing mask. As a consequence, if some discriminative facial parts are removed, models then turn to other discriminative regions, reducing the risk of redundant responses around similar facial regions among different local branches. Different attention centers which aim at different facial parts are learned. Therefore, multiple local branches are more likely to focus on different facial parts. We demonstrate better performance is achieved by utilizing the ACL and AE simultaneously, which is 98.27% on LFW and 87.03% on CALFW, respectively. Besides, when the AE is applied, training data would exhibit rich occlusion changes, which well prepares models towards pose variations. Therefore, the performance on CPLFW is boosted from 78.33% to 78.72%. As illustrated in Fig. 7.7, compared with ACL (Column 3) and AE (Column 4), putting ACL and AE together in Column 5 can obtain more comprehensive discriminative local patches. This demonstrates the excellent complementarity between the AE and ACL.

Varying number of local branches b

The number of local branches b is compared in Table 7.5. The performance is improved from 98.05% to 98.27% on LFW and from 78.58% to 78.72% on CPLFW when b is increased from 3 to 6. This outcome can be attributed to the fact that more discriminative facial regions are located by different local branches. But accuracy drops slightly when b is changed to 9, which is 98.08%, 77.85%, and 86.85% on LFW, CPLFW, and CALFW, respectively. This is because useless or noisy information are extracted in some branches when b is too large.

Table 7.5: Performance comparison (%) of varying number of local branches (**b**), probability (**p**) of conducting the AE, **AE locations**, **AE types**, and **ACL types**.

b	p	AE Locations	AE Types	ACL Types	LFW	CPLFW	CALFW
3	0.50	AE-L	AE-R	ACL-F	98.05	78.58	**87.06**
6	0.50	AE-L	AE-R	ACL-F	**98.27**	**78.72**	87.03
9	0.50	AE-L	AE-R	ACL-F	98.08	77.85	86.85
6	**0.25**	AE-L	AE-R	ACL-F	**98.38**	78.28	86.78
6	**0.50**	AE-L	AE-R	ACL-F	98.27	**78.72**	**87.03**
6	**0.75**	AE-L	AE-R	ACL-F	98.22	76.88	86.95
6	0.50	**AE-N**	-	ACL-F	98.13	78.33	86.18
6	0.50	**AE-I**	AE-R	ACL-F	97.78	78.58	85.87
6	0.50	**AE-G**	AE-R	ACL-F	98.20	77.75	86.95
6	0.50	**AE-GL**	AE-R	ACL-F	98.23	78.05	**87.37**
6	0.50	**AE-L**	AE-R	ACL-F	**98.27**	**78.72**	87.03
6	0.50	AE-L	**-**	ACL-F	98.06	77.73	86.87
6	0.50	AE-L	**AE-0**	ACL-F	98.00	77.53	86.43
6	0.50	AE-L	**AE-M**	ACL-F	**98.38**	78.28	86.68
6	0.50	AE-L	**AE-R**	ACL-F	98.27	**78.72**	**87.03**
6	0.50	AE-L	AE-R	**ACL-M**	98.06	78.72	86.72
6	0.50	AE-L	AE-R	**ACL-FS**	98.10	76.22	85.95
6	0.50	AE-L	AE-R	**ACL-F**	**98.27**	**78.72**	**87.03**

Different values of probability p to conduct the AE

Different probability values p (0.25, 0.5, and 0.75) to conduct the AE operation are compared in Table 7.5. When p is increased from 0.25 to 0.5, the performance is boosted to from 78.28% to 78.72% on CPLFW and from 86.78% to 87.03 % on CALFW. However, if p is increased to 0.75, the performance is slighted dropped to 76.88% on CPLFW and 86.95% on CALFW.

Different locations of the AE

Varying locations of the AE are compared in Table 7.5: **AE-N** represents no AE is used; **AE-I**, **AE-G**, **AE-L**, and **AE-GL** denote the AE is used in the image, global, local, and both global and local branches, respectively.

On one hand, because the AE-L is applied on high-level attention maps where one unit corresponds to one facial part, discriminative facial parts tend to be removed. As a result, the AAN-Face model would explore other potentially discriminative facial parts. On the other hand, since the AE-I is applied on images, it is more likely to erase less informative facial parts. As a consequence, models are less likely to explore other discriminative parts and only focus on few facial parts. Therefore, AE-L can better encourage models to learn diverse discriminative local representations, which achieves 98.27% on LFW, 78.72% on CPLFW, and 87.03 on CALFW, respectively, compared with 97.78%, 78.58%, and 85.87% of the AE-I.

For FR, if the model only focuses on few facial parts, it is very likely to predict the wrong identity when these parts are occluded because of pose and viewpoint variations. Therefore, it is crucial to extract diverse local features from different facial parts. When the AE is used in local CNNs, some facial regions located by attention maps in local branches are erased, encouraging models to extract potentially discriminative features from non-erased facial regions. This explains why AE-L (98.27%, 78.72%, 87.03%) has a better performance than AE-N (98.13%, 78.33%, 86.18%), and AE-GL (98.23%, 78.05%, 87.37%) obtains a higher accuracy than AE-G (98.20%, 77.75%, 87.35%).

Besides, in the AE-G, units are erased within selected facial regions in feature maps of the global CNN, which makes models robust to various occlusions. Meanwhile, in the AE-L,

the ACL cooperates with the AE to focus on diverse local information. In this way, the AE-L not only better prepares models with various occlusion levels, but also pushes models to generate diverse discriminative local representations beyond occluded facial regions. This explains why the AE-L has a better performance than the AE-G, which is 98.27% to 98.20% on LFW, 78.72% to 77.75% on CPLFW, and 87.03% to 86.85% on CALFW, respectively.

Different types of random values in the AE

Three types of random values are used in the AE to erase units in the selected rectangle region: 1) Units are assigned with 0, denoted as AE-0; 2) Units are erased with mean values 0.5, referred to AE-M; 3) Units are replaced with random values ranging in $[0, 1]$, represented as AE-R. As shown in Table 7.5, we notice that AE-M and AE-R outperform the baseline; AE-R and AE-M have slightly better performance than AE-0; AE-R obtains a higher accuracy on AE-M. If not specified, AE-R is the default setting.

Table 7.6: The settings of methods in Table 5.3. Specifically, Center loss uses web-collected training data, including CASIA-WebFace [11], CACD2000 [12], and Celebrity+ [13]. After removing the images with identities appearing in testing datasets, it roughly goes to 0.7M images of 17,189 unique persons.

Method	Backbone	Dataset	Batch size	Image size
Center loss [29]	ResNet34	Combined	256	112×96
SphereFace [36]	ResNet64	WebFace	128	112×96
LS-CNN [5]	LS-CNN	VGGFace2	256	128
MV-Softmax [138]	AttentionNet	MS-v1c-R	128	144
ArcFace [31]	ResNet100	MS1MV2	512	112
DDL [150]	ResNet100	VGGFace2	1536	112
HPDA [14]	LS-CNN	MS1MV2	256	112
DBM [140]	ResNet50	MS1MV2	-	112
AAN-Face*	ResNet100	MS1MV2	512	112
AAN-Face†	LS-CNN	MS1MV2	256	112
AAN-Face‡	LS-CNN	MS1MV2	256	112
AAN-Face$^{\circ}$	LS-CNN-177	MS1MV2	256	112

Table 7.7: The settings of methods in Table 5.3. Specifically, Center loss uses web-collected training data, including CASIA-WebFace [11], CACD2000 [12], and Celebrity+ [13]. After removing the images with identities appearing in testing datasets, it roughly goes to 0.7M images of 17,189 unique persons.

Method	Learning rate	Epochs or iters
Center loss [29]	$0.1^{(1+floor(iters/[16K,24K]))}$	28K
SphereFace [36]	$0.1^{(1+floor(iters/[16K,24K]))}$	28K
LS-CNN [5]	$0.1^{(1+floor(epoch/[5,9]))}$	10
MV-Softmax [138]	$0.1^{(1+floor(iters/[4,8,10]))}$	12
ArcFace [31]	$0.1^{(1+floor(iters/[100K,160K]))}$	180K
DDL [150]	$0.1^{(3+floor(iters/10K))}$	20K
HPDA [14]	$3 * 0.1^{(2+floor(epoch/[7,10]))}$ 2 warm-up epochs; 1e-4 if epoch=11	12
DBM [140]	$0.1^{(1+floor(epoch/[5,8,11]))}$	15
AAN-Face*	$0.1^{(1+floor(epoch/[13,19,23]))}$	24
AAN-Face†	$3 * 0.1^{(2+floor(epoch/[7,10]))}$ 2 warm-up epochs; 1e-4 if epoch=11	12
AAN-Face‡	$3 * 0.1^{(2+floor(epoch/[13,21]))}$ 2 warm-up epochs; 1e-4 if epoch=23	24
AAN-Face$^{\circ}$	$3 * 0.1^{(2+floor(epoch/[13,21]))}$ 2 warm-up epochs; 1e-4 if epoch=23	24

Varying types of the ACL

The ACL is used to encourage the same local branch to focus on the same facial part. When the ACL is used with the AE, diverse facial regions can be automatically learned. Three types of ACL are evaluated in Table 7.5. **ACL-M** and **ACL-F** denote that the ACL is applied on attention maps and feature embeddings of local branches, respectively. Notice that each subject learns its corresponding attention centers in both ACL-M and ACL-F. We also compare the **ACL-FS** where feature embeddings of attention centers are shared among different subjects.

ACL-M penalizes the variances of locations about responded facial areas and their centers. Consequently, it is easily affected by location changes of facial regions, especially scenarios under heavy occlusions or large pose variations. In contrast, ACL-F is proposed to efficiently

Table 7.8: Performance comparison (%) on cross-quality and cross-pose face matching.

Method	IJB-A quality		CPLFW	CFP-FP	VGGFace2-FP
	FAR=0.01	FAR=0.001			
Center loss [29]	52.5	31.3	77.48	-	75.10
SphereFace [36]	54.8	39.6	81.40	-	20.10
LS-CNN [5]	87.5	75.5	88.03	97.17	69.92
MV-Softmax [138]	-	-	89.69	95.70	-
ArcFace [31]	68.6	65.7	92.08	98.37	46.20
DDL [150]	-	-	93.43	98.53	-
HPDA [14]	87.6	80.3	92.35	-	95.32
DBM [140]	-	-	92.63	-	-
AAN-Face*	**90.6**	**85.9**	93.25	98.39	95.72
AAN-Face†	89.2	82.6	92.68	97.94	95.62
AAN-Face‡	89.4	82.9	92.97	98.26	95.68
AAN-Face$^{\circ}$	90.0	84.5	**93.57**	**98.63**	**95.82**

pull the deep embeddings of the same semantic facial parts to their centers. Since embeddings in ACL-F are outputs of the GAP operation, ACL-F is encouraged to be more robust to changes of facial parts locations. This is the reason why ACL-F has a better performance than ACL-M (98.27% to 98.06% on LFW and 87.03% to 86.72% on CALFW).

Different from ACL-FS which has shared attention centers among different subjects, ACL-F has its attention centers for each subject. In such a way, each subject has the ability to flexibly locate its discriminative facial regions. This allows ACL-F models to learn informative features flexibly than ACL-FS models. Therefore, ACL-F achieves 98.27% on LFW, 78.72% on CPLFW, and 87.03% on CALFW, exceeding ACL-FS which obtains 98.10%, 76.22%, and 85.95%, respectively.

7.3.3 Experiments on Cross-Quality Face Matching

This task matches a low-quality face with a high-quality face, which is very common in real-world applications. Different methods are compared on IJBA-quality dataset, as shown in Table 7.8. A tuple (a, b) is used to efficiently represent verification results when FAR=0.01

Table 7.9: Performance comparison (%) on cross-age and general face matching.

Method	CALFW	AgeDB-30	LFW
Center loss [29]	85.48	-	99.13
SphereFace [36]	90.30	-	99.42
LS-CNN [5]	92.00	-	99.52
MV-Softmax [138]	95.63	98.11	99.79
ArcFace [31]	95.45	**98.15**	99.83
DDL [150]	-	-	99.68
HPDA [14]	95.90	-	99.80
DBM [140]	96.08	97.90	99.78
AAN-Face*	96.03	97.95	99.85
AAN-Face†	96.00	97.83	99.80
AAN-Face‡	95.97	98.10	99.82
AAN-Face$^{\circ}$	**96.13**	**98.15**	**99.87**

and 0.001, respectively.

Several models design advanced loss functions, like Center loss [29], SphereFace [36], and ArcFace [31]. However, they employ global faces to extract representations, which may suffer from performance degradation due to dramatic quality changes of global face images. In contrast, LS-CNN [5] designs a spatial attention module to weigh important parts and suppress noisy ones, obtaining a higher performance, i.e. (87.5%, 75.5%).

By extracting rich local representations, HPDA [14] achieves a result of (87.6%, 80.3%). Despite with the same training setting as the HPDA, the proposed AAN-Face† outperforms the HPDA, obtaining a result of (89.2%, 82.6%). Similarly, as illustrated in Tables 7.6 and 7.7, both ArcFace and AAN-Face* train ResNet-100 on the same MS1MV2 dataset. However, the latter achieves the state-of-the-art result, i.e. (90.6%, 85.9%) as shown in Table 5.3. This is because our AAN-Face model is robust to occlusions or pose changes which have significant influences on face image quality values. AAN-Face‡ doubles training epochs, achieving a performance of (89.4%, 82.9%). Employing a more powerful network (LS-CNN-177), AAN-Face$^{\circ}$ achieves a competitive performance, i.e. (90.0%, 84.5%).

7.3.4 Experiments on Cross-Pose Face Matching

We compare the AAN-Face model with state-of-the-art methods in Table 7.8 where a tuple (a, b, c) is used to represent results on CPLFW, CFP-FP, and VGGFace2-FP datasets. VGGFace2-FP results of Center loss, SphereFace, and ArcFace models are from [139].

Firstly, AAN-Face* is trained on ResNet-100 and achieves a competitive result of (93.25%, 98.39%, 95.72%). Secondly, experimental results demonstrate that our AAN-Face† model achieves a higher overall performance, which is (92.68%, 97.94%, 95.62%). This is because AAN-Face can well prepare models with pose variations by generating various occlusions levels using the AE. For instances, as demonstrated in Fig. 7.2, different discriminative facial parts are erased, simulating various levels of pose variations. Furthermore, when the AE is incorporated with the ACL, the AAN-Face model can extract rich diverse local representations. If some facial parts are invisible in profile faces, representations from other facial parts can contribute. Take faces in Fig. 7.1 as examples, although eyes in Row 2, Columns 1&4 and Row 3, Column 4 are occluded, the similar noses in Row 2 and similar lips in Row 3 located by AAN-Face model contribute the verification. Thirdly, AAN-Face‡ further increases the performance to (92.97%, 98.26%, 95.68%). Besides, AAN-Face$^\circ$ achieves the state-of-the-art accuracy, which is (93.57%, 98.63%, 95.82%).

7.3.5 Experiments on Cross-Age Face Matching

We compare the results of AAN-Face model with the state-of-the-art approaches on CALFW and AgeDB-30 in Table 5.3.

It demonstrates that our approach surpasses the state-of-the-art. Although faces with large age gaps have remarkable appearance changes, some facial regions remain similar which play important roles in cross-age face matching. For example, faces in Fig. 7.1 Row 4 have dramatic appearance changes, but we can still match them by the similar noses and mouths. Guided by the AE and ACL, AAN-Face model can extract multiple local representations. This is the reason why AAN-Face$^\circ$ model has a higher accuracy than other models, achieving 96.13% on CALFW and 98.15% on AgeDB-30, respectively.

Table 7.10: Performance comparison (%) on cross-modality face matching.

Method	CASIA NIR-VIS 2.0			
	FAR=0.01	FAR=0.001	FAR=0.0001	Rank-1
IDR [151]	98.90	95.70	-	97.30
WCNN [152]	99.40	97.60	-	98.40
MFR [153]	99.32	95.97	81.92	96.92
PCFH [154]	99.60	97.70	-	98.80
PACH [155]	99.60	98.30	-	98.90
HPDA [14]	99.47	98.28	94.56	98.68
AAN-Face*	89.98	75.10	55.90	81.11
AAN-Face†	**99.67**	98.94	96.29	99.18
AAN-Face‡	99.40	98.00	93.54	98.54
AAN-Face$^{\circ}$	99.65	**99.06**	**97.06**	**99.27**

7.3.6 Experiments on LFW Dataset

Because most faces in LFW dataset are frontal or near-frontal, this a relatively simple task. Although our AAN-Face model is designed to solve many challenging face matching tasks, it still obtains the highest accuracy on the LFW.

As demonstrated in Table 7.9, AAN-Face* and AAN-Face† achieve 99.85% and 99.80% accuracy. Although it has the same stem CNN as AAN-Face†, AAN-Face‡ doubles the training epochs, leading to a better convergence and obtaining 99.82% accuracy. Furthermore, since AAN-Face$^{\circ}$ uses a powerful stem CNN (LS-CNN-177), which obtains a 99.87% accuracy. Since there are 6 mislabeled pairs in the verification protocol, the upper bound of accuracy is 99.90%. These results show a excellent generalization ability of the proposed model.

7.3.7 Experiments on Cross-Modality Face Matching

We compare different methods on the CASIA NIR-VIS 2.0 database in Table 7.10. In order to show the results efficiently, we use a tuple (a, b, c, d) to represent TAR results when FAR=0.01, 0.001, 0.0001, and Rank-1 accuracy, respectively.

Both IDR [151] and WCNN [152] are pre-trained on MS-Celeb-1M dataset and fine-tuned on the NIR-VIS domain, which obtain (98.90%, 95.70%, -, 97.30%) and (99.40%, 97.60%, -, 98.40%), respectively, where - represents that the result is unavailable. Without training on NIR face images, MFR [153] meta-leans transferable knowledge across domains to improve generalization, achieving a competitive performance (99.32%, 95.97%, 81.92%, 96.92%). PCFH [154] and PACH [155] solve the heterogeneous face recognition problem via face hallucination, which achieves (99.60%, 97.70%, -, 98.80%) and (99.60%, 98.30%, -, 98.90%), respectively.

Although our previous HPDA [14] does not use any NIR images during training or learns transferable knowledge in different domains, it still achieves great performance of (99.47%, 98.28%, 94.56%, 98.68%). We attribute this to the fact that different local representations are extracted to match faces at different modalities. With the same training settings as HPDA [14], AAN-Face† improves the performance to (99.67%, 98.94%, 96.29%, 99.18%). This again shows the better localization ability of diverse local patches in our AAN-Face model, despite of the modality discrepancy problem. Meanwhile, compared with AAN-Face†, AAN-Face‡ doubles the training epochs, but obtains an inferior performance of (99.40%, 98.00%, 93.54%, 98.54%). This may be explained by the fact that AAN-Face‡ tends to be over-fitting on the VIS domain, while limiting its generalization capacity of extracting discriminative information on NIR images. Using a more powerful stem CNN (i.e. LS-CNN-177), AAN-Face$^{\circ}$ achieves the overall best performance, i.e. (99.65%, 99.06%, 97.06%, 99.27%). This is achieved without fine-tuning on NIR face images, which again demonstrates the strong generalization ability of our AAN-Face model.

7.3.8 Experiments on Masked Face Matching

The AAN-Face is compared with different models in Tables 7.11 and 7.12 on several challenging protocols about masked faces, including M2N, M2M, and MLFW. The publicly available ArcFace [31] and CurricularFace [143] models are evaluated. Besides, our previous HPDA [14] is also compared. In order to compare results fairly, AAN-Face* and AAN-Face† are used. The former is trained on ResNet-100 which also is used in ArcFace and

Figure 7.8: Qualitative illustration of the AAN-Face model [8] on the masked face matching. **Rows 1&3** Input faces with quality changes (**Column 1**), pose variations (**Column 2**), aging (**Column 3**), almost frontal views (**Column 4**), and heavy occlusions (**Columns 5, 6&7**). **Rows 2&4** show CAMs generated by our AAN-Face model.

CurricularFace, and the latter is trained on LS-CNN which is employed in HPDA.

M2N refers to matching faces between masked and normal faces in RMFRD. Notice that either masked or normal faces are captured under the real-world scenarios. As shown in Fig. 7.8, faces have dramatically different appearances, like low-quality faces (Column 1), profile faces (Column 2), faces with large age gaps (Column 3), frontal faces (Column 4), and heavily occluded faces (Columns 5, 6, &7). Therefore, M2N is a very challenging task. It contains a verification and an identification protocols. To show the results efficiently, a tuple (a, b, c, d) is used where a, b, and c represent TAR values on FAR=0.1, 0.01, and 0.001, respectively, and d denotes the the Rank-1 accuracy. ArcFace [31] and CurricularFace [143] obtain performances of (31.53%, 8.54%, 1.45%, 15.73%) and (33.43%, 13.21%, 4.42%, 17.38%), respectively. Wearing face masks makes some parts of a face occluded (i.e. noses and mouths). Since global faces are directly employed to train ArcFace and CurricularFace models, distracted masks tend to be represented, resulting in inferior representations. In masked face matching, it is expected that models can recognize the identity from non-occluded parts. To this aim, HPDA [14] can extract diverse local representations, so non-occluded facial parts can contribute beyond the occluded ones, boosting the performance to

Table 7.11: Performance comparison (%) of our AAN-Face models with several models on M2N and M2M protocols of masked face matching.

Method	M2N				M2M		
	FAR=0.1	FAR=0.01	FAR=0.001	Rank-1	FAR=0.1	FAR=0.01	FAR=0.001
ArcFace [31]	31.53	8.54	1.45	15.73	29.22	7.25	2.06
CurricularFace [143]	33.43	13.21	4.42	17.38	31.55	9.79	2.95
HPDA [14]	57.41	30.94	14.98	34.76	58.99	37.68	**22.01**
AAN-Face*	45.62	19.56	7.49	23.65	47.41	21.14	7.27
AAN-Face†	61.28	34.74	17.18	40.57	60.31	38.18	19.79
AAN-Face‡	61.24	36.36	**18.87**	**42.31**	**60.59**	**38.23**	**20.13**
AAN-Face$^{\circ}$	**62.11**	**37.18**	18.71	41.54	59.39	35.95	16.54

Table 7.12: Performance comparison (%) of our AAN-Face models with several models on MLFW protocol of masked face matching.

Method	MLFW			
	FAR=0.1	FAR=0.01	FAR=0.001	Acc.
ArcFace [31]	44.92	12.85	1.81	69.25
CurricularFace [143]	59.24	23.45	3.62	74.61
HPDA [14]	89.06	85.51	76.65	92.34
AAN-Face*	80.87	66.19	29.26	86.10
AAN-Face†	89.44	86.22	80.71	92.58
AAN-Face‡	**89.77**	86.28	83.13	92.65
AAN-Face$^{\circ}$	89.67	**86.95**	**84.17**	**92.91**

(57.41%, 30.94%, 14.98%, 34.76%).

With the same backbone as ArcFace and CurricularFace, AAN-Face* boosts the result to (45.62%, 19.56%, 7.49%, 23.65%). Compared with HPDA which is trained on LS-CNN [5], AAN-Face† uses the same backbone, but boosts the performance to (61.28%, 34.74%, 17.18%, 40.57%). There are two reasons that can explain the results. First, the AE simulates occlusions, which well prepares the model for masked faces. Second, the ACL controls the centers of attention maps, extracting diverse local representations, making it possible that non-occluded parts contribute beyond the masked parts. Compared with AAN-Face†, AAN-Face‡ doubles the training epochs and has a better convergence, improving the performance

to (61.24%, 36.36%, 18.87%, 42.31%). Using the LS-CNN-177, AAN-Face$^{\circ}$ achieves 62.11%, 37.18% on FAR=0.1, 0.01, respectively, but drops slightly to 18.71% on FAR=0.001 and rank-1 accuracy of 41.54%.

M2M refers to the matching between masked faces and masked faces. A tuple (a, b, c) is used where a, b, and c represent the TAR values on FAR=0.1, 0.01, and 0.001, respectively. As demonstrated in Table 7.11, ArcFace and CurricularFace have performances of (29.22%, 7.25%, 2.06%) and (31.55%, 9.79%, 2.95%). On one hand, both ArcFace and CurricularFace are trained on global face images. On the other hand, some facial parts in either gallery or probe faces in testing are occluded. The large gap in train and testing data leads to the inferior performance. Differently, HPDA [14] automatically locates diverse facial parts, boosting the performance to (58.99%, 37.68%, 22.01%). Furthermore, due to its well preparation towards occlusions and excellent ability to extract diverse local representations, AAN-Face* and AAN-Face† achieve performances of (47.41%, 21.14%, 7.27%) and (60.31%, 38.18%, 19.79%), respectively. AAN-Face‡ further increases the results to (60.59%, 38.23%, 20.13%). However, the accuracy of AAN-Face$^{\circ}$ is slightly decreased to (59.39%, 35.95%, 16.54%). This may because AAN-Face$^{\circ}$ is over-fitted on non-occluded facial images, leading to a poor generalization ability.

The MLFW denotes a verification protocol on masked LFW dataset. A tuple (a, b, c, d) is used where a, b, and c represent TAR values on FAR=0.1, 0.01, and 0.001, respectively, and d denotes the accuracy. Although the publicly available ResNet100 based ArcFace dataset[1] and CurricularFace models achieve 99.77% and 99.80% on the original LFW, their performance is dramatically reduced to (44.92%, 12.85%, 1.81%, 69.25%) and (59.24%, 23.45%, 3.62%, 74.61%) on MLFW, as illustrated in Table 7.12. In contrast, AAN-Face* obtains a performance of (80.87%, 66.19%, 29.26%, 86.10%). Both HPDA [14] and AAN-Face† obtain 99.80% accuracy on the original LFW. Due to the localization of multiple local representations in HPDA [14], it obtains (89.06%, 85.51%, 76.65%, 92.34%). AAN-Face† further increases the performance to (89.44%, 86.22%, 80.71%, 92.58%). This demonstrates the importance and effectiveness of the local representations in masked face recognition. In comparison, AAN-Face‡ yields (89.77%, 86.28%, 83.13%, 92.65%). Lastly, AAN-Face$^{\circ}$ achieves the best overall

[1]https://github.com/deepinsight/insightface/wiki/Model-Zoo

performance of (89.67%, 86.95%, 84.17%, 92.91%)

7.4 Summary

We have presented a new model called AAN-Face for face recognition, which may be the first attempt to augment the training images for various face recognition tasks. The proposed attention erasing (AE) scheme in applied in attention maps to simulate various occlusion levels, making models robust to pose or occlusion changes. The proposed attention center loss (ACL) can control the same attention map to focus on the same facial parts, capturing critical local patches and ignoring the uninformative. When the ACL is fused with the AE scheme, models are encouraged to learn diverse local representations. Experimental results on several challenging tasks, especially on masked face datasets, have demonstrated that our new approach outperforms the state-of-the-art models.

Chapter 8

CQA-Face: Contrastive Quality-aware Attentions for Face Recognition

8.1 Motivations

Face recognition (FR) has many practical applications. Previous works can be divided into two categories: global-based approaches and local-based methods. The former learns features on global face images [31, 140]. However, they rarely consider representations on local patches to improve discrimination of face features. Three positive pairs are shown in Fig. 8.2 (a)&(f), the holistic faces change dramatically by blur changes in (1), pose variations in (2), and age gaps in (3). However, some local patches remain similar which can help the verification. For example, the similar eyes in (1) or the similar noes in (2)&(3).

There are mainly two groups of methods to extract discriminative local representations: landmark-based and attention-based methods. Landmark-based works detected face landmarks first and then extract local features on cropped regions centered around landmarks [37,38,42]. However, these methods largely depend on accuracy of landmark detection and may even suffer from detection failure under dramatic pose variations or heavy occlusions. For example, due to the ongoing outbreak of the COVID-19, people are wearing face masks. Because some facial parts (e.g. noses or mouths) are invisible, landmark detection may be inaccurate or failed. Besides, even if face landmarks are detected, the cropped parts may include masks which would inevitably deteriorate the extracted face features. Without

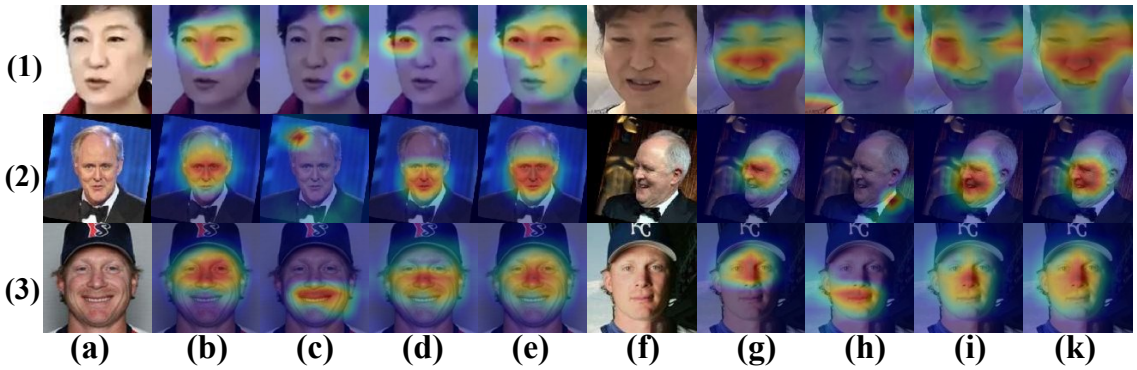

Figure 8.1: Effects of the CQA-Face model on three positive pairs. **(a)**&**(f)**: Input faces are affected by blur changes **(1)**, pose variations **(2)**, and aging **(3)**. **(b-d)**&**(g-i)**: 1st, 2nd, and 3rd local branches of local CNNs where different local patches are located in each local branch. **(e)**&**(k)**: Discriminative local patches are emphasized in local CNNs. For a good illustration, only three local branches are used to show class activation maps (CAMs) [3].

relying on landmarks, some models employ attention modules to automatically locate useful parts [2, 5]. However, no mechanism is designed to select the local patches effectively, which may miss some important facial parts, and thus limit the performance.

FR is challenging mainly because of two reasons: 1) High intra-class variations. Faces from the same subject may appear at different poses or occlusion levels. In such a case, if only few facial parts are located, the extracted features may not be sufficient when these parts are invisible under occlusions (e.g. face masks) or pose variations. 2) Small inter-class differences. Faces from different subjects may have similar local appearances, especially considering a large number of subjects. The performance would decline if the few emphasized facial parts across different subjects look similar. To alleviate the above issues, it is necessary to capture local representations that are as rich as possible. In such a way, more potentially useful facial parts can contribute to FR if the few emphasized parts are invisible or remain similar across different subjects. To achieve this goal, a contrastive attention learning (CAL) module is devised to encourage the diversity among different attention maps. Consequently, it is ensured that varying local patches are well-explored across face images and diverse discriminative facial parts are emphasized.

Local representations can provide discriminative information for FR. However, concatenating local representations directly without considering relations between different facial

parts may not be optimal, as discussed in previous paragraph. To address the limitations of separated facial parts, the structural correlation of facial parts is built to boost the discriminative ability of local representations. Besides, it is observed that different facial parts have various quality under occlusion, blur, or pose variations, as illustrated in Fig. 8.2 (b-d)&(g-i). The performance would be deteriorated if concatenating these local features directly. To address these two issues, we devise a quality-aware network (QAN). It introduces a part-level quality-aware module to explore the relation between an individual facial part and the rest parts by using graph convolutional networks. In such a manner, it has two benefits. First, each local part itself and its relations with other parts are considered simultaneously. This encourages each part-level feature to utilize information from other parts, making them more discriminative. Second, it can estimate the quality of located facial parts, emphasizing informative parts and suppressing noisy ones.

By putting the CAL and QAN together, a CQA-Face model is developed to learn rich quality-aware local representations. Our main contributions are three-fold:

1. A contrastive attention learning (CAL) is designed to ensure the localization of comprehensive facial parts, especially the less discriminative but still useful facial parts;

2. A region-level quality-aware network (QAN) is proposed to generate quality scores for each facial part by exploring its relation with the rest, emphasizing important facial parts and suppressing noisy ones;

3. By combining the CAL and QAN, the CQA-Face model is developed, which outperforms the state-of-the-art methods on many challenging datasets.

8.2 Proposed Method

The overall framework of our approach is shown in Fig. 8.2, mainly composed of the stem CNN, Contrastive Attention Learning (CAL), and Quality-Aware Networks (QAN). The stem CNN extracts high-level feature maps. Because HSNet-61 model [5] has an excellent generalization ability, it is used as the default stem CNN.

If without a proper signal, multiple attention maps tend to focus on few facial parts, while neglecting other important parts. Motivated by this observation, the CAL is devised

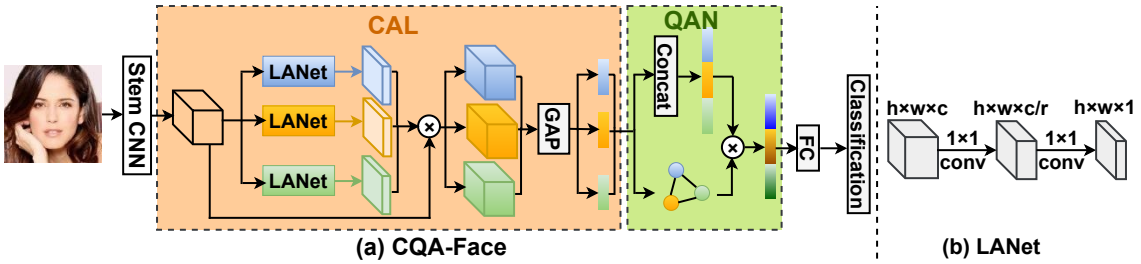

Figure 8.2: **(a) CQA-Face**: The CAL pushes models to explore comprehensive local representations. The QAN learns a quality score for each local patch in a global scope. GAP and FC refer to the global average pooling and fully-connected layers, respectively. Notice that only three local branches are used for illustration. **(b) LANet**: The spatial attention in [5] is used where h, w, and c is the height, width, and number of channels. r is the reduction ratio.

to guide multiple local branches to extract diverse local representations based on high-level feature maps.

However, since comprehensive image details are explored thoroughly across an image, it is possible that some distracted regions (e.g. face masks) may also be represented. To alleviate this, the QAN is proposed to learn quality scores among different local patches from a global scope. Finally, diverse quality-aware local representations are concatenated for face matching. Details are illustrated as follows.

8.2.1 Contrastive Attention Learning

Existing attention-based methods do not perform localization of less discriminative but still useful ones. If models focus on few facial parts, they would suffer from performance drop if facial parts are invisible or remain similar across different subjects. To address this, a contrastive attention learning (CAL) mechanism is proposed to encourage diversity in multiple attention maps. Consequently, learned models can retrieve comprehensive local patches across face images.

Suppose that $X \in R^{\mathrm{h\times w\times c}}$ denote the input of the CAL, where h, w, and c refer to the height, width, and number of channels, respectively. Next, it is expected to learn diverse attention maps in different local branches, i.e. $[M_1, M_2, ..., M_{\mathrm{b}}]$, where b is the number of local branches. As shown in Fig. 8.2 (b), the spatial attention (i.e. LANet) in [5] is used

to weigh different regions, assigning high weights to important parts and small weights to useless ones. There are two convolutional layers in LANet in which the first layer uses the ReLU function and the second layer adopts the Sigmoid function.

For the i^{th} local branch, the output O_i is calculated by the product of the attention map M_i and input X as follows:

$$O_i = X \circ M_i, \tag{8.1}$$

where $\circ$ denotes Hadamard product.

However, if without a proper guidance, multiple attention maps $[O_1, O_2, ..., O_{\mathrm{b}}]$ tend to have similar responses around the same facial parts, limiting the representational ability.

To address this issue, we propose a contrastive attention learning (CAL) using a divergence loss as follows:

$$L_{\mathrm{CAL}} = \frac{2}{b(b-1)} \sum_{i=1}^{b} \sum_{j=i+1}^{b} max(0, -t + exp^{\text{-}(\mathrm{M_i}\text{-}\mathrm{M_j})^2/\sigma}), \tag{8.2}$$

where M_{i} and M_{j} mean attention maps in the i^{th} and j^{th} local branches, respectively. t is a hyper-parameter margin. σ is a positive value to control the shape of the Gaussian function.

The loss L_{CAL} measures the correlation between M_{i} and M_{j}. When M_{i} and M_{j} are similar, L_{CAL} tends to be large, pushing M_{i} and M_{j} to be dissimilar. In such a way, M_{i} and M_{j} would have responses around different facial parts. Comprehensive facial parts are well-explored across face images. Consequently, models can rely on more useful parts if some parts are invisible for faces with occlusions or pose variations.

After that, a global average pooling (GAP) layer is used to pool feature maps $[O_1, O_2, ..., O_{\mathrm{b}}]$ in different local branches, generating local representations in each local branch $[P_1, P_2, ..., P_{\mathrm{b}}]$.

It should be mentioned that the diverse learning proposed in [14] can also extract diverse local patches. The divergence loss is defined as follows:

$$L_{\mathrm{D}} = \frac{2}{b(b-1)} \sum_{i=1}^{b} \sum_{j=i+1}^{b} max(0, t - dist(M_i, M_j)), \tag{8.3}$$

where $dist(M_{\mathrm{i}}, M_{\mathrm{j}})$ is the Euclidean distance between M_{i} and M_{j}. As a result, varying attention maps are encouraged to be different from each other, and thus focus on different facial parts. Compared with L_{D} [14], our proposed L_{CAL} is more flexible to measure distances of two attention maps. Experimental results also verify the superiority of our L_{CAL}.

8.2.2 Quality-Aware Networks

Local patches appear at facial images can be divided into three categories: Few important local patches (e.g. the nose, mouth, both eyes, or mouth) which can significantly contribute to FR; Some unimportant regions, like the cheek or forehead; Distracted background or occlusions which may be represented. This is because these parts are close to important parts (e.g. eyes) in profile or masked faces. It is highly desired that discriminative parts should be emphasized and distracted regions should be suppressed.

The attention module automatically determines which regions should be highlighted. However, few existing approaches design a mechanism to refine attention responses. Consequently, some distracted regions may be extracted, such as noisy background or face masks, which deteriorate the performance. This issue is especially serious when the CAL is used, because every face detail is explored to extract comprehensive features. For example, for masked face matching, because face masks are very close to discriminative parts, noisy information tends to be represented.

Therefore, there are two issues: Local patches may exhibit varying quality scores; It is necessary to infer the relations between one facial part and others because the relations provide important clues to refine attention maps. To alleviate these issues, a quality-aware network (QAN) is designed. Specifically, it is implemented by a graph convolutional network where relationships between different local patches are captured and quality scores are learned simultaneously.

Since each local branch generates a local patch, the number of local patches is b. Assume that $\mathcal{G}(\mathcal{V}, \xi)$ is the constructed graph with b nodes where $\mathcal{V} = [P_1, P_2, ..., P_{\mathrm{b}}]$ and the edge ξ models the relation between two patches. The adjacent matrix $A \in R^{\mathrm{b}\times\mathrm{b}}$ represents pairwise relations of local patches.

Inspired by the work [156], pairwise relations between every two local patches are represented as follows:

$$e(P_{\mathrm{i}}, P_{\mathrm{j}}) = \phi(P_{\mathrm{i}})^{\top}\varphi(P_{\mathrm{j}}), \tag{8.4}$$

where ϕ and φ represent two different projects of local features. More specifically, $\phi = W^{\phi}P_{\mathrm{i}}$ and $\varphi = W^{\varphi}P_{\mathrm{j}}$ where $W^{\phi}, W^{\varphi} \in R^{\,\mathrm{c/r}\times\mathrm{c}}$. r is a dimension reduction ratio to reduce the num-

ber of parameters. W^{ϕ} and W^{φ} are implemented by two 1×1 convolutional layers followed by batch normalization (BN) and ReLU operations. In such a manner, the transformations allow models to learn correlations between different local patches adaptively.

The element $A_{i,j}$ in adjacency matrix A represents the relation between P_i and P_j. For the i^{th} feature node, pairwise relations with all the nodes are stacked in a fixed order to obtain a relation vector $A_i = [A(i,:), A(:,i)] \in R^{2b}$. Since the relations are stacked into a vector with a fixed order, spatial information is also represented in the relation vector A_i.

Each P_i represents original local features, while A_i denotes the structural relations. They complement and reinforce each other but in different embedding spaces. Therefore, we combine them in their respective embedding space and jointly learn an importance S_i of feature node P_i by the following formulation:

$$S_i = \theta([\mu(P_i), \nu(A_i)]), \tag{8.5}$$

where μ and ν represent projection functions for P_i and A_i, respectively. Specifically, they consist of a spatial 1×1 convolutional layer followed by BN and ReLU operations. Then, an embedding function is conducted to mine rich information from them for inferring quality scores through two 1×1 convolutional layers.

For all nodes, we have a quality score list $S = [S_1, S_2, ..., S_b]$. We normalize the quality scores by the Sigmoid function to obtain normalized quality scores $\hat{S} = [\hat{S}_1, \hat{S}_2, ..., \hat{S}_b]$ as follows:

$$\hat{S}_i = \frac{1}{1 + e^{-\mathrm{S_i}}}. \tag{8.6}$$

Then, each local representation is multiplied by its corresponding quality score as follows:

$$\hat{P}_i = \hat{S}_i \cdot P_i. \tag{8.7}$$

For an image, we first concatenate local representations $[\hat{P}_1, \hat{P}_2, ..., \hat{P}_b]$ from b local branches, which are followed by a FC layer with 512 units before the classification layer.

8.2.3 Overall Loss

L_{Cls} is the CosFace loss [30], guiding models to learn discriminative features. CosFace loss is formulated as follows:

$$L_{\text{Cls}} = -\frac{1}{N}\sum_{i=1}^{N} log\frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j\neq y_i} e^{s\cos\theta_{j,i}}}, \tag{8.8}$$

where N represents the number of samples. m is the cosine margin to maximize the decision margin in the angular space. The sample x_i is normalized and re-scaled to s, belonging to the y_i class.

The overall loss is defined as follows:

$$L_{\text{Overall}} = L_{\text{Cls}} + \lambda L_{\text{CAL}}, \tag{8.9}$$

where L_{CAL} encourages multiple attention maps to locate diverse facial parts, learning comprehensive local representations. λ is a coefficient to control the balance in these two losses.

8.3 Experiments

8.3.1 Implementation Details

CosFace loss [30] is used. The number of warm-up epochs is 2. The batch-size is set to 256. During training on VGGFace2, learning rate starts at 0.03 and is divided by 10 at the 7^{th} and 10^{th} epochs, respectively. The learning rate is set to 1e-4 at the 12^{th} epoch. Training stops at the 12^{th} epoch. During training on MS1MV2, learning rate is 0.03 and is divided by 10 at the 13^{th} and 19^{th} epochs, respectively. It is set to 1e-4 at the 23^{rd} epoch. Training stops at the 24^{th} epoch. The ResNet-100 used as the stem CNN.

After comparative experiments, the number of local branches (b) is set to 4. The σ and t in Eqn. (8.2) are set to 0.01 and 0.2, respectively. The λ in Eqn. (8.9) is 0.5.

8.3.2 Ablation Study

In ablation study, experiments are conducted, as shown in Tables 8.1 and 8.2 where a tuple (-, -, -) refers to results on LFW, CALFW, and CPLFW, respectively.

Table 8.1: Effectiveness of different modules. L_D refers to the diverse learning in [14].

CAL	QAN	L_D	LFW	CALFW	CPLFW
			99.35	92.55	89.55
√			99.47	92.90	89.85
	√	√	99.33	93.20	89.82
√	√		**99.52**	**93.37**	**90.20**

Effects of the proposed components

The proposed CQA-Face model mainly consists of two components: the CAL and QAN. Their performances are shown in Table 8.1.

Without an explicit guidance, multiple attention maps tend to focus on only few facial parts and miss other important regions. As shown in Fig. 8.2 (b)&(g), attention-based methods can learn some important face parts automatically. However, since there is not an explicit signal, some important regions are ignored. To alleviate this problem, the CAL is proposed to locate diverse local patches by encouraging distances between each two attention maps. For face pairs from the same subject, it is possible that some local patches are distinctive across large age gaps. This is the reason why the accuracy is boosted from 92.55% to 92.90% on CALFW. For profile faces, some facial parts are occluded due to viewpoint changes. This would deteriorate the performance if only few facial parts are located. In contrast, the CAL can generate diverse local patches. Consequently, other important regions can help the matching besides occluded ones, leading to a moderate improvement on CPLFW which is from 89.55% to 89.85%. Since rich local representations are mined across face images, the performance on LFW is slightly increased from 99.35% to 99.47%. Besides, we also qualitatively show localization ability of diverse local patches in Fig. 8.2 (b-d)&(g-i). Guided by the CAL, diverse discriminative local patches are located.

While the CAL generates diverse local patches by exploring every image detail, this may raise two questions: Noisy information may be extracted, like face masks; Different facial parts may appear at various quality under occlusion, blur, or illumination changes. For instances, since background is very close to discriminative parts for profile faces, noisy information tends to be represented. The QAN alleviates these issues by learning a quality

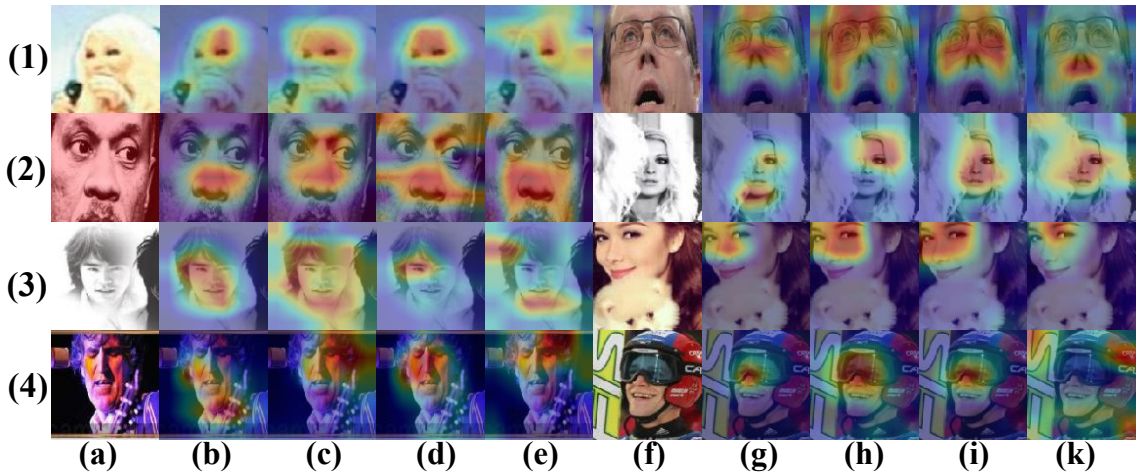

Figure 8.3: Effects of the CQA-Face model on several challenging faces. **(a)**&**(f):** Input faces which are affected by illumination changes, occlusions, or pose variations where MTCNN fails to detect landmarks. **(b)**&**(g):** Without an explicit signal, multiple attention maps emphasize few face parts, while missing some important parts. **(c)**&**(h):** The CAL can locate rich local patches. However, different located regions may exhibit different discriminative ability, like background and snow goggles. **(d)**&**(i):** The QAN can emphasize important face parts and suppress noisy regions. **(e)**&**(k):** Located regions if the L_{D} in Eqn. (8.3) is used.

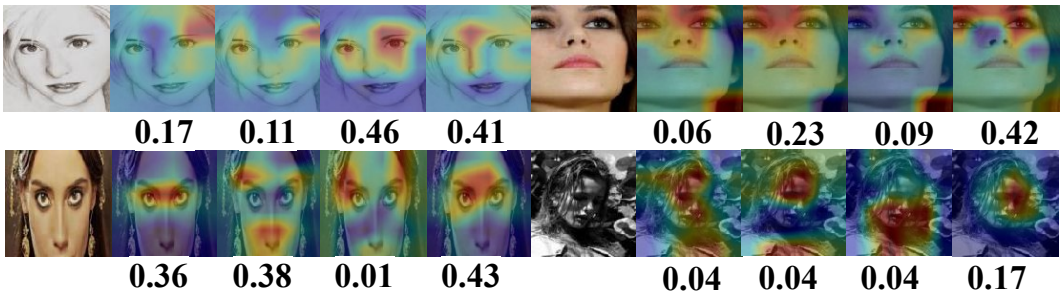

Figure 8.4: Quality scores learned by the QAN on 4 local branches.

score for each face part automatically from a global view. As shown in Fig. 8.4, discriminative local patches are assigned with large quality scores, while less important regions have small quality scores. Consequently, discriminative parts are highlighted and useless regions are suppressed. Therefore, the accuracy is increased from (99.47%, 92.90%, 89.85%) to (99.52%, 93.37%, 90.20%). Second, some background information may be represented under the guidance of the CAL, the QAN can assign low weights to distracted regions and large weights to discriminative parts, which well prepares our models for pose changes. The large accuracy gain (0.35%) on CPLFW verifies the effectiveness of the QAN.

As shown in Fig. 8.2 (b)&(g), attention-based methods can learn some important face parts automatically. However, since there is not an explicit signal, some important regions are ignored. This would inevitably deteriorate the performance if only some face parts are visible under pose variations or occlusions. The CAL alleviates this issue by enlarging pairwise distances about each two attention maps. As demonstrated in (c)&(h), rich local regions are located under the guidance of the CAL. However, it can be observed that some noisy regions are emphasized, like background, or snow goggles. The QAN can adaptively assign weights to different regions through learning from a global perspective. As observed in (d)&(i), some important regions are emphasized and noisy regions are suppressed. As a result, more discriminative features are extracted to improve the performance.

Comparison between CAL and diverse learning and dropout

The diverse learning in [14] encourages the dissimilarities among multiple attention maps by the L_D in Eqn. (8.3). Our proposed CAL is compared with the L_D in Table 8.1. If the

L_D is used, its performance is (99.33%, 93.20%, 89.82%). In contrast, our CAL is able to achieve a higher performance of (99.52%, 93.37%, 90.20%). This is because our CAL adopts a more flexible divergence loss where the σ controls the variation around its mean value.

Besides quantitative results in Table 8.1, we run a experiment by adding only L_D into our baseline, achieving (99.42%, 92.67%, 89.68%), which is inferior than only adding CAL (99.47%, 92.90%, 89.85%). Besides, as shown in Fig. 8.2, qualitative results between L_D (e&k) and CAL (d&i) also demonstrate the superiority. This is because L_D uses Euclidean distance to calculate the similarity of two attention maps. However, it is sensitive to scales due to characteristics of Euclidean distances. This makes it less effective under the scenario of large scale variations in attention maps where some noise may have high responses. In contrast, CAL takes advantages of Gaussian function and the diversity penalization is affected by the Gaussian distance, preventing more penalization variances and achieving better performance than Euclidean distance [157]. Furthermore, we thoroughly explore different values of σ which can control the Gaussian distances flexibly. Three local branches locate different local patches in Fig. 8.2 (b-d)&(g-i).

If dropout is used instead of CAL, it achieves (99.42%, 92.72%, 89.52%). Although dropout can encourage the learning of more parts, this signal is too weak to achieve diversity. In contrast, our CAL enforces attention maps away from each other, locating diverse local patches.

Different number of local branches b

Different number of local branches (i.e. 2, 4, 8, and 16) are compared in Table 8.2. If b is set to 2, only few facial parts are located, while missing some important local patches. b is improved to 4, boosting the results from (99.32%, 93.15%, 90.02%) to (99.52%, 93.37%, 90.20%). However, if the b is 8, some unnecessary information may be represented. Consequently, the result is dropped to (99.45%, 93.13%, 90.00%). This issue is becoming more serious when b is 16, which achieves the result of (99.40%, 92.80%, 89.78%).

Table 8.2: Performance comparison (%) of different hyper-parameters: Varying number of local branches (b) and values of λ Eqn. (8.9), and t and σ in Eqn. (8.2).

b	λ	σ	t	LFW	CALFW	CPLFW
2	0.50	0.01	0.2	99.32	93.15	90.02
4	0.50	0.01	0.2	**99.52**	**93.37**	**90.20**
8	0.50	0.01	0.2	99.45	93.13	90.00
16	0.50	0.01	0.2	99.40	92.80	89.78
4	**5.00**	0.01	0.2	99.43	93.10	**90.22**
4	**1.00**	0.01	0.2	99.42	93.18	90.00
4	**0.50**	0.01	0.2	**99.52**	**93.37**	90.20
4	**0.10**	0.01	0.2	99.47	92.90	89.85
4	**0.05**	0.01	0.2	99.40	92.88	90.18
4	0.50	**0.001**	0.2	99.40	93.17	**90.22**
4	0.50	**0.005**	0.2	**99.55**	93.23	90.07
4	0.50	**0.010**	0.2	99.52	**93.37**	90.20
4	0.50	**0.050**	0.2	99.40	93.27	89.92
4	0.50	**0.100**	0.2	99.47	93.20	90.12
4	0.50	0.01	**0.6**	99.35	93.18	90.07
4	0.50	0.01	**0.4**	99.35	93.07	90.17
4	0.50	0.01	**0.2**	**99.52**	**93.37**	**90.20**
4	0.50	0.01	**0.0**	99.45	93.18	90.17

Different values of λ

As illustrated in Table 8.2, we compare the performance of different values of λ in Eqn. (8.9): 5, 1, 0.5, 0.1, and 0.05. If we reduce the value of λ from 5, 1 to 0.5, the overall accuracy is boosted to (99.52%, 93.37%, 90.20%). This is because some noisy information is extracted under the strong guidance of the ACL if λ is too large. On the other hand, if λ is decreased from 0.5 to 0.1 and 0.05, the performance declines. This means that the signal of the ACL is too weak to miss some important features. Therefore, λ is assigned to 0.5, which obtains a good trade-off between the L_{Cls} and L_{CAL}.

Different values of σ

The σ controls the shape of the Gaussian function. When σ becomes larger, more variances are allowed around the mean; As σ becomes smaller, the less variances allow. It is observed that the highest accuracy is achieved when $\sigma = 0.01$.

Varying values of t

We investigate varying values of the hyper-parameter margin t in Eqn. (8.2) in Table 8.2: 0.6, 0.4, 0.2, and 0. It shows that the best margin $t = 0.2$ which achieves the best accuracy.

8.3.3 Experiments on Masked Face Matching

Table 8.3: Performance comparison (%) of our CQA-Face models with several models on M2N and M2M protocols of masked face matching.

Method	M2N				M2M		
	FAR=0.1	FAR=0.01	FAR=0.001	Rank-1	FAR=0.1	FAR=0.01	FAR=0.001
ArcFace [31]	31.53	8.54	1.45	15.73	29.22	7.25	2.06
CurricularFace [143]	33.43	13.21	4.42	17.38	31.55	9.79	2.95
CQA-Face	**59.40**	**34.22**	**16.72**	**40.46**	**57.84**	**34.93**	**17.75**

Table 8.4: Performance comparison (%) of our CQA-Face models with several models on MLFW protocol of masked face matching.

Method	MLFW			
	FAR=0.1	FAR=0.01	FAR=0.001	Acc.
ArcFace [31]	44.92	12.85	1.81	69.25
CurricularFace [143]	59.24	23.45	3.62	74.61
CQA-Face	**89.33**	**86.68**	**84.23**	**92.78**

The most recent publicly available models are compared using the M2N, M2M, and MLFW protocols, respectively, in Tables 8.3 and 8.4. For a fair comparison, CQA-Face adopts ResNet-100 as the stem CNN which is used in both ArcFace [31] and CurricularFace [143].

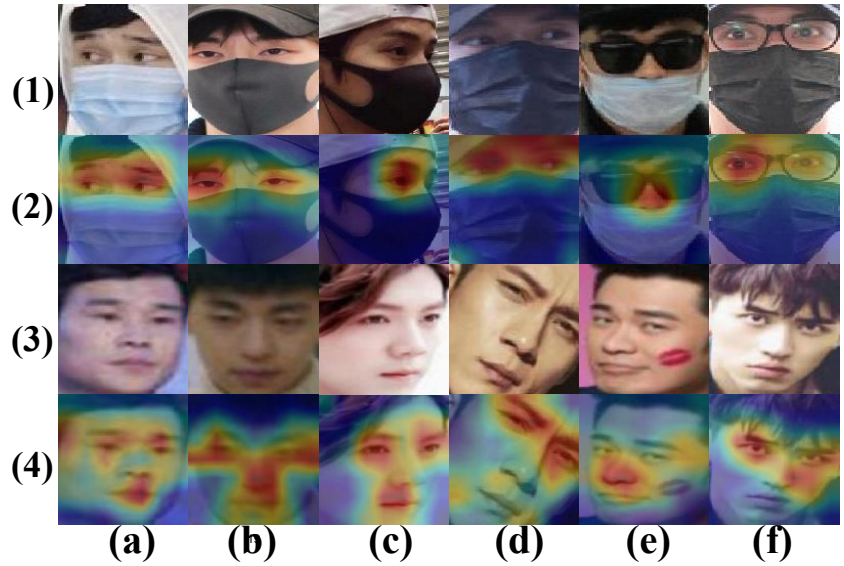

Figure 8.5: Qualitative illustration of the CQA-Face model [9] on masked face matching. **(1)**&**(3)**: Input faces with blur changes in **(a)**&**(b)**, pose variations in **(c)**&**(d)**, and heavy occlusions in **(e)**&**(f)**). **(2)**&**(4)** show CAMs on these faces.

M2N refers to matching between masked and normal faces. For a clear presentation, a tuple (a, b, c, d) is used to denote the performance where a, b, c is the TAR values when FAR=0.1, 0.01, 0.001, respectively. d refers to the Rank-1 accuracy. Because both masked and normal faces are captured in the real-world scenarios, this is a very challenging task. Since ArcFace and CurricularFace are trained on global faces without considering locating discriminative local representations, the performance is significantly decreased on this task. Specifically, ArcFace obtains the accuracy of (31.53%, 8.54%, 1.45%, 15.73%) and CurricularFace achieves the performance of (33.43%, 13.21%, 4.42%, 17.38%). However, due to its localization ability of diverse local patches under the supervision of the CAL and its part-level quality-aware capacity among multiple local patches with the QAN, our CQA-Face model boosts the performance to (59.40%, 34.22%, 16.72%, 40.46%). Although our model uses an inferior CosFace, it still outperforms ArcFace and CurricularFace, showing the superiority.

The M2M protocol refers to the verification between masked and masked faces. For clarity, a tuple (a, b, c) is used to denote the performance where a, b, c is the TAR values when FAR=0.1, 0.01, 0.001, respectively. Since both faces in gallery and probe sets are masked faces where some facial clues are occluded in M2M, thus M2M is more challenging than

Table 8.5: Performance comparison (%) of our CQA-Face model with the state-of-the-art on different face matching tasks.

Methods	IJB-A quality		CPLFW	CFP-FP	CALFW	LFW
	FAR=0.01	FAR=0.001				
Center loss [29]	52.5	31.3	77.48	-	85.48	99.13
SphereFace [36]	54.8	39.6	81.40	-	90.30	99.42
LS-CNN [5]	87.5	75.5	88.03	97.17	92.00	99.52
MV-Softmax [138]	-	-	89.69	95.70	95.63	99.79
ArcFace [31]	68.6	65.7	92.08	98.37	95.45	**99.83**
HPDA [14]	87.6	80.3	92.35	-	95.90	99.80
DBM [140]	-	-	92.63	-	96.08	99.78
EQFace [158]	-	-	92.60	98.34	95.98	99.82
CQA-Face	**91.1**	**86.4**	**93.00**	**98.49**	96.12	99.83

M2N. Consequently, global-based models (i.e. ArcFace, CurricularFace) have inferior performances, with results of (29.22%, 7.25%, 2.06%) and (31.55%, 9.79%, 2.95%), respectively. Compared with them, our CAL module can locate rich local representations. Consequently, non-occluded facial parts can contribute to the FR even if some face clues are invisible. Besides, the QAN learns quality scores for multiple local patches in a global scope, emphasizing important local patches and suppressing noisy regions. This is especially important for masked face matching where face masks cover some important face regions. In such a manner, the QAN assigns low weights to unimportant located parts. Therefore, our CQA-Face achieves the highest overall performance of (57.84%, 34.93%, 17.75%).

MLFW means the verification protocol on masked LFW. A tuple (a, b, c, d) is used where a, b, and c represent TAR values on FAR=0.1, 0.01, and 0.001, respectively, and d denotes the accuracy. As illustrated in Table 8.4, although ArcFace [31] model[1] achieves 99.77% on the original LFW, its accuracy is reduced significantly to (44.92%, 12.85%, 1.81%, 69.25%). Similarly, although CurricularFace [143] obtains the 99.80% on the original LFW, it achieves the accuracy of (59.24%, 23.45%, 3.62%, 74.61%) on MLFW. Our CQA-Face model improves over them, achieving (89.33%, 86.68%, 84.23%, 92.78%).

[1]https://github.com/deepinsight/insightface/wiki/Model-Zoo

As shown in Fig. 8.5, global faces change significantly under different challenging factors, like blur changes in (a)&(b), pose variations in (c)&(d), and heavy occlusions in (e)&(f). However, our CQA-Face model can locate discriminative parts effectively. As demonstrated in the above analysis, our CQA-Face model is more robust to face masks, which shows great potentials for recognizing masked faces under the pandemic of the COVID-19.

8.3.4 Experiments on Cross-Quality Face Matching

In some real-world applications, the captured face images may be of low-quality, which are matched with enrolled high-quality faces. Here cross-quality face matching is conducted to simulate the above scenarios. Several public models are compared on IJB-A quality in Table 8.5. TAR values are reported when FAR=0.01 and 0.001, respectively.

Several global-based approaches employed full face images as the training input: Center loss [29], SphereFace [36], and ArcFace [31]. However, they fail to extract discriminative local patches, which lead to sub-optimal representations. Differently, LS-CNN [5] used spatial attention modules to focus on important local parts, and boosted the performance to (87.5%, 75.5%). Furthermore, HPDA model [14] enlarged pairwise attention distances, emphasizing rich facial parts to make models robust to blur changes. It obtains the performance of (87.6%, 80.3%). Like ArcFace [31] which uses ResNet-100 as the stem, CQA-Face boosts the performance from (68.6%, 65.7%) to (91.1%, 86.4%).

8.3.5 Experiments on Cross-Pose Face Matching

CPLFW and CFP-FP datasets are used to evaluate the performance where a tuple (a, b) is used to show the results.

Since some face clues are missed in profile view, it is inevitable that global-based methods (i.e. Center loss [29], SphereFace [36], and MV-Softmax [138]) suffer from performance drops. In contrast, LS-CNN [5] locates discriminative local patches by spatial attentions. Thus, it achieves competitive performances on these datasets, which shows great potentials of local descriptions in this task.

However, some important local patches may be missed. HPDA [14] alleviates this issue

by maximizing pairwise attention Euclidean distances, and thus outperforms LS-CNN, which is (92.35%, -) versus (88.03%, 97.17%). − denotes that the result is unreported. Meanwhile, CQA-Face adopts ResNet-100 as stem CNN, but further increases the overall performance to (93%, 98.49%). Our explanation is that the CAL enlarges pairwise attention distances, and thus make models locate diverse facial parts. Non-occluded parts can contribute to FR if the occluded parts are invisible, making models robust to pose variations. And also, the QAN learns a quality score for each local representation in a global scope, assigning high weights to important parts and low weights to noisy parts. Although EQFace [158] learns a quality score for a whole facial image and achieves (92.60%, 98.34%), it fails to explore quality scores for different facial parts. This is important for cross-pose face matching because different facial parts have different importance.

8.3.6 Experiments on Cross-Age Face Matching and LFW Dataset

We investigate the performance on cross-age face matching. As shown in Table 8.5, our CQA-Face achieves the best result on CALFW (96.12%). Finally, our CQA-Face model obtains the competitive result on LFW, achieving 99.83% accuracy.

8.4 Summary

We have proposed a new method, called CQA-Face model, which is mainly composed of contrastive attention learning (CAL) and quality-aware network (QAN). First, the CAL encourages the diversity among different attention maps. In such a manner, it is guaranteed that comprehensive facial regions are explored. Second, the QAN learns quality scores for each local representations from a global scope. Our CQA-Face model outperforms the state-of-the-art methods on several challenging tasks, illustrating the importance and usefulness of our proposed approach.

Chapter 9

Conclusion and Future Work

In this dissertation, the recent developments of FR have been reviewed firstly, which include traditional approaches and deep learning approaches. Then, my research works are introduced where effective attention mechanisms are designed for FR. Since rich local patches are located to benefit FR, decent performances have been obtained on various face recognition tasks, including cross-age, cross-pose, cross-quality, cross-modality, and masked face matching.

A comprehensive comparison is conducted between all our proposed models in terms of accuracy and inference time. As shown in Tables 9.1, 9.2, and 9.3, a comprehensive performance comparison is conducted between our proposed LS-CNN, HPDA, CQA-Face, DSA-Face, AAN-Face and other SOTA models. It can be observed that DSA-Face performs the best on cross-quality and cross-pose datasets. AAN-Face is the best on cross-age and LFW datasets. DSA-Face performs the best on M2N and M2M. AAN-Face is the best on MLFW dataset. Besides, we also conduct a inference time comparison in terms of running time. This experiment is conducted with a batch size of 16 on NVIDIA TITAN Xp. The running time is calculated by averaging 2,000 iterations. As shown in Table 9.4, our proposed LS-CNN, HPDA, CQA-Face, DSA-Face, and AAN-Face models are compared. It can be observed that CQA-Face, DSA-Face[a], and AAN-Face[a] have a similar running time because they use the same backbone, namely ResNet-101. Similarly, LS-CNN, HPDA, DSA-Face[b], DSA-Face[c], AAN-Face[b], and AAN-Face[c] use the same backbone (LS-CNN) where the HPDA has a higher running time because of its more complex network structure. Besides,

Table 9.1: Performance comparison (%) between different models on cross-quality, cross-pose, cross-age face matching and LFW dataset.

Method	IJB-A quality		CPLFW	CFP-FP	VGGFace2 -FP	CALFW	AgeDB-30	LFW
	FAR=0.01	FAR=0.001						
Center loss [29]	52.5	31.3	77.48	-	75.10	85.48	-	99.13
SphereFace [36]	54.8	39.6	81.40	-	20.10	90.30	-	99.42
MV-Softmax [138]	-	-	89.69	95.70	-	95.63	98.11	99.79
ArcFace [31]	68.6	65.7	92.08	98.37	46.20	95.45	**98.15**	99.83
DDL [150]	-	-	93.43	98.53	-	-	-	99.68
DBM [140]	-	-	92.63	-	-	96.08	97.90	99.78
LS-CNN	87.5	75.5	88.03	97.17	69.92	92.00	-	99.52
HPDA	87.6	80.3	92.35	-	95.32	95.90	-	99.80
CQA-Face	91.1	86.4	93.00	98.49	-	96.12	-	99.83
DSA-Face[a]	87.3	80.4	93.70	98.53	95.96	-	-	99.83
DSA-Face[b]	89.5	83.2	92.73	98.14	95.66	-	-	99.75
DSA-Face[c]	**91.2**	**86.7**	93.33	98.66	95.96	-	-	99.82
DSA-Face[d]	90.5	85.3	93.62	**98.69**	**96.24**	-	-	99.85
AAN-Face[a]	90.6	85.9	93.25	98.39	95.72	96.03	97.95	99.85
AAN-Face[b]	89.2	82.6	92.68	97.94	95.62	96.00	97.83	99.80
AAN-Face[c]	89.4	82.9	92.97	98.26	95.68	95.97	98.10	99.82
AAN-Face[d]	90.0	84.5	**93.57**	98.63	95.82	**96.13**	**98.15**	**99.87**

the running time of DSA-Face[d] and AAN-Face[d] increases significantly. This is because the LS-CNN-177 is adopted as the backbone which has 177 layers compared with 97 layers in the LS-CNN.

As validated in my research works, local features are proved very important in FR. It is highly expected to achieve decent performance with local features on other face-related tasks:

Face Anti-Spoofing. Face anti-spoofing (FAS) is important for ensuring the security of face recognition systems. Deep learning has obtained great success in this area, however, most existing approaches fail to consider comprehensive relation-aware local representations of live and spoof faces. To solve this issue, we propose a Transformer-based Face Anti-Spoofing (TransFAS) model to explore comprehensive facial parts for FAS [159]. Besides the multi-head self-attention which explores relations among local patches in the same layer, we propose Cross-layer Relation-aware Attentions (CRA) to adaptively integrate local patches from different layers. Furthermore, to effectively fuse hierarchical features, we explore the best Hierarchical Feature Fusion (HFF) structure, which can capture the complementary

Table 9.2: Performance comparison (%) of our models with several publicly available models on M2N and M2M protocols of masked face matching.

Method	M2N				M2M		
	FAR= 0.1	FAR= 0.01	FAR= 0.001	Rank-1	FAR= 0.1	FAR= 0.01	FAR= 0.001
ArcFace [31]	31.53	8.54	1.45	15.73	29.22	7.25	2.06
CurricularFace [143]	33.43	13.21	4.42	17.38	31.55	9.79	2.95
Subcenter Arcface [144]	24.60	4.51	0.56	12.49	28.95	8.57	3.15
HPDA	57.41	30.94	14.98	34.76	58.99	37.68	**22.01**
CQA-Face	59.40	34.22	16.72	40.46	57.84	34.93	17.75
DSA-Face[a]	48.04	21.53	8.09	22.67	47.96	21.47	7.93
DSA-Face[b]	62.09	36.59	18.55	43.08	**60.81**	**38.92**	19.58
DSA-Face[c]	48.28	21.60	7.42	23.91	51.05	25.54	11.16
DSA-Face[d]	**62.83**	**39.21**	**21.14**	**43.50**	59.28	38.14	17.08
AAN-Face[a]	45.62	19.56	7.49	23.65	47.41	21.14	7.27
AAN-Face[b]	61.28	34.74	17.18	40.57	60.31	38.18	19.79
AAN-Face[c]	61.24	36.36	18.87	42.31	60.59	38.23	20.13
AAN-Face[d]	62.11	37.18	18.71	41.54	59.39	35.95	16.54

information of both low-level artifacts and high-level semantic features of the spoofing patterns. With these novel modules, TransFAS not only improves the generalization ability of classical Vision Transformer, but also achieves SOTA performance on multiple benchmarks, demonstrating the superiority of the Transformer-based model for FAS.

Despite decent performance has been obtained, few existing works fully leverage temporal information. This would inevitably lead to inferior performance because real and fake faces share highly similar spatial appearances while important temporal features between different frames are neglected. In this work, we propose a temporal transformer network (TTN) to learn multi-granularity temporal information for FAS [160]. It mainly consists of temporal difference attentions (TDA), a pyramid temporal aggregation (PTA), and a temporal depth difference loss (TDL). Firstly, the Vision Transformer (ViT) is used as the backbone where comprehensive local patches are utilized to learn subtle differences between real and spoof faces. Then, instead of learning temporal features on global faces which may miss some

Table 9.3: Performance comparison (%) of our proposed models with several publicly available models on MLFW protocol of masked face matching.

Method	MLFW			
	FAR=0.1	FAR=0.01	FAR=0.001	Acc.
ArcFace [31]	44.92	12.85	1.81	69.25
CurricularFace [143]	59.24	23.45	3.62	74.61
Subcenter Arcface [144]	50.05	30.32	18.82	69.89
HPDA	89.06	85.51	76.65	92.34
CQA-Face	89.33	86.68	84.23	92.78
DSA-Face[a]	82.81	69.33	36.77	87.47
DSA-Face[b]	89.20	85.84	79.77	92.46
DSA-Face[c]	85.68	74.91	51.12	88.63
DSA-Face[d]	89.74	86.72	82.25	**92.91**
AAN-Face[a]	80.87	66.19	29.26	86.10
AAN-Face[b]	89.44	86.22	80.71	92.58
AAN-Face[c]	**89.77**	86.28	83.13	92.65
AAN-Face[d]	89.67	**86.95**	**84.17**	**92.91**

Table 9.4: Inference time comparison of our proposed models.

Method	LS-CNN	HPDA	CQA-Face	DSA-Face				AAN-Face			
				[a]	[b]	[c]	[d]	[a]	[b]	[c]	[d]
Time (ms)	238.24	345.78	155.83	162.16	259.08	262.55	614.49	152.90	233.48	235.52	530.41

important local cues, the TDA is used to extract temporal local attentions on comprehensive local patches from adjacent frames. Besides, the TDA is inserted into different layers of the ViT. Consequently, the TDA can extract multi-scale motion-sensitive local cues to improve the FAS performance. Secondly, it is observed that different subjects may have different temporal speeds in some actions, making it necessary to learn different temporal speeds. To solve this issue, the PTA aggregates temporal features at various tempos, which could build short-range and long-range relations among multiple frames. Thirdly, depth maps for real parts may change continuously, while they remain to be zeros for spoof regions. In order to locate motion features on spoof parts, the TDL is proposed to guide the network to

locate spoof facial parts where motion patterns between neighboring frames are set as the ground truth. To the best of our knowledge, this work may be the first attempt to learn temporal information via Transformers. Both qualitative and quantitative results on several challenging tasks demonstrate the usefulness and effectiveness of our proposed modules.

Facial Expression Recognition. Facial expression recognition (FER) has received increasing interest in computer vision. We propose the TransFER model which can learn rich relation-aware local representations [161]. It mainly consists of three components: Multi-Attention Dropping (MAD), ViT-FER, and Multi-head Self-Attention Dropping (MSAD). First, local patches play an important role in distinguishing various expressions, however, few existing works can locate discriminative and diverse local patches. This can cause serious problems when some patches are invisible due to pose variations or viewpoint changes. To address this issue, the MAD is proposed to randomly drop an attention map. Consequently, models are pushed to explore diverse local patches adaptively. Second, to build rich relations between different local patches, the Vision Transformers (ViT) are used in FER, called ViT-FER. Since the global scope is used to reinforce each local patch, a better representation is obtained to boost the FER performance. Thirdly, the multi-head self-attention allows ViT to jointly attend to features from different information subspaces at different positions. Given no explicit guidance, however, multiple self-attentions may extract similar relations. To address this, the MSAD is proposed to randomly drop one self-attention module. As a result, models are forced to learn rich relations among diverse local patches. Our proposed TransFER model outperforms the state-of-the-art methods on several FER benchmarks, showing its effectiveness and usefulness.

Besides, many other related tasks may also be explored, such as face detection, deefake detection, facial landmark detection, and so on.

References

[1] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1701–1708, 2014.

[2] B.-N. Kang, Y. Kim, B. Jun, and D. Kim, "Attentional feature-pair relation networks for accurate face recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5472–5481, 2019.

[3] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2921–2929, 2016.

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

[5] Q. Wang and G. Guo, "Ls-cnn: Characterizing local patches at multiple scales for face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 1640–1653, 2019.

[6] Q. Wang and G. Guo, "Dsa-face: diverse and sparse attentions for face recognition robust to pose variation and occlusion," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4534–4543, 2021.

[7] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in Neural Information Processing Systems*, pp. 487–495, 2014.

[8] Q. Wang and G. Guo, "Aan-face: attention augmented networks for face recognition," *IEEE Transactions on Image Processing*, vol. 30, pp. 7636–7648, 2021.

[9] Q. Wang and G. Guo, "Cqa-face: Contrastive quality-aware attentions for face recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, pp. 2504–2512, Jun. 2022.

[10] P. Li, X. Wu, Y. Hu, R. He, and Z. Sun, "M2fpa: a multi-yaw multi-pitch high-quality dataset and benchmark for facial pose analysis," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10043–10051, 2019.

[11] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Learning face representation from scratch," *arXiv preprint arXiv:1411.7923*, 2014.

[12] B.-C. Chen, C.-S. Chen, and W. H. Hsu, "Cross-age reference coding for age-invariant face recognition and retrieval," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 768–783, Springer, 2014.

[13] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

[14] Q. Wang, T. Wu, H. Zheng, and G. Guo, "Hierarchical pyramid diverse attention networks for face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8326–8335, 2020.

[15] L. M. Mayron, Y. Hausawi, and G. S. Bahr, "Secure, usable biometric authentication systems," in *International Conference on Universal Access in Human-Computer Interaction*, pp. 195–204, Springer, 2013.

[16] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pp. 586–587, IEEE Computer Society, 1991.

[17] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 711–720, 1997.

[18] J. Yang, D. Zhang, A. F. Frangi, and J.-y. Yang, "Two-dimensional pca: a new approach to appearance-based face representation and recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 26, no. 1, pp. 131–137, 2004.

[19] L. Wiskott, N. Krüger, N. Kuiger, and C. Von Der Malsburg, "Face recognition by elastic bunch graph matching," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 7, pp. 775–779, 1997.

[20] T. Ahonen, A. Hadid, and M. Pietikainen, "Face description with local binary patterns: Application to face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 28, no. 12, pp. 2037–2041, 2006.

[21] C. Liu and H. Wechsler, "Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition," *IEEE Transactions on Image processing*, vol. 11, no. 4, pp. 467–476, 2002.

[22] W. Zhang, S. Shan, W. Gao, X. Chen, and H. Zhang, "Local gabor binary pattern histogram sequence (lgbphs): A novel non-statistical model for face representation and recognition," in *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, vol. 1, pp. 786–791, IEEE, 2005.

[23] Z. Cao, Q. Yin, X. Tang, and J. Sun, "Face recognition with learning-based descriptor," in *2010 IEEE Computer society conference on computer vision and pattern recognition*, pp. 2707–2714, IEEE, 2010.

[24] Z. Lei, M. Pietikäinen, and S. Z. Li, "Learning discriminant face descriptor," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 36, no. 2, pp. 289–302, 2013.

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

[26] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[27] O. M. Parkhi, A. Vedaldi, A. Zisserman, *et al.*, "Deep face recognition.," in *BMVC*, vol. 1, p. 6, 2015.

[28] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 815–823, 2015.

[29] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European conference on computer vision*, pp. 499–515, Springer, 2016.

[30] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5265–5274, 2018.

[31] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2019.

[32] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, "Vggface2: A dataset for recognising faces across pose and age," in *FG*, pp. 67–74, IEEE, 2018.

[33] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, "Ms-celeb-1m: A dataset and benchmark for large-scale face recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 87–102, Springer, 2016.

[34] Y. Sun, Y. Chen, X. Wang, and X. Tang, "Deep learning face representation by joint identification-verification," in *Advances in neural information processing systems*, pp. 1988–1996, 2014.

[35] X. Wu, R. He, Z. Sun, and T. Tan, "A light cnn for deep face representation with noisy labels," *Transactions on Information Forensics and Security*, vol. 13, no. 11, pp. 2884–2896, 2018.

[36] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 212–220, 2017.

[37] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1891–1898, 2014.

[38] C. Ding and D. Tao, "Trunk-branch ensemble convolutional neural networks for video-based face recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2017.

[39] J. Liu, Y. Deng, T. Bai, Z. Wei, and C. Huang, "Targeting ultimate accuracy: Face recognition via deep embedding," *arXiv preprint arXiv:1506.07310*, 2015.

[40] J. Lei, B. Zhang, and H. Ling, "Deep learning face representation by fixed erasing in facial landmarks," *Multimedia Tools and Applications*, vol. 78, no. 19, pp. 27703–27718, 2019.

[41] W. Xie, L. Shen, and A. Zisserman, "Comparator networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 782–797, 2018.

[42] B.-N. Kang, Y. Kim, and D. Kim, "Pairwise relational networks for face recognition," *arXiv preprint arXiv:1808.04976*, 2018.

[43] H. Ling, J. Wu, J. Huang, J. Chen, and P. Li, "Attention-based convolutional neural network for deep face recognition," *Multimedia Tools and Applications*, vol. 79, no. 9, pp. 5595–5616, 2020.

[44] J. Huang and C. Ding, "Attention-guided progressive mapping for profile face recognition," in *2021 IEEE International Joint Conference on Biometrics (IJCB)*, pp. 1–8, IEEE, 2021.

[45] H. Uppal, A. Sepas-Moghaddam, M. Greenspan, and A. Etemad, "Two-level attention-based fusion learning for rgb-d face recognition," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 10120–10127, IEEE, 2021.

[46] A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *arXiv preprint arXiv:1404.5997*, 2014.

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

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

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016.

[50] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[51] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks.," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, p. 3, 2017.

[52] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[53] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on faces in'Real-Life'Images: detection, alignment, and recognition*, 2008.

[54] L. Wolf, T. Hassner, and I. Maoz, *Face recognition in unconstrained videos with matched background similarity*. IEEE, 2011.

[55] Y. Sun, X. Wang, and X. Tang, "Deep learning face representation from predicting 10,000 classes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1891–1898, 2014.

[56] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks," *arXiv preprint arXiv:1502.00873*, 2015.

[57] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv preprint arXiv:1706.03762*, 2017.

[58] A. Bansal, A. Nanduri, C. D. Castillo, R. Ranjan, and R. Chellappa, "Umdfaces: An annotated face dataset for training deep networks," in *IJCB*, pp. 464–473, IEEE, 2017.

[59] S. Sengupta, J.-C. Chen, C. Castillo, V. M. Patel, R. Chellappa, and D. W. Jacobs, "Frontal to profile face verification in the wild," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1–9, IEEE, 2016.

[60] T. Zheng and W. Deng, "Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments," *Beijing University of Posts and Telecommunications, Tech. Rep*, vol. 5, 2018.

[61] T. Zheng, W. Deng, and J. Hu, "Cross-age lfw: A database for studying cross-age face recognition in unconstrained environments," *arXiv preprint arXiv:1708.08197*, 2017.

[62] S. Moschoglou, A. Papaioannou, C. Sagonas, J. Deng, I. Kotsia, and S. Zafeiriou, "Agedb: the first manually collected, in-the-wild age database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 51–59, 2017.

[63] B. F. Klare, B. Klein, E. Taborsky, A. Blanton, J. Cheney, K. Allen, P. Grother, A. Mah, and A. K. Jain, "Pushing the frontiers of unconstrained face detection and recognition: Iarpa janus benchmark a," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1931–1939, 2015.

[64] H.-W. Ng and S. Winkler, "A data-driven approach to cleaning large face datasets," in *ICIP*, pp. 343–347, IEEE, 2014.

[65] G. Guo and N. Zhang, "What is the challenge for deep learning in unconstrained face recognition?," in *FG*, pp. 436–442, IEEE, 2018.

[66] S. Li, D. Yi, Z. Lei, and S. Liao, "The casia nir-vis 2.0 face database," in *CVPR-W*, pp. 348–353, 2013.

[67] Z. Wang, G. Wang, B. Huang, Z. Xiong, Q. Hong, H. Wu, P. Yi, K. Jiang, N. Wang, Y. Pei, *et al.*, "Masked face recognition dataset and application," *arXiv preprint arXiv:2003.09093*, 2020.

[68] T. I. Dhamecha, R. Singh, M. Vatsa, and A. Kumar, "Recognizing disguised faces: Human and machine evaluation," *PloS one*, vol. 9, no. 7, p. e99212, 2014.

[69] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," tech. rep., Technical Report 07-49, University of Massachusetts, Amherst, 2007.

[70] B. Maze, J. Adams, J. A. Duncan, N. Kalka, T. Miller, C. Otto, A. K. Jain, W. T. Niggel, J. Anderson, J. Cheney, *et al.*, "Iarpa janus benchmark-c: Face dataset and protocol," in *2018 International Conference on Biometrics (ICB)*, pp. 158–165, IEEE, 2018.

[71] J. Lu, V. E. Liong, and J. Zhou, "Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 8, pp. 1979–1993, 2017.

[72] Y. Duan, J. Lu, J. Feng, and J. Zhou, "Context-aware local binary feature learning for face recognition," *TPAMI*, vol. 40, no. 5, pp. 1139–1153, 2018.

[73] M. Yang, Q. Wang, W. Wen, and Z. Lai, "Multi-feature joint dictionary learning for face recognition," in *ACPR*, pp. 629–633, IEEE, 2017.

[74] J. Yangqing and et al., "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 675–678, ACM, 2014.

[75] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[76] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[77] F. Chollet, "Monthly arxiv.org mentions for frameworks," 2018.

[78] W. Xiang and et al., "A light cnn for deep face representation with noisy labels," *arXiv preprint arXiv:1511.02683*, 2015.

[79] Q. Wang, G. Guo, and M. I. Nouyed, "Learning channel inter-dependencies at multiple scales on dense networks for face recognition," *arXiv preprint arXiv:1711.10103*, 2017.

[80] J. Lu, J. Hu, and Y.-P. Tan, "Discriminative deep metric learning for face and kinship verification," *TIP*, vol. 26, no. 9, pp. 4269–4282, 2017.

[81] Y. Duan, J. Lu, and J. Zhou, "Uniformface: Learning deep equidistributed representation for face recognition," in *CVPR*, pp. 3415–3424, 2019.

[82] C. Qiong and et al., "Vggface2: A dataset for recognising faces across pose and age," *arXiv preprint arXiv:1710.08092*, 2017.

[83] K.-S. Ira and et al., "The megaface benchmark: 1 million faces for recognition at scale," in *CVPR*, pp. 4873–4882, 2016.

[84] S. Shaohuai and et al., "Benchmarking state-of-the-art deep learning software tools," in *Cloud Computing and Big Data (CCBD), 2016 7th International Conference on*, pp. 99–104, IEEE, 2016.

[85] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *IJCV*, vol. 115, no. 3, pp. 211–252, 2015.

[86] D. Sandberg, "Facenet," 2019.

[87] F. Claudio and et al., "Investigating nuisances in dcnn-based face recognition," *TIP*, vol. 27, no. 11, pp. 5638–5651, 2018.

[88] Y. Wu and K. He, "Group normalization," *arXiv preprint arXiv:1803.08494*, 2018.

[89] H. Kaiming and et al., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *ICCV*, pp. 1026–1034, 2015.

[90] C. Sharan and et al., "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[91] N. Silberman and S. Guadarrama, "Tensorflowslim image classification model library," 2017.

[92] K. Heehoon and et al., "Performance analysis of cnn frameworks for gpus," *Performance Analysis of Systems and Software (ISPASS)*, 2017.

[93] H. Kaiming and et al., "Mask r-cnn," in *ICCV*, pp. 2980–2988, IEEE, 2017.

[94] Y. Ding and et al., "Depth-aware saliency detection using convolutional neural networks," *Journal of Visual Communication and Image Representation*, vol. 61, pp. 1–9, 2019.

[95] P. He and et al., "Frame-wise detection of relocated i-frames in double compressed h. 264 videos based on convolutional neural network," *Journal of Visual Communication and Image Representation*, vol. 48, pp. 149–158, 2017.

[96] Q. Wang, Y. Zheng, G. Yang, W. Jin, X. Chen, and Y. Yin, "Multiscale rotation-invariant convolutional neural networks for lung texture classification," *IEEE journal of biomedical and health informatics*, vol. 22, no. 1, pp. 184–195, 2018.

[97] M. Jiang and G. Guo, "Body weight analysis from human body images," *IEEE TIFS*, vol. 14, pp. 2676–2688, Oct 2019.

[98] C. Zalluhoglu and N. Ikizler-Cinbis, "Region based multi-stream convolutional neural networks for collective activity recognition," *Journal of Visual Communication and Image Representation*, vol. 60, pp. 170–179, 2019.

[99] C. Jianmin and et al., "Revisiting distributed synchronous sgd," *arXiv preprint arXiv:1604.00981*, 2016.

[100] D. Dipankar and et al., "Distributed deep learning using synchronous stochastic gradient descent," *arXiv preprint arXiv:1602.06709*, 2016.

[101] Google, "Google's remote procedure call library (grpc)."

[102] NVIDIA, "Nccl."

[103] A. A. Ahmad and et al., "Scalable distributed dnn training using tensorflow and cuda-aware mpi: Characterization, designs, and performance evaluation," *arXiv preprint arXiv:1810.11112*, 2018.

[104] H. Chen and et al., "Learning deep representation for imbalanced classification," in *CVPR*, pp. 5375–5384, 2016.

[105] A. Hasnat, J. Bohné, J. Milgram, S. Gentric, and L. Chen, "Deepvisage: Making face recognition simple yet with powerful generalization skills," in *ICCV*, pp. 1682–1691, 2017.

[106] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 818–833, Springer, 2014.

[107] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *arXiv preprint arXiv:1709.01507*, vol. 7, 2017.

[108] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[109] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2018.

[110] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 630–645, Springer, 2016.

[111] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[112] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Advances in Neural Information Processing Systems*, pp. 4467–4475, 2017.

[113] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[114] S. Sankaranarayanan, A. Alavi, C. Castillo, and R. Chellappa, "Triplet probabilistic embedding for face verification and clustering," *arXiv preprint arXiv:1604.05417*, 2016.

[115] J.-C. Chen, J. Zheng, V. M. Patel, and R. Chellappa, "Fisher vector encoded deep convolutional features for unconstrained face verification," in *Image Processing (ICIP), 2016 IEEE International Conference on*, pp. 2981–2985, IEEE, 2016.

[116] J. Zhao, Y. Cheng, Y. Xu, L. Xiong, J. Li, F. Zhao, K. Jayashree, S. Pranata, S. Shen, J. Xing, *et al.*, "Towards pose invariant face recognition in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2207–2216, 2018.

[117] L. Tran, X. Yin, and X. Liu, "Disentangled representation learning gan for pose-invariant face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 3, p. 7, 2017.

[118] L. Q. Tran, X. Yin, and X. Liu, "Representation learning by rotating your faces," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[119] J. Zhao, L. Xiong, J. Li, J. Xing, S. Yan, and J. Feng, "3d-aided dual-agent gans for unconstrained face recognition," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[120] X. Yin and X. Liu, "Multi-task convolutional neural network for pose-invariant face recognition," *TIP*, vol. 27, no. 2, pp. 964–975, 2018.

[121] W. Hu, Y. Huang, F. Zhang, and R. Li, "Noise-tolerant paradigm for training face recognition cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 11887–11896, 2019.

[122] J. Deng, J. Guo, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *arXiv preprint arXiv:1801.07698*, 2018.

[123] J. Deng, Y. Zhou, and S. Zafeiriou, "Marginal loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern RecognitionW*, vol. 4, 2017.

[124] B. Chen, W. Deng, and J. Du, "Noisy softmax: Improving the generalization ability of dcnn via postponing the early softmax saturation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[125] X. Qi and L. Zhang, "Face recognition via centralized coordinate learning," *arXiv preprint arXiv:1801.05678*, 2018.

[126] Y. Wen, Z. Li, and Y. Qiao, "Latent factor guided convolutional neural networks for age-invariant face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4893–4901, 2016.

[127] H. Li, H. Hu, and C. Yip, "Age-related factor guided joint task modeling convolutional neural network for cross-age face recognition," *IEEE TIFS*, vol. 13, no. 9, pp. 2383–2392, 2018.

[128] Y. Wang, D. Gong, Z. Zhou, X. Ji, H. Wang, Z. Li, W. Liu, and T. Zhang, "Orthogonal deep features decomposition for age-invariant face recognition," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 738–753, 2018.

[129] H. Wang, D. Gong, Z. Li, and W. Liu, "Decorrelated adversarial learning for age-invariant face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3527–3536, 2019.

[130] J. Chen, Y. Deng, G. Bai, and G. Su, "Face image quality assessment based on learning to rank," *IEEE signal processing letters*, vol. 22, no. 1, pp. 90–94, 2015.

[131] H. Larochelle and G. E. Hinton, "Learning to combine foveal glimpses with a third-order boltzmann machine," in *Advances in Neural Information Processing Systems*, pp. 1243–1251, 2010.

[132] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, "Feature transfer learning for deep face recognition with long-tail data," *arXiv preprint arXiv:1803.09014*, 2018.

[133] B.-N. Kang, Y. Kim, B. Jun, and D. Kim, "Hierarchical feature-pair relation networks for face recognition," in *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0, 2019.

[134] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

[135] Z. Tan, Y. Yang, J. Wan, G. Guo, and S. Z. Li, "Deeply-learned hybrid representations for facial age estimation," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3548–3554, International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[136] C. Yu, X. Zhao, Q. Zheng, P. Zhang, and X. You, "Hierarchical bilinear pooling for fine-grained visual recognition," in *Proceedings of the European Conference on Computer Vision*, pp. 574–589, 2018.

[137] X. Wang, S. Wang, J. Wang, H. Shi, and T. Mei, "Co-mining: Deep face recognition with noisy labels," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 9358–9367, 2019.

[138] X. Wang, S. Zhang, S. Wang, T. Fu, H. Shi, and T. Mei, "Mis-classified vector guided softmax loss for face recognition," *arXiv preprint arXiv:1912.00833*, 2019.

[139] Q. Wang and G. Guo, "Benchmarking deep learning techniques for face recognition," *Journal of Visual Communication and Image Representation*, vol. 65, p. 102663, 2019.

[140] D. Cao, X. Zhu, X. Huang, J. Guo, and Z. Lei, "Domain balancing: Face recognition on long-tailed domains," *arXiv preprint arXiv:2003.13791*, 2020.

[141] G. Guo and N. Zhang, "A survey on deep learning based face recognition," *Computer vision and image understanding*, vol. 189, p. 102805, 2019.

[142] J. Zhai, "Facial mask: A necessity to beat covid-19," *Building and Environment*, 2020.

[143] Y. Huang, Y. Wang, Y. Tai, X. Liu, P. Shen, S. Li, J. Li, and F. Huang, "Curricularface: adaptive curriculum learning loss for deep face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5901–5910, 2020.

[144] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Sub-center arcface: Boosting face recognition by large-scale noisy web faces," in *European Conference on Computer Vision*, pp. 741–757, Springer, 2020.

[145] Q. Meng, S. Zhao, Z. Huang, and F. Zhou, "Magface: A universal representation for face recognition and quality assessment," *arXiv preprint arXiv:2103.06627*, 2021.

[146] Y. Kim, W. Park, M.-C. Roh, and J. Shin, "Groupface: Learning latent groups and constructing group-based representations for face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 5621–5630, 2020.

[147] Y. Kim, W. Park, and J. Shin, "Broadface: Looking at tens of thousands of people at once for face recognition," in *European Conference on Computer Vision*, pp. 536–552, Springer, 2020.

[148] R. Huang, S. Zhang, T. Li, and R. He, "Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis," in *ICCV*, pp. 2439–2448, 2017.

[149] J. Zhao, L. Xiong, P. K. Jayashree, J. Li, F. Zhao, Z. Wang, P. S. Pranata, P. S. Shen, S. Yan, and J. Feng, "Dual-agent gans for photorealistic and identity preserving profile face synthesis," in *NeurIPS*, pp. 66–76, 2017.

[150] Y. Huang, P. Shen, Y. Tai, S. Li, X. Liu, J. Li, F. Huang, and R. Ji, "Distribution distillation loss: Generic approach for improving face recognition from hard samples," *arXiv preprint arXiv:2002.03662*, 2020.

[151] R. He, X. Wu, Z. Sun, and T. Tan, "Learning invariant deep representation for nir-vis face recognition," in *AAAI*, 2017.

[152] R. He, X. Wu, Z. Sun, and T. Tan, "Wasserstein cnn: Learning invariant features for nir-vis face recognition," *IEEE TPAMI*, vol. 41, no. 7, pp. 1761–1773, 2018.

[153] J. Guo, X. Zhu, C. Zhao, D. Cao, Z. Lei, and S. Z. Li, "Learning meta face recognition in unseen domains," *arXiv preprint arXiv:2003.07733*, 2020.

[154] J. Yu, J. Cao, Y. Li, X. Jia, and R. He, "Pose-preserving cross spectral face hallucination.," in *IJCAI*, pp. 1018–1024, 2019.

[155] B. Duan, C. Fu, Y. Li, X. Song, and R. He, "Cross-spectral face hallucination via disentangling independent factors," in *CVPR*, pp. 7930–7938, 2020.

[156] X. Wang and A. Gupta, "Videos as space-time region graphs," in *ECCV*, pp. 399–417, 2018.

[157] Z. Gong, P. Zhong, and W. Hu, "Diversity in machine learning," *IEEE Access*, vol. 7, pp. 64323–64350, 2019.

[158] R. Liu and W. Tan, "Eqface: A simple explicit quality network for face recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1482–1490, 2021.

[159] Z. Wang, Q. Wang, W. Deng, and G. Guo, "Face anti-spoofing using transformers with relation-aware mechanism," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2022.

[160] Z. Wang, Q. Wang, W. Deng, and G. Guo, "Learning multi-granularity temporal characteristics for face anti-spoofing," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 1254–1269, 2022.

[161] F. Xue, Q. Wang, and G. Guo, "Transfer: Learning relation-aware facial expression representations with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3601–3610, 2021.