國立臺灣大學電機資訊學院電信工程學研究所
碩士論文

Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

跨域掌紋辨識的資料擴增

Data Augmentation for Cross-Domain Palmprint Recognition

李政旻
Cheng-Min Lee

指導教授：謝宏昀 博士
Advisor: Hung-Yun Hsieh, Ph.D.

中華民國 111 年 9 月
September, 2022

doi:10.6342/NTU202203861

致謝

研究所的這段時光經歷了許多重大轉變，有了非常多新的體悟，也放棄了很多過去的執著。非常感謝在這段期間給我包容、陪伴和一起奮鬥的人們。按順序來說，首先要感謝我的家人和老闆，支持我在工作之後才重新回來求學，也在研究所期間提供很多經濟協助。再來要感謝　謝宏昀教授，願意做我的指導教授。給予我許多修課和研究上的建議，從論文題目的制定、文獻閱讀的技巧、研究方向的討論、簡報表達的方法、到最後的論文撰寫。一步一步教導我每個環節真正重要的部分，以及溝通表達時要把讀者當作無知的人，重要的概念一再重複，並且緊密的連結，這些觀念使我受益良多。剛進實驗室的時候，覺得充滿老舊古董，想要更換也面臨老師的挑戰，為什麼要換？是個哲學問題，也是個玄學問題。看著這一年不斷更換的設備，我不禁心想，是時代變了？還是老師變了？我想這些都只能歸咎於自己不夠積極。面對現實的各種挑戰，都應該勇於面對。人生在世就一個「爭」字，非常感謝老師讓我學到重要的一課。

再來我要感謝實驗室的夥伴們。真的非常感謝承翰，這三年來給予我非常多的幫助。從最開始的５Ｇ計畫，到最後一年一起為了畢業論文奮鬥。也在我低潮無助的時候使我振作起來，還會把各種事情都規劃的很好，也因此我們才能順利畢業。感謝梓豪提供的模板使我在畢業的路上有跡可循，也在題目搜索和研究問題上給予我很多建議。感謝啟仁，論文檢索的影片我看了好幾遍，終於理解如何研究題目。感謝國嘵，總是讓我覺得事情好像就這麼簡單，你的效率讓我很嚮往。感謝俊翔，總是待在實驗室不辭辛勞地做研究，真的很敬佩你的耐心與毅力，激勵我不能一直爛下去。感謝取暖團的學長和學弟們，共同為了口試和畢業而努力真的很棒，非常慶幸能夠跟你們一起畢業。感謝實驗室的同學們，你們都非常有特色，雖然久久才見到一次，但每次都覺得很有趣。

漫長的三年過去，遙想當年仍歷歷在目，，住院的煎熬，以及研究的低潮，期許經歷過各種苦難的我能夠持續進步，再次感謝在這段期間給我支持與鼓勵的家人朋友們，讓我能夠順利完成碩士學業，祝各位都能平安順心，也祝福各位實驗室的學弟妹們都能夠順利畢業。

摘要

基於生物特徵信息的身份認證系統在近年被廣泛的運用，大多數人已經接受它帶來的便利性、安全性和隱私性，可以避免使用者的密碼被盜或忘記。其中掌紋識別是人性化且衛生的生物識別，無需觸摸設備，對手勢也沒有太大的限制。隨著當今影像識別技術的飛速發展，深度學習方法在多樣化的影像資料處理上表現優於傳統方法，然而大多數掌紋識別中的深度學習方法僅在單個資料集上進行訓練和測試，一旦使用不同的成像設備，辨識的效果就會嚴重受到影像。但在實際應用場景中，輸入影像來自不同的相機是很常見的，因此如何解決不同成像設備帶來的影像差距至關重要。因此我們提出了三種影像增強策略和 ResNeSt 的縮減版本來解決跨域的掌紋識別。首先我們在訓練期間使用 Optuna 框架和 TPE 採樣器對影像隨機轉換進行超參數優化搜索。其次，將 ROI 影像旋轉到四個方向作為不同類別的訓練影像來 oversample 訓練資料集。這也可以用於增強我們提出的基於 test-time augmentation 的多轉換特徵比對方法。在有約束的資料集（PolyU-M）上訓練並在無約束的資料集（MPD）上測試的困難條件下，三種增強方法可以分別提高準確度 12.55%、5.34% 和 4.66%，總共可以實現 22.55% 的準確度提升。 此外，使用 MPD 訓練的模型在所有測試資料集中都可以達到 99.66% 以上的準確度。

doi:10.6342/NTU202203861

ABSTRACT

Identity authentication systems based on biometric information have been widely used in recent years. Most people accept its convenience, security, and privacy, avoiding the user's password being stolen or forgotten. Among them, palmprint recognition is user-friendly and hygienic biometric, without touching the device and too many pose restrictions. With the rapid development of today's image recognition technology, deep learning methods perform better than traditional methods on diverse image data. Although most deep learning methods in palmprint recognition perform well on a single dataset, it will seriously affect performance once applied to different image acquisition devices. In practical application scenarios, it is common for input images to come from different cameras, so how to bridge the gap between different imaging conditions is extremely crucial. Therefore, we propose three data augmentation strategies and a reduced version of ResNeSt to solve the cross-domain palmprint recognition problem. First, we perform a hyperparameter optimization search for random transformations during training with the Optuna framework and the TPE sampler. Second, the training dataset is oversampling augmented, which the ROI images are rotated into four orientations as training images of different classes. This can also be used to enhance our proposed multi-transform matching based on a test-time augmentation technique. Under the difficult conditions of training on a constrained acquisition dataset (PolyU-M) and testing on an unconstrained dataset (MPD), the three augmentation methods can improve by 12.55%, 5.34%, and 4.66%, respectively, so a total of 22.55% improvement can be achieved. Furthermore, the model trained on both MPD can achieve more than 99.66% accuracy in all test datasets.

doi:10.6342/NTU202203861

TABLE OF CONTENTS

doi:10.6342/NTU202203861

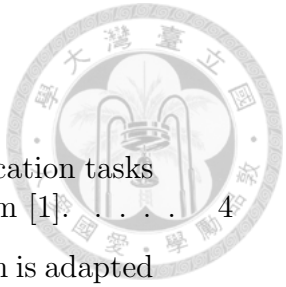
doi:10.6342/NTU202203861

doi:10.6342/NTU202203861

LIST OF TABLES

doi:10.6342/NTU202203861

LIST OF FIGURES



doi:10.6342/NTU202203861

doi:10.6342/NTU202203861

doi:10.6342/NTU202203861

CHAPTER 1

INTRODUCTION

Biometric recognition has attracted more and more attention in recent years due to the increasing awareness of privacy concerns and information security. Furthermore, image recognition technology has improved with the development of deep learning, and the performance of biometric recognition can reach a practical level. Compared with traditional token-based or knowledge-based methods, biometric recognition can effectively prevent forgetting and theft, offering higher security. However, there has been relatively little effort to explore deep learning for palmprint recognition. The benefits of palmprint recognition are convenience and user-friendliness. No touching of equipment is required, so it is hygienic and easy to use. It is not like face recognition would need to take off the mask or iris recognition can be uncomfortable. In addition, it can be easily combined with other hand-related features for multi-modal biometric identification, such as palm veins, finger-knuckle-prints, or hand shape.

A palmprint recognition usually has the following procedures: image acquisition, image preprocessing, feature extraction, and feature matching. The most important step in image preprocessing is to extract the region of interest (ROI) so that the feature extraction step can receive images of uniform size and orientation. The ROI refers to obtaining the main palmprint area without background. Feature extraction includes many different types, the most commonly used in traditional methods are texture-based features. The latest methods mostly use deep learning to train a feature extraction model, and the resulting feature is a vector. Compared with traditional methods, deep learning methods have better tolerance for the variation of external environments and different hand postures. As for feature matching, it is to calculate the difference between the features to indicate the similarity, and finally judge whether it is the same person according to this similarity.

However, most deep learning methods for palmprint recognition only focus on a single dataset. Once identified on other datasets, the performance will be greatly degraded. In a practical application, it is common for training and real input images to come from different imaging acquisition devices because it is difficult to collect enough training images using the same device. Therefore, how to bridge the gap between images of different domains is of great research value.

doi:10.6342/NTU202203861

To address such a cross-dataset recognition problem, most of the past methods use the target dataset to assist, for example, model fine-tuning with a small number of target data, domain adaptation, or even training a generative adversarial network (GAN). These methods only focus on a single target dataset and are not helpful for data in other domains except GAN. However, the model training process of GAN is cumbersome and difficult to converge. Therefore, we focus on image augmentation to improve model adaptability fairly without the need for complex model training.

We observed that the main differences in the ROIs from different datasets lie in skin color, brightness, and shadow because the ROI cropping algorithm can well correct the orientation and scale of the images. Most of these changes can be simulated through image transformation. In addition, the drastic image changes allow the model to focus more on the texture and ignore other parts. Therefore, we decided to validate this idea by searching for an data warping augmentation strategy suitable for palmprint recognition through an optimization method. In addition, we have also improved the feature extraction model and feature matching method to make them more suitable for the palmprint recognition task. The main contributions of our work are summarized as follows:

1. We adopt Optuna, a search optimization framework, and TPE, a sampler, to perform hyper-parameter search and selection for image transformation. It turns out that **Brightness**, **Contrast**, **Hue**, and **Noise injection** are the most useful transformations among our pre-defined functions. This result is very consistent with our conjecture. Moreover, the transformations obtained through this search can improve the accuracy by more than 12.55% compared to the baseline in cross-dataset evaluation.

2. We refer to the technique of test-time augmentation and propose a **multi-transform matching**. This method generates rotated and mirror images from the input ROI for matching, and then combines the results to obtain the final similarity. On this basis, we propose an **oversampling** data augmentation method, which rotates ROI images to 4 orientations as training data. This training method can not only enhance the performance of multi-transform matching, but also make the model produce more discriminative features. Oversampling and multi-transform matching can improve the accuracy by more than 5.34% and 4.66% in the case of cross-dataset recognition, respectively.

3. We also propose a reduced version of **ResNeSt** with the last layer removed. According to experiments, such a structure is more suitable for palmprint

doi:10.6342/NTU202203861

recognition. The last layer usually has the largest number of parameters, so such adjustment can also make the model lighter. A lightweight model means less training data is required and faster convergence. The feature extraction model is proposed first because data augmentation can only make the model more familiar with existing tasks, but cannot improve the ability of the model itself, so we need to ensure that the model has sufficient capability.

The thesis consists of 6 chapters. Chapter 2 introduces the background knowledge and reviews the related works. Chapter 3 describes the details of baseline system implementation and the motivation. Chapter 4 explains our proposed methods, including the three points listed above. The evaluation of each method is shown in Chapter 5. Finally, Chapter 6 concludes this thesis.

doi:10.6342/NTU202203861

CHAPTER 2

BACKGROUND AND RELATED WORK

This chapter first describes the fundamental components of a palmprint recognition system in Section 2.1, along with an overview of the previous works in each part. In Section 2.2, we detail the definition of the recognition task, as well as their assumptions and respective evaluation methods. Then, an introduction to deep metric learning is provided in Section 2.3, since our work is based on this approach. Finally, we reviewed the related literature about cross-dataset palmprint recognition.

2.1 Palmprint Recognition System

A biometric system must include both registration and recognition. Registration converts images into features and stores them in the database, and the recognition can be operated either in verification mode or identification mode, depending on the application context [1]. The matching behaviors of these two modes differ in the number of templates for comparison, as depicted in Figure 1.

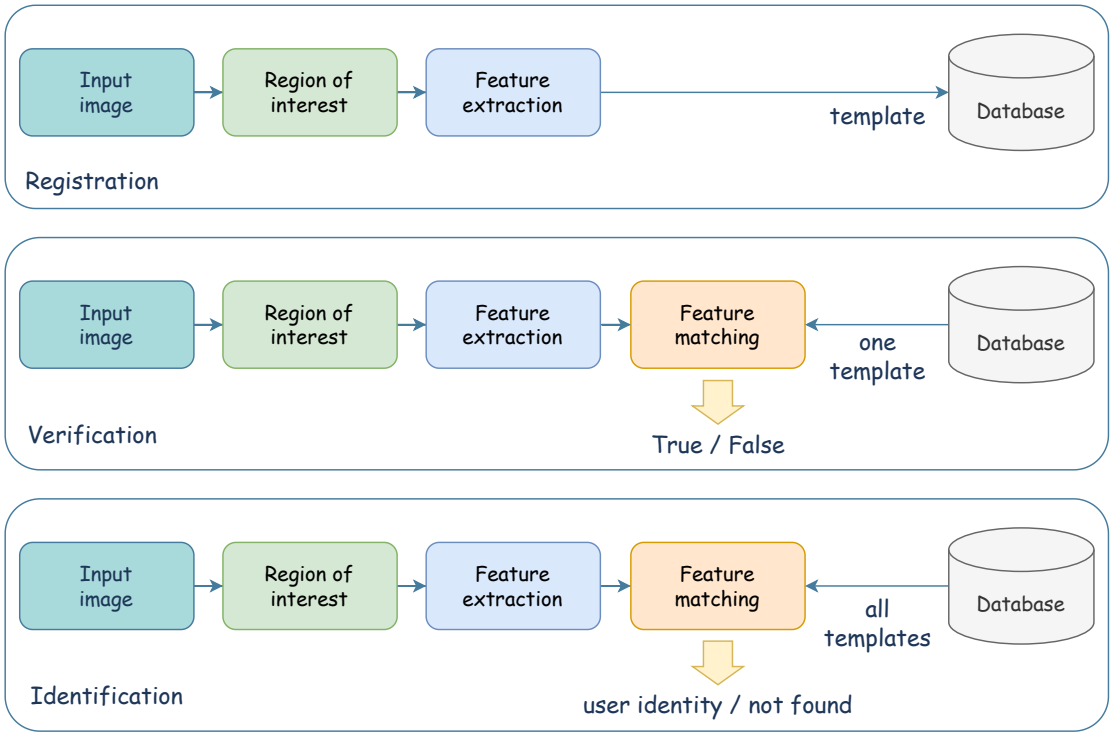


Figure 1: Block diagrams of registration, verification, and identification tasks in a palmprint recognition system, which is adaped from [1].

doi:10.6342/NTU202203861

The Palmprint recognition system is also a biometric system, including the above three operations, and they can be integrated into a flow chart as Figure 2. It consists of four major steps: image acquisition, region of interest cropping, feature extraction, and feature matching. This section will provide a comprehensive introduction to these four steps and review related literature on each step.

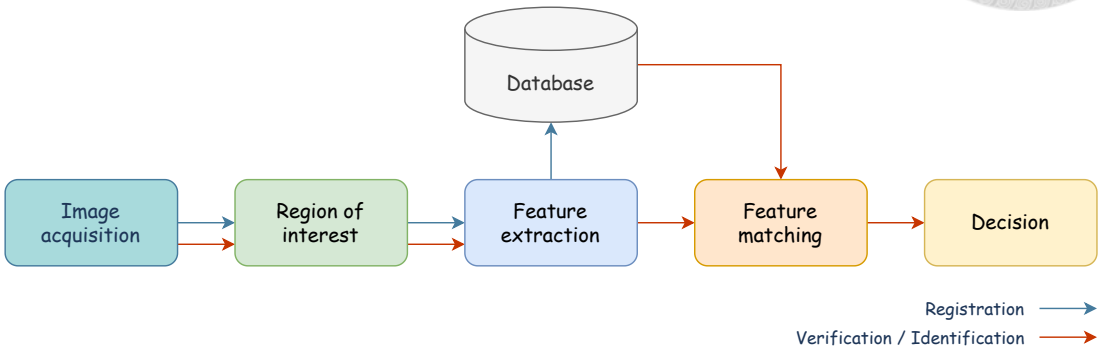


Figure 2: The flow diagram of palmprint recognition system, which is adapted from Zhong et al. [2].

2.1.1 Image acquisition

The palmprint image acquisition type was briefly categorized as follows: constrained acquisition, partly unconstrained acquisition, and unconstrained acquisition [18]. These three categories are classified according to the pose and the background.

- **Constrained acquisition:** The constrained acquisition means that users can only take one valid pose, and the image must be captured on a uniform background. Most of them will have the user place their hands directly on the device and secure with hooks. An example from the PolyU dataset [3] is shown in Figure 3.

- **Partly unconstrained acquisition:** Only one of the background and the pose would be limited by the partly constrained acquisition. Tongji contactless palmprint dataset [4] falls into this category. The picture of their device is shown in Figure 4. They allow the user to change the gesture slightly but must be inside the device.

- **Unconstrained acquisition:** The unconstrained acquisition is the opposite of the constrained acquisition, with no restrictions on the pose and the environment. Generally speaking, the palm will be opened as much as possible to make the part of the palmprint flat and clear.

doi:10.6342/NTU202203861

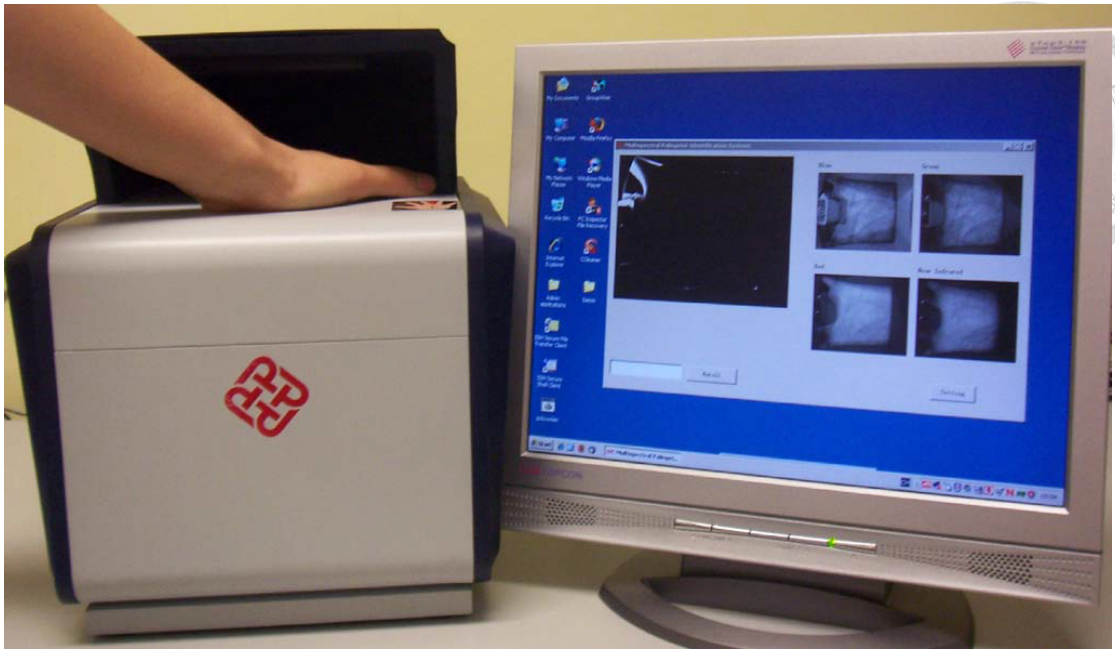


Figure 3: The image acquisition device of PolyU-M palmprint dataset from Zhang et al. [3]

On the other hand, sensing techniques have also been developed to improve accuracy, reduce error rate, or be suitable in specific scenarios. Grayscale and RGB images are the most common choices. There is also some research using multispectral images, 3D images, or minutiae palmprint images [2]. Multispectral images can provide more information about the palm. A common multispectral image is to utilize near-infrared light. It can penetrate subcutaneous tissues and is absorbed in the vein vessels. As a result, the near-infrared image would reveal more vein patterns than the palmprint, which is another unique human pattern. When these two patterns are combined, they may produce better performance. A 3D hand image can generate a lot of 2D palm images with different angles, which is very useful in an unconstrained scenario. The minutiae palmprint image is similar to the fingerprint image, which is a high-resolution image that contains the whole palm. Nevertheless, it can only be used in contact devices. All the image examples captured with the techniques mentioned above are demonstrated in Figure 5.

A variety of palm-related image datasets have been proposed in previous studies. Ungureanu et al. [18] have detailed a lot of palmprint datasets and classified them based on the restrictions imposed on the user during the acquisition process. They focus on the palmprint images in the visible spectrum, and the other types are not included because the visible spectrum is the most commonly used in the research. From the paper, it can be found that the research is moving away from constrained to unconstrained acquisition and from a fixed camera device to

doi:10.6342/NTU202203861

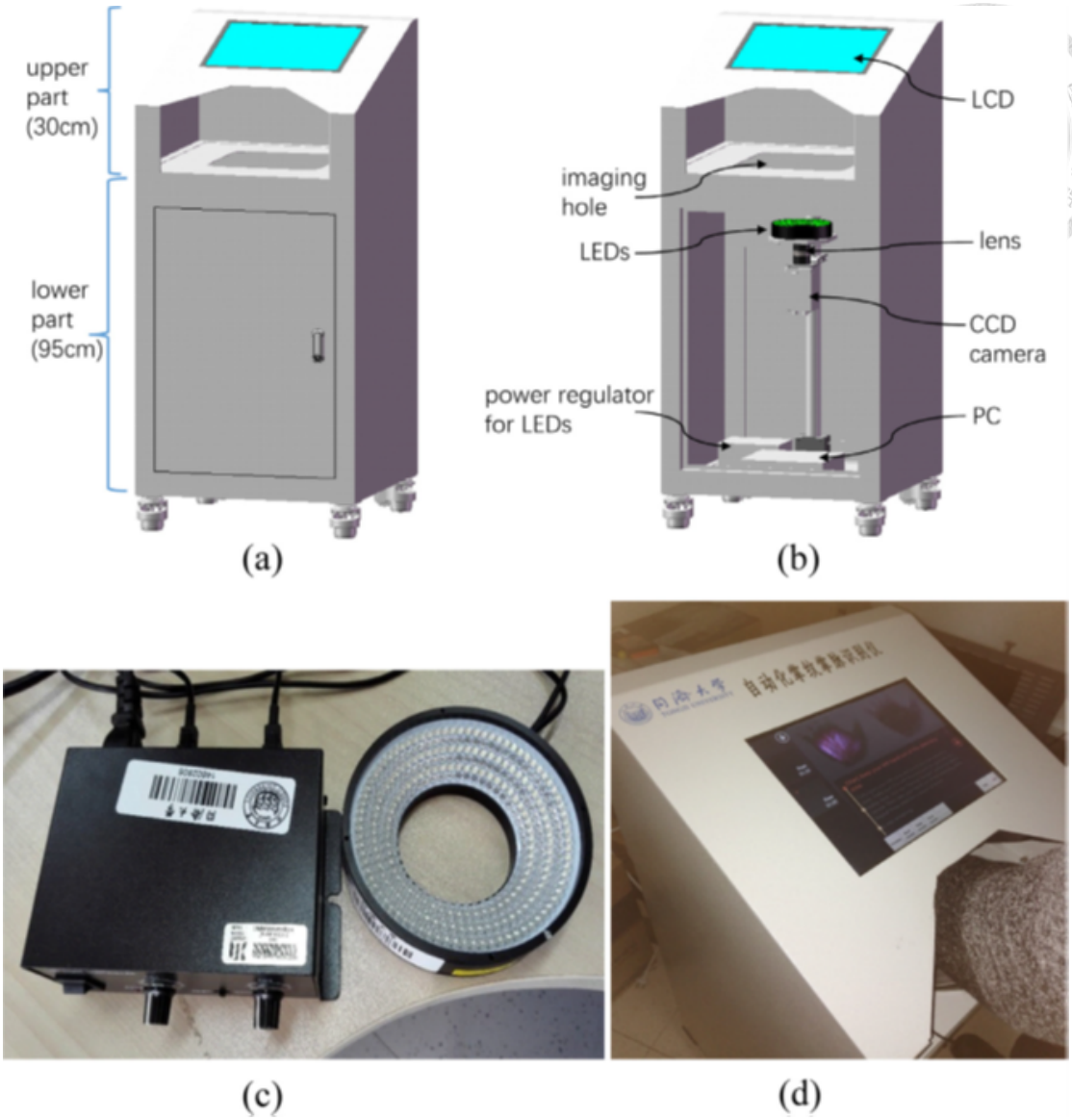


Figure 4: The image acquisition device of Tongji contactless palmprint dataset from [4]

a mobile phone. The experiment setup is getting more general and practical.

Among these datasets, there are several popular choices in past research: CASIA-M [19], PolyU-M [3], and IIT-D v1 [23]. They are all constrained palmprint datasets, whereas CASIA-M and PolyU-M are multispectral palm image datasets that contain spectral images other than RGB. Recently, many large datasets have appeared, such as Tongji datasets [16, 20] and HFUT [7]. They collected over 10,000 hand images, providing a richer basis for testing. In addition, datasets obtained through mobile phones have gradually become the main research material, because their conditions are more life-like and complex. As long as there are good results on such a dataset, it can be easily applied to various scenarios. For example, Tongji MPD [24] and XJTU-UP [25].

doi:10.6342/NTU202203861

(a) Grayscale image [19]

(b) RGB image [16]

(c) NIR image [20]

(d) 3D image [21]

(e) High-resolution palmprint image [22]

Figure 5: Images acquired using different sensing techniques

2.1.2 Region of interest (ROI)

Preprocessing is essential to provide a stable and enhanced image for feature extraction. Because any small issue, such as illumination conditions, backgrounds, closed fingers, and others, significantly impacts the quality of the extracted features. As a result, ensuring that all input images are at the same level is indispensable. In palmprint recognition, extracting the region of interest (ROI) is the main step apart from other procedures like image enhancement, image filtering, etc [2].

Palmprint features are mainly distributed in the palm area, and the most prominent are the three main lines and several wrinkles [26]. However, a palm image contains a lot of redundant information, which has a negative effect on feature extraction. For instance, the background may differ, and the pose of fingers may also affect the recognition result. Therefore, it is suitable to crop the area that contains most of the information we are interested in, which is also the region of interest (ROI). Another benefit of the cropped ROI is to adjust hand image rotation, translation, and scale normalization [7]. Therefore, obtaining a stable ROI is crucial to the performance of recognition results. In actual scenes, a successful ROI extraction algorithm requires obtaining stable ROI from palms of different poses and sizes under a complex background [26].

The classical ROI extraction algorithms set up a coordinate system based on

doi:10.6342/NTU202203861

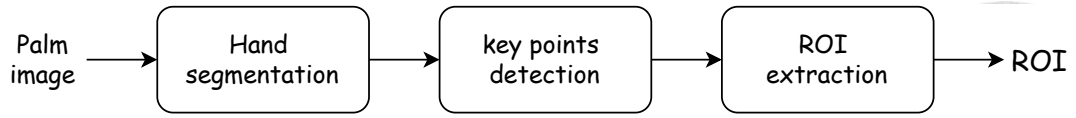


Figure 6: Key steps in segmenting ROI images for contactless palm images, which is adapted from Zhou and Kumar [5].

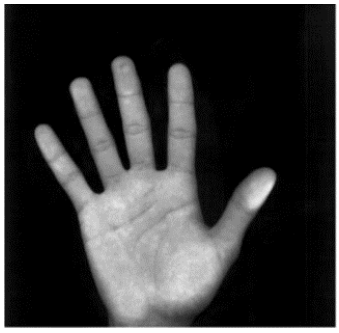
detected key points, which are between fingers [27]. They have four common steps: (i) hand segmentation, (ii) key points detection, and (iii) ROI extraction. The flow diagram is shown in Figure 6.

One example is the work of Lin et al. [6]. To separate the palm area from the background area, they binarized the palm image first. Then the hand contour is discovered, the sample is shown in Figure 7(c). After that, the distance between the binarized palm's wrist point and the palm's boundary is calculated. They can then use the computed distance to find four local minimum points. Among these points, they only use two key points: one is the valley point between the index and middle finger, and the other is the valley point between the ring and little finger. So far, the key points are located as shown in Figure 7(e). Based on the length between these two key points, they calculate the distance apart from this line and the side length of the ROI. Using a scale of the length between selected key points is natural for dealing with potential scale changes in the contactless environment. Finally, the region of interest could be located and cropped. Figure 7 depicts the details.

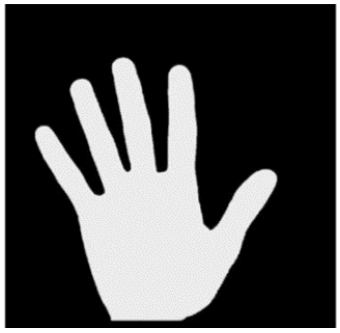
One of the advantages of the ROI segmentation described earlier is that it removes areas of the image that are badly affected. However, this process may drop too much information because the cropped area only contains the center of the palm area. Thus, some work is devoted to maximizing the ROI size and accommodating more biometric information. Ma et al. [28] employed the widely used method of extracting hand contours and locating peak and valley points to discover key points. However, instead of selecting the valley point between the ring and little fingers, they choose the valley point between the thumb and index finger. Then, they draw a line from the valley point between the middle finger and index finger through that point to the boundary of the hand contours. A square ROI box can be obtained by this line segment.

Another work proposed by Nikisins et al. [29] extracts an irregular shape of ROI by excluding hand edges and parts near fingers because these regions can be affected by ambient light and shadows. They employ a filter to detect all half-moon shapes on the binarized image, then compute the squared distances between each pair of points to see if there are exactly four valley points. The ROI may be derived if done correctly.

doi:10.6342/NTU202203861

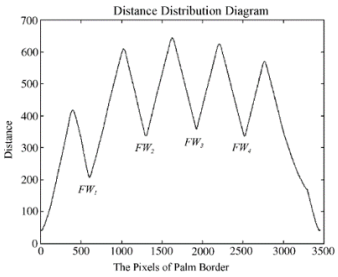


(a) original hand image

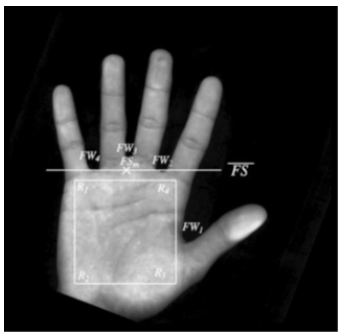


(b) binarized hand image

(c) the boundaries of the hand



(d) the contour profile distance distribution



(e) the new coordinate system in hand image

(f) cropped ROI image

Figure 7: Illustration of segmentation of the ROI, which is adapted from Lin et al. [6].

From the example described in Figure 7(e), we know that the ROI extraction is based on the key points. If we can get key points, then we can get the ROI. Preprocessing techniques such as binarizing and edge detection aim to obtain the key points, but this kind of operation is sensitive to the background and illumination issues. As a result, the ROI extraction study is mostly focused on detecting key points. Xiao et al. [7] is a novel method. They consider that straight line clusters can easily detect the fingers. They first binarized the hand image and defined several rules to form line clusters. If any line has eight intersection points between a straight line and the hand region, they will conclude that this line passes through four fingers. They then used K-means clustering to obtain the center of each cluster. The advantage of this approach is that it does not need to extract the hand's contour, making it more robust.

Deep learning methods are good at adapting to a different environment, so they have been used in several studies to detect key points [30, 31]. Zhang et al. [24] proposed DeepMPV+, which contains an ROI extraction module and an ROI matching module. They use an object detection method to solve the ROI extraction problem. Modern CNNs achieve good performance and are highly

doi:10.6342/NTU202203861

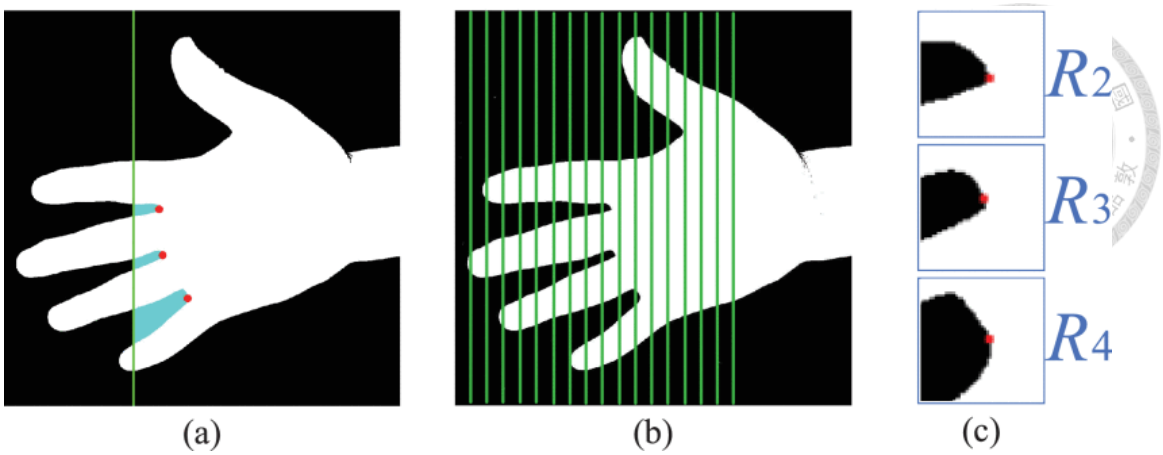


Figure 8: A example to find key points with line clusters from Xiao et al. [7]

robust to complex backgrounds. Luo and Zhong [26] further combined object detection and key point detection to extract ROI, and added an auxiliary network to estimate the palm angle. Several ROI extraction methods introduced above are listed in Table 1.

Some research avoids using ROI extraction. They provide a guide on the screen of the phone during acquisition, indicating reference points [32] or specific hand pose [33]. Moreover, Afifi [34] and Kuzu et al. [35] use the whole image as an input to a CNN, so they do not need the step of ROI extraction. However, this kind of approach is more sensitive to the acquisition environment.

Table 1: Comparison of different ROI extraction methods.

Method	**Hand segmentation**	**Key points detection**	**ROI extraction**
Lin et al. [6]	threshold	distance distribution	P2,P4
Ma et al. [28]	threshold	-	P1,P2
Nikisins et al. [29]	threshold	morphological image filter	P1,P2,P3,P4
Xiao et al. [7]	threshold	line clusters	P2,P4
Zhang et al. [24]	-	Tiny-YOLOv3	Palm center and the midpoints of (P2,P3) and (P3,P4)
Luo and Zhong [26]	Tiny-YOLOv3	MobilenetV2	P2,P4

Note: P1, P2, P3, and P4 are four valley points from the thumb to the little finger.

doi:10.6342/NTU202203861

2.1.3 Feature Extraction

Feature extraction aims to generate a template that represents the identity. It will be stored in the database and compared with the input images. The main objective is to make the features as discriminative as possible when they come from different identities, and make the features similar if they are from the same identity. Secondary considerations include reducing storage costs and shortening recognition time. A lot of feature extraction algorithms have been developed, not only for the source of features but also for ways to record and compare features efficiently. According to the survey [2], there are five main categories of feature extraction methods: encoding-based, structure-based, statistics-based, subspace-based, and learning-based. Without a doubt, some methods are a fusion of these methods. Note that the feature data produced by these methods generally fall into the image and code template regardless of fusion or not. The following is a brief introduction to these five categories.

- **Encoding-based Methods:** The encoding-based methods transfer an image to coded information with predefined filters and generate code according to certain principles. The codes will be compared using binary arithmetic operations and output a matching score. For examples, RLOC [36], BOCV [37], Ordinal code [38], PalmCode [39], CompCode [40] and OLOF [3] are representative methods. These ways are more suitable for time-critical applications.

- **Structure-based Methods:** The structure-based methods are traditional recognition techniques that come from fingerprint recognition. They make use of edge detection algorithms to discover the orientation and location of ridges, lines, and feature points. Then they construct a template with these lines and points. The shortcoming of these methods is the demand for photo quality. They require a high-resolution image source. One recent work is proposed by Chen and Guo [41].

- **Statistics-based Methods:** This statistics-based method is based on statistical conceptions of an image such as mean, variance, standard deviation, density, and invariant moment. They can be divided into two categories according to whether a transformation is used. Often used transformations are the wavelet transform and Fourier transform. They can represent the multiscale information of a palmprint image in the frequency domain, but the work should be done in a small region. That means the methods of transformation type are local-based. A novel approach [42] combined Local

doi:10.6342/NTU202203861

Binary Pattern Histogram (LBPH) and Dual-Tree Complex Wavelet Transform (DT-CWT). The final representation of the feature was a weighted histogram set.

- **Subspace-based Methods:** The subspace-based methods originate from face recognition. This method will convert a palmprint image into a low-dimensional representation by projection or mathematical transformation. There are three common subspace algorithms: principal component analysis (PCA), independent component analysis (ICA), and locality preserving projection (LPP). Unlike the previous, linear discriminant analysis (LDA) will use label information to improve performance. Gabor filters, noise reduction, image modification, kernel function, and other approaches could be simply integrated with this type of method to increase robustness and efficiency. One example that uses PCA is Bai et al. [43]. They combined surface type (ST) features and PCA for 3D palmprint identification.

- **Learning-based Methods:** Machine learning and deep learning have been widely used for image-related tasks in recent years, as well as the palmprint recognition task. Many researchers apply machine learning not only for feature extraction but also for classification. In machine learning methods, clustering and SVM are more commonly used in the palmprint recognition task. As for deep learning methods, there are three major components: model structure, loss function, and optimization strategy. The model structure determines the model capability, and the loss function is the optimization objective. The advantage of deep learning is that it can adapt to changes in the environment, such as illumination and shadows. However, they require more training data along with accurate labels. Deep learning can be used in a variety of ways, including feature extraction with CNN, classification with DNN, and comparison with the Siamese network. One of a kind is deep metric learning, which aims to maximize the distance between features belonging to different identities and the similarity of features belonging to the same identity. A recent method is called C-LMCL [8], which was proposed by Zhong and Zhu. They developed a new loss function that combined center loss and large margin cosine loss, making the model a more discriminative feature. Section 2.3 will go into greater detail.

Combining the methods mentioned above is also common. Zhang et al. [4] proposed CR_CompCode, which combines CompCode and block-wise statistics. This method generates a CompCode map from the ROI and divides it into uniform blocks. The output feature vector is the histogram that is computed from

doi:10.6342/NTU202203861

these blocks. The CompNet [44] is developed for contactless palmprint recognition, which uses a competitive convolutional neural network with constrained learnable Gabor filters to reduce the number of parameters in typical CNN. PalmNet [11] is also the fusion of CNN and other methods, aiming to achieve high accuracy with touchless palmprint samples captured using different devices. They proposed a novel CNN that uses unsupervised learning based on Gabor responses and principal component analysis (PCA) to tune palmprint-specific filters.

2.1.4 Feature Matching

Feature matching is the final step of palmprint recognition, which aims to find the person corresponding to the input image or reject the imposters. Usually, this is highly correlated with feature extraction, since the feature extraction method would pre-determines the target matching method to optimize. Generally, there are two common types of matching. One is to apply a classifier that would directly output the result regardless of who the user is or whether these two features belong to the same person. The classifier could be a decision tree, support vector machine, multilayer perceptron, and so on. The other is to measure the similarity or distance of each feature pair with a metric function and find the most similar pair or compute the similarity score for the decision. Among the metric functions, Hamming distance, Euclidean distance, and cosine similarity are the most popular choices. Below is a brief explanation of these three metric function.

- **Hamming distance:** The Hamming distance between two feature vectors is the number of positions at which the corresponding indexes differ. If the vectors are in the binary format, they can be computed with the Exclusive-OR operation, which is an extremely fast method. However, the shortcomings are also obvious. This data format will cause a large amount of information loss, and its discriminative power is also weaker than others.

- **Euclidean distance:** The Euclidean distance is a widely used function, which could be used in most cases. It calculates the mean square error between the two feature vectors, which can also be seen as the length of the straight line between these two points in the hyperspace. Given two feature vectors $x, y \in \mathbb{R}^n$, the formula is expressed as:

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}. \tag{2.1}$$

- **Cosine similarity:** The cosine similarity is getting more and more attention recently due to the development of face recognition. This function has several useful properties discovered in recent face recognition research, which will

doi:10.6342/NTU202203861

be discussed in Section 2.3. The cosine similarity can be derived by using the Euclidean dot product formula. Given two feature vectors $A, B \in \mathbb{R}^n$, the formula is represented as:

$$\cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{2.2}$$

where A and B are two input vectors. Note that cosine similarity is not a proper metric. Another formal distance metric is the angular distance, which could be derived from cosine similarity, but the computing cost is much higher than using the cosine similarity. Therefore, cosine similarity is more common.

2.2 Evaluation of the Palmprint Recognition System

In Figure 1, we briefly mentioned the two recognition modes of the palmprint recognition system but did not describe them in detail. Therefore we will explain their differences and define their recognition process in Section 2.2.1. The next subsection is evaluation methods for the above two tasks, but we mainly focus on identification mode. We first define the open-set identification problem, and then two simulation methods will be introduced according to different situations.

2.2.1 Verification and Identification

In the verification mode, users have to claim an identity and provide their biometric data. They may use a personal identification number, a user name, or other tokens that can represent their identity. Then, the system will compare the stored template of that identity with the captured biometric data, which is called an *one-to-one comparison*. Verification is typically used for positive recognition in order to prevent multiple people from using the same identity.

In the identification mode, users only need to provide their biometric data. The system will compare it with all the identities in the database, called a *one-to-many comparison*. There are two meanings or two usages for it. One is to provide convenience to users. They no longer need to bring their token or remember any information. The other is used as negative recognition to prove the person is not someone else.

The verification problem can be formulated as follows [1]: given an input feature vector X_Q, which is extracted from user's biometric data, and a claimed identity I, determine if (X_Q, I) belongs to class w_1 or w_2, where w_1 indicates that the claim is true (a genuine user) and w_2 indicates that the claim is false (an imposter). Typically, X_Q is matched against X_I, which is the biometric template

doi:10.6342/NTU202203861

of the claimed identity I, to determine its class. Thus

$$(X_Q, I) \in \begin{cases} w_1 & if\ S(X_Q, X_I) \geq t \\ w_2 & otherwise, \end{cases} \tag{2.3}$$

where S is the function that measures similarity between feature vectors X_Q and X_I, and t is a predefined *threshold*. The output of the similarity function S could be seen as a similarity or matching score between the two input vectors. Then, the threshold t is used to determine whether the two feature vectors are similar enough to prevent imposters. Therefore, every claimed identity is classified into w_1 and w_2 based on the variables X_Q, I, X_I, a predefined threshold t and a similarity function S.

On the other hand, the identification problem can be stated as follows [1]: give an input feature vector X_Q, determine the identity $I_k, k \in \{1, 2, ..., N, N+1\}$ corresponding to the input feature vector. The identities from I_1 to I_N are the identities enrolled in the system, and I_{N+1} indicates the rejected case where the system can not find a suitable identity corresponding to the input feature vector. Hence

$$X_Q \in \begin{cases} I_k & if\ \max\limits_k \{S(X_Q, X_{I_k})\} \geq t, k = 1, 2, \ldots, N \\ I_{N+1} & otherwise, \end{cases} \tag{2.4}$$

where X_{I_k} is the biometric template corresponding to identity I_k, S is a similarity function, and t is a predefined threshold. The output of the similarity function is denoted as the matching score, just like verification.

2.2.2 Evaluation Methods

We have covered the entire process of a recognition system, which contains registration and recognition. To evaluate such a system, we first need to split the dataset to simulate the registration and input images. The straightforward idea is to split the palm images of the same people in half. The first half is regarded as training set, while the second half is considered testing input. This kind of testing scenario is the same as a classification problem, also known as a *close-set* recognition problem. However, this is not sufficient to test a palmprint recognition system. Thus, we will introduce a more realistic setting, the *open-set* recognition problem.

The main difference between the open-set and the close-set recognition problem is the testing identities. In the close-set recognition problem, all the testing identities are seen in the training phase. In contrast, we do not have complete knowledge of the world at the training phase of the open-set recognition problem,

doi:10.6342/NTU202203861

and unknown identities will be submitted to the system during testing. The open-set recognition problem is closer to the real application. As a result, the open-set recognition problem may be carried out in various forms. First, we need to clarify that known identities in the open-set recognition problem are not trained identities. It indicates the identities in the registration set, also known as the enrolled identities. From this point of view, the unknown identities are not only the identities that have never been seen but also include the training identities that do not appear in the registration set. The following is a brief explanation.

- **Close-set Recognition Problem:** Assuming the world has only a finite number of classes, so defined as *close-set* recognition. Hence, we can tackle all the classes and construct a classifier to recognize them.

- **Open-set Recognition Problem:** Considering the world has an unlimited number of classes, we can only know a part of them. This is, therefore, defined as *open-set recognition*. We do not need to recognize the unknown classes and do not care about the type of unknown class. We only need to focus on the known classes and the boundaries between the known and the unknown.

To simulate the open-set recognition problem, we have to split the test set into registration images and test images. However, the image sources for newly registered biometric templates may come from different acquisition equipment. Therefore, for different registration situations, we can simulate in different ways. Cross-subject evaluation can be applied when the newly registered identity images are from the same acquisition device as the training images. If they are different, cross-dataset evaluation may be more appropriate [8]. An overview of these two evaluations is described below and shown in Figure 9.

- **Cross-Subject Evaluation:** The unknown identities are simulated with the image data that comes from the same dataset.

- **Cross-Dataset Evaluation:** The testing set for validating the system does not contain the image data that comes from the same dataset as training data.

The cross-dataset evaluation can test the system more comprehensively, both in variance and volume. Normally, a dataset would be divided into a training dataset and a testing dataset, but the amount of testing data is much smaller. Especially in palmprint datasets, most of the publicly available datasets are not so big compared to others, like face recognition or image classification, and so on. For that, the cross-dataset evaluation can further combine multiple datasets to look into the robustness of the system.

doi:10.6342/NTU202203861

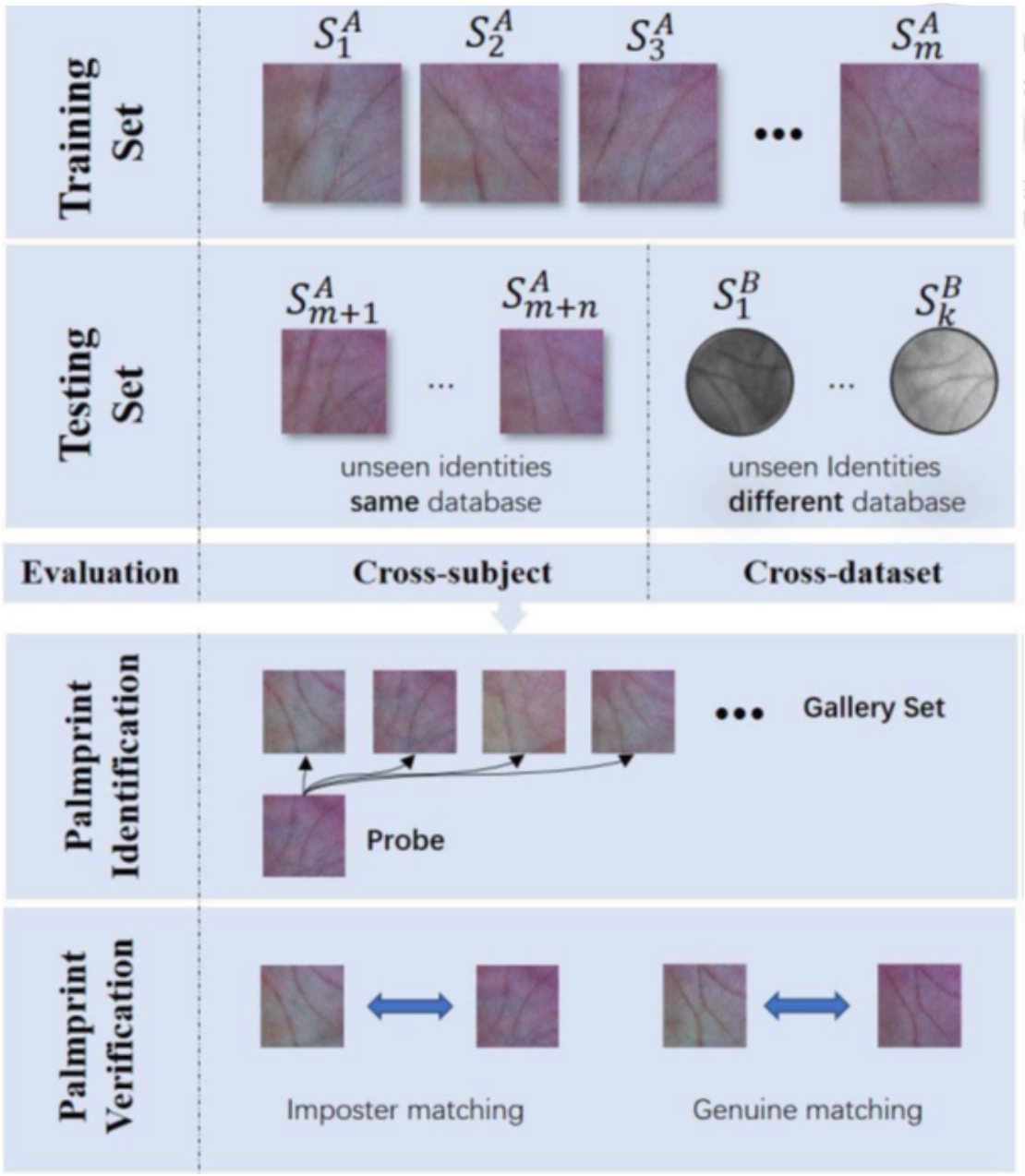


Figure 9: An overview of the open-set recognition from Zhong and Zhu [8].

2.3 Deep Metric Learning

Deep metric learning uses a deep learning method to train an embedding neural model, which is also known as a feature extractor. The model can minimize the distance between the samples that belong to the same class, and maximize the distance of different class samples. An example demonstrating the process of feature extraction with a CNN model is shown in Figure 10. This technique has gained great success in face recognition [45] and image retrieval [46], while it is still not used a lot in the field of palmprint recognition.

doi:10.6342/NTU202203861

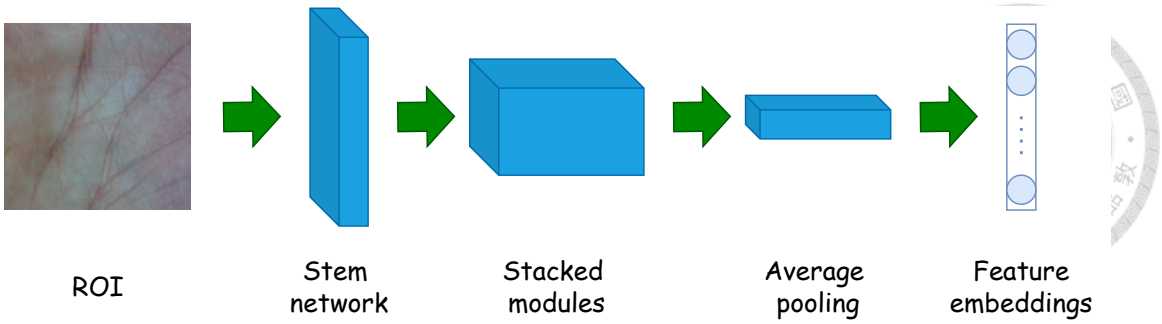


Figure 10: An example of extracting feature template using deep metric learning model.

A *metric* is a non-negative function that measures the so-called notion of "distance" between two points. Thus, given a metric function $d : \mathbb{R}^n \to \mathbb{R}$ (which is usually predefined), two data samples $x_1, x_2 \in X$ along with their labels $y_1, y_2 \in Y$, and a feature extractor $f : X \to \mathbb{R}^n$, the combination $d(f(x_1), f(x_2))$ should produce small value while the labels y_1 and y_2 are equal, and larger value if they are not [47].

Deep learning has three main components: model, loss function, and optimizer, and so does deep metric learning. To obtain the goal described above, deep metric learning mainly focuses on designing the loss function to form different feature distributions. The most popular loss functions are all from face recognition, and the development trend is from contrastive-based to cosine-based loss functions. For example, contrastive loss [48] and triplet loss [49] are well-known contrastive-based loss functions. And the CosFace [50] and ArcFace [10] are state-of-the-art cosine-based loss functions. Since cosine-based loss functions are the mainstream today, we will only briefly introduce cosine-based loss functions here.

All the cosine-based loss functions originate from softmax loss. The softmax loss is a combination of the softmax function and cross-entropy loss. It was used for classification problems, so it would only produce separable features. However, deep metric learning aims to obtain discriminative features because there are inestimable categories to classify in the recognition tasks. If the distance between each category is not large enough, it is more likely to cause recognition errors when dealing with unseen samples. An illustration of the difference between these two types is shown in Figure 11. Hence, those improved loss functions added restrictions on the softmax loss to enlarge the decision margins. The following are some critical cosine-based loss functions.

- **Center Loss:** Center loss adds an L2 distance regularization term to the softmax loss. In order to solve the issues in the previous contrastive-based

doi:10.6342/NTU202203861

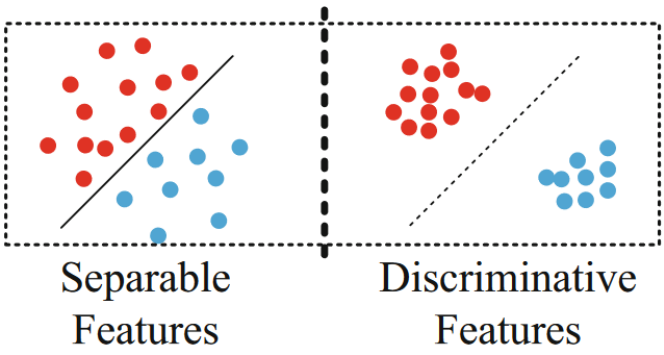


Figure 11: The difference of feature distribution requirement in classification task and recognition task from Wen et al. [9]

losses, such as sampling issues and so on. It was one of the successful attempts to combine softmax loss in deep metric learning. Although it relies on the softmax loss, the metric function in feature matching is still L2 distance. It starts the research based on softmax loss in deep metric learning. The loss function of the center loss for a single sample can be expressed as:

$$L_{center} = L_{softmax} + \frac{\lambda}{2}\|X_i - C_{y_i}\|_2^2, \tag{2.5}$$

where X_i is the feature extracted from the input sample and C_{y_i} is the center vector of the class of X_i.

- **SphereFace:** The center loss still has no guarantee of large inter-class variability since the clusters closer to zero will benefit less from the regularization term. To solve this issue, SphereFace has improved the softmax loss and uses cosine similarity as a matching metric. First, it regularizes the weight norm $\|W_i\| = 1$ to avoid the feature cluster close to zero. That is, it enforces the class centers to be at the same distance from the center. Second, it introduces the concept of margins into the cosine function. The loss function of the SphereFace for a single sample can be expressed as:

$$L_{SphereFace} = -\log \frac{e^{\|X_i\| \cos(m\theta_{y_i,i})}}{e^{\|X_i\| \cos(m\theta_{y_i,i})} + \sum_{j \neq y_i} e^{\|X_i\| \cos(\theta_{j,i})}} \tag{2.6}$$

where the angle $\theta_i \in [0, \frac{\pi}{m}]$. They create another function to avoid this annoying restriction to replace the cosine function, but we will not introduce it here. The detail is in [51].

- **CosFace:** The decision margin of SphereFace depends on θ, which leads to different margins for different classes. As a result, in the decision space, some inter-class features have a larger margin while others have a smaller margin, which reduces the discriminating power. CosFace proposes a simpler

doi:10.6342/NTU202203861

yet more effective way to define the margin. First, they regularize not only the weight norm but also the feature norm $\|X_i\| = 1$. Second, the margin changed to be a subtraction term of cosine value. The loss function of the CosFace for a single sample can be expressed as:

$$L_{CosFace} = -\log \frac{e^{s(\cos(\theta_{y_i,i})-m)}}{e^{s(\cos(\theta_{y_i,i})-m)} + \sum_{j \neq y_i} e^{s\cos(\theta_{j,i})}}, \tag{2.7}$$

where the s is scale parameter and m is the margin parameter.

- **ArcFace:** ArcFace is very similar to CosFace and addresses the same limitations of SphereFace as mentioned in the CosFace description. However, instead of defining the margin in the cosine space, it defines the margin directly in the angle space. The loss function of the ArcFace for a single sample can be expressed as:

$$L_{ArcFace} = -\log \frac{e^{s(\cos(\theta_{y_i,i}+m))}}{e^{s(\cos(\theta_{y_i,i}+m))} + \sum_{j \neq y_i} e^{s\cos(\theta_{j,i})}}, \tag{2.8}$$

where s is the scaling parameter and m is referred to as the margin parameter.

The comparison of the decision boundary and decision margin is shown in Figure 12. There is not much difference in performance between CosFace and ArcFace. Although there are several improved losses, for example, in parameter choosing, dynamic margin, or multiple centers, both of them are still widely used in various tasks nowadays.

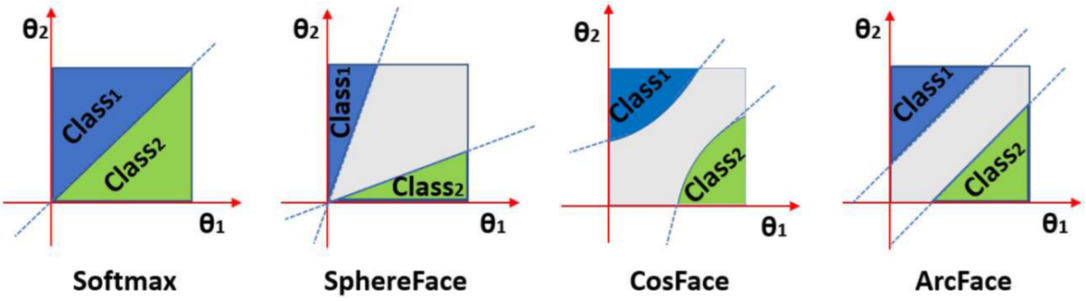


Figure 12: Decision margins of different loss functions under binary classification case from Deng et al. [10]. The dashed line represents the decision boundary, and the grey areas are the decision margins.

2.4 Related Work

Research on palmprint recognition has been going on for a long time, at least before the 2000s. It can be divided into two periods according to the rise of

doi:10.6342/NTU202203861

CNN. This is because most methods after that will use CNN, whether it only uses CNN for feature extraction or combines CNN with previous methods. In contrast, the method before CNN is usually called the conventional method. Many conventional methods have been introduced in Section 2.1.3, of which the encoding-based method is the most effective. For instance, RLOC [36], CompCode [40], OLOF [3] and so on.

Y. Liu and A. Kumar have summarized the limitations of the conventional methods in [52]. First, although these methods offer quite accurate performance, they need to be further improved on large contactless datasets. Second, they applied hand-crafted filters for the generation of features. Therefore, they heavily rely on parameter selection when operating in different imaging environments.

The CNN-based approaches have the potential to address the above limitations. Since deeply learned architecture is known to offer higher generalization capabilities for image-related tasks. Compared to the empirical selection of hand-crafted filter parameters for palmprint matching, the parameters in CNN can be learned from training data. Therefore, it is now widely used in the field of palmprint recognition.

Deep metric learning is one of the CNN-based methods, and it has achieved good performance in face recognition. It indicates that it can identify a large number of images in a complex environment. The details of deep metric learning have been introduced in Section 2.3. However, there is a huge difference between the images of the face and hand. Moreover, the training data for the hand is also much less than the data in the face dataset.

Therefore, it is difficult for us to directly train a powerful model with a large number of complex images. In practical application scenarios, it is very likely to receive input images from different imaging conditions. If we use a dataset different from the training data for testing, the accuracy will suffer from the gap between different datasets. As a result, how to use this technique to solve the problem of palmprint recognition in complex environments remains to be studied. So far, there are three main directions to increase the robustness of deep metric learning models in palmprint recognition: metric loss, domain adaptation, and generative adversarial networks.

2.4.1 Metric Loss

These kinds of methods enhanced the metric losses originally used in face recognition based on the domain knowledge of palmprint recognition. They optimize the feature space distribution of data samples by adding some regularization into the loss functions to restrict the relationship between input samples. If the model

doi:10.6342/NTU202203861

can better classify the training data, it is more likely to be able to classify images from different environments.

Zhu et al. [53] proposed an adversarial metric learning methodology to make different categories of palmprints uniformly distributed in the hypersphere embedding space. They discovered that the features in the embedding space would be grouped into two clusters. As a result, they added a confusion term to the contrastive-based losses to optimize the feature distribution. The confusion term will shorten the distance between different classes, which is the opposite of the contrastive-based loss. Hence, they named it adversarial metric learning. However, the two clusters may be the left and right hand. They did not delve deeply into the causes of the feature distribution. Moreover, such contrastive-based losses suffered the sampling issue, so most of the current deep metric learning methods use cosine-based losses instead.

D. Zhong and J. Zhu [8] developed a new loss function, centralized large margin cosine loss (C-LMCL), which is a combination of the center loss and large margin cosine loss (LMCL). The LMCL is the same as CosFace, which we have explained in Section 2.3. This combination provides a stronger gathering ability and will result in a very small circular region in the embedding space for each category. They aim to train a model with C-LMCL that has the ability to generalize across unseen subjects and different datasets. Although the loss function is well designed, their model is not that well. Our reproduction can not achieve the result shown in the paper. There will be a very large difference in cross-dataset evaluation.

2.4.2 Domain Adaptation

Domain adaptation is a group of techniques to fit the deep learning model onto a new dataset with a domain shift from the training dataset. As we mentioned above, we usually use a second dataset to evaluate the robustness of palmprint recognition. Thus, some research regards it as a cross-domain problem. While this assumption is somewhat different from the original goal, it is still a way to strengthen feature extraction models, albeit on only one dataset at a time. This idea is straightforward. Deep learning relies on vast and various training data, but there is insufficient data with only one dataset. Hence, they take part of the data in the target dataset and try to unify the feature distributions of the source and target datasets. To be precise, it is to pull the feature distribution of the target data closer to the source data.

H. Shao and D. Zhong [54] propose a Learning with Partners (LWP) framework to improve multi-source cross-dataset palmprint recognition. This framework requires multiple source datasets with labels and an unlabeled target dataset, all of

doi:10.6342/NTU202203861

which have the same identities. Moreover, the scenario is to find the corresponding identities in the source dataset. They first train a teacher network with the source datasets. The same target extractors then process the source and target datasets and can be viewed as students or partners. These target extractors can not only learn from the teacher but also get guidance from other partners. They use multiple losses to optimize such a class-like framework. However, this application scenario is not practical because it is unlikely that a new target dataset has exactly the same identity as the training data. In addition, this method needs to train multiple models, which requires a lot of computing and time costs.

Du et al. [55] follow the ADDA approach [56], which is a two-step training process, and combine it with Maximum Mean Discrepancy (MMD). They built it on top of the Deep Hashing Network (DHN) to solve such a cross-domain palmprint recognition problem. They named it the Regularized Adversarial Domain Adaptative Hashing method (R-ADAH). It first trains a source feature extractor in a supervised manner and then trains a target feature extractor in an unsupervised manner with the help of the source feature extractor. However, they not only build a domain discriminator for adversarial training but also use an MMD loss to shorten the feature distribution distance between the outputs of the source and target feature extractor. Although it does not need the identity labels of the target dataset, it still requires the domain labels and adequate target images for domain adaptation. Otherwise, the domain discriminator will suffer from the data unbalance and cause a great performance drop.

H. Shao and D. Zhong [57] solve the problem of data unbalance by introducing FADA to replace the ADDA. This method can only use one labeled sample per subject in the target palmprint dataset to make the model adapt to the new domain. FADA and ADDA are mostly the same, but they can afford a relatively small amount of target data. They train the domain discriminator by combining the features from both source and target data and label them as four types according to the domain and identity. Even though this method no longer requires many target images, it requires the target images to have the same identities as the training data. It is still a very restricted situation, which rarely happens in real-world applications.

2.4.3 Generative Adversarial Network

Since the main factor limiting the performance of the model is the lack of training data, if we could generate a lot of fake training data, it might improve the generalization ability of the model. Generative adversarial network (GAN) is one of these kinds of techniques that can produce realistic images. Although the

doi:10.6342/NTU202203861

improvement is not limited to a specific domain, these kinds of methods need to use more data than the domain adaptation methods mentioned above to train the generative model. Furthermore, the training of GAN is more difficult and requires more parameter adjustment skills. Otherwise, it will not be able to generate usable images.

Shao et al. [58] propose PalmGAN to improve the performance of cross-domain palmprint recognition. PalmGAN is based on CycleGAN [59], a bi-direction mapping. The source domain samples can be mapped to the target domain, and the target domain samples can also be transferred to the style of the source domain. They first train this generator with all the source and target images without the domain labels. They then train a Deep Hashing Network (DHN) with these fake target domain images. In this way, the feature extractor can be improved in the target domain without directly using the target domain images. Although PalmGAN is not strictly required balanced samples from the source and target domain, they still need plenty of training data. Otherwise, it can not produce useful fake target domain images.

H. Shao and D. Zhong [60] propose the Joint Pixel and Feature Alignment (JPFA) framework, which combines both GAN and domain adaptation for cross-dataset palmprint recognition scenarios. They first use CycleGAN to transfer the style of sample images between the source and target domains. For two sample images, source and target domains, respectively, there will be four input images for training three feature extractors.

The domain adaptation loss MK-MMD is applied to both the source and target feature extractors. For the source feature extractor, it is a common adaptation process that closes the distance between the distributions of source and target samples. In the case of the target feature extractor, it is applied to both the target sample and fake target images. Furthermore, there is also a consistency loss between the source and target feature extractors.

However, these kinds of methods need the help of the target dataset, and they cannot be directly deployed in new environments. In addition, their "cross-dataset" experiments are conducted on different splits of the same dataset, such as different spectrums or different phones. They are not even tested in the new environment because they require all identities of the target dataset to be the same as the training dataset.

doi:10.6342/NTU202203861

CHAPTER 3

IMPLEMENTATION OF THE BASELINE SYSTEM

This chapter will go through the implementation of the baseline system and the issues discovered. The system is based on the work [8], whose paper has reported great recognition results in the cross-dataset evaluation. Its recognition processes follow Figure 2. The only difference is that we use the ROI extraction algorithm from [61]. We will explain the **ROI**, **feature extraction**, and **feature matching** in the first three sections.

Then, we evaluate the system with the same datasets as the reference work: the PolyU multispectral palmprint dataset (PolyU-M) and the Tongji palmprint dataset. In addition, we also use the Tongji mobile palmprint dataset (MPD) to examine its performance in an unconstrained environment. We will show the shortcomings of this system in Section 3.4 and propose motivations for improvement.

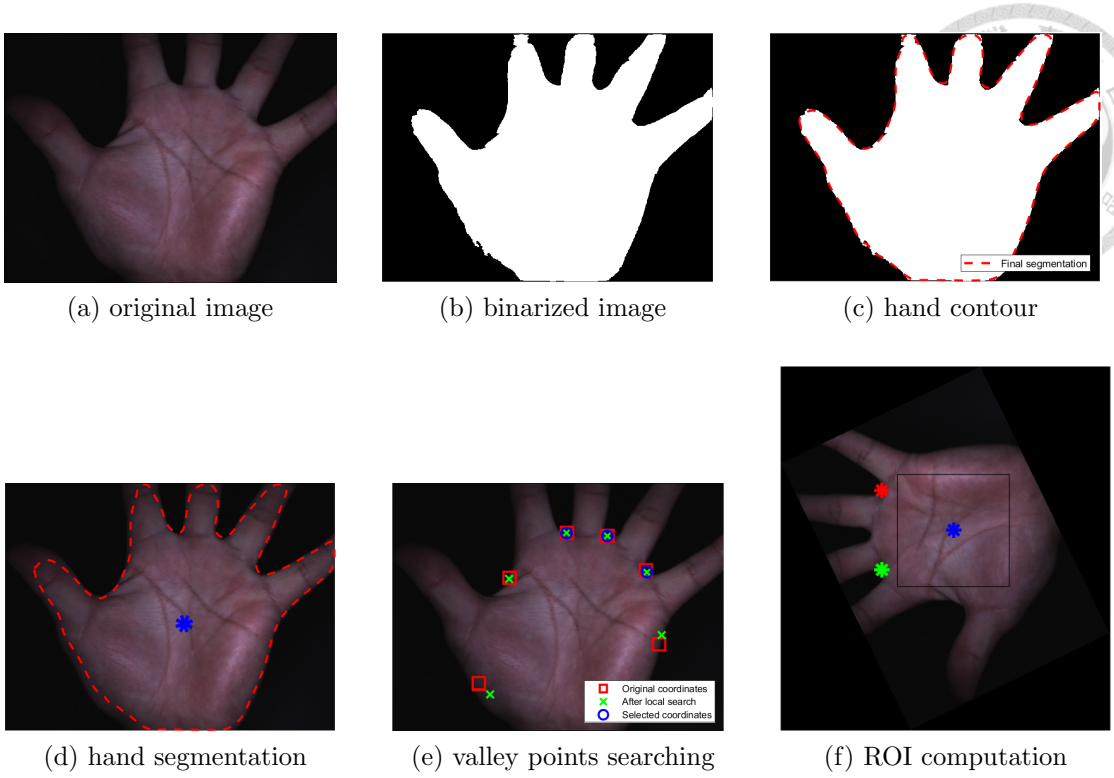
3.1 Region of interest (ROI)

We use the ROI extraction method proposed by Genovese et al. [11] for the Tongji palmprint dataset, which is partially based on [62]. This ROI extraction approach is composed of three steps: hand segmentation, valley points searching, and ROI computation.

1. **Hand segmentation:** The method begins by converting the RGB image to grayscale and performing several preprocessing steps before thresholding, including enhancing, normalization, and smoothing. And then, Otsu's thresholding [63] is executed to estimate the palm region. An example is shown in Figure 13(b). After binarization, they extract the edges from the binarized image by Kirsch's edge detection [64] method. The edges will be divided into horizontal and vertical to help obtain the palm outline. The example of a hand contour is shown in Figure 13(c). On the other hand, the binarized image would also be used to calculate the centroid, which is the center of mass. The centroid is illustrated as a blue dot in Figure 13(d)

2. **Valley points searching:** The valley points could be found by calculating the distance between the centroid and all the hand boundaries, which is a

doi:10.6342/NTU202203861



(a) original image (b) binarized image (c) hand contour

(d) hand segmentation (e) valley points searching (f) ROI computation

Figure 13: The process of ROI extraction

similar process as we introduced in Figure 7(d). This is followed by a local search algorithm [65] to optimize the coordinates of those points, which is depicted in Figure 13(e). And then, these points are checked by several pre-defined rules for rejecting the points that are not appropriate, such as the angles and distances between points. If a set contains exact three points that pass all the rules, the proper valley points for ROI computation are obtained. If not, it can not extract ROI from this image. This is necessary because the input images may be various, and the segmentation methods may not be able to extract accurate hand contours either.

3. **ROI computation:** If the three valley points between the fingers are properly located, then we can compute the ROI. The image will be rotated first so that the line between the valley points is perpendicular to the horizontal line and the fingers point to the left. Finally, a square area with sides 1.4 times the length of the line and 0.2 times the length of the line away is found. The reference system of ROI computation is demonstrated in Figure 14.

The process details are shown in Figure 13, and this method is publicly available at [61]. For the cases of PolyU and Tongji MPD, we use the ROI originating from the datasets.

doi:10.6342/NTU202203861

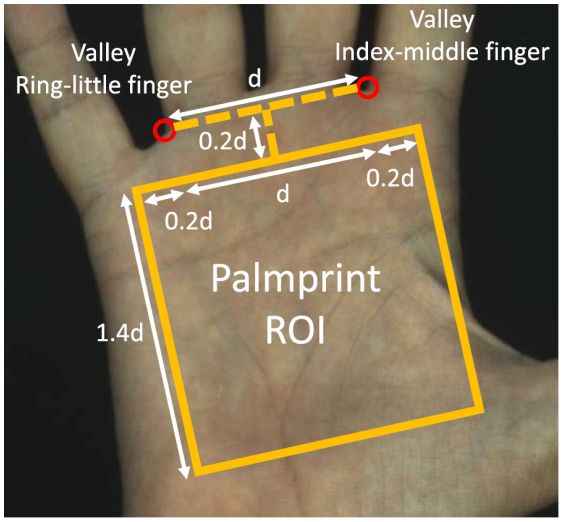


Figure 14: Example of the reference system of ROI computation from Genovese et al. [11]

3.2 Feature Extraction

Feature extraction is achieved by training a deep learning model as a feature extractor. The training process is the same as the general process. The following will be introduced in the order of data augmentation, model structure, and loss function. Data augmentation refers to the random transformation during the training process, and the transformed image will keep the same label as the original image.

3.2.1 Data Augmentation

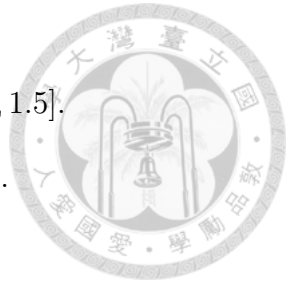
Data Augmentation is a common deep learning technique in image-related tasks. It will randomly perform a list of predefined transformations to the input images. Every single transformation is independent. They could all be executed, or none of them be executed. The main purpose is to produce more training data to improve the generalization ability and prevent overfitting.

The extracted ROI images will be resized to 224×224 first, then randomly transformed. All of these transformations are applied with the probability $p = 0.6$. Before being fed into the feature extractor, each pixel (in $[0, 255]$) in RGB images is normalized to the range of $[0, 1]$. Our implementation of those transformations follows the reference work, but there are some slight modifications due to the implementation library. The list of the transformations we used is as follows:

- **Rotation:** The range of the rotated degrees is $[-5°, +5°]$, and the blank pixels will be filled with black.
- **Resized cropping:** The range of resizing is $[1.07, 1.14]$, then a 224×224

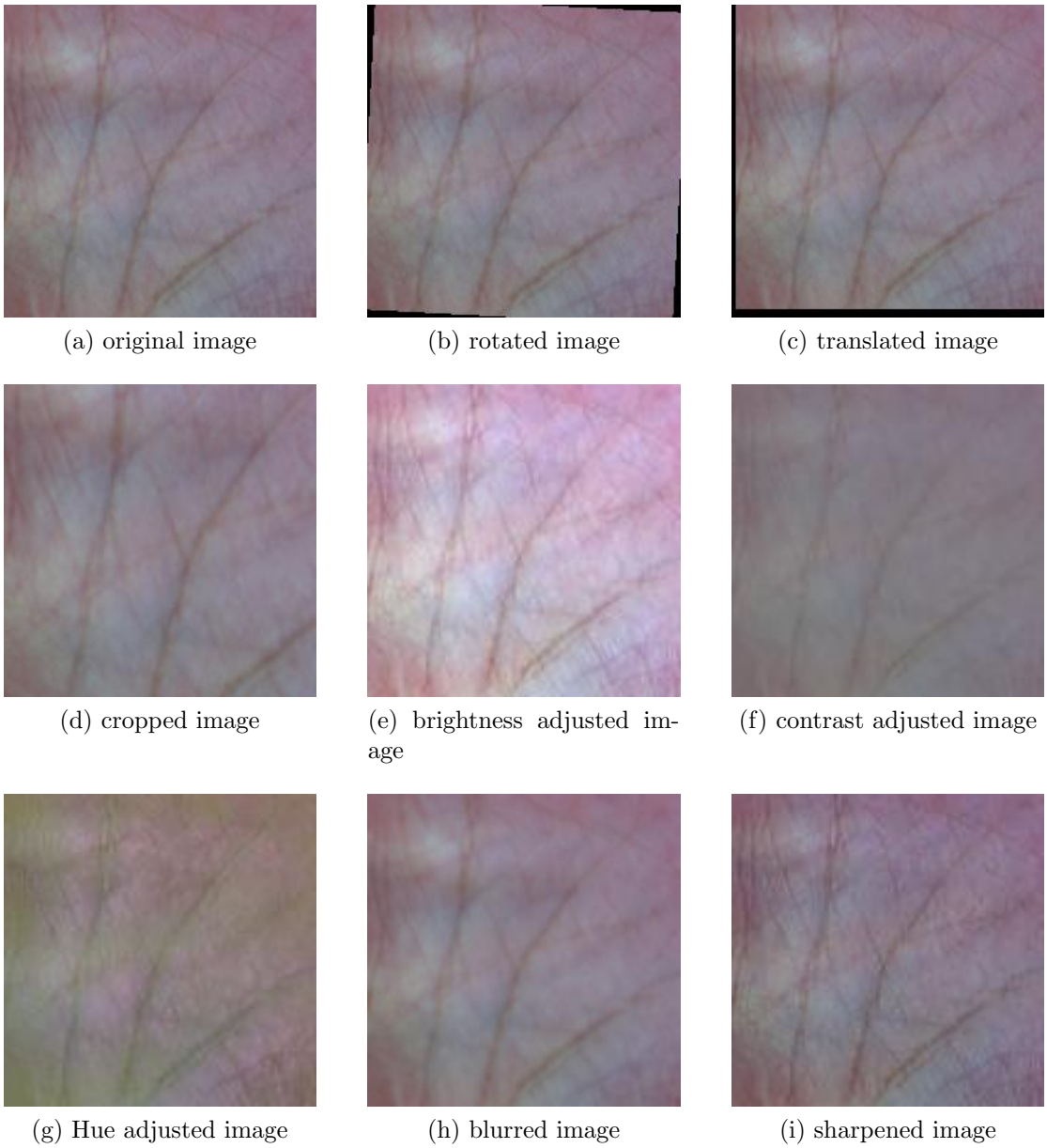
doi:10.6342/NTU202203861

image patch is cropped.



- **Brightness:** The range of the brightness variation is $[0.5, 1.5]$.
- **Contrast:** The range of the contrast variation is $[0.5, 1.5]$.
- **Hue:** The range of the saturation variation is $[0.75, 1.25]$.
- **Smoothing:** Different sizes $(1 \times 1, 3 \times 3, 5 \times 5)$ of Gauss filters are randomly chosen for smoothing.
- **Sharpening:** The sharpness is set to 1.5, while 1 gives the original image and 2 increases the sharpness by a factor of 2.

doi:10.6342/NTU202203861



(a) original image (b) rotated image (c) translated image

(d) cropped image (e) brightness adjusted image (f) contrast adjusted image

(g) Hue adjusted image (h) blurred image (i) sharpened image

Figure 15: Examples of the transformed images

doi:10.6342/NTU202203861

3.2.2 Model Structure

3.2.2.1 ResNet-20

The ResNet-20 is the model proposed in the reference work, which seems to be a modified version of the ResNet-18. There are three main differences from ResNet-18: the residual block stack, the downsampling CNN layer, and the fully connected layer. The original ResNet-18 has 2 residual blocks in each convolution layer, while the block numbers of ResNet-20 are in the order of 1,2,4,1. Second, the ResNet-20 has a downsampling CNN layer before each convolution layer, but the ResNet-18 downsamples in the first residual block in each convolution layer. At last, the ResNet-20 removes the average pooling after all the convolution layers from the ResNet-18. The detailed model structure of the ResNet-20 is shown in Table 2.

Table 2: The architecture of ResNet-20

Layers	**Output sizes**	**Parameters**
input	$3 \times 224 \times 224$	data
conv1.x	$64 \times 112 \times 112$	$3 \times 3, 64$, stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 1$, stride 1
conv2.x	$128 \times 56 \times 56$	$3 \times 3, 128$, stride 2 $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$, stride 1
conv3.x	$256 \times 28 \times 28$	$3 \times 3, 256$, stride 2 $\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 4$, stride 1
conv4.x	$512 \times 14 \times 14$	$3 \times 3, 512$, stride 2 $\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 1$, stride 1
FC	128	128 dimensional feature vector

Note: The square bracket indicates the residual block, and the number outside the block means the number of this block. For example, $[3 \times 3, 128] \times 2$ represent 2 concatenated residual blocks with 128 filters of kernel size 3×3. [8]

doi:10.6342/NTU202203861

3.2.2.2 ResNet-18

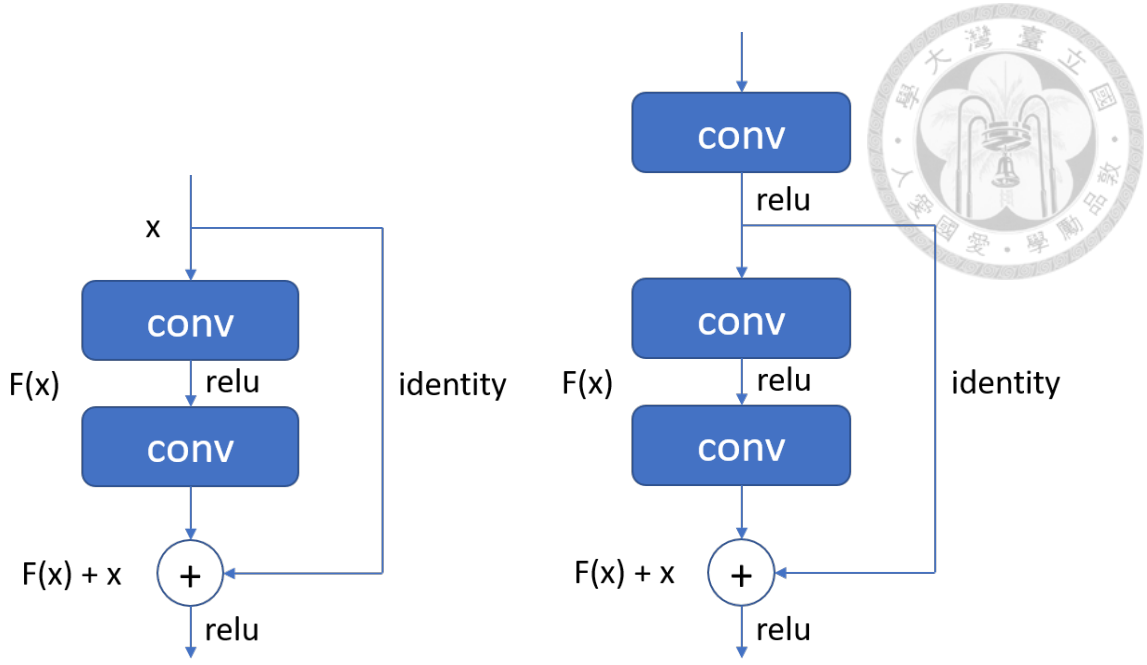
The ResNet was developed by Kaiming et al. [66]. Since ResNet won the ILSVRC in 2015, it has become one of the most popular CNN models in the world. Even though several advanced CNN models have evolved in recent years, it is still a popular choice for image-related tasks. We will utilize ResNet-18 as another backbone model to see how the modifications from ResNet-18 to ResNet-20 affect performance. The network architecture of ResNet-18 is as follows, and we can see the difference of block structure between it and ResNet-20 in Figure 16.

Table 3: The architecture of ResNet-18

Layers	**Output sizes**	**Parameters**
input	$3 \times 224 \times 224$	data
conv1	$64 \times 112 \times 112$	$7 \times 7, 64$, stride 2
conv2.x	$64 \times 56 \times 56$	3×3 max pool, stride 2 $\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3.x	$128 \times 28 \times 28$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4.x	$256 \times 14 \times 14$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5.x	$512 \times 7 \times 7$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
avgpool	$512 \times 1 \times 1$	average pooling
FC	1000	softmax 1000 class

Note: The square bracket indicates the residual block, and the number outside the block means the number of this block. For example, $[3 \times 3, 128] \times 2$ represent 2 concatenated residual blocks with 128 filters of kernel size 3×3. If this layer has to down scale the image size, the down-sampling CNN layer will be placed in the first residual block.

doi:10.6342/NTU202203861



(a) ResNet-18 [66] (b) ResNet-20 [8]

Figure 16: The block structure of models

3.2.3 Centralized Large Margin Cosine Loss (C-LMCL)

The centralized large margin cosine loss (C-LMCL) is the major breakthrough in the reference work. It is a combination of two deep metric learning losses: Center loss and large margin cosine loss (LMCL). To the best of our knowledge, the LMCL is the basis for maximizing the inter-class variance. The Center loss is regarded as a constraint to reduce the intra-class variance. In other words, the feature vectors belonging to the same class could be gathered in the hyperspace. In this section, we will first describe the deviation of the LMCL, followed by the formula of the Center loss. At last, these two losses are combined to formulate the centralized large margin cosine loss (C-LMCL).

3.2.3.1 Large Margin Cosine Loss (LMCL)

In the multiclassification tasks, the original softmax loss seeks to maximize the posterior probability of the true class. This is the same as minimizing the cross entropy between the true class and the softmax of the model output. Because the true class label will be expanded as a one-hot vector with only a value at the index representing the true class, the cross entropy formulation could be simplified as $-\log p_{y_i}$. The p_{y_i} indicates the softmax output at the index representing the true class. Suppose the output feature vector by CNN is X_i whose true label is y_i along with the output of the classification result o_j, then the softmax loss can be

doi:10.6342/NTU202203861

written as

$$
\begin{aligned}
L_s &= \frac{1}{N}\sum_{i=1}^{N} -\log p_{y_i} \qquad (3.1)\\
&= \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{o_{y_i}}}{\sum_{j=1}^{M} e^{o_j}}\\
&= \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{W_{y_i}^T X_i + b_{y_i}}}{\sum_{j=1}^{M} e^{W_j^T X_i + b_j}}\\
&= \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{\|W_{y_i}\|\,\|X_i\|\cos\theta_{y_i} + b_{y_i}}}{\sum_{j=1}^{M} e^{\|W_j\|\,\|X_i\|\cos\theta_j + b_j}}, \qquad (3.2)
\end{aligned}
$$

where N denotes the total count of training samples and M indicates the number of classes. W_j and b_j are the weight vector and bias of the jth class in the last full connected layers. And θ_j is the angle between weight vector W_j and feature vectors X_i.

In order to minimize the cross-entropy loss, the model should optimize in three directions: increasing the weight norm of the true class $\|W_{y_i}\|$, increasing feature norm $\|X_i\|$, and reducing the angle between feature and weight vector θ_{y_i}. In the first case, it will increase the weight norm of the class that has more samples or easier samples [51]. The second will help to increase the norm of the easier samples and reduce the norm of the harder samples [67]. These three directions will be optimized at the same time, and result in a sector distribution of the output feature.

Although the classification accuracy can be high, it is sensitive to hard samples. To illustrate, if there are two samples from different classes, they may both be at the position near the origin [68]; that is, they have a smaller norm. It also demonstrates that using Euclidean distance as the metric between features is not precise, because it relies on the norm of the features. If we use angle as a metric for determining the difference between two features, it would not be influenced by the norm of the features. Hence, we hope the model can focus on optimizing the angle between the weight vector and the feature vector. As a result, it is natural to block another two optimization directions, which is to fix the value of both the weight norm and the feature norm. And the loss can be further simplified as

$$
\begin{aligned}
L_{ns} &= \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{\|W_{y_i}\|\,\|X_i\|\cos\theta_{y_i}}}{\sum_{j=1}^{M} e^{\|W_j\|\,\|X_i\|\cos\theta_j}}\\
&= \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{s\cos\theta_{y_i}}}{\sum_{j=1}^{M} e^{s\cos\theta_j}}, \qquad (3.3)
\end{aligned}
$$

doi:10.6342/NTU202203861

where the s is a parameter that we can tune. This is also known as normalized softmax.

In the biometric recognition tasks, it is not sufficient for the model to merely classify the known classes. We need the capability to measure the similarity between unknown classes in palmprint recognition. For better generalization capability, large inter-class variance and small intra-class variance are necessary, while these are not the optimization targets of the softmax loss. A classic strategy is to add a margin between classes. The feature extractor trained with the normalized softmax loss may produce a feature close to the decision boundary. In order to further increase the margin between classes, a more stringent constraint for classification should be added to the training time:

$$\cos\theta_{y_i} - m > \cos\theta_j, \quad \forall \quad j = 1, 2, \ldots, M \quad \text{and} \quad j \neq y_i. \tag{3.4}$$

The constraint forces the feature vector to be much closer to its true weight vector than others with a certain "advantage" - m. In this way, the inter-class distance can be expanded so that it is less likely to induce incorrect recognition despite small disturbances [8]. After adding this constraint to the normalized softmax loss, the formulation would be

$$L_{lmc} = \frac{1}{N}\sum_{i=1}^{N} -\log\frac{e^{s(\cos\theta_{y_i}-m)}}{e^{s(\cos\theta_{y_i}-m)} + \sum_{j=1,j\neq y_i}^{M} e^{s\cos\theta_j}}, \tag{3.5}$$

where the scale s and margin m are parameters that can be adjusted individually. This loss function is proposed by Wang et al. [50], which is named the large margin cosine loss (LMCL).

3.2.3.2 Center loss

In contrast to LMCL, which enlarges the gaps of decision boundaries to produce a more discriminative feature vector, the main objective of Center loss is to minimize the intra-class distance in another way. As the name suggests, the Center loss is the distance between the feature vector and the center vector of its class. Thus, the function of the Center loss could be expressed by

$$L_c = \frac{1}{2}\sum_{i=1}^{N} \|X_i - C_{y_i}\|_2^2, \tag{3.6}$$

where N denotes the number of training samples, X_i is the output feature vector, y_i is the ground-truth class that X_i belongs to, and C_{y_i} is the center vector of the y_ith class. In our implementation, the center vector is not calculated from the mean of all the feature vectors for each class. With the help of the LMCL, we can use its weight vector as the center vector.

doi:10.6342/NTU202203861

3.2.3.3 *Centralized Large Margin Cosine Loss (C-LMCL)*

The cooperation of the previous two loss functions results in a new loss function named centralized large margin cosine Loss (C-LMCL) [8]. The formula of the C-LMCL is written as follows

$$\begin{aligned} L &= L_{lmc} + \lambda L_c \\ &= -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos\theta_{y_i}-m)}}{e^{s(\cos\theta_{y_i}-m)}+\sum_{j=1,j\neq y_i}^{M}e^{s\cos\theta_j}} \\ &\quad +\frac{\lambda}{2}\sum_{i=1}^{N}\|X_i - C_{y_i}\|_2^2, \end{aligned} \tag{3.7}$$

where λ is a parameter balancing LMCL and Center loss. There are a total of 3 hyperparameters that could be adjusted: the balancing parameter λ, the scale s, and the margin m. We will follow the best parameters that the original paper [8] provides: $\lambda = 0.1, s = 30, m = 0.65$.

3.3 *Feature Matching*

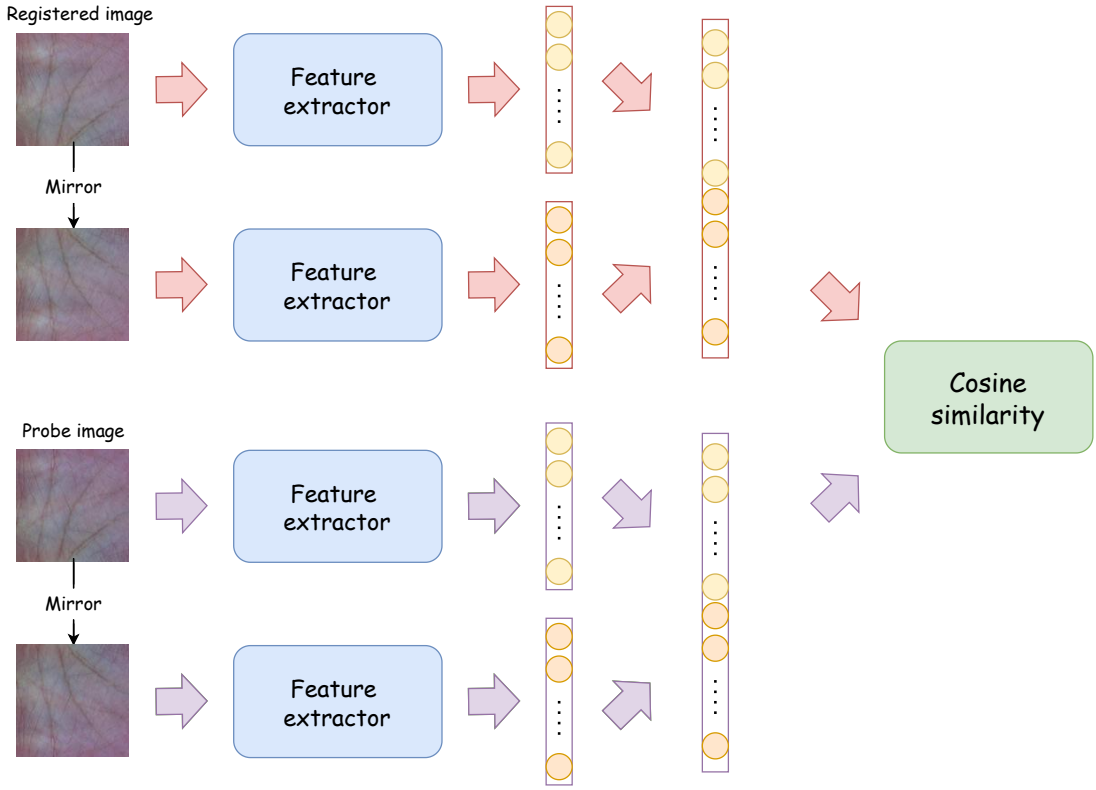


Figure 17: Illustration of feature matching which uses the concatenated feature of the origin image and its mirror image.

In the reference work, they mentioned that the final feature is the feature concatenation of the input image and its mirror image. Based on these features,

doi:10.6342/NTU202203861

the cosine similarity is calculated. An example is illustrated in Figure 17. In identification mode, the input image will be compared with multiple templates. The template with the highest similarity score with the input image will be selected as the recognition result. In the verification mode, a one-to-one comparison is performed. A preset threshold is usually used to verify the similarity score of the two to determine whether the two are the same person.

3.4 Observation and Discussion

In this section, we first derive the limitations of the feature matching method in Section 3.4.1, which they did not explain in the reference work. And then show the results of the cross-subject evaluation on several datasets in Section 3.4.2. We pay special attention to the unconstrained dataset, MPD, because our goal is to develop an approach that can handle complex environments. The MPD would be represented as MPD(h) and MPD(m) in the following tables, indicating Huawei and Xiaomi, according to the mobile phone used to take the images. We also conducted a cross-dataset evaluation between the Tongji and PolyU datasets like the reference work in Section 3.4.3. Both evaluations will compare ResNet-18 to ResNet-20. We found that the performance of ResNet-20 on the cross-dataset evaluation trained on the PolyU dataset and directly applied to the Tongji dataset is far worse than the results in the paper. Finally, we will summarize the observed shortcomings of the current system and analyze the possible improvements in Section 3.4.4.

We performed a different dataset split from the reference work in these evaluations. We keep one-tenth of the source dataset for evaluation. These parts of image data are from totally different classes than the training classes. In detail, the heading "source" in the following tables means that we use 90% of the classes in that dataset to train the model. Note that the number of image data samples in each class is the same within each dataset, so there will be the same number of training images regardless of which classes are used for training. Moreover, the heading "target" in the following tables means the dataset we used for testing. As for the split of the registration set, 5 images in each class are regarded as the registration set and the others are the probe images so that the experiment is more realistic while having a larger number of testing samples.

3.4.1 Feature Matching by Concatenated Features

In the original paper, they did not explain why they used such a matching method, and of course, they did not explain the advantages and disadvantages of this method. They only have one sentence saying they use this way to generate

doi:10.6342/NTU202203861

the final palmprint representation. However, we suffer an accuracy drop due to this. So we derive the mathematical formulation of this matching method and figure out its effects and limitations for improvement.

Since this feature matching method is based on cosine similarity, we start from its mathematical representation. The formula for the cosine similarity can be derived from the Euclidean dot product, and it could be written as

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \, \|B\|}. \tag{3.8}$$

Then, we need to define the notation of the concatenation of two vectors. If the feature dimension is N, the feature vector extracted from a registered image can be denoted as $G \in \mathbb{R}^{\mathbb{N}}$, as well as the feature vector $G^M \in \mathbb{R}^{\mathbb{N}}$ is from its mirror image. According to the matrix notation, the concatenated vector can be expressed as $[\,G \;\; G^M\,]_{1\times 2N}$. As a result, given the feature vector G, G^M along with the features P, P^M obtained from an input image, the cosine similarity of these two input images can be defined as

$$\cos(\theta) = \frac{[\,G \;\; G^M\,] \cdot [\,P \;\; P^M\,]}{\|[\,G \;\; G^M\,]\| \, \|[\,P \;\; P^M\,]\|}. \tag{3.9}$$

Since the dot product is an element-wise multiplication, the dot product of two concatenated vectors equals adding the dot product of each vector. Hence, the formula can be derived as follows:

$$\cos(\theta) = \frac{G \cdot P + G^M \cdot P^M}{\|[\,G \;\; G^M\,]\| \, \|[\,P \;\; P^M\,]\|}. \tag{3.10}$$

Then we expand it, and if all the feature norms are assumed to be equal, we can get a more simplified form as shown below:

$$\cos(\theta) = \frac{\|G\| \, \|P\| \cos(\theta^I) + \|G^M\| \, \|P^M\| \cos(\theta^M)}{\sqrt{\|G\|^2 + \|G^M\|^2} \times \sqrt{\|P\|^2 + \|P^M\|^2}} \tag{3.11}$$

$$= \frac{\cos(\theta^I) + \cos(\theta^M)}{2}, \quad if \; \|G\| = \|P\| = \|G^M\| = \|P^M\|, \tag{3.12}$$

where the θ^I is the angle between the vectors extracted from input image and registrated image, and θ^M is the angle between their mirror images.

From Equation (3.12), we know that if the norm of every feature vector is the same, the final cosine similarity would be the median value of the cosine similarity between the input image and registered image and between their mirror images. Since the Center loss cluster the features of the same class closely together, the feature norm of the same class will be very close. Although the feature norm of different classes is not very close, they are usually not too far apart for a well-trained model. Consequently, the cosine similarity of the concatenated features

doi:10.6342/NTU202203861

might be between the cosine similarity of the input image and registered image and the cosine similarity of their mirror images in an ideal case.

In the above descriptions, we assume the feature norm of different palmprints might be close, but this is not always the case. If one of the feature norms is much higher than the others, it will cause the final similarity to be reduced to a value lower than both of the similarities. This is because once the feature norm is high, the cosine similarity between it and others is very likely to be low. Since the high feature norm indicates the model was unfamiliar with the source image and produced an abnormal feature, the similarity between it and others is more like a random value. According to Equation (3.11), if $\|G^M\|$ is large, the denominator will also become larger. However, $\|G^M\| \cos(\theta^M)$ may remain at a similar value or lower, and the final similarity will become lower.

The lower bound becomes especially important since the final similarity will fall between the two similarities. Thus, not only should the feature extracted from the two input images be similar, but the feature extracted from their mirror images also needs to be similar. Otherwise, the final performance will also be affected greatly. Because the number of matching is very large in identification mode, it is more likely that recognition errors will occur due to the addition of features from mirror images. Therefore, we need to ensure the recognition performance of mirror images.

In summary, concatenating the feature from the input image and its mirror image would result in a performance gain under two constraints. First, the norm of any feature cannot be significantly greater than the norms of the others. Second, the model must also be able to identify by the mirror images. Hence, this technique boosts the accuracy of a system that already performs well, while in other cases, it has a negative impact. We will remove these negative effects, ensuring that the matching method can use additional images to improve accuracy or at least remain the same.

3.4.2 Cross-subject Observation

Table 4: Comparison of Rank 1 accuracy(%) on different datasets

Dataset	PolyU	Tongji	MPD(h)	MPD(m)
ResNet-20	100	99.56	72.67	70.17
ResNet-18	100	96.11	91.83	77.33

The result of cross-subject evaluation on four palmprint datasets is shown in

doi:10.6342/NTU202203861

Table 4. ResNet-20 performs perfectly on the PolyU and Tongji datasets, but the accuracy on both the MPD datasets is much worse than on these datasets. One of the reasons for the large drop in accuracy is the feature matching method. In the previous section, we have explained the requirements of this matching method. The ResNet-18 performed much better in the MPD datasets, despite having lower accuracy on the Tongji dataset than ResNet-20. It is also affected by the matching method on the MPD(m) result, but it is still higher than ResNet-20 when matching with only input images.

We consider that the key reason ResNet-20 performed so poorly on MPD datasets was that it used a fully connected layer rather than global average pooling. Since Lin et al. [69] proposed to replace the fully connected layer with the global average pooling, it has become a dominant choice to use the pooling layer instead of the fully connected layer. The advantage of using a pooling layer is to prevent overfitting and provides more robust features. Hence, in such an unconstrained environment like the MPD dataset, the features extracted by ResNet-18 have better performance. However, this is still not enough. We must find a model structure more suitable for unconstrained palmprint recognition.

3.4.3 Cross-dataset Observation

Table 5: Comparison of Rank 1 accuracy(%) on the cross-dataset evaluation

Source	Target	ResNet-20	ResNet-18
Tongji	PolyU	99.57	83.94
PolyU	Tongji	60.68	75.64

The cross-dataset evaluation result between the PolyU and Tongji datasets is shown in Table 5. For simplicity, we denote "train on the source dataset and directly test on the target dataset" as the right arrow in the following descriptions, e.g., PolyU $\rightarrow$ Tongji. According to the table, the performance of ResNet-20 on Tongji $\rightarrow$ PolyU is satisfactory, though it does not reach the level of the original paper. In contrast, it is far worse on PolyU $\rightarrow$ Tongji. Although the matching method also decreases accuracy, if we do not use the mirror image, the accuracy would still only be 67%. This result is far away from what is reported in the paper, let alone if the model is used on an unconstrained dataset.

The ResNet-18 outperformed the ResNet-20 on PolyU $\rightarrow$ Tongji but was lower in reverse. This is similar to cross-subject evaluation. The ResNet-20 wins in the easier scenario, whereas the ResNet-18 wins in the more difficult scenario. Because

doi:10.6342/NTU202203861

the PolyU dataset is captured in a constrained manner, the pose and environment of images are almost consistent. Thus, the variance of the images in the PolyU dataset is less than that of the Tongji dataset. That is why it is more difficult on PolyU → Tongji. This shows that ResNet-18 has better generalization ability.

As mentioned in the previous section, the advantage of global average pooling is that it can prevent overfitting. Conversely, the fully connected layer fits the training data more closely. When the patterns in the training data can somewhat cover the patterns in the testing data, it can get excellent recognition results. That is the case of Tongji → PolyU, so ResNet-20 has very high accuracy. However, it has three apparent disadvantages:

1. The testing data is always unpredictable in the real application. Training data rarely cover the testing data. A scenario like Tongji → PolyU is unlikely to happen. Thus, there is not much chance for him to show his strengths.

2. The accuracy drop is particularly huge when the palmprint images differ from the training data. In practical applications, this is always the case.

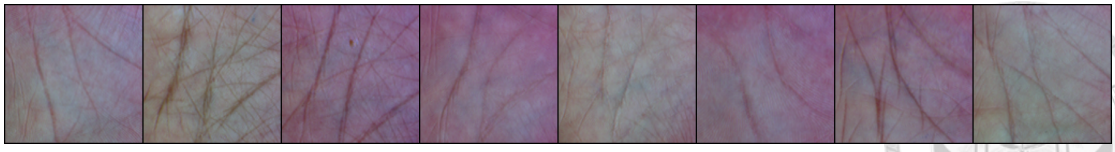
3. If the amount and variation of training data grow, the model may not be able to fit the training data well. Alternatively, the number of parameters may need to grow with the amount of data.

On the other hand, the more important part is that there are huge gaps in the images of different datasets. Some samples of each dataset we used are shown in Figure 18. Many studies treat such situations as cross-domain problems mentioned in the related work section. Most of them tried to solve this problem by using more existing data. Whether it is to adapt the model to the target domain or to generate training data in the style of the target domain, using target domain images for learning is required. However, those methods only focus on one target domain, while our goal is to enhance the generalization ability of the model so that it can identify in complex environments. The characteristic of deep learning is that it can model the problem better as long as it is given more diverse training data. Therefore, we will focus on data augmentation and find better augmentation strategies to solve the cross-datasets palmprint recognition problem.

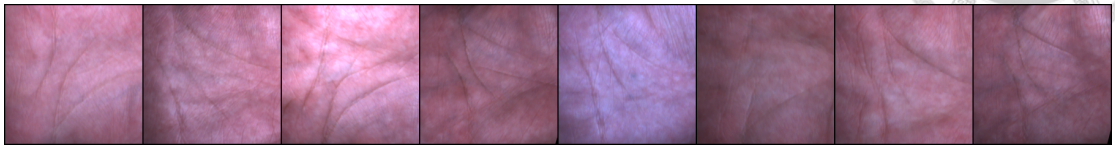
3.4.4 Summary and Motivation

After reviewing these several evaluations, we can sort out some problems and improve directions in the current method. From the cross-subject evaluation, we know that the current matching method is unstable. Although it sometimes improves the accuracy, it worsens when the model cannot extract discriminative features from the mirror images. While the number of registered images increases,

doi:10.6342/NTU202203861

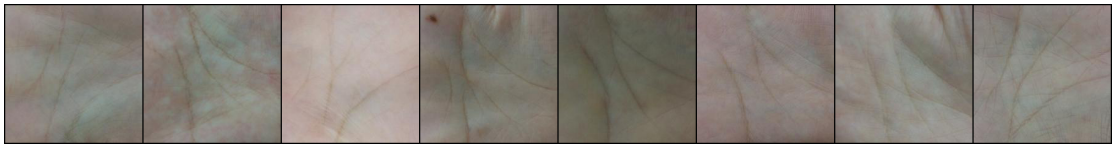


(a) PolyU [3]



(b) Tongji [16]

(c) MPD(h) [17]



(d) MPD(m) [17]

Figure 18: Example ROIs of each datasets

it is more likely that the wrong identities will produce a higher similarity score than the true identity due to the mirror image, resulting in an identification error.

We can make it better by separating features. Since the feature norm of the input image and its mirror image will affect each other, we can perform feature matching separately and then recombine the results, thereby eliminating the negative impact of instability. In addition, the mirror transformation can be done in different directions, such as vertical and horizontal. Moreover, the rotations could also be taken into account. Because when we perform both vertical and horizontal mirror conversion, it is equal to the 180 degrees rotation. The fusion of these additional features could make the matching more robust. We could design a better feature matching method based on these two points.

In the cross-subject evaluation of the MPD dataset, both models perform poorly due to the diversity of input images. Suppose we increase the network width to expand the receptive field and introduce an attention mechanism to enhance the generalization ability of the model while avoiding overfitting. In that case, the model may handle these more challenging environments.

As for cross-dataset evaluation, training data is important. Previous research has focused on cross-domain approaches to bridge the gap between palmprint

doi:10.6342/NTU202203861

datasets. However, this is insufficient to handle the complex environment as they only consider one target domain. We will optimize the existing data augmentation strategy so that models can be trained on only one dataset and applied to other datasets. We can organize the above into the following three points:

1. The current feature matching method is unstable. Using mirror of input images to improve accuracy must preserve at least the same level of accuracy as not using mirror images.

2. ResNet-20 cannot produce sufficiently discriminative features in an unconstrained environment like MPD. We will use a wider model structure to handle these diverse input images.

3. To overcome the cross-datasets palmprint recognition problem, providing more various training data for deep learning models is necessary. We would improve the data augmentation strategy to help models perform better in various environments.

doi:10.6342/NTU202203861

CHAPTER 4

DATA AUGMENTATION ON TRAINING AND RECOGNITION

In this chapter, we address the problem of cross-datasets palmprint recognition through three data augmentation techniques: data warping, oversampling, and test-time augmentation. Data warping is a method of randomly transforming images during training, and we optimize this method for palmprint recognition. Our oversampling method is to expand the training set through rotation, and each image would be rotated to the other 3 orientations as new training data. Both of these are applied during training to enable the model to reach its better potential, while test-time augmentation improves performance through image transformation during recognition. Based on test-time augmentation, We propose a method to generate multiple images for comparison by rotating and mirroring, called multi-transform matching. In addition to data augmentation, we have mentioned the shortcomings of existing feature extraction models at the end of Chapter 3, so we propose a model stacking that is more suitable for palmprint recognition.

Since our method includes feature extraction model training and feature matching, we will introduce it in two parts according to the flow chart shown in Figure 2. First, we organize our proposed training method in Figure 19. The data augmentation in the red dash lines includes the oversampling and data warping we mentioned above, which we will detail in Section 4.1. The feature extractor in the blue block is the deep learning model. We will introduce our proposed structure in the section 4.2. The rest of the training process will be describe later.

As for feature matching, we have explained the advantages and disadvantages of using mirror-concatenated feature matching in Section 3.4.1. While the matching performance can be improved when the features are very distinguishable, it also increases the risk of recognition errors. To solve this problem, we match the mirror images independently, and add another mirror transformation and rotations to increase reliability. This method benefits from our proposed oversampling method, which we will explain in more detail in Section 4.3.

The training process in Figure 19 is our improvement based on [8]. The main difference is the part of data augmentation. The random transformation during training is called data augmentation in the original paper, but this method is only one of data augmentation according to the definition in [70]. Such an approach

doi:10.6342/NTU202203861

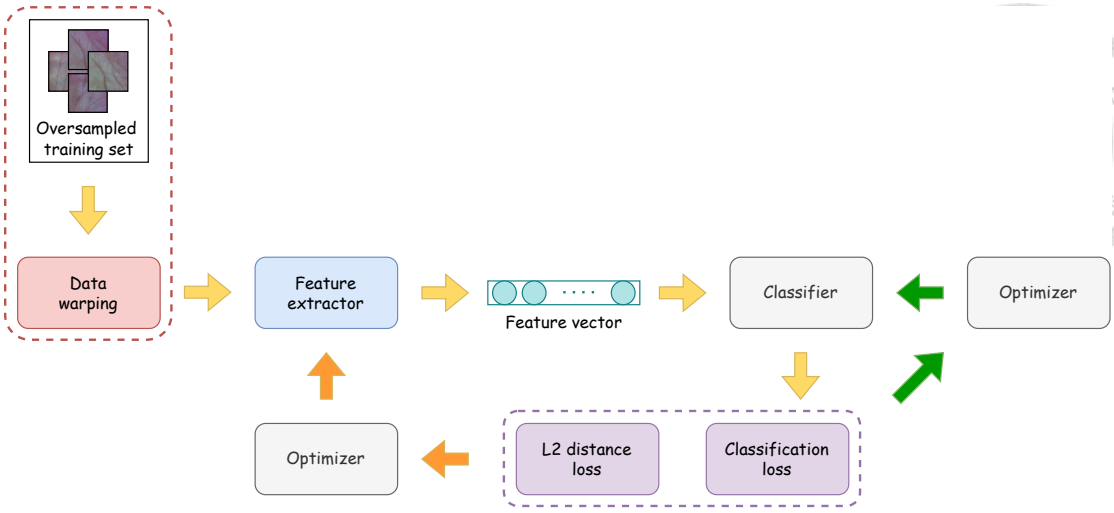


Figure 19: Feature extractor training process

is called data warping in their definition, as shown in the red block in the figure. In addition, oversampling is to expand the training set before training, so it is directly displayed on the training set in the figure.

The rest are based on the original process to improve the components separately. In addition to the feature extraction model, we also make the loss function more suitable for our proposed augmentation strategy. The original CLMCL is divided into three parts: classifier, classification loss, and L2 distance loss.

The classifier maintains a set of weight vectors with as many classes as the training set to find the class closest to the input feature vector. It originated from the last fully connected layer of the model in the image classification task. However, it is also used for L2 distance calculation here in addition to the original classification. It can be imagined that it uses cosine similarity and L2 distance for classification, and there are also two types of loss functions for parameter update.

The classification loss refers to the margin-based cosine loss based on the dot product, such as LMCL [50], ArcFace [10], CurricularFace [71]. These losses are based on softmax loss and add some restrictions to its input to create a more compact feature space. Hence, these losses consist of two parts, of which we use LMCL for the restriction part. As for the L2 distance loss, it is a very important part. We proposed the oversampling method based on the assumption that this loss can closely gather samples of the same class.

Since our data warping strategy produces extreme samples, the original loss function may lead to unstable training. Therefore, we use Focal loss and Huber loss to replace the softmax loss and center loss. These two have relatively smooth gradient curves, and will not generate excessive gradients due to extreme samples, resulting in excessive parameter updates. We will discuss more in Section 4.1.4. For the classifier, we keep the original LMCL.

doi:10.6342/NTU202203861

Finally, we use the two optimizers to update the parameters of the feature extractor and classifier because they require different learning rates. Since the weights in the classifier represent the center vectors of each class, a relatively large learning rate is required at the beginning of training to spread these vectors evenly. If the same optimizer is used, the parameters will not converge well. Here we use SGDM and Adam optimizer for the feature extractor and classifier, respectively. Although it has a great impact on model training, in this study, we only focus on the components that improve palmprint recognition across datasets. In the following sections, we will introduce the colored blocks in Figure 19.

4.1 Palmprint ROI Augmentation

C. Shorten and T. M. Khoshgoftaar [70] have stated that data augmentation is a very powerful method for enhancing generalizability. The augmented data will represent a more comprehensive set of possible data points to bridge the gap between the training set, the validation set, and any other testing data in the future. Although this is not the only technique for enhancing generalization ability, we will focus on developing a better data augmentation strategy for palmprint recognition in this work. Other solutions such as designing more complex architecture, batch normalization, and pre-training are covered in next section.

The image-related recognition algorithm must overcome the issues of lighting, background, occlusion, scale, viewpoint, and more. The idea of data augmentation is to use domain knowledge or experience to imagine possible situations so that the model can overcome these challenges and achieve good performance in real-world applications. Especially when we do not have a sufficiently diverse and large amount of training data, as is the situation with palmprint recognition. Therefore, we will invest more effort in developing better data augmentation strategies to simulate various imaging conditions.

Data augmentation tries to solve the problem from the root cause, which is the training dataset. It is generally accepted that larger datasets result in better deep learning models. Data augmentation methods assume we can extract more information from the original dataset and expand the training dataset by either oversampling or data warping. Oversampling creates synthetic instances and adds them to the training set. For example, mixing images, generative adversarial networks (GANs), and feature space augmentations. Data warping transforms existing images during each training epoch, including geometric and color transformations, noise injection, random erasing, and neural style transfer. Oversampling and data warping are not mutually exclusive, and they can work on the images simultaneously.

doi:10.6342/NTU202203861

In Section 4.1.1, We explain our **oversampling** method, which expands the training data size through rotation. Due to the constraints of L2 distance loss, palmprint ROIs in different orientations can be regarded as new training data. It can not only directly improve recognition performance but also make the features generated by the model more compact, giving the matching algorithm greater development flexibility.

Second, **data warping** is another augmentation approach that generates images through various transformations during training. These transformations are performed randomly, and the intensity of the transformations is also randomly chosen. Since there are many kinds of transformations, each has several parameters. The most suitable transform combination and their hyper-parameters cannot be found by manual search alone, so we adopt hyper-parameter optimization to search for the best solution. We detail the search method in Section 4.1.3.

We mentioned earlier that the oversampling augmentation we used requires L2 distance loss, but Center loss is a loss with a very large numerical change. Therefore, it will make the training difficult to converge, especially when the transformation used by data warping is relatively extreme. These excessively transformed training images will cause a large gradient in the Center loss, so we use a smoother loss instead. We will introduce these **loss functions** in Section 4.1.4, which make training stable.

4.1.1 Oversampling with rotation

We found that even if there is only one orientation of palmprint ROI in the training data, the model can still distinguish palmprints in different orientations, as well as the left and right hands in these orientations. Although they are not very tightly clustered in the feature space, a clear distinction can still be seen. However, the recognition accuracy of palmprint ROI images in these orientation is relatively poor, and the improvement of feature matching by using them is relatively small. This explanation will be described in Section 4.3. We extracted the registrated images different from the training dataset to features and drew them into 3D feature space by UMAP [72] as shown in Figure 20.

If we use palmprint ROI images in different orientations as training images, it can improve the model's generalization ability and enhance our ensembled feature matching. Instead of only using one mirror image to improve the performance of the comparison, various rotated and mirrored images can be added to assist feature matching effectively. The implementation is to generate images rotated by 90, 180, and 270 degrees for each training image so that the training set will become four times the original size. It can also be randomly transformed in combination with

doi:10.6342/NTU202203861

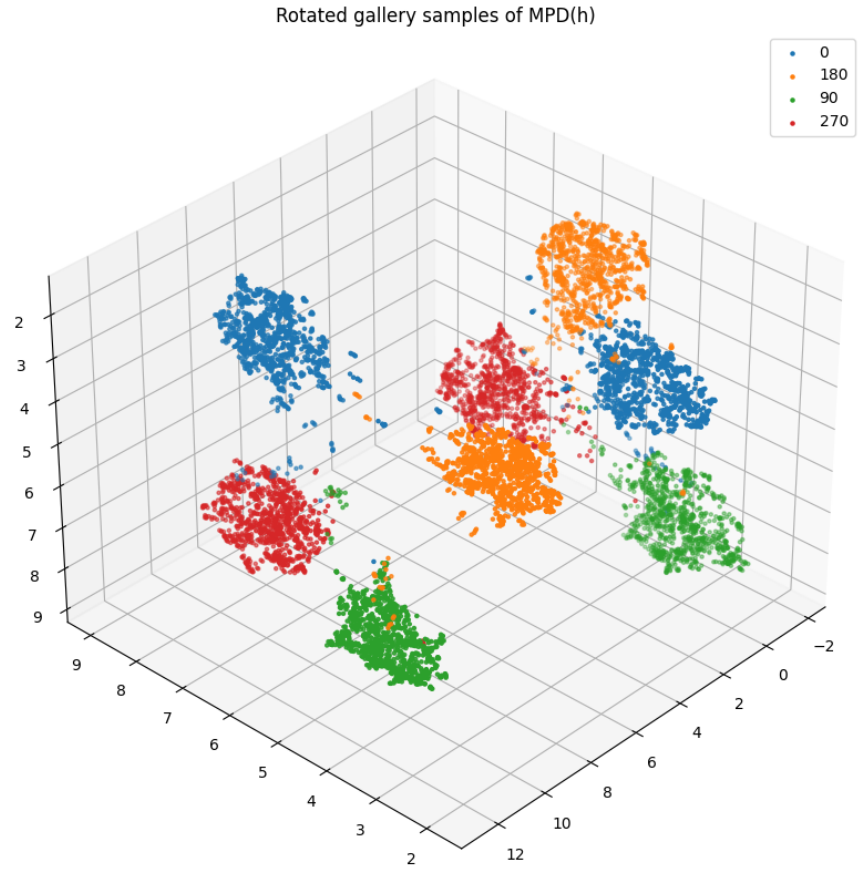


Figure 20: An illustration of the features extracted from different palmprint ROI orientations. 0 is for the original input image, 180 refers to the input image being rotated 180 degrees, and so on.

the data warping augmentation.

What class these newly generated training images fall into is still to be determined. According to the facts, these images belong to the same person. However, they are significantly distinguished in the feature space. The loss function we use during training includes an L2 distance loss, which is very restrictive. It gathers images of the same class very closely together because these cropped ROI images of the same class will definitely be very similar. Figure 21 is an example of ROI images from the same class. Therefore, these rotated training images are regarded as different classes during training. Overall, applying such rotational augmentation will increase the size of the training set and the number of classes by a factor of 4.

doi:10.6342/NTU202203861

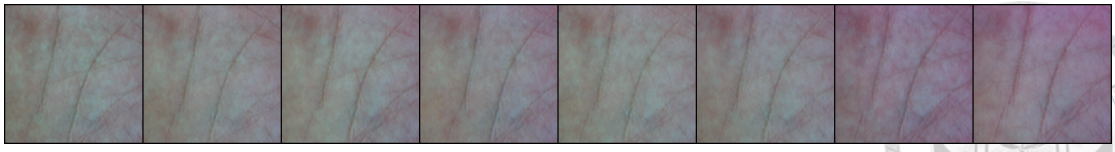


Figure 21: Sample images of the same palm from the PolyU-M dataset [3]. All samples are very similar, so palmprints that are not in the same orientation, even if they are the same hand, should be regarded as different classes from the perspective of loss.

4.1.2 Data Warping Transformations

Data warping augmentation is a fundamental choice for image-related tasks, as it is easy to implement and brings huge performance gains, especially when the training dataset is small. A common practice is pre-determining a set of image transformation functions suitable for the task and randomly executing these transformations during training to make the same image different for each training iteration. These transformations are independent to each other. That is, an input image may have no transformations or every transformation. In order to make the training images more diverse, each transformation usually defines an intensity range in addition to the probability. In other words, one transformation needs to decide on at least two hyper-parameters. When many transformations are used, there are a large number of hyper-parameters to decide. Therefore, we aim to find a set of image transformation functions and their operating hyper-parameters that are most suitable for palmprint recognition.

We first review the seven transformation functions used in Section 3.2.1. Of these, resized cropping and sharpening are useless, in our opinion. Resized cropping will scale up the image and randomly crop a portion of it, but this will make it a different scale from the normal image. In the ROI cropping algorithm, the distance between the finger valleys has been used to unify the scale of each palmprint image. It can also be seen from Figure 18 that the scales between the sample images are almost the same. Therefore, the image generated by resized cropping is not suitable for use in the current scenario, so we remove this conversion function. As for sharpening, instead of providing more difficult images, it provides the images with a clearer texture. It does not help much in training, so we do not need a useless function to increase training time.

In addition, we added two transformation methods: noise injection and translation. The translation is intuitive. Although the anchor point fixes the ROI, there will still be a slight error. Translation can simulate such a situation. Noise injection is to add diversity to the training image, and the model is expected to focus more on texture features. The following are the transformation functions

doi:10.6342/NTU202203861

and their intensity parameters that we will search for:

- **Brightness:** The brightness factor is uniformly sampled from $[\,max(0, 1 - a), 1 + a\,]$ to adjust brightness.
- **Contrast:** The contrast factor is uniformly sampled from $[\,max(0, 1-a), 1+a\,]$ to adjust contrast.
- **Saturation:** The saturation factor is uniformly sampled from $[\,max(0, 1 - a), 1 + a\,]$ to adjust saturation.
- **Hue:** The hue factor is uniformly sampled from $[\,-a, a\,]$ to adjust hue, but the bounds of the hue factor is $[\,-0.5, 0.5\,]$.
- **Smoothing:** The smoothing is performed by Gaussian blur, which randomly chooses the kernel size from $(3 \times 3, 5 \times 5)$. The standard deviation for creating a kernel is in the range of $[\,0.1, a\,]$.
- **Noise injection:** We randomly sample from the standard normal distribution to form a matrix with the same size and number of channels as the input image. Then, multiply it by a magnification factor, and add it to the original image. The magnification factor is an integer uniformly sampled from the range of $[\,0, a\,]$.
- **Rotation:** The range of the rotated degrees is uniformly sampled from $[\,-a^\circ, +a^\circ\,]$, and the blank pixels will be filled with black.
- **Translation:** If the images size is $[\,W, H\,]$, the range of the number of the shift pixels is $[\,-aW, aW\,]$ along the x axis and $[\,-aH, aH\,]$ along the y axis. The blank pixels will also be filled with black.

Note that all the intensity parameters in each transformation are non-negative values and have been reduced to only one.

The execution order of these transformations also needs to be considered, otherwise, the effect of some transformations will not be brought into play. For a simple example, performing noise injection followed by smoothing will eliminate the noise. First of all, rotation and translation will not be affected in any way, so put them last. Second, color-related transformations (brightness, contrast, saturation, and hue) require RGB values for color space conversion, so they come first. Finally, smoothing and noise are placed in the middle. This order is the same as we introduced earlier.

doi:10.6342/NTU202203861

4.1.3 Hyper-Parameter Optimization

We adopt the hyper-parameter optimization framework Optuna [73] to optimize data warping hyper-parameters. Optuna mainly consists of two parts: searching strategy and performance estimation strategy. The search strategy defines the search space of the parameter set and a sampling algorithm. The sampling algorithm includes grid search, various evolutionary algorithms, etc. On the other hand, the performance estimation strategy estimates the values of the hyper-parameters of the current search process based on the learning curve. It determines whether this hyper-parameter set should be discarded, also known as automated early stopping. These two are called sampler and pruner, respectively, in Optuna.

Sequence Model-Based Global Optimization (SMBO) is one of the sampling algorithms, which is a formal definition of Bayesian optimization. This kind of method will try to find a better sample to reduce the number of sampling, so it is more efficient than grid search or random search. Each time a set of parameters is sampled, a model must be trained to obtain the objective value, which is a very time-consuming process, so efficiency is a very important consideration. Compared with evolutionary algorithms that require a large number of initial samples, SMBO is a more efficient choice. SMBO consists of the following four core elements:

1. **Domain:** Each hyper-parameter has its own value range and prior distribution, such as uniform distribution, log uniform distribution, and so on.

2. **Objective function:** The objective function is what we want to optimize, which is the loss function in our case. Given a set of hyper-parameter, fitting a model to the training set after instantiating the hyper-parameters, and the loss on the validation set is obtained as the output value.

3. **Surrogate function:** A substitution function, also called a response surface, is a probabilistic representation of an objective function based on the previous history. It is called a response surface because it is the probability that given a hyper-parameters to obtain such an objective function score at a high dimensional level.

4. **Selection function:** The selection function is how we choose a new set of hyper-parameters to experiment with based on the surrogate function, which is our expectation for the next set of parameters.

To sum up, SMBO is to construct a probabilistic model based on the historical trial records. Since calculating the objective function is too time-consuming, the surrogate function is used to represent the probability response between the

doi:10.6342/NTU202203861

hyper-parameter set and the objective value. With this surrogate function, we can calculate the probability that any hyper-parameter set will reach a specific objective value. Then define the goal of the next hyper-parameter set, which is the selection function, to find a hyper-parameter set closer to our goal.

The detailed process is shown in Algorithm 1 [74], where the λ is a hyper-parameter set and S_t is the surrogate function. There are four steps in total. The first step is to find a hyper-parameter set that is more likely to have a smaller objective function value based on the existing historical trial records. Second, calculate the value of the objective function. Third, add current result to the trial record collection. Finally, update S with the new trial record collection.

Algorithm 1 Sequential model-based global optimization (SMBO)

Input: Objective function L
Output: Best parameter λ^*
Initializing S_0
$H = \emptyset$
for $t = 1$ **to** T **do**
  $\lambda^* = argmin S_{t-1}(\lambda)$
  $c = L(\lambda^*)$
  $H = H \cup (\lambda^*, c)$
  fit new model S_t according to updated H
end for
return λ with minumum c in H

Here we use the **Tree-structured Parzen Estimator (TPE)** [74] as the sampler, which is also an SMBO method. The advantage is that it converges faster and only focuses on searching near better parameters rather than performing a global sampling. We will go into detail in Section 4.1.3.1. We then describe our **optimization process** and setup in Section 4.1.3.2. Finally, we use the TPE results to analyze the utility of each transformation. We remove transformations with low execution rates and search again with the remaining transformations, which are covered in Section 4.1.3.3.

4.1.3.1 Tree-structured Parzen Estimator

The idea of TPE is to search for hyper-parameters close to better samples when we already have some sample points, instead of global sampling like a Gaussian process. The demonstration is shown in Figure 22. When we already have several samples, we naturally think to sample the points near the samples that exceed the threshold because it is more likely to produce a better objective value. This is the assumption of TPE so that it can converge faster.

We have mentioned the basic elements of the sampling algorithm earlier, and

doi:10.6342/NTU202203861

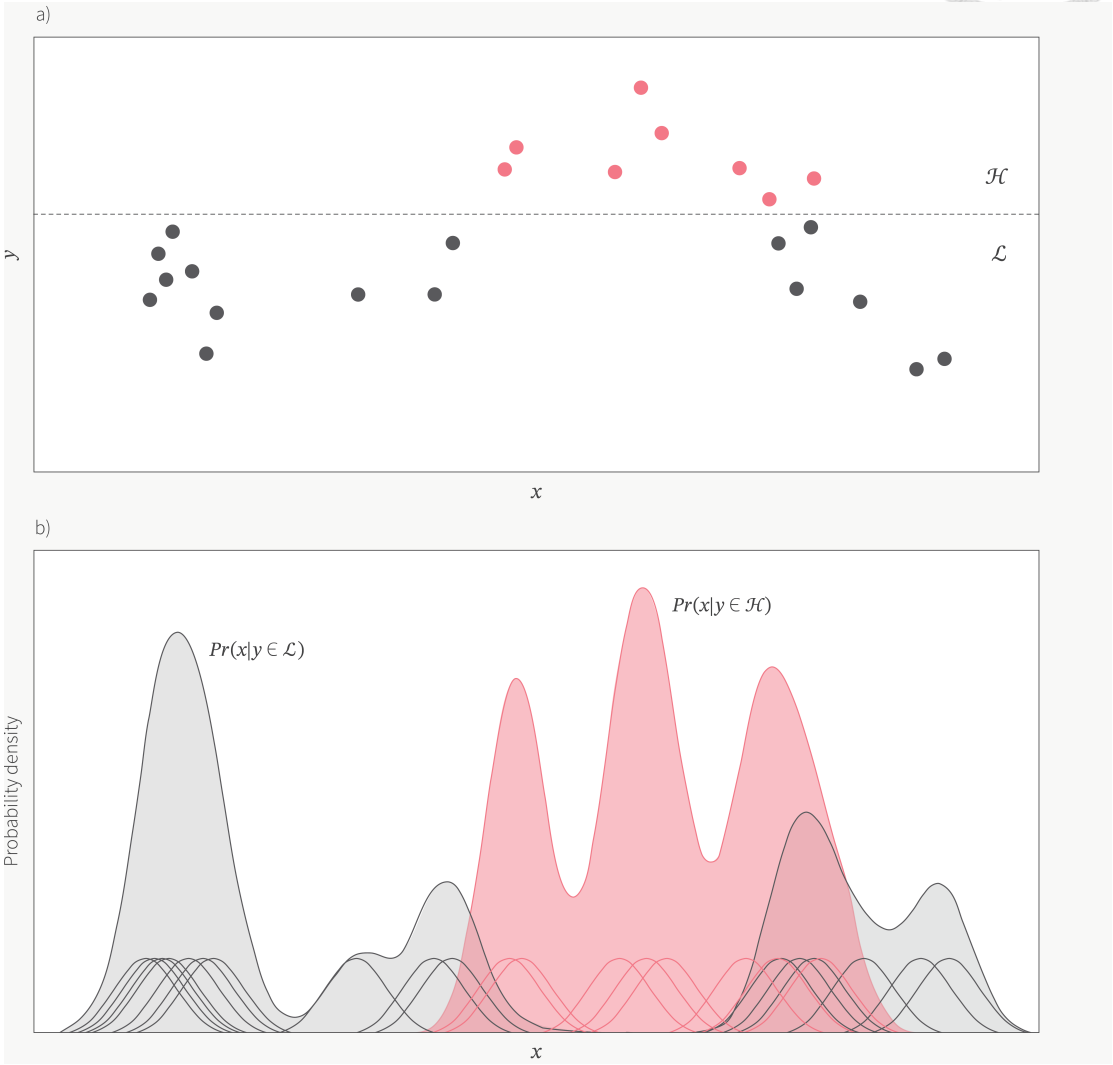


Figure 22: Tree-structured Parzen Estimator. a) Tree-structured Parzen Estimator divides the data into two sets $\mathcal{L}$ and $\mathcal{H}$ by thresholding the observed function values. b) They then build a probability density of each set using a Parzen kernel estimator — they place a Gaussian at each data point and sum these Gaussians to get the final data distribution. A point is desirable to sample next if the probability of $P_r(x|y \in \mathcal{H})$ being in the set $\mathcal{H}$ is large and the probability of $P_r(x|y \in \mathcal{L})$ being in the set $\mathcal{L}$ is low [12].

doi:10.6342/NTU202203861

we will introduce them one by one except for the objective function. The first one is **Domain**. Besides the probability distribution of hyper-parameters in the domain, the relationship between hyper-parameters should be considered. In TPE, they restrict the domain to tree-structured configuration spaces. Configuration spaces are tree-structured in the sense that some leaf variables are only well-defined when node variables take particular values.

Moreover, they remark that not all hyper-parameters are correlated. Thus it is not enough to place one Gaussian process over the entire space of hyper-parameters. They chose to group the hyper-parameters by common use in a tree-like fashion and place different independent Gaussian processes over each group. That is, they sample hyper-parameters from each group independently, and different groups are also independently optimized.

As for the **surrogate function**, they use the Parzen window to model the probability density function between samples and objective function values. The Parzen–Rosenblatt window is proposed by Emanuel Parzen and Murray Rosenblatt in the kernel density estimation problem, which can estimate the probability density of the estimated value based on the currently observed value and the prior distribution type. The definition of the Parzen window is that the probability within the window is divided by the window volume V, and its high-dimensional expression is as follows:

$$p(x) = \frac{k}{nh^D} = \frac{1}{nh^D} \sum_{i=1}^{n} K(\frac{x_i - x}{h}), \tag{4.1}$$

where the k is the number of samples in the window, and n is the total number of samples. The D refers to dimension, and the h^D equals V if the dimension is three. The function K is used to calculate k and is called the window function, or kernel function. In TPE, the kernel function uses a Gaussian kernel function, in which the closer the distance is, the greater the count weight.

Whereas the Gaussian-process based approach modeled $p(y|x)$ directly, TPE models $p(x|y)$ and $p(y)$. As mentioned earlier, TPE establish two probability distributions based on the threshold of the observed loss, so the $p(x|y)$ can be defined as

$$p(x|y) = \begin{cases} l(x) & if\ y < y^* \\ g(x) & if\ y \geq y^*, \end{cases} \tag{4.2}$$

where $l(x)$ is the density formed by the observations that the corresponding loss was less than y^* and $g(x)$ is the density formed by the remaining observations. In the Gaussian-process based approach, the y^* is typically less than the best-observed loss, while the TPE algorithm depends on a y^* larger than the best-observed loss so that some points can be used to form $l(x)$. The TPE algorithm

doi:10.6342/NTU202203861

chooses y^* to be some quantile γ of the observed y values, so that $p(y < y^*) = \gamma$, but no specific model for $p(y)$ is necessary. By maintaining sorted lists of observations, the number of samples used to estimate $l(x)$ can be computed through γ.

The **selection function** used in TPE is Expected Improvement (EI) [75], but they redefine EI according to the surrogate function. EI is the expected value that the objective value of this hyper-parameter set exceeds a certain threshold y^*. If it is in the form of minimizing the loss, the mathematical formula can be written as

$$EI_{y^*}(x) = \int_{-\infty}^{\infty} max(y^* - y, 0)p(y|x)dy. \tag{4.3}$$

Since the surrogate function defines $p(x|y)$, rewrite the formula as

$$EI_{y^*}(x) = \int_{-\infty}^{y^*} (y^* - y)p(y|x)dy = \int_{-\infty}^{y^*} (y^* - y)\frac{p(x|y)p(y)}{p(x)}dy, \tag{4.4}$$

where p(x) can be expressed in Equation 4.2 as follows:

$$\begin{aligned} p(x) &= \int_{\mathbb{R}} p(x|y)p(y)dy \\ &= \int_{-\infty}^{y^*} p(x|y)p(y)dy + \int_{y^*}^{\infty} p(x|y)p(y)dy \\ &= \gamma l(x) + (1-\gamma)g(x), \end{aligned} \tag{4.5}$$

and the remaining molecular part can also be rewritten with Equation 4.2 as the following:

$$\begin{aligned} \int_{-\infty}^{y^*} (y^* - y)p(x|y)p(y)dy &= l(x)\int_{-\infty}^{y^*} (y^* - y)p(y)dy \\ &= y^*\gamma l(x) - l(x)\int_{-\infty}^{y^*} yp(y)dy. \end{aligned} \tag{4.6}$$

Hence, the complete EI function can be derived as

$$\begin{aligned} EI_{y^*}(x) &= \frac{y^*\gamma l(x) - l(x)\int_{-\infty}^{y^*} yp(y)dy}{\gamma l(x) + (1-\gamma)g(x)} \\ &\propto \left(\gamma + (1-\gamma)\frac{g(x)}{l(x)}\right)^{-1}. \end{aligned} \tag{4.7}$$

Equation 4.7 shows that to maximize improvement we would like points x with high probability under $l(x)$ and low probability under $g(x)$. The tree-structured form of l and g makes it easy to obtain many candidates through l and evaluate them according to $g(x)/l(x)$. The algorithm returns the candidate x^* with the greatest EI on each iteration.

doi:10.6342/NTU202203861

To make the sampling process of TPE clearer, we can summarize it into three steps:

1. Set the tree formed by all hyper-parameters. Each group of hyper-parameters must remain independent.

2. The Parzen estimator is used to construct the probability distribution of each group and then sample from the $l(x)$ distribution of each group several times.

3. After sampling the hyper-parameter subset $\bar{x}_i$ from each group , we can get the $l(\bar{x}_i)$ and $g(\bar{x}_i)$ of this subset. Multiplying the probability of each subset to get the joint probability $l(\bar{x})$ and $g(\bar{x})$ of the current hyper-parameter set, and then calculate the $l(\bar{x})/g(\bar{x})$. The hyper-parameter set with maximum $l(\bar{x})/g(\bar{x})$ would be chosen as the sample of this iteration.

4.1.3.2 Optimization Process

In Section 4.1.2, we have already listed the transformation set to search, as well as the hyper-parameters and their bounds. There are a total of 8 transformation functions, each containing an intensity and a probability hyper-parameters, and these 16 hyper-parameters form the search space. Except for hue, none of these intensity hyper-parameters have upper bounds, so we need to define upper bounds for them. To ensure the range is large enough is to check if the image generated using the upper bound is sufficiently indistinguishable.

To evaluate the effect of a hyper-parameter set, we train the model according to the process in Figure 19. All components will use the same as described in this chapter. The validation loss is then calculated using the validation set from a different dataset than the training set, since the main purpose of our augmentation is to adapt the model to different environments. However, the classifier used for training cannot be directly used to classify non-training data, so the validation loss cannot be calculated. Therefore, we average the feature template from registrated images in the validation set to generate the center vector, which is used to simulate the weight vector in the classifier.

We use cross-entropy loss and Center loss when calculating the validation loss instead of the Focal loss and Huber loss mentioned in Section 4.1.4. This is because the loss generated by Focal loss and Huber loss will be smaller than the original, which is not conducive to our observation of the difference between hyper-parameter sets. Their meanings are similar, but the impact on training is different. Therefore, we choose the more intuitive and significant cross-entropy loss and center loss.

doi:10.6342/NTU202203861

Furthermore, the execution probability and the sampling of intensity are both random, indicating that the model trained with the same hyper-parameter set may lead to a very different result. Therefore, we will perform training twice for the same hyper-parameter set during searching. The result representing this hyper-parameter set is the average of the validation losses computed by the two models.

Pruner requires the validation loss of each training epoch to decide whether this hyper-parameter set should be dropped early. Using too much data as a validation set will increase the training time, so we only use half of the classes of the target dataset as a validation set. Note that some images of each category need to be divided into a registration set to calculate the center vector. As a consequence, the image split of each category will also affect the loss calculation, and we use 5 images as a registration set here.

The pruner we use will calculate the median of the validation loss reported on the same epoch for each trial to judge whether the current hyper-parameter set is promising. In other words, it will be discarded if this hyper-parameter set performs less than half of the existing hyper-parameter sets at a certain step. Since we will train twice in total, the definition of the same epoch will be confused, so we only use the pruner for the first training. Moreover, if this hyper-parameter set can complete the first training, it is worth the expense.

We have covered all the tools needed in the hyper-parameter optimization process, including a set of transformation functions and their search space, objective function, sampler, and pruner. The detailed optimization process is presented in Algorithm 2, which can be briefly described as follows:

1. Define the bounds of every hyper-parameter to be searched for the sampler (line 1).

2. At the beginning of each trial, the sampler will sample all the hyper-parameters from the search space (line 4).

3. Train the model with the hyper-parameters to convergence, then calculate the validation loss and record it (line 8 $\sim$ 10).

4. Check if the current hyper-parameter set is worth continuing. If not, terminate the training and start another trial (line 11 $\sim$ 18).

5. After the pre-defined trial numbers, the best hyper-parameter set with the smallest loss could be found (line 23).

doi:10.6342/NTU202203861

Algorithm 2 Optuna optimization process

Input:
1) Pre-defined transformations: $T = \{ T_1, T_2, \cdots, T_N \}$
2) Objective function: O
3) Sampler: S
4) Pruner: P

Output: Best hyper-parameters p^*

```
Define the hyper-parameter bounds of T in S
trialLosses ← [ ]
for i = 0 to number of trials do
    p ← S.sample()
    modelLosses ← [ ]
    for j = 0 to 2 do
        for e = 0 to number of epoch do
            Training model M with T, p
            l ← O(M)
            Save (i, e, l) into P
            if l < P.median(e) then
                P.prune()
                break
            end if
        end for
        if P.pruned then
            break
        end if
        modelLosses.append(l)
    end for
    trialLosses.append(p, average(modelLosses))
end for
p* ← min(trialLosses)
return p*
```

4.1.3.3 Transformation Selection with TPE

In the TPE algorithm, the existing parameters are only searched for optimization, and no selection is made. However, these selected transformations are just guesswork based on the possible scenario and have no credible basis. A common hyper-parameter analysis method uses functional ANOVA [76] to evaluate the importance, but this method requires using random sampling to have more accurate results. Therefore, we directly decide whether or not to use this transformation based on the results of the TPE search.

Each transformation has a hyper-parameter of execution probability, which can be a good indicator of the utilization of the transformation. With this, we can use a threshold to filter out those transformations with low usage. First, we take the top $n\%$ samples from the TPE search results and average the probability of

doi:10.6342/NTU202203861

each transformation to represent their usage. Then set a threshold U to remove transformations with usage lower than this value. In this way, we can find those transformations that are more suitable for palmprint recognition in the current workflow.

Additionally, we can narrow down the intensity hyper-parameters through the TPE search results for a more precise search. We take the upper and lower bounds of the best half samples as the new hyper-parameter intensity range. We perform a iterative search with these newly selected transformations and a new range of hyper-parameters with the same process described earlier.

4.1.4 Loss Function for Extreme Samples

We have found that using the 0.1 lambda (coefficient of Center loss) and the learning rate of 0.01 provided by the original paper would cause the training to fail. The loss value would be too large to learn at the beginning of training. We successfully train by adjusting the learning rate afterward. If we adjust lambda, the impact on the final result will vary greatly, which means that the parameter lambda is very sensitive. In addition, the augmentation method we use may produce a large transformation of the image. These samples will cause Center loss to generate a large loss value, which in turn affects the stability in the later stage of training.

To solve the above problems, we turn to **Huber loss**. It is the synthesis of squared loss and absolute loss, which we will introduce in detail in Subsection 4.1.4.1. In addition, we also replace the original cross-entropy loss with Focal loss. Focal loss is characterized by reducing the weight of easy samples and increasing the weight of hard samples so that the model pays more attention to these hard samples. This may seem to conflict with the original purpose, but they are compatible and reinforce the effect of data augmentation. Because Focal loss mainly balances the gradient proportions of hard and easy samples, it will not cause the hard samples to generate excessive gradients, leading to unstable training. Conversely, it reduces the loss value of hard samples, and we will explain **Focal loss** in more detail in Subsection 4.1.4.2.

4.1.4.1 Huber Loss

Huber loss [77] is proposed to improve the robustness of the squared loss function to outliers. The form of Center loss is the same as square loss, so it can solve the problem that Center loss is too sensitive to outliers. Given the prediction $\hat{y}$

doi:10.6342/NTU202203861

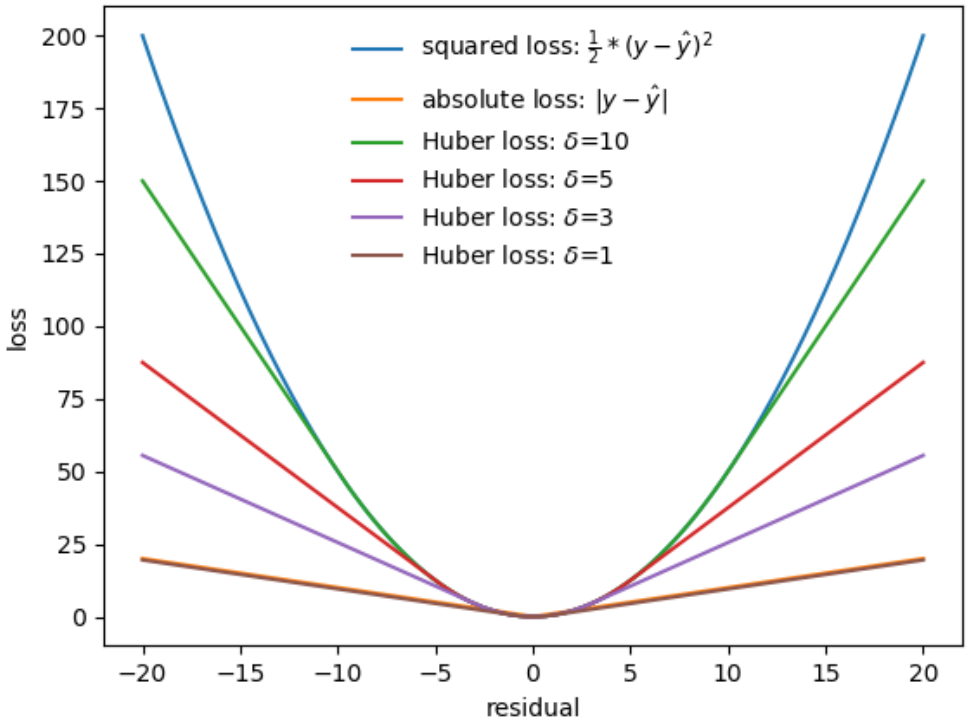


Figure 23: Residual-loss curve of Huber loss between different parameters from Huang [13]

and the ground truth y, the function could be formulated as

$$L_\delta(y, \hat{y}) = \begin{cases} \frac{1}{2}(y - \hat{y})^2 & if\ |y - \hat{y}| \leq \delta \\ \delta(|y - \hat{y}| - \frac{1}{2}\delta) & otherwise, \end{cases} \tag{4.8}$$

where the δ is a hyper-parameter needed to be tuned. The residual-loss curves of different δ are shown in Figure 23.

From Equation (4.8), we can know that Huber loss consists of two parts. One is the part where the absolute value of the error is less than δ, which is equivalent to squared loss, and the rest is equivalent to absolute loss. This design can make the whole curve smoother and combine both advantages. The merit of squared loss is that the gradient will gradually decrease as the loss value approaches its minimum value, which benefits convergence. However, its problem is that when the residual is large, it will bring too large gradients, which was mentioned earlier. As for absolute loss, it is the other way around. Although it will not cause an excessive gradient, the gradient remains large even when the residual is close to its minimum value, which is not conducive to convergence. Thus, combining these two can perfectly keep the benefits and do away with the drawbacks.

Another benefit is the ease of tuning parameters. From Figure 23, it can be

doi:10.6342/NTU202203861

found that the residual-loss curve does not change drastically between different δ. It means that there would not be a significant variation in performance between similar δ, and thus no parameter tuning issues mentioned above. We directly set the coefficient of Huber loss to 1, and balance the ratio between it and the classification loss through δ.

4.1.4.2 Focal Loss

Focal loss [14] was originally proposed for object detection tasks. Because most of the candidate objects found in a picture are from the background, these are all simple negative samples. Therefore, there will be an extreme imbalance in the calculation of loss, resulting in the optimization direction of the model is not what we want. In response to such an imbalance problem, Lin et al. [14] proposed Focal loss based on cross-entropy loss, which multiplies the original cross-entropy loss by a modulating factor. The purpose is to reduce the weight of easy samples so that the model can focus more on hard samples during training.

The cross-entropy loss is the same as the softmax loss. Thus, the formulation could be found in Equation (3.1). Take the binary classification as an example. The original classification loss is the direct summation of the cross-entropy of each training sample. That is, the weight of each sample is the same. The formula is as follows:

$$CE(p, y) = \begin{cases} -\log(p) & if\ y = 1 \\ -\log(1-p) & otherwise, \end{cases} \tag{4.9}$$

where p represents the model's estimated probability for the class with label $y = 1$, ranging from 0 to 1. For notational convenience, they define p_t:

$$p_t = \begin{cases} p & if\ y = 1 \\ 1-p & otherwise, \end{cases} \tag{4.10}$$

and the cross-entropy loss could be simplified as

$$CE(p, y) = CE(p_t) = -\log(p_t). \tag{4.11}$$

Since the problem is that easily classified negatives comprise most of the loss and dominate the gradients, they first proposed the alpha-balanced cross-entropy loss, which uses a weighting factor α to balance the weights between positive and negative samples. The definition of α is almost the same as p_t, $\alpha \in [0, 1]$ for $y = 1$, $1 - \alpha$ otherwise. The α-balanced CE loss could be written as

$$CE(p_t) = -\alpha_t \log(p_t). \tag{4.12}$$

doi:10.6342/NTU202203861

Because alpha can only balance the importance of positive and negative samples, it does not differentiate between easy and hard samples. Therefore, they propose another loss function to down-weight easy examples and thus focus training on hard samples. They used a modulating factor to replace the alpha coefficient and defined the Focal loss as

$$FL(p_t) = -(1-p_t)^{\gamma}\log(p_t), \tag{4.13}$$

where the γ is the focusing parameter greater than 0. The focusing parameter γ smoothly adjusts the ratio of reducing the weight of easily classified samples. When $\gamma = 0$, the Focal loss is equal to the traditional cross-entropy loss, and the influence of the modulating factor will increase while γ increases. The Focal loss curves of different γ are visualized in Figure 24.

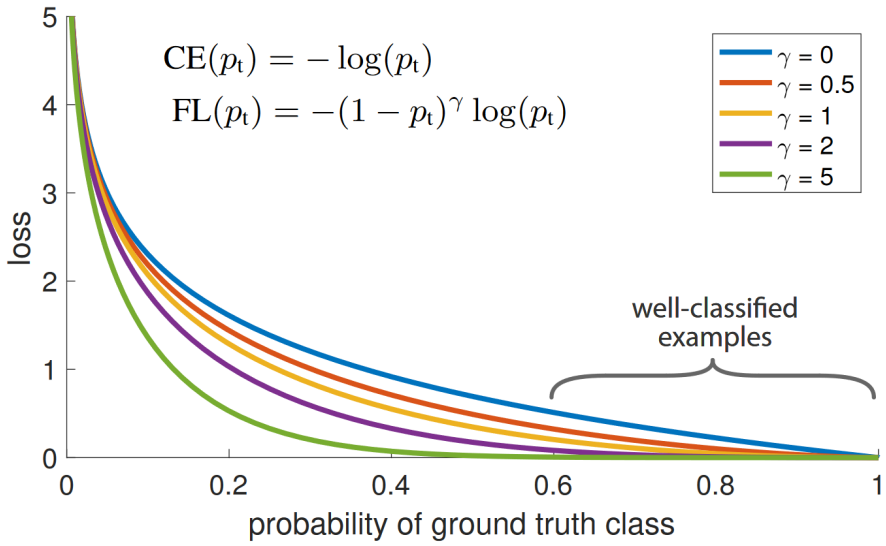


Figure 24: Probability-loss curve of Focal loss between different gamma from Lin et al. [14]

When a sample is misclassified, p_t is small. The modulating factor $(1-p_t)$ is close to 1, so the loss value is slightly reduced. However, when p_t is close to 1, the modulating factor $(1-p_t)$ will be close to 0, so the weights of well-classified samples will be significantly reduced. All in all, Focal loss is to balance the importance between hard samples and easy samples by adding a coefficient, which down-weights hard samples and easy samples with different ratios.

Although the authors also mentioned combining the two into alpha-balanced Focal loss, the choice of parameters will become complicated. Our main purpose is to control the impact of hard samples on the training process, because our augmentation may generate hard samples. Hence, we only use modulating factor to make the model pay more attention to hard samples without being overly sensitive.

doi:10.6342/NTU202203861

4.2 Reduced ResNeSt-50

As seen from Section 3.4.2, the ResNet-18 performs well in the constrained dataset but not in an unconstrained dataset like MPD. Therefore, we need to improve the generalization ability of the model in an unconstrained dataset. Otherwise, even if we use the data augmentation technique to generate many samples, the model will not be able to fit on such a complex training set.

After ResNet, many models have been proposed. Given the characteristics of palmprint ROI images, we think that increasing the model width would be a better direction, namely split-transform-merge. The design concept behind split-transform-merge is to make use of the separable characteristic of convolution to increase the network width, thereby increasing the representation ability of the network without requiring more computing power. This kind of design focus on the fusion of channel features from different dimensions, and it starts from Inception [78] and converges to ResNeXt [79]. ResNeXt perfectly interprets it in a simple way, and it also does a very good job in palmprint recognition. However, ResNeSt [15] combines the design concept of squeeze-and-attention on this basis, called the Split-Attention network. Because of its good integration of the two mainstream design concepts for image classification models, it is also the best-performing model of the ResNet series. Therefore, we will use **ResNeSt** as the backbone model and introduce it in Section 4.2.1.

Furthermore, we wonder if the effectiveness of the deep structure. We only need the texture information of the palmprint ROI, not the semantic information like image classification. The advantage of the deep structure of deep learning is that it can use the hierarchical convolution structure to convert the original image into meaningful feature embeddings. Therefore, shallower models may be more suitable for palmprint recognition, so we propose **reduced ResNeSt**, which will be discussed in Section 4.2.2.

4.2.1 ResNeSt

As a member of the ResNet family, ResNeSt also uses classic architecture, which is shown in Figure 6. This architecture comprises three parts: stem, building blocks, and output head. They have proposed several tweaks to network structure and training strategies, but we will only explain the core part, which is the building block.

ResNeSt block is a bottleneck structure. First, reduce the number of channels through 1×1 CNN, then use 3×3 CNN for feature processing in the middle, and finally restore the number of channels through 1×1 CNN. The 1×1 CNN at the bottom of Figure 25(a) is the part that used for channel restoration. As

doi:10.6342/NTU202203861

Table 6: ResNeSt architectures

Output	ResNeSt-26	ResNeSt-50	ResNeSt-101
112×112	3×3, 32, stride 2		3×3, 64, stride 2
56×56	3×3, 32, stride 1 3×3, 64, stride 1 3×3 max pool, stride 2		3×3, 64, stride 1 3×3, 128, stride 1 3×3 max pool, stride 2
56×56	$\begin{bmatrix} 1 \times 1, 64 \\ \text{SA[C=1,R=2]}, 128 \\ 1 \times 1, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 64 \\ \text{SA[C=1,R=2]}, 128 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ \text{SA[C=1,R=2]}, 128 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
28×28	$\begin{bmatrix} 1 \times 1, 128 \\ \text{SA[C=1,R=2]}, 256 \\ 1 \times 1, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 128 \\ \text{SA[C=1,R=2]}, 256 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ \text{SA[C=1,R=2]}, 256 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
14×14	$\begin{bmatrix} 1 \times 1, 256 \\ \text{SA[C=1,R=2]}, 512 \\ 1 \times 1, 1025 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 256 \\ \text{SA[C=1,R=2]}, 512 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ \text{SA[C=1,R=2]}, 512 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
7×7	$\begin{bmatrix} 1 \times 1, 512 \\ \text{SA[C=1,R=2]}, 1024 \\ 1 \times 1, 2048 \end{bmatrix} \times 2$	$\begin{bmatrix} 1 \times 1, 512 \\ \text{SA[C=1,R=2]}, 1024 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ \text{SA[C=1,R=2]}, 1024 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
1×1	global average pool		
128	fully connected layer		

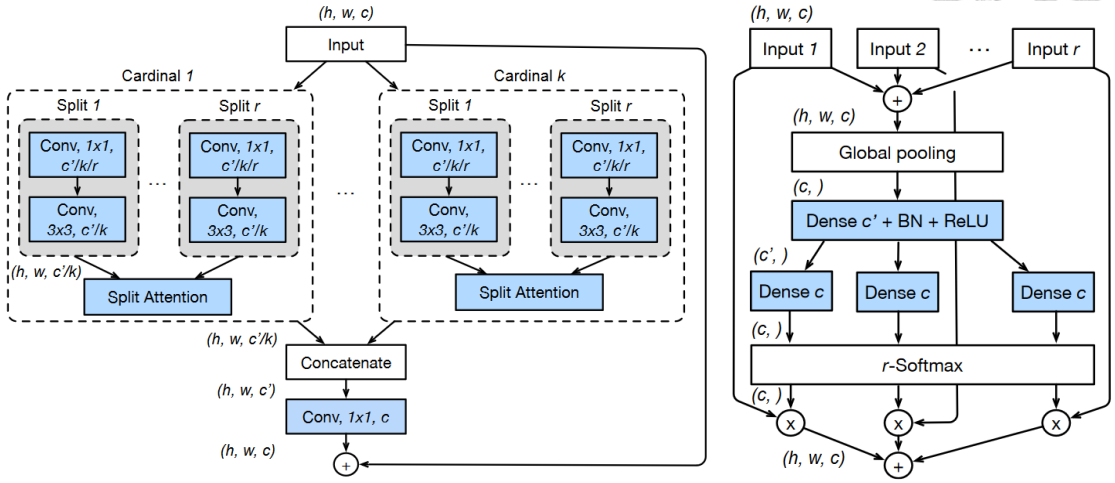
Note: The SA refers to the split attention module, where the C and R are cadinality and radix, respectively.

for the upper part, it draws on the multi-path of ResNeXt [79] and the feature map Attention of SKNet [80]. The input is first divided into k cardinal groups, which is the same as ResNeXt. Each cardinal group is further divided into r radix, and the output of the splits would be processed by the split-attention module and concatenated at the end, where k and r are hyper-parameters.

The purpose of the split attention module is to re-weight each feature channel by the channel attention mechanism. It captures the channel correlation by the following processes. The first step is squeezing the global context by the global average pooling. The output of this step would be a vector of the same length as the number of channels. Then perform an operation similar to squeeze-and-excitation on this vector, first reducing the vector and then restoring the length to achieve cross-channel feature fusion. This channel attention is conducted group-wise, so it is more nuanced than previous methods. Finally, the attention factors of different groups are weighted into the original grouping features, and feature fusion is performed to achieve attention assignment. The operation details are depicted in Figure 25(b).

Although the structure is straightforward and intuitive, which adds channel attention to each group, it is not easy to modularize and accelerate using standard CNN operators [15]. So they came up with another structure called radix-major implementation. The overview of the radix-major ResNeSt block is illustrated

doi:10.6342/NTU202203861



(a) ResNeSt block (b) Split-attention module

Figure 25: The block structure of ResNeSt from Zhang et al. [15]

in Figure 26. The input feature is first divided into rk groups, in which each group has a cardinality index and a radix index. After the convolution operation, the summation is conducted across different splits. The feature map groups with the same cardinality index, but different radix indexes are fused. The global average pooling layer keeps the channel dimension separated, which is the same as conducting pooling for each cardinal group and then concatenating the results. The next two dense layers have the number of groups equal to the cardinality, which produces the attention for each split as in the original structure. Then, the attention is turned into weights through softmax. Finally, the weighted sum of the features in each cardinal group is also conducted.

Such an implementation can greatly simplify the block construction. The first 1×1 convolutional layer can be unified into one layer, and the 3×3 convolutional layer can be replaced by a single grouped convolution with the number of groups equal to rk. As for the remaining attention and convolution layers, it is also not a problem. Therefore, we will use this ResNeSt implementation as our backbone model for palmprint recognition.

4.2.2 Reduced ResNeSt

While the standard ResNeSt has performed well, we will make some adjustments to make it more suitable for palmprint recognition tasks. Y. Liu and A. Kumar [52] have mentioned that palmprint patterns do not reveal rich structural information or meaningful hierarchies like facial images. Among conventional methods, the texture-based methods are considered to be more accurate methods, which often use small-sized filters and block-based operations to extract palmprint

doi:10.6342/NTU202203861

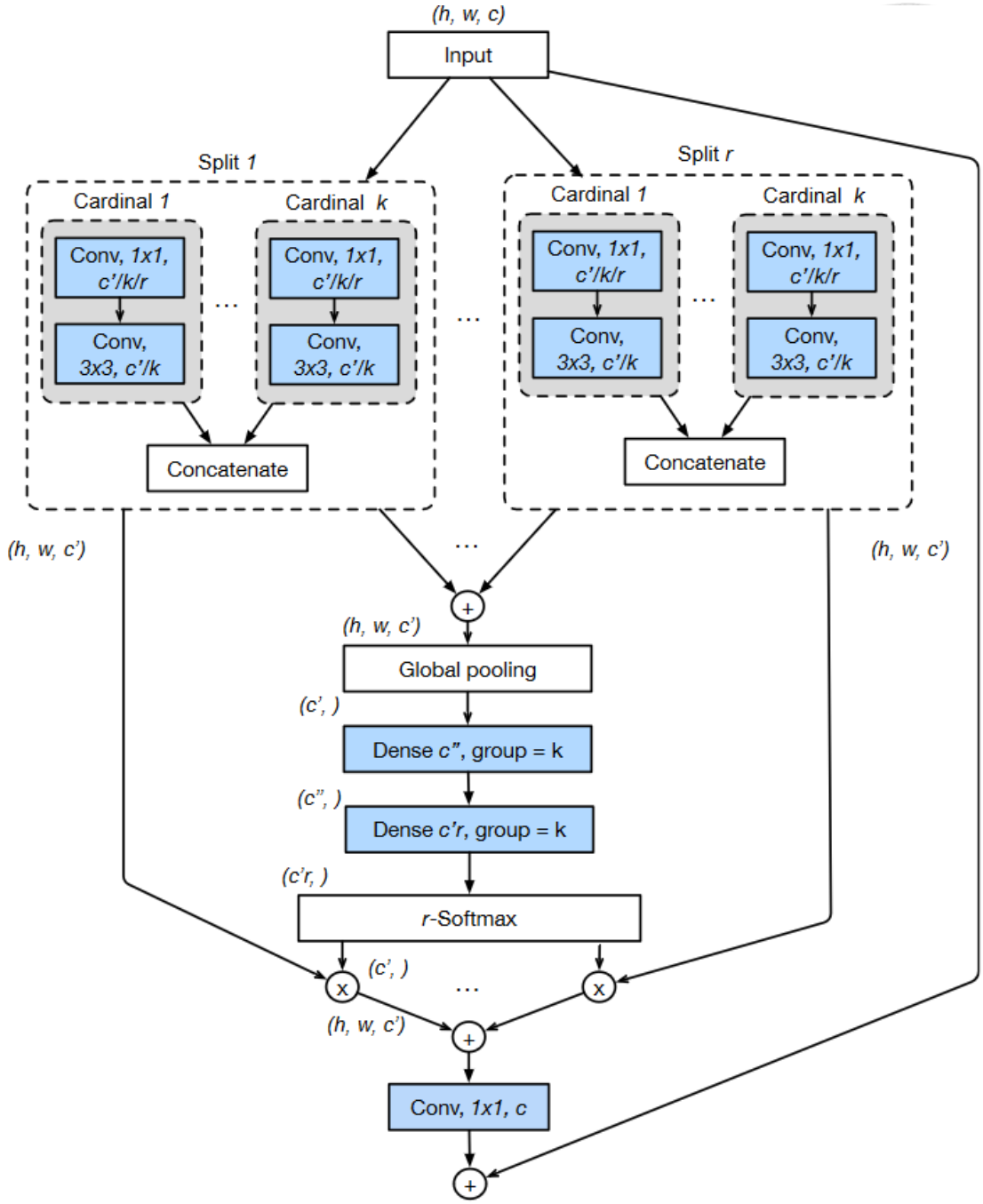


Figure 26: Radix-major implementation of ResNeSt block from Zhang et al. [15]

features. Therefore, they infer that the most discriminative information in a palmprint image is from the local intensity distributions in the ROI image rather than global features. However, the deep structure of the CNN models is used to be effective in capturing high-level or global information, so we believe that the model does not need to be too deep. The ResNeSt-26 and ResNeSt-50 would be better choice. Furthermore, we also wonder if such a four-layer structure is necessary.

Due to the characteristics of deep CNN models, it is difficult for us to prove the role of a single layer directly. Even if we use visualization techniques to draw

doi:10.6342/NTU202203861

the outputs and gradient amounts of the intermediate layers or the activations of CNN kernels, we cannot be sure whether there are useless blocks. Therefore, we try to remove one model layer to verify the effect of depth. Because the entire model is tightly linked, we can not remove the middle layers directly. The only choice is the layer before the global pooling layer because the global pooling layer will only leave one value per channel regardless of the length and width of the input feature map. In this way, we examine the training process and performance between models with different layers, including the original model, the model with one layer removed, and the model with two layers removed.

We found that the validation loss of the model with the last layer removed is lower than the other two, and the training speed is quite fast. In particular, the ResNeSt-50 with one layer removed is not only much better than others, but also has the fastest training speed. Furthermore, we observe that the model with the last layer removed performs the best on ResNet. Therefore, we infer that using such a three-layer architecture for palmprint recognition would be a better choice. Consequently, we used ResNeSt-50 without the last layer as the backbone model for palmprint feature extraction and call it reduced ResNeSt-50. The detailed model structure is shown in Figure 27.

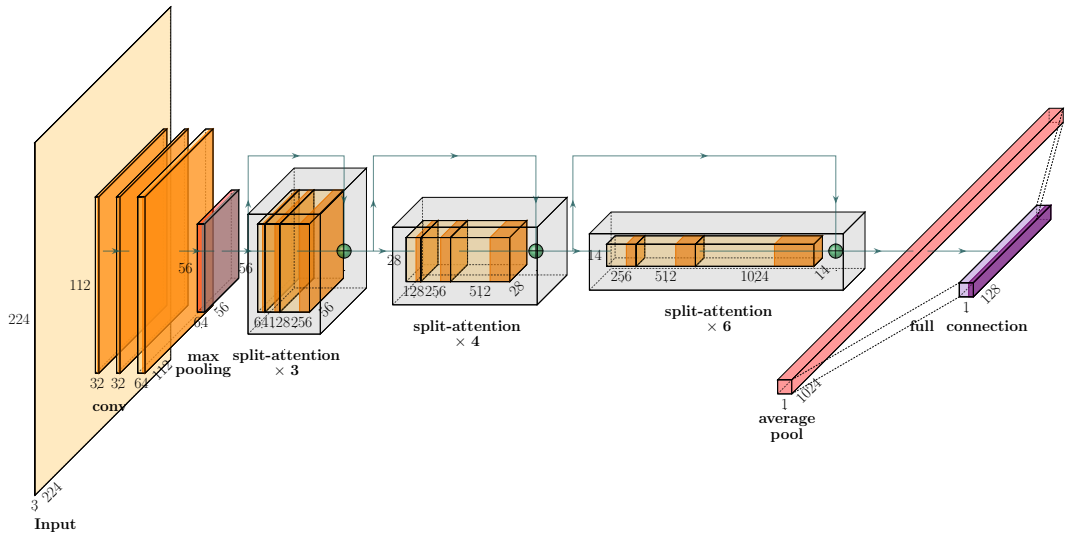


Figure 27: The architecture of reduced ResNeSt-50

4.2.3 Pre-trained Model

Pre-trained models have many benefits, such as transfer learning or speeding up training time, especially in few-shot image classification, which can achieve surprising results [81]. He et al. [82] have summarized the effect of pre-training

doi:10.6342/NTU202203861

on ImageNet. Their conclusions show that similar results can be achieved on ImageNet with or without pre-training and that using ImageNet pre-trained models does not necessarily improve the accuracy of the target task. However, they also mention that the target dataset must be sufficiently large for direct training. In addition, they support the use of pre-trained models to speed up the convergence of the target task, and such an advantage reduces research cycles, leading to easier access to encouraging results. On the other hand, Hendrycks et al. [83] shows that although pre-training does not improve performance on traditional tasks, it can improve robustness and uncertainty estimates.

In view of the above, we believe that the use of pre-trained models is necessary and important. The advantages of the pre-trained model can be organized into the following four points:

1. There is no current study pointing out the disadvantages of using pre-trained models.

2. It has been widely proven that pre-trained models can speed up training time.

3. Pre-trained models may improve accuracy when the target dataset is not large enough. The existing palmprint datasets are not large enough in terms of volume and variation.

4. The pre-trained model may improve the performance across datasets because it can improve robustness and produce better feature representation.

4.3 Multi-Transform Matching

Test-time augmentation is a technique that also uses transformation at recognition time, and the matching methods we mentioned in Section 3.3 also belong to this category. In [70], they collected several research studies showing that incorporating augmented data during testing is effective and reduces highly confident but inaccurate predictions. Such a method has been around for a long time, as early as in the Alexnet paper [84], they average the predictions on ten randomly cropped patches. In this section, we will first introduce our ensembled matching method in Section 4.3.1. Then, we discuss the transformations used in recognition time in Section 4.3.2.

4.3.1 Ensembled Matching

From Section 3.4.1 we have discovered that existing matching methods allow the original input image and its mirror image to influence each other through

doi:10.6342/NTU202203861

feature norms. If the norm of the features extracted by the model from this mirror image is large, it is likely to cause identification errors. Since we aim to use these additional images to aid recognition, we want to avoid negative effects like this.

Instead of combining features, we perform feature matching separately so that they do not affect each other. We compare the original images of the two input images to generate a similarity score, then compare their mirror images to generate another similarity score, and finally average the two scores to get the final similarity score. In addition, this approach can be more flexible to add other kinds of transformations to increase the confidence of identification results further. We refer to this method as multi-transform matching, and the matching process is shown in Figure 28. Thus, given a set of transformations T, the final similarity score can be denoted as

$$S(A, B) = \frac{1}{N} \sum_{n=1}^{N} D(M(T_n(A)), M(T_n(B))), \tag{4.14}$$

where A and B are two input images, N is the number of transformations we use, and the D and M are distance function and feature extraction model. The distance function we use here is cosine similarity.

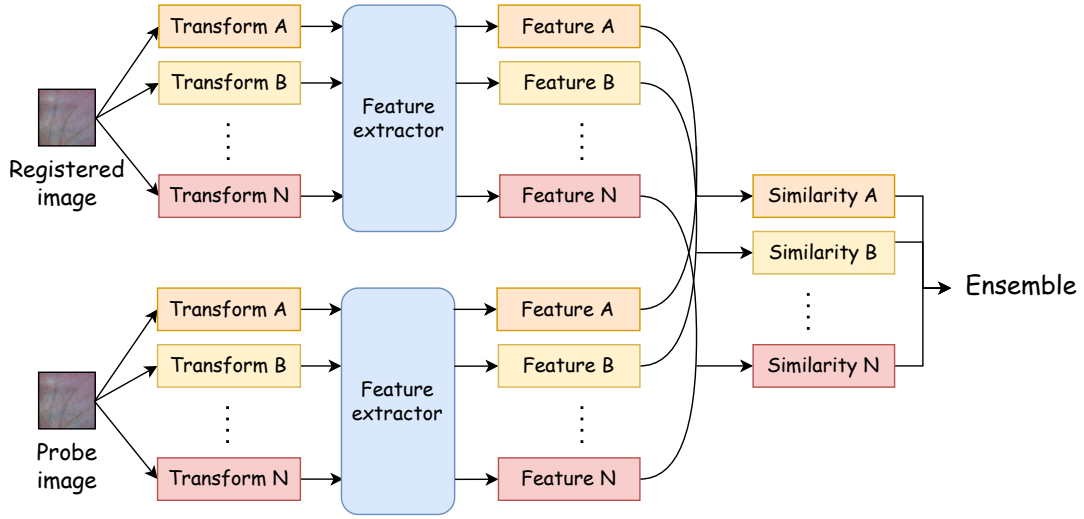


Figure 28: Illustration of our multi-transform feature matching

We further organize this approach into an algorithm to make it clearer. A palmprint recognition system generally includes the following three behaviors: registration, identification, and verification, which we have already introduced in Figure 1. The registration procedure is detailed in Algorithm 3, which mainly converts the input image into features and stores them. Here, due to the use of test-time augmentation technique, several features will be stored according to the number of transformations used.

doi:10.6342/NTU202203861

The feature matching part of identification and verification is the same; the main difference is the number of matches. Because the verification process is simpler, we can focus more on the feature matching part, so we only describe the verification process here. The verification procedure is presented in Algorithm 4. The most important part is in line 3 to 7, which generate the features from transformed images respectively and calculate the cosine similarity with the registration templates. Note that the T_0 in the pre-defined transformation set is an identity function that takes input as output directly. Finally, the similarity of all pairs are averaged to produce the final similarity score (line 9).

Algorithm 3 Registration procedure

Input:
1) Pre-defined transformations: $T = \{ T_0, T_1, \cdots, T_N \}$
2) Feature extraction model: M
3) Registered templates: G
4) Input image: X

Output: Registered templates: G

```
1: templates ← [ ]
2: for t ∈ T do
3:     img ← t(X)
4:     feature ← M(img)
5:     templates.append(feature)
6: end for
7: G.append(templates)
8: return G
```

Algorithm 4 Verification procedure

Input:
1) Pre-defined transformations: $T = \{ T_0, T_1, \cdots, T_N \}$
2) Feature extraction model: M
3) Registered templates: G
4) Input image: X
5) Query identity: I

Output: Similarity score

```
1: similarities ← [ ]
2: Find index i of query identity I in the registered templates G
3: for j = 0 to N do
4:     img ← T_j(X)
5:     feature ← M(img)
6:     similarity ← cos(feature, G_ij)
7:     similarities.append(similarity)
8: end for
9: score ← average(similarities)
10: return score
```

doi:10.6342/NTU202203861

4.3.2 Transformations for Recognition Time

There are three points to consider about the transformation used in recognition. First, we must be sure that it is reasonable to use such an image transformation for this task. That is, the model needs to be able to recognize these transformed images. Second, the model must be robust enough with a low variance in feature extraction across transformations. Finally, the computational cost of these transformations and the impact on recognition speed are application considerations. We will cover these points in the following explanation.

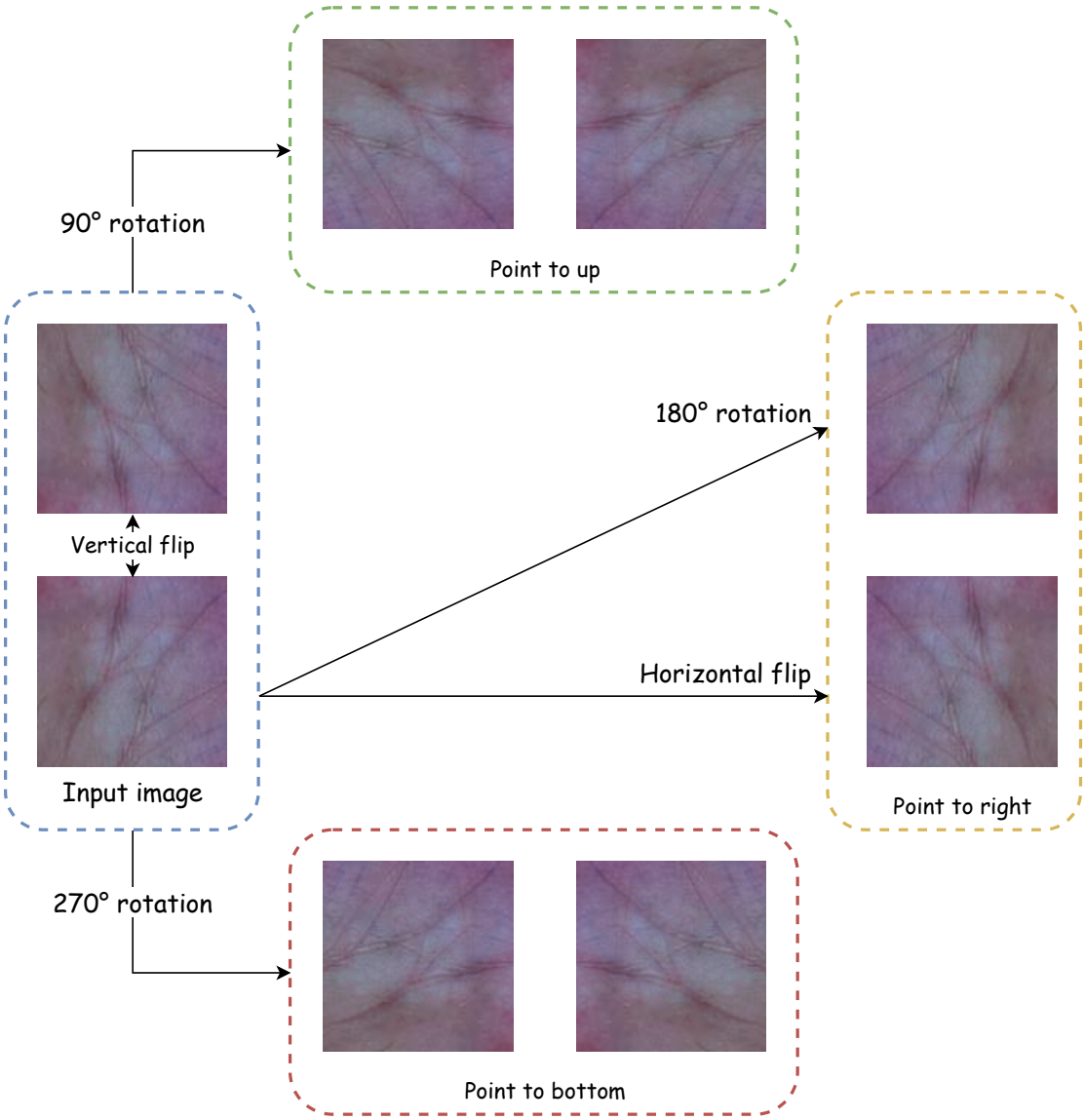


Figure 29: A left hand image can be converted to form left and right hand images in four orientations, a total of eight images.

We already know that adding mirror image for feature matching is helpful, and rotation is a similar transformation. In Section 4.1.1, we also showed that these rotated images can be recognized by the model and are also helpful for

doi:10.6342/NTU202203861

feature matching. Therefore, we can infer that both rotation and mirroring are transformations that improve the recognition effect. These two transformations can expand an input image into 8 images in total, including left and right hand images in four orientations, as shown in Figure 29.

Although these transformed images can improve the matching performance, the vertical flip is the best. The reason it is the best comes from the ROI cropping algorithm, where the last step rotates the image so that the fingers point to the left. It makes a left hand image form a texture similar to the right hand in the same orientation after a vertical flip. Because we have trained the left and right hand images in this orientation, it is natural that flipping vertically works best.

To satisfy the aforementioned conditions, make the model robust enough to produce similar recognition results under these transformations. Therefore, we employ the oversampling augmentation method described in Section 4.1.1. This approach successfully enables the model to recognize through these transformed images and maintain the same level of improvement from these transformed images.

Since no matter which of these newly added images is used, the improvement in recognition accuracy is similar, so we only need to consider the number of additional images. Although using more transformations can lead to more performance gains, both transformations and additional matching increase computational cost, which in turn reduces the speed of identification. In practical application scenarios, more consideration should be given to hardware specifications and user experience, so adjustments should be made according to the situation.

Only 5 additional images were used in our experiments. Because the more we use, the less the improvement will be. Moreover, The remaining two need to convert the input image twice, so the computation cost is relatively high. Therefore, we totally compare 6 feature pairs for every two input images in our multi-transform matching.

doi:10.6342/NTU202203861

CHAPTER 5

PERFORMANCE EVALUATION

In this chapter, we evaluate the three improvements proposed in Chapter 4. Before that, we introduce the datasets used in these evaluations, which are detailed in Section 5.1. Since the model is the basis of all methods, we will first demonstrate the advantages of our proposed Reduced ResNeSt in Section 5.2. This is followed by the results of our hyper-parameter optimization search for data warping augmentation and the improvements it brings are described in Section 5.3. Then, We evaluate our proposed multi-transform matching in Section 5.4. Since our oversampling augmentation is developed from it, its effect is also presented in this section. Details about the metrics and implementation are explained separately in each section. Finally, we summarize the performance improvements brought by our three proposed data augmentation methods.

5.1 Datasets

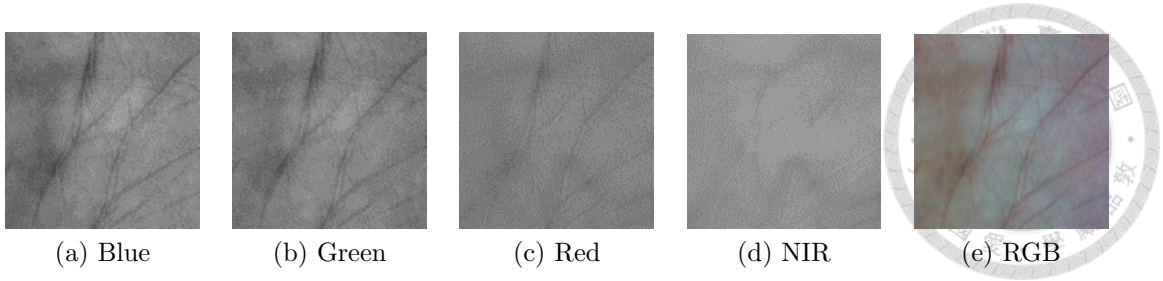
5.1.1 PolyU Multispectral Palmprint Dataset

The PolyU multispectral palmprint dataset [3] has been widely adopted in palmprint recognition for a long time. The images of the dataset were acquired under four channels of the image spectrum, i.e., the blue channel, the green channel, the red channel, and the NIR channel. They recruited a total of 250 volunteers, including 55 women and 195 men, between the ages of 20 and 60. The images were collected in an enclosed space with a fixed position and from two sessions at a time interval of 9 days. During each session, the volunteers were asked to provide 6 images for each palm. Hence, 12 images were collected from a single palm under one certain channel. In total, the database contains 6000 (500 palms * 6 shots * 2 sessions) images for one channel. In order to standardize the application, a set of cropped ROI images with a size of 128 $\times$ 128 pixels was released for public use. Some examples are shown in Figure 30. The red, green, and blue channels were concatenated in our experiment to form RGB palmprint images.

5.1.2 Tongji Contactless Palmprint Dataset

The Tongji contactless palmprint dataset [4] is the first large-scale contactless palm image dataset. They designed image acquisition equipment and collected both palmprint and palm vein images. Although the images were divided into

doi:10.6342/NTU202203861



(a) Blue (b) Green (c) Red (d) NIR (e) RGB

Figure 30: The sample images of PolyU-M dataset [3]. (e) is the concatenated RGB image.

two datasets, they were captured at the same time. During acquisition, only one instruction is given to the subjects: they need to stretch their hands and naturally make the finger gaps observed on the screen. There were 300 volunteers in total, with 192 men and 108 women. 235 of them were between the ages of 20 and 30, and the rest were between 30 and 50. The images were collected in two separate sessions, and the average time interval between the two acquisition sessions was about two months. In each session, 10 images of each palm were captured from each volunteer. In total, the dataset contains 12000 (600 palms * 10 shots * 2 sessions) images from 600 different palms. Figure 31 shows the examples from their public website [16].

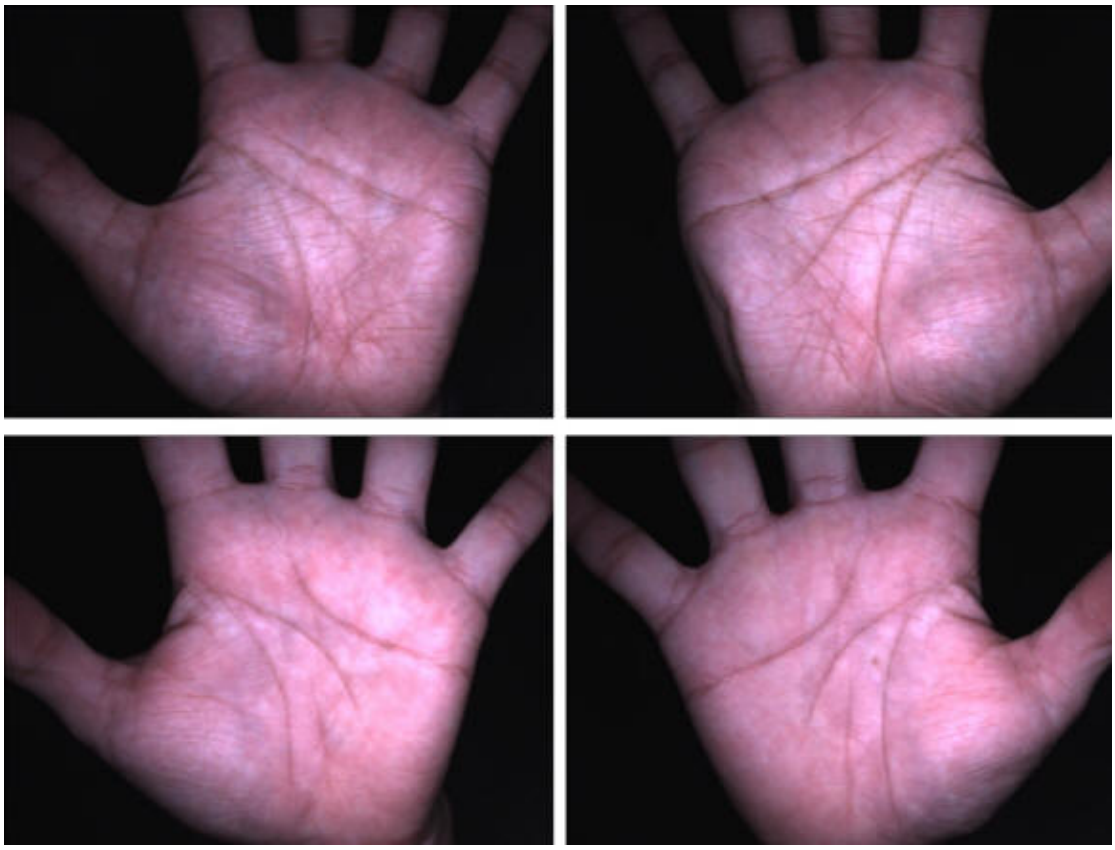


Figure 31: The sample images of Tongji contactless palmprint dataset from [16]

doi:10.6342/NTU202203861

5.1.3 Tongji Mobile Palmprint Dataset

The palmprint images of the Tongji mobile palmprint dataset [24] were collected by two kinds of smartphones, Huawei and Xiaomi, in an unconstrained manner. Thus, they have a variety of backgrounds and lighting environments. MPD comprises 16,000 palmprint images from 200 volunteers in two sessions at a time interval of half a year. Among those volunteers, 195 subjects were 20 to 30 years old, and the others were from 30 to 50 years old, with a balanced gender ratio. Each volunteer is asked to provide 10 palm images of each hand in each session using each smartphone, which means there are 40 images of each hand. They also provide a set of cropped ROI images for public use. Figure 32 shows the examples from their public website [17].

Figure 32: The sample images of Tongji mobile palmprint dataset from [17]

In our experiment, the images collected by different mobile phones were selected as different datasets. Although the images captured by the two phones may not have a huge difference, they come from different acquisition devices. Our goal is to bridge the gap between the different environments, and the different acquisition devices are regarded as different environments in our application context. So there are two datasets denoted as MPD(h) and MPD(m) that indicate Huawwi and Xiaomi, respectively. Each contains 8,000 (400 palms * 10 shots * 2 sessions) palmprint images belonging to 400 categories. The details of the datasets we used in the experiments are summarized in Table 7.

doi:10.6342/NTU202203861

Table 7: Some details of different palmprint datasets

Database	Acquisition type	Palms	Images	Images per palm
PolyU [3]	Constrained	500	6000	12
Tongji [16]	Partly unconstrained	600	12000	20
MPD(h) [17]	Unconstrained	400	8000	20
MPD(m) [17]	Unconstrained	400	8000	20

5.2 Evaluation of the Reduced ResNeSt-50

This section mainly examines our proposed improvements to the feature extraction model, the details of which are introduced in Section 4.2. We first demonstrate the advantages of the pre-trained model in Section 5.2.1, then examine the effect of ResNeSt layer reduction in Section 5.2.2, and finally compare our proposed Reduced ResNeSt-50 with the baseline models introduced in Chapter 3 in Section 5.2.3.

In this section, the experiments follow the training process in Figure 19, and we only change the feature extraction model. To evaluate the model, we mainly consider training and validation loss curves to compare convergence speed and cross-dataset recognition performance. The significance of verification loss as performance is that it includes L2 distance loss and classification loss. L2 distance loss can be regarded as the aggregation degree of the classes, and classification loss is directly related to the accuracy of recognition.

Note that the calculation for training loss and validation loss is different. The computation of the loss requires the assistance of the classifier, but the classifier can only be used for classes in the training set. Therefore we cannot directly calculate the validation loss. To address this problem, we average the feature template of each registered image as the center vector to replace the weights in the classifier, because the weights in the classifier are regarded as the center vector in the L2 distance loss during training.

We mentioned using softmax loss and center loss as our metrics during data warping search. Because their values are relatively large, and have better discrimination. Here we also use these two as validation loss to maintain consistency. For training, we use the Focal loss and Huber loss we introduced, because they can stabilize the training convergence.

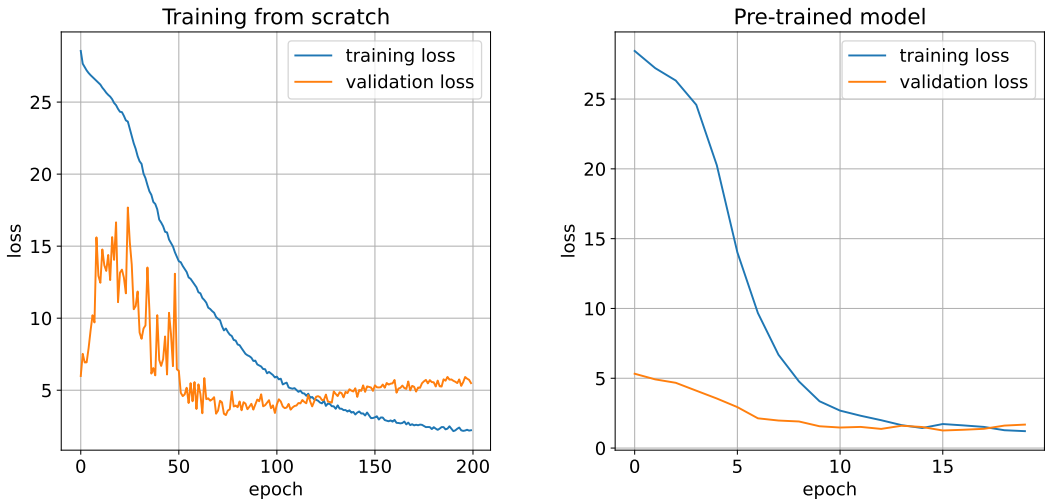
PolyU-M and MPD(h) are used for the training set and validation set, respectively. The validation set only contains half of the classes in the dataset.

doi:10.6342/NTU202203861

Moreover, five images in each class are regarded as registration set, and the rest are probe images. Since our main objective is to improve recognition accuracy across datasets, we choose a relatively difficult setting to evaluate the proposed method: train from the constrained dataset and apply it to the unconstrained dataset.

5.2.1 Pre-trained Model

We have introduced the benefits of pre-trained models in Section 4.2.3. There are three main advantages: faster convergence, improved performance on small training datasets, and more robustness across datasets. In this section, we will compare the loss records during training between the pre-trained model and training from scratch to show their training speed and loss on the target dataset. The model used in the following experimental is ResNeSt-26.



(a) Training from scratch

(b) Pre-trained model

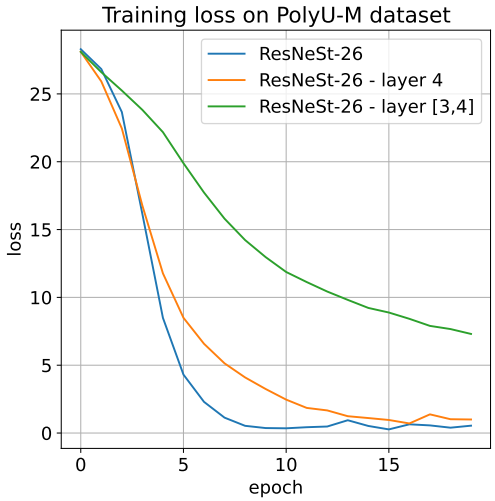
Figure 33: Comparison between pre-trained model and training from scratch

From the figure 33 we can clearly see that there is a huge gap between the two in terms of training speed and validation loss. First of all, it can be found from the number of training iterations that the pre-training model converges extremely fast. Although it does not seem obvious from the illustration, overfitting will still occur on the pre-trained model, so it is necessary to pay attention to the number of training iterations. The validation loss represents the performance on the target dataset. The smaller the loss, the smaller the distance between the registration set and the probe images. The reason why the validation loss is smaller than the training loss is that they are computed differently, which is explained at the beginning of the section. Since here we use different datasets for training and

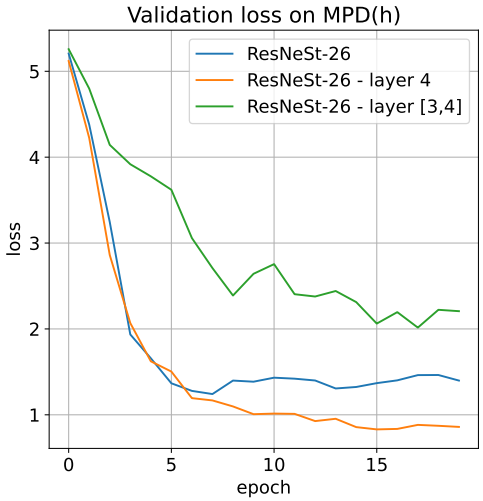
doi:10.6342/NTU202203861

validation, it shows that the pre-trained model is indeed helpful for cross-dataset recognition.

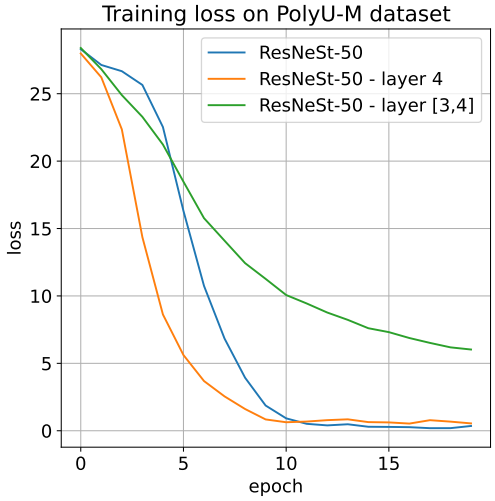
5.2.2 Reduced ResNeSt



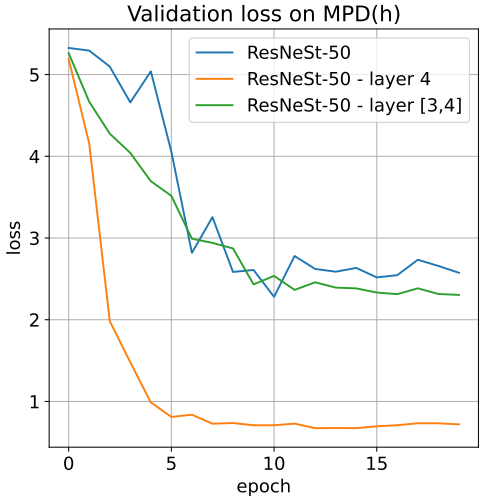
(a) Training loss for ResNeSt-26



(b) Validation loss for ResNeSt-26



(c) Training loss for ResNeSt-50



(d) Validation loss for ResNeSt-50

Figure 34: Comparison between different depths of ResNeSt-26 and ResNeSt-50

This experiment is to answer the question, is the deep structure model really suitable for palmprint recognition? Therefore, we show a comparison of training records for models with a different number of layers. The layer here refers to a row of Table 6 rather than an individual ResNeSt block, which we mentioned in Section 4.2.2.

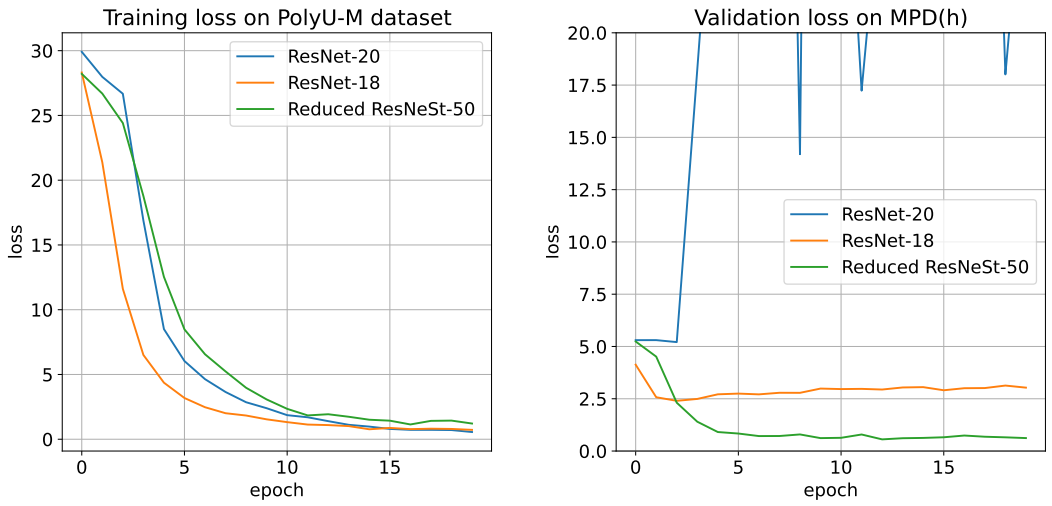
doi:10.6342/NTU202203861

Figure 34(a) and (b) shows the comparison of the training records with different depths of ResNeSt-26, including the original ResNeSt-26 and its removal of the last layer and the last two layers. From Figure 34(a), it can be found that the model with fewer layers converges slower. This is because we use pre-trained models. The more layers a pre-trained model has, the better its modularization ability to adapt to new tasks. Otherwise, in general, the larger the number of parameters, the more data and training time it will take for the model to converge. As for the validation loss demonstrated in Figure 34(b) that the model with the last layer removed is the best.

The results of ResNeSt-50 are even more surprising, which is shown in Figure 34(c) and (d). ResNeSt-50 with the last layer removed far outperforms the other two, and has the fastest convergence speed. Compared with ResNeSt-26, this reduced ResNeSt-50 also has better performance.

In addition, we also observed the same phenomenon on ResNet. Therefore, we infer that in palmprint recognition, it is more suitable to use the ResNet series of models without the last layer.

5.2.3 Model Comparison



(a) Training loss on PolyU-M dataset

(b) Validation loss on MPD(h)

Figure 35: Comparison between different models

Finally, we compare the proposed Reduced ResNeSt-50 with ResNet-20 and ResNet-18 mentioned in Chapter 3. It is obvious from Figure 35(b) that Reduced ResNeSt-50 outperforms the other two. Since the validation loss of ResNet-20 is too high, the range of the loss in the figure is limited. It shows that this model

doi:10.6342/NTU202203861

performs poorly across datasets and cannot extract discriminative features in such situation. In contrast, Reduced ResNeSt-50 has superior discrimination ability.

Then we show the model's ability to generate discriminative features from another intuitive point of view. We extract features from the registration images in the validation set through different models and use UMAP [72] for dimensionality reduction visualization, as shown in Figure 36. A total of 1000 (400 class * 5 samples) samples are included in the registration set. We combine samples with very close distances through UMAP, so the separation between features should be clearly seen on the graph. If the sample distance on the graph is very close, the margin between categories would also be small, and it is more likely to produce a recognition error.

It can be seen from the figure that the features extracted by Reduced ResNeSt-50 are more evenly distributed, while ResNet-20 is tightly clustered. This result is consistent with the validation loss curve that Reduced ResNeSt-50 has better feature extraction ability. Therefore, we use this proposed Reduced ResNeSt-50 as the backbone model to provide sufficient capacity for data augmentation.

5.3 Evaluation of the Data Warping Search

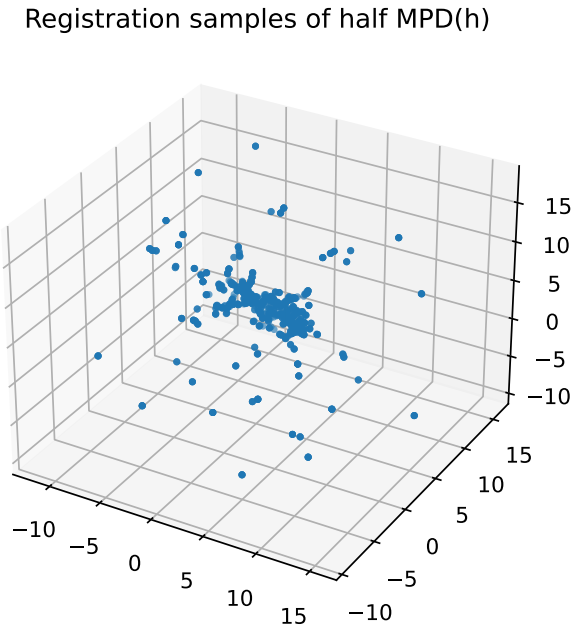
We will examine the results of our proposed hyper-parameter search method in this section, while the evaluation of oversampling will be presented in the next section because it is very deeply related to multi-transform matching. The results of hyper-parameter optimization will be shown in Section 5.3.1, and the comparison with the baseline augmentation method is in Section 5.3.2.

The experimental setup performed in this section is basically the same as in the previous section. Refer to Figure 19 for the specific training process. Similarly, Focal loss and Huber loss are training losses, while softmax loss and center loss are used for validation. PolyU-M dataset and half of MPD(h) are training and validation sets, and does not oversample the training dataset. Reduced ResNeSt-50 is used as a feature extraction model.

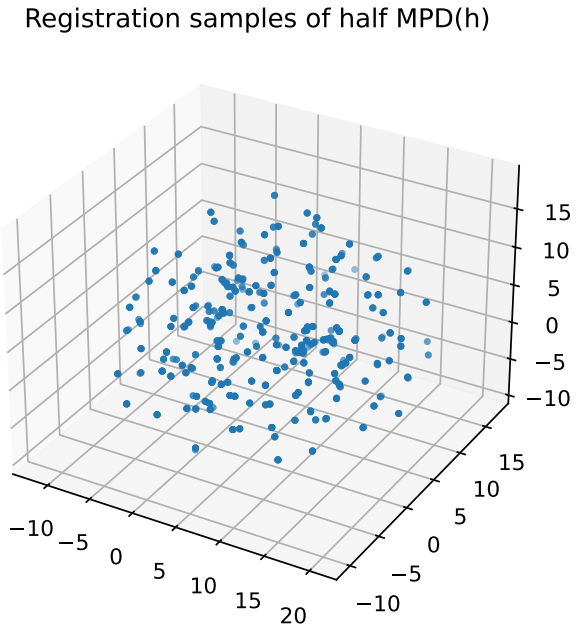
The image transformations to search have been introduced in Section 4.1.2, and we organize them in Table 8. In order to reduce the interaction between image transformations, we first perform brightness, contrast, saturation, and hue adjustments that require precise values for their color space transformations. Then smoothing and noise injection are performed to avoid noise being eliminated by smoothing. Rotation and translation are done last, as they are not affected by the others.

Our goal is to search for the best values of the intensity and probability parameters. The intensity parameter defines the range of intensity sampling for each

doi:10.6342/NTU202203861



(a) ResNet-20



(b) Reduced ResNeSt-50

Figure 36: Visualization of the features extracted by different models. The dots on the illustration are the images of the registration set, and the more scattered, the less potential to produce recognition error.

doi:10.6342/NTU202203861

Table 8: Image transformations and its intensity parameter

Transformation	Intensity range	Description
Brightness	$[\,max(0, 1-a), 1+a\,]$	magnification factor
Contrast	$[\,max(0, 1-a), 1+a\,]$	magnification factor
Saturation	$[\,max(0, 1-a), 1+a\,]$	magnification factor
Hue	$[\,-a, a\,], a \in [\,0, 0.5\,]$	magnification factor
Smoothing	$[\,0.1, a\,]$	Gaussian kernel σ
Noise injection	$[\,0, a\,]$	magnification factor
Rotation	$[\,-a^\circ, +a^\circ\,]$	counterclockwise degree
Translation	$[\,-aW, aW\,]$, $[\,-aH, aH\,]$	ratio of the width and height

Note: Transformations are performed in the order shown in the table to reduce the mutual influence. The intensity range represents the sampling range defined by the intensity parameter a.

transformation performed, and the probability parameter determines the probability of performing the transformation on each training sample. Next, we will search for the hyper-parameter values of each transformation and whether they are necessary by our proposed method, thereby finding a data warping strategy that is more suitable for palmprint recognition across datasets.

5.3.1 TPE Searching Result

In this section, we first introduce the details of the TPE implementation, and then show the history of the optimization process, as well as the selection of transformations based on the results.

The first step is to form a tree-structured domain. All transformation hyper-parameters are sampled independently, so instead of creating groups, we form probability models for each hyper-parameter. We then need to define the search space for each hyper-parameter, which is listed in Table 9. The maximum value of these ranges is set to a fairly large value to ensure that the search range is large enough. If the image is transformed with the maximum value, it will result in indistinguishable images. Taking brightness as an example, it will produce a completely white image. As for the probability parameters, all of them are set as [0,1].

doi:10.6342/NTU202203861

Table 9: Search range of each intensity parameter

Transformation	**Search range**	**type**
Brightness	$[0, 2]$	continuous
Contrast	$[0, 3]$	continuous
Saturation	$[0, 4]$	continuous
Hue	$[0, 0.5]$	continuous
Smoothing	$[1, 10]$	discrete integer
Noise injection	$[0, 20]$	discrete integer
Rotation	$[0, 10]$	discrete integer
Translation	$[0, 0.1]$	discrete

Second, the TPE sampler needs enough start-up samples to compute the computational model. Therefore, 20 trials were randomly sampled before sampling according to TPE, and gamma was set to 0.1, indicating that at least two samples were used to form $l(x)$. In each sampling, 24 candidates will be obtained from each $l(x)$ to calculate $l(x)/g(x)$, and the highest one will be taken as the current sampling.

In addition, we only use pruner in the search phase of TPE to speed up the search process, so the first 20 trials are not terminated early. Because the random sampling at the beginning aims to explore the parameter space as much as possible, once the sampling stage of TPE is entered, only the best part will be explored. The pruner does not come into play until after the 10th epoch of the first training, called warm-up steps. If the first training is complete, the second training will not be stopped early as well.

According to the above implementation details, we have performed 50 trials, and the results are recorded in Figure 37. It can be found that the verification loss obtained after the 20th trial is almost a small value, indicating that TPE can effectively find a good value, so it is more likely to find a better value.

Then we make the transformation selection based on the search results of TPE. The probability parameter of a transformation indicates how often this transformation is performed. Suppose the TPE search result shows that the probability parameter of this transformation should be as small as possible. In that case, this transformation obviously provides very little help and may even have a negative impact.

doi:10.6342/NTU202203861

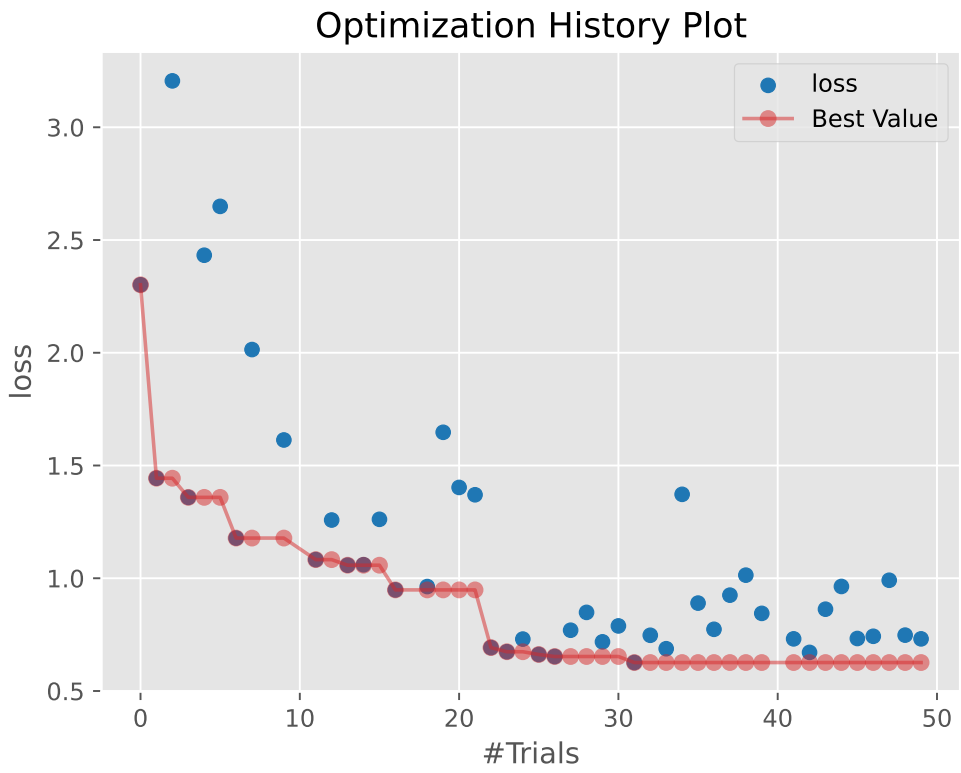


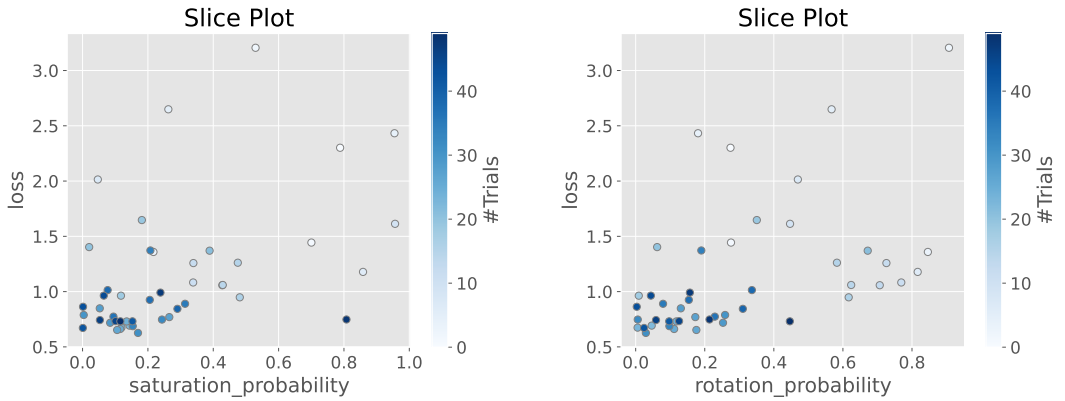
Figure 37: The sampling history and corresponding validation loss. TPE sampling starts at the 20th trial, and before this is random sampling.

Consequently, we take the best top 10% samples to represent the best parameters. If the mean of the probability parameter of any transformation is lower than 0.1, remove the transformation. This operation was done iteratively until no transformations were filtered out. Because removing some transformations, the mutual influence will also change, causing the distribution of the parameters to change as well.

Figure 38 is a plot of the probability parameter and validation loss for the two removed transformations: saturation and rotation. The deeper the sample in the figure represents the later sampling, that is, the parameter that TPE considers to be better. It can be found that they all tend to be smaller and smaller. In addition to these two, smoothing and translation are also similar.

We performed a total of three searches with 50 trials each. In the end, only **brightness**, **contrast**, **hue**, and **noise injection** are left. The optimal value of the noise intensity parameter is only 1, which has little effect on the image. Therefore, we infer that adjusting color is the most effective way to augment data for palmprint recognition across datasets. From Figure 18, it can also be found that the biggest gap between different palmprint ROIs is the skin color, ambient light and shadow.

doi:10.6342/NTU202203861



(a) Saturation probability samples (b) Rotation probability samples

Figure 38: The relationship between the transformation probability parameter and the validation loss, the darker the color represents the later sampled value. Both saturation and rotation would be removed because their execution probability is too low.

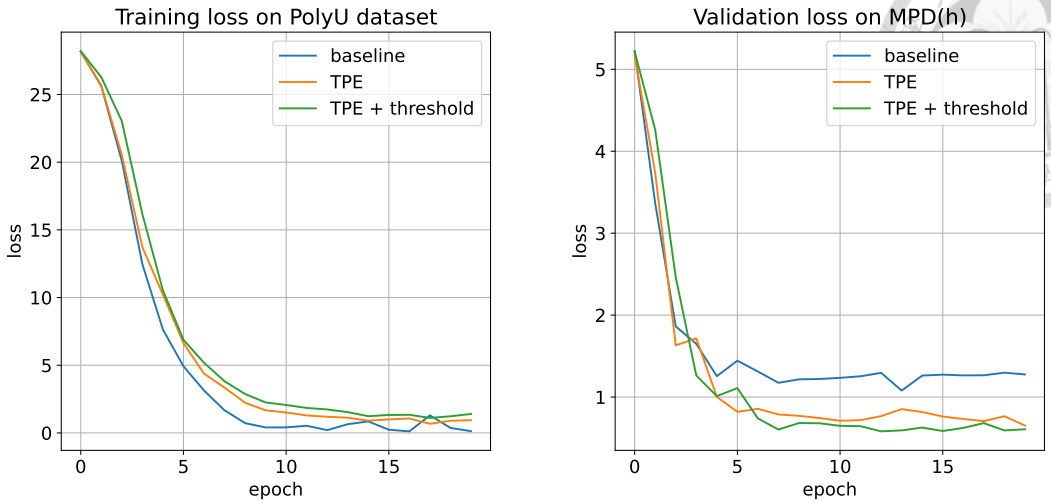
5.3.2 Cross-dataset Evaluation of Different Data Warping

In this subsection, we compare the data warping augmentation searched by TPE with the baseline, including both eight transformations and the remaining four transformations after filtering through the threshold. We first compare the effect of different data warping augmentation on the model's feature extraction ability across datasets by validation loss curves. The accuracy gain of the data warping augmentation approach discovered through hyper-parameter optimization is then demonstrated in comparison to the baseline.

Figure 39(b) shows the validation loss curves of these three data warping augmentations on MPD(h). The validation loss curves searched by TPE are lower than the baseline, and the TPE with threshold filter is even better. Such results suggest that adding more transformations does not lead to better data warping augmentation. More transformations may bring not only more computational cost but also a negative impact. Therefore, the choice of transformation is critical, and how to evaluate the contribution of transformation more precisely deserves further research.

Figure 40 shows the cross-dataset accuracy of these three data warping augmentations, and we first focus on the results trained with the PolyU-M dataset. Since the training set is a constrained dataset, it lacks a lot of light and shadow variation compared to the unconstrained dataset. While baseline augmentation is a relatively moderate data warping, most of the hyper-parameter selections are close to the middle value or common value, lacking sufficient image changes. Hence it performs poorly on MPD.

doi:10.6342/NTU202203861



(a) Training loss on PolyU-M dataset

(b) Validation loss on MPD(h)

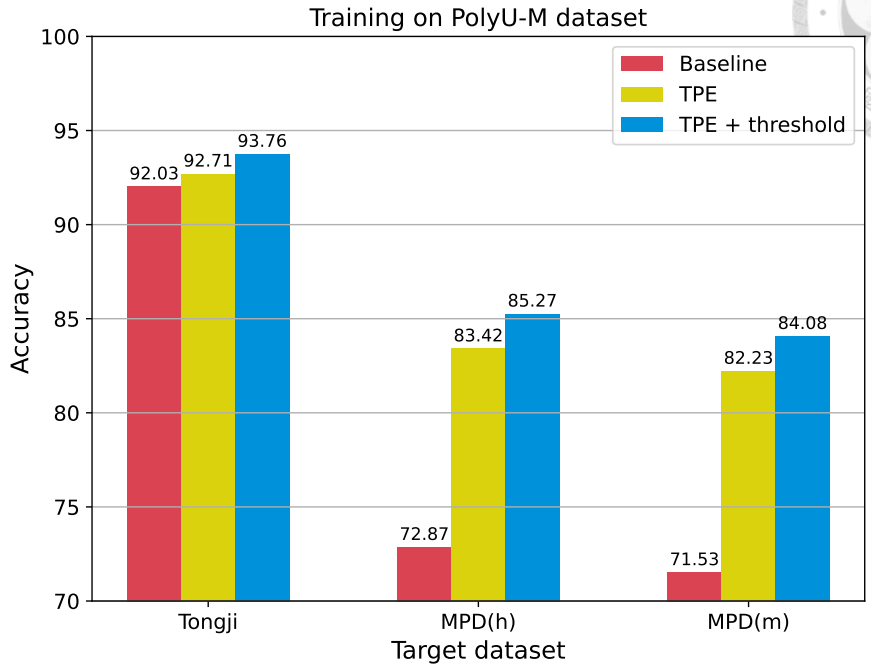
Figure 39: Comparison between different augmentations

By optimization search, we can avoid the influence of stereotypes and find the most suitable combination of transformations and hyper-parameters. The combination obtained through the search greatly improves the accuracy of cross-dataset identification. The accuracy of our proposed method increases by **12.4%** from the baseline on MPD(h), as seen in the Figure.

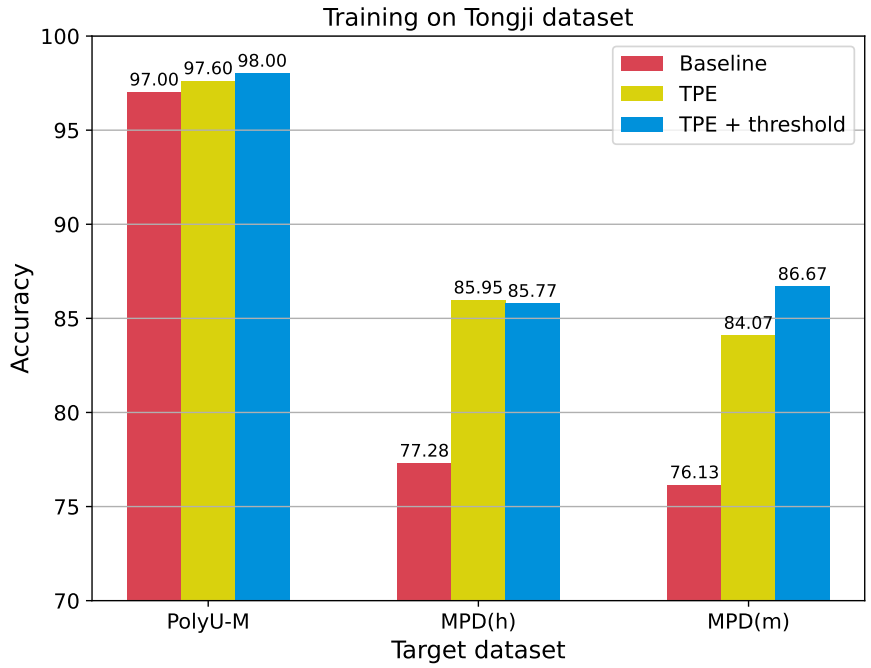
Since the search is performed on PolyU-M and MPD(h), their improvement is particularly large. However, MPD is the most challenging among these datasets, so this data warping approach can also be helpful for training on other datasets. We show the cross-dataset accuracy trained on the Tongji dataset in Figure 40(b). Our method is still superior to the baseline. It can be noted that training on the Tongji dataset and identifying on the MPD(m), which is completely different from our search setup, still achieves an accuracy improvement of **10.54%**.

The limitation of this method lies in the target dataset to be searched, and its diversity determines the upper bound of the search results. Once more complex data emerges, more complex datasets must be searched to cope with those new data. For example, images have lots of shadows or painting on palms. Another limitation is the transformation chosen, which must be able to cover changes in the target dataset. For example, the issue of occlusion is not considered in current transformation set. In addition, the current way of transformation selection still needs to be improved to become more efficient and accurate.

doi:10.6342/NTU202203861



(a) Training on PolyU-M dataset



(b) Training on Tongji dataset

Figure 40: Rank 1 Accuracy(%) comparison of different kinds of data warping in cross-dataset scenarios

doi:10.6342/NTU202203861

5.4 Evaluation of the Multi-Transform Matching and Oversampling

Inspired by [8] introduced in Chapter 3, we have proposed to add more transformations in recognition time. A palmprint ROI image can form up to 8 images in four orientations, up, down, left, and right of the left and right hands. We evaluate the feature matching improvement brought by using these additional images in this section.

Since the model only inputs images in the left orientation during training, the improvement in feature matching of images in other orientations is limited. Therefore, we have proposed an oversampling augmentation method that transforms images to different orientations as training images. It not only improves the recognition ability in other directions, but also makes the improvement brought by different transformations consistent. Therefore, the selection of transformations is reduced to only the number of transformations need to be chosen. We will demonstrate their advantages in Section 5.4.1. Finally, we compare the recognition performance across datasets of the multi-transform matching with the mirror-concatenated matching introduced earlier in Section 5.4.2.

The feature extraction model used in the following evaluations was trained according to Figure 19. The data warp augmentation uses the four transformations found with TPE and threshold filter along with the optimal parameters. The feature extraction model is pre-trained reduced ResNeSt-50, and the loss function is Focal loss and Huber loss. The parameters of the reduced ResNeSt-50 and the classifier are updated with SGDM and Adam, respectively.

5.4.1 Evaluation of the Transformations within Multi-Transform Matching

Table 10 shows the evaluation of the multi-transform matching and oversampling. The values in the table are all rank 1 accuracy, calculated by feature matching to identify the most similar registered images. Feature matching is all performed with multi-transform matching, differing only in the transformations used. Here we select only five transformations for comparison. Note that the most common way of matching using only the input image can also be considered one kind of multi-transform matching, i.e., including only one transformation that does no transformation.

This table is divided into two columns; the difference is whether the training data is expanded with oversampling. We start from "without oversampling". Using only the input image I can be considered as a basis for calculating the amount of improvement. It can be seen from the table that the improvement of vertical

doi:10.6342/NTU202203861

Table 10: Comparison of Rank 1 Accuracy(%) of multi-transform matching using different transformations on MPD(h)

Matching Source	Without Oversampling	With Oversampling
Input I	**85.27**	**90.03**
$I + H$	88.77	92.20
$I + V$	90.25	92.28
$I + R$	88.80	92.32
$I + U$	86.50	92.02
$I + B$	86.07	92.55
All	91.62	94.33

Note: "All" refers to match with total 6 palmprint templates, including input, horizontal flip (H), vertical flip (V), 180° rotation (R), 90° rotation (U), and 270° rotation (B).

flip is the largest, because the output image of the ROI cropping algorithm is left-oriented, and the vertical flip is equivalent to converting into a style similar to the texture of the other hand. For example, the ROI image of a left hand will become the texture of a right hand after being vertically flipped. This texture is included in the training data, so the model has a better ability to identify it, and thus has a better accuracy improvement.

Although the model has not been trained on images in other orientations, it still has a part of the ability to identify images in these orientations, so it can improve the accuracy. Among them, the flip type (180° rotation is equal to horizontal and then vertical flip) is better, and the improvement of rotation is less. However, adding all five of these transformations is even better, which means that adding multiple transformations has its advantages.

Since the best improvement is the vertical transformation which produces images similar to training images, we generate images in all orientations to train the model to enhance the multi-transform matching further. It can be seen from the table that oversampling not only boost up all transformations to the same level but also improves the basic identification accuracy. The accuracy improvement is even as high as **4.76%**.

We mentioned earlier that multi-transform matching could afford up to eight images. However, the accuracy gain decreases as more generated images are added. Figure 41 depicts the accuracy of adding transitions in different orders. Among

doi:10.6342/NTU202203861

them, 0 means that no additional images are added, which is "input I" in the above table, and 5 means that five additional images are used to assist the matching, that is, "All" in the above table. The figure shows that the effect of different orders on the accuracy is not much different, and the number used is critical. Moreover, regardless of the order, the accuracy gain is gradually decreasing.

We, therefore, reasoned that adding too many generated images might be a waste of computational cost, so we chose only six kinds. The remaining two require two transformations to achieve, which consumes more computation than the others, so they are not taken into account.

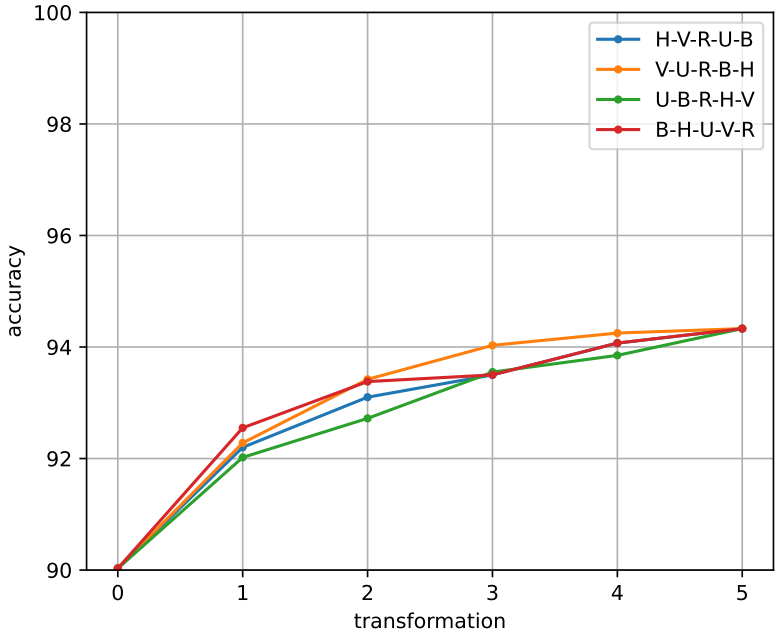
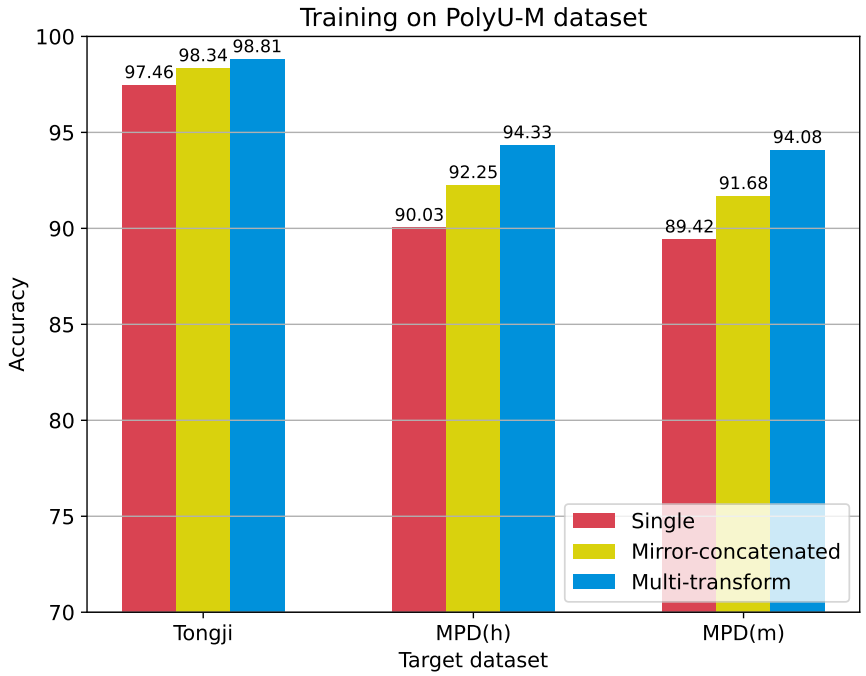


Figure 41: Accuracy of cumulative transformations in different order. The symbols are the same as Table 10. From the left to the right of the X-axis, each grid adds a transformation. The Y-axis corresponds to the accuracy using cumulative transformations. For $x = 0$ and $x = 5$ is the same as the "I" and "All" in Table 10.

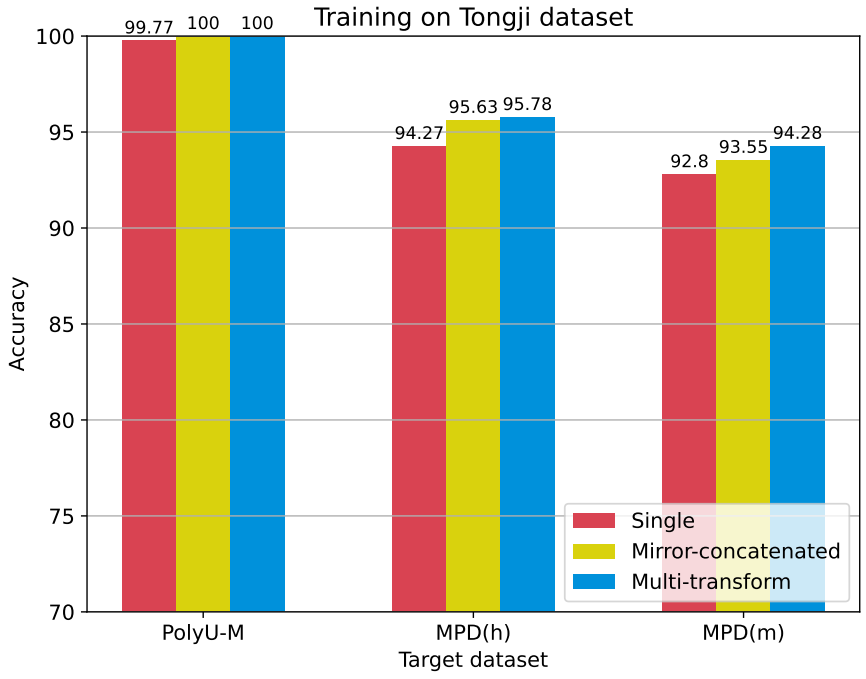
5.4.2 Comparison of Different Matching Methods

In this section, we compare the proposed multi-transform matching with mirror-concatenated matching [8] and single matching. Single matching is to match only through the input image. Because the other two will use multiple images, it is called single matching here. Single matching is the most common method, so we consider it as the baseline. Next, we compare the accuracy and time consuming of these three matching methods in several datasets that are different from the training dataset.

doi:10.6342/NTU202203861



(a) Training on PolyU-M dataset



(b) Training on Tongji dataset

Figure 42: Rank 1 Accuracy(%) comparison of different kinds of matching in cross-dataset scenarios

doi:10.6342/NTU202203861

Figure 42 shows the rank 1 accuracy comparison of different matching methods. Different matching methods use the features extracted by the same model but are trained with different **oversampled** source datasets. Since training on the PolyU-M dataset and identifying on other datasets is the most difficult situation, we use this situation as a representative to examine the performance of multi-transform matching.

From the figure, we can see that the multi-transform matching significantly improved compared with single matching on MPD. The improvement can reach up to **4.66%**, which is better than the mirror-concatenated matching.

We mentioned earlier that single matching is a special case of multi-transform matching, and mirror-concatenated matching under a well-trained model is very similar to multi-transform matching using a single mirror reflection. Therefore, it is intuitive that the results of the multi-transform matching are superior. As can be seen from Figure 41, the more kinds of transformations are used, the higher the accuracy. Moreover, it is not limited to a certain dataset. It can be seen from Figure 42(b) that significant accuracy improvements are also achieved on each target dataset when trained on the Tongji dataset.

Table 11: Comparison of execution time(s) of different matching methods

Matching method	**Matching time(s)**
Single	0.363
Mirror-Concatenated	0.516
Multi-Transform	1.041

Finally, the execution time of the matching is also a matter of concern. Therefore, we compare the execution times of different matching methods in Table 11, and the experiments are conducted with one-to-one validation.

We exclude the feature extraction time of the registration images and only compare the time from inputting the test ROI image to generating the similarity score. Taking the multi-transform matching as an example, the calculated time interval includes the transformation of the input ROI image, the feature extraction of the six images, the calculation of their respective cosine similarity, and averaging the similarities. Furthermore, the matching is only calculated by CPU, without parallel acceleration through the GPU.

Compared with the calculation of similarity, feature extraction takes much more time. Taking the single matching as an example, almost all the time is used for feature extraction, and the calculation of cosine similarity is only a few

doi:10.6342/NTU202203861

milliseconds. Hence, the multi-transform matching is more than six times the amount of operations compared to single, but only three times worse in execution time. Moreover, the multi-transform matching can also be greatly accelerated through parallel computing, so such a matching method can effectively improve the accuracy.

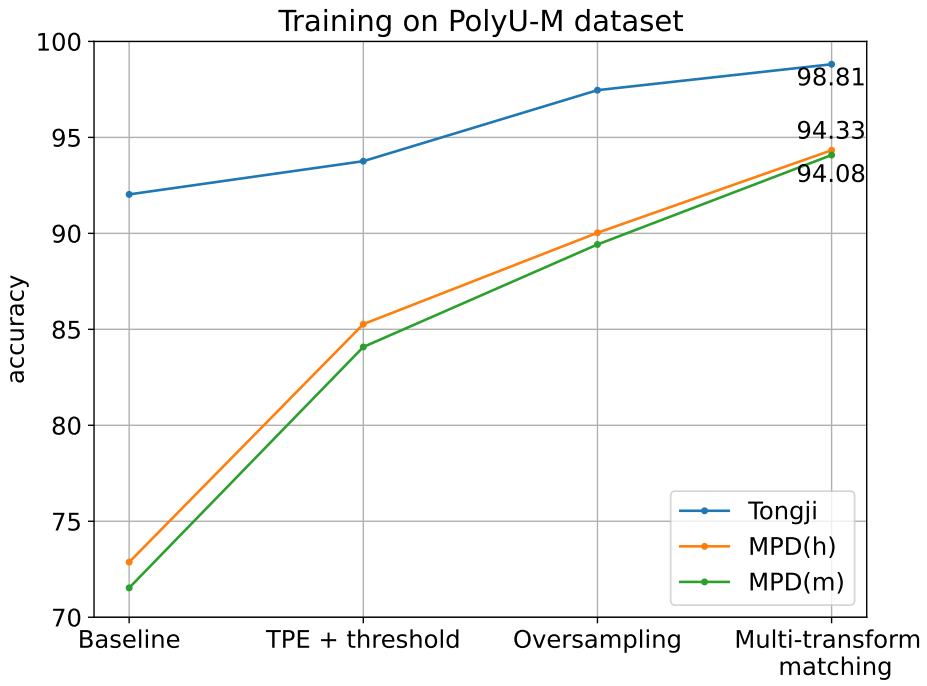
5.5 Summary

In previous evaluations, we individually demonstrated the advantages of our proposed method, including Reduced ResNeSt-50, hyper-parameter search with optuna and TPE, oversampling by rotation, and multi-transform matching. As mentioned earlier, these data augmentation methods can all be used simultaneously to improve performance. We previously focused on comparisons with other methods, so here we will summarize the effects of combining our proposed data augmentation methods.

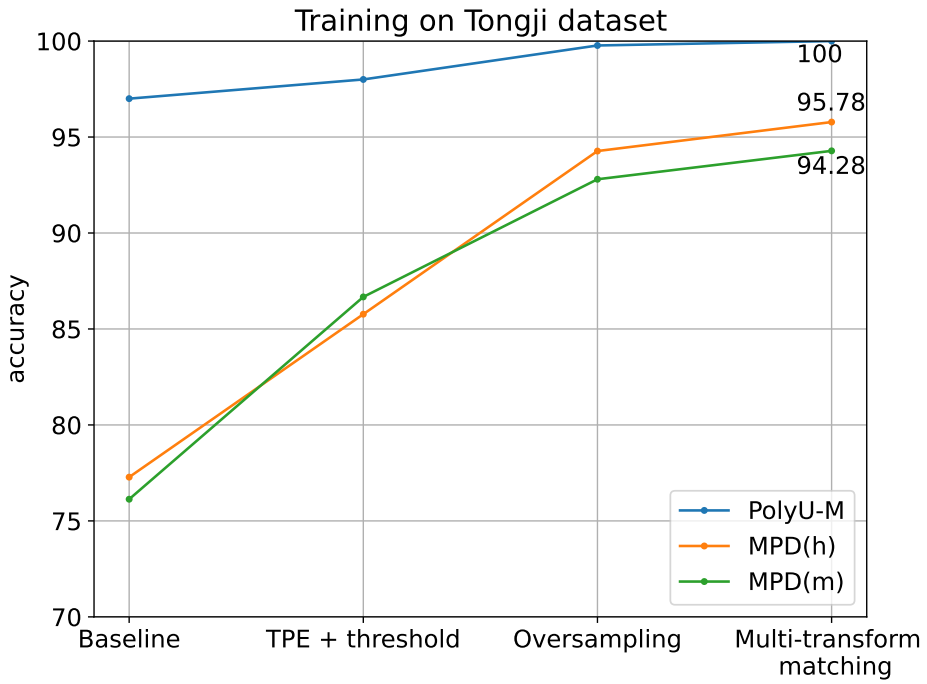
Figure 43 depicts a difficult cross-dataset situation, using the PolyU-M and Tongji datasets as training sets, and testing the effect of the trained model on other datasets. The methods we propose are added sequentially from left to right in the figure. Testing on the unconstrained dataset MPD is critical. When we use PolyU-M datasets as the training set, our method can achieve a total of 21.46% and 22.55% on MPD(h) and MPD(m), respectively. As for using the Tongji dataset as a training set, it is increased by 18.5% and 18.15%, respectively. It indicates that our proposed method has a very significant improvement for datasets with less variation.

The previous figure is intended to show that even if not the PolyU-M dataset, which is used for search, such an approach can also improve substantially when other datasets are used as training sets. In addition, such methods are not limited to improving the accuracy of MPD. Figure 44 shows the improvement in accuracy using MPD(h) and MPD(m) as the training set. The images in MPD are more various, so it has a better effect as a training set. However, our proposed method can still improve the accuracy to very close to 100% on this basis. It can be seen from the figure that using MPD as the training set has the lowest accuracy of 99.66% on all datasets. It indicates that our method is very robust in improving palmprint recognition under various training conditions, and can achieve excellent rank 1 accuracy.

doi:10.6342/NTU202203861



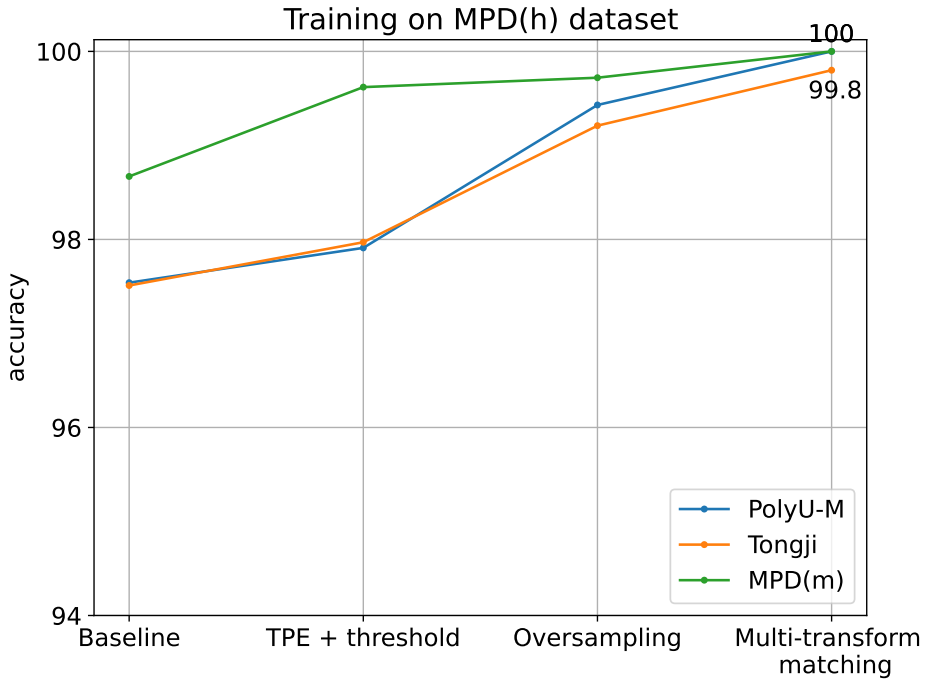
(a) Training on PolyU-M dataset



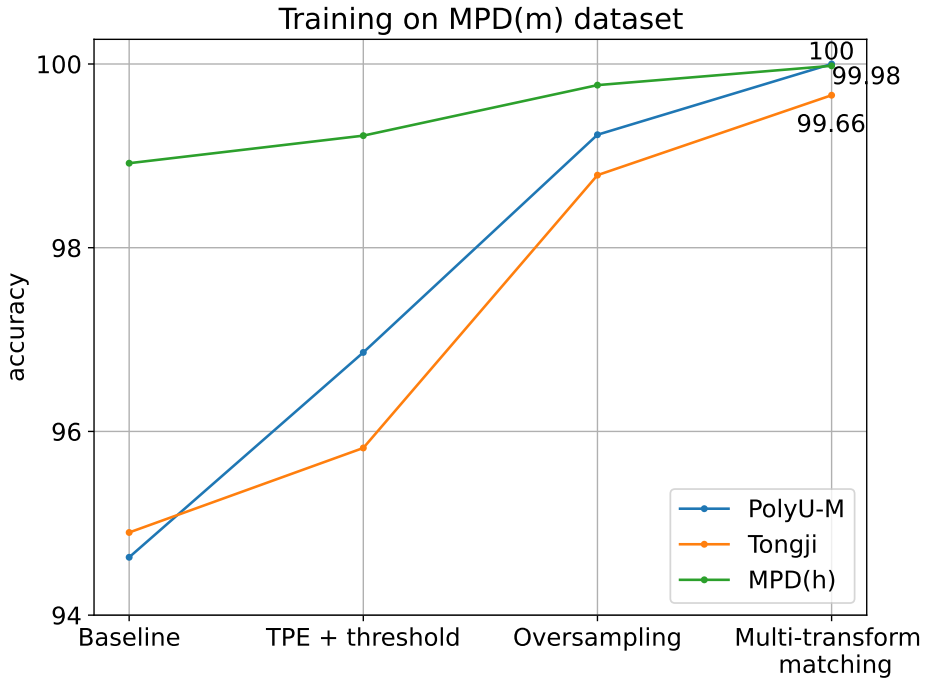
(b) Training on Tongji dataset

Figure 43: Cumulative accuracy(%) improvement of our data augmentation methods in cross-dataset scenarios.

doi:10.6342/NTU202203861



(a) Training on MPD(h)



(b) Training on MPD(m)

Figure 44: Cumulative accuracy(%) improvement for training on MPD.

doi:10.6342/NTU202203861

CHAPTER 6

CONCLUSION AND FUTURE WORK

In palmprint recognition across datasets, past methods mostly focus on a single target dataset. Such an approach is still not robust enough. The variation of input images in practical applications may be very large, not limited to two domains. Therefore, we aim to enable the feature extraction model to learn a more general feature representation.

Given that the ROI cropping algorithm provides a very uniform input image, we reasoned that this problem could be effectively addressed by enhancing random image transformations during training. Based on the observation, we conduct a search for random image transformations in a situation where the image condition gap is the largest. We use PolyU-M as the training dataset and half of MPD(h) as the validation set. The transformations found under such conditions can best reflect the kind of transformation most suitable for palmprint ROI images. The results show that brightness, contrast, hue, and noise injection are the most effective transformations, with up to 12.55% improvement in accuracy on cross-dataset recognition.

For feature matching, we propose a test-time augmentation method suitable for palmprint recognition, called multi-transform matching. Rotate the input image into four orientations or mirror it, and up to eight images can be formed. We use 6 of them for matching and then combine the results as the final similarity. Although such a matching method has a good improvement, the images from different orientations have different degrees of improvement.

Inspired by this concept, we propose an oversampling method that rotates the images into 4 orientations and adds them to the training set as different classes of training images. This data augmentation method can not only strengthen the feature extraction ability of the model, but also make the improvement brought by images from different orientations similar.

This oversampling method can improve the accuracy by up to 5.34% base on the data warping proposed above, and a further improvement of up to 4.66% with multi-transform matching. Combining these three, we can improve the accuracy by up to 22.55% in total compared to the augmentation used in the baseline system on cross-dataset recognition.

There are some issues that need to be considered in such a multi-transformation matching. The first is the cost of increasing the matching. Our comparison of

doi:10.6342/NTU202203861

the time cost in the situation of using only the CPU shows that matching with 6 images takes about three times as long as a single image. However, such an algorithm can be highly parallelized, greatly reducing the computational time. The second is about the performance of the matching. Our experiments show that the improvement in accuracy decreases as more transformations are used in the matching. Therefore, even if we use all of them, the benefits are not high. In the case of sufficient computing power, it is enough to use 6 transformations.

Although our search method also requires additional datasets to assist, only a small number of target dataset for validation is sufficient. Moreover, the process of searching is quite automated and does not require a tedious tuning process. Such a method is not limited to improving the target dataset, but can fairly improve the overall capability of the model. However, there are still some issues that can be investigated deeply.

First, we only study the identification accuracy rate, and have not yet had a deep understanding of the validation error rate like FAR and FRR. Reducing the error rate is the key to making palmprint recognition closer to the application level. Second, the current transformation selection algorithm is still simple and requires multiple iterations, which is inefficient. If there is a more accurate method will be able to speed up the search process. Third, if more complex image conditions are present, such as graffiti on palms or strong shadows, our current selection of image transformations may be inadequate. Therefore, more practical scenarios are still to be studied.

doi:10.6342/NTU202203861

REFERENCES

[1] A. Jain, A. Ross, and S. Prabhakar, "An introduction to biometric recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 4–20, 2004.

[2] D. Zhong, X. Du, and K. Zhong, "Decade progress of palmprint recognition: A brief survey," *Neurocomputing*, vol. 328, pp. 16–28, 2019, chinese Conference on Computer Vision 2017. Online Available at: https://www.sciencedirect.com/science/article/pii/S0925231218309597

[3] D. Zhang, Z. Guo, G. Lu, L. Zhang, and W. Zuo, "An online system of multispectral palmprint verification," *IEEE Transactions on Instrumentation and Measurement*, vol. 59, no. 2, pp. 480–490, 2010.

[4] L. Zhang, L. Li, A. Yang, Y. Shen, and M. Yang, "Towards contactless palmprint recognition: A novel device, a new benchmark, and a collaborative representation based identification approach," *Pattern Recognition*, vol. 69, pp. 199–212, 2017. Online Available at: https://www.sciencedirect.com/science/article/pii/S0031320317301681

[5] Y. Zhou and A. Kumar, "Human identification using palm-vein images," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 4, pp. 1259–1274, 2011.

[6] C.-L. Lin, T. C. Chuang, and K.-C. Fan, "Palmprint verification using hierarchical decomposition," *Pattern Recogn.*, vol. 38, no. 12, p. 2639–2652, dec 2005. Online Available at: https://doi.org/10.1016/j.patcog.2005.04.001

[7] Q. Xiao, J. Lu, W. Jia, and X. Liu, "Extracting palmprint roi from whole hand image using straight line clusters," *IEEE Access*, vol. 7, pp. 74 327–74 339, 2019.

[8] D. Zhong and J. Zhu, "Centralized large margin cosine loss for open-set deep palmprint recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 6, pp. 1559–1568, 2020.

[9] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 499–515.

[10] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," 2018. Online Available at: https://arxiv.org/abs/1801.07698

doi:10.6342/NTU202203861

[11] A. Genovese, V. Piuri, K. N. Plataniotis, and F. Scotti, "Palmnet: Gabor-pca convolutional networks for touchless palmprint recognition," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 12, pp. 3160–3174, 2019.

[12] M. O. Ahmed and S. Prince. Tutorial 8: Bayesian optimization. Online Available at: https://www.borealisai.com/research-blogs/tutorial-8-bayesian-optimization/

[13] T. Huang. Huber loss and focal loss. Online Available at: https://chih-sheng-huang821.medium.com/%E6%A9%9F%E5%99%A8-%E6%B7%B1%E5%BA%A6%E5%AD%B8%E7%BF%92-%E6%90%8D%E5%A4%B1%E5%87%BD%E6%95%B8-loss-function-huber-loss%E5%92%8C-focal-loss-bb757494f85e

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," 2017. Online Available at: https://arxiv.org/abs/1708.02002

[15] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, and A. Smola, "Resnest: Split-attention networks," 2020. Online Available at: https://arxiv.org/abs/2004.08955

[16] "Tongji contactless palmprint dataset." Online Available at: https://cslinzhang.github.io/ContactlessPalm/

[17] "Tongji mobile palmprint dataset." Online Available at: https://cslinzhang.github.io/MobilePalmPrint/

[18] A.-S. Ungureanu, S. Salahuddin, and P. Corcoran, "Toward unconstrained palmprint recognition on consumer devices: A literature review," *IEEE Access*, vol. 8, pp. 86 130–86 148, 2020.

[19] "Casia palmprint database." Online Available at: http://biometrics.idealtest.org/

[20] "Tongji contactless palmvein dataset." Online Available at: https://sse.tongji.edu.cn/linzhang/contactlesspalmvein/

[21] "The hong kong polytechnic university contact-free 3d/2d hand images database version 1.0'." Online Available at: http://www.comp.polyu.edu.hk/~csajaykr/myhome/database_request/3dhand/Hand3D.htm

[22] S. Chen, Z. Guo, J. Feng, and J. Zhou, "An improved contact-based high-resolution palmprint image acquisition system," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 9, pp. 6816–6827, 2020.

[23] "Iit delhi palmprint image database version 1.0." Online Available at: https://www4.comp.polyu.edu.hk/~csajaykr/IITD/Database_Palm.htm

[24] Y. Zhang, L. Zhang, R. Zhang, S. Li, J. Li, and F. Huang, "Towards palmprint verification on smartphones," *ArXiv*, vol. abs/2003.13266, 2020.

doi:10.6342/NTU202203861

[25] H. Shao, D. Zhong, and X. Du, "Efficient deep palmprint recognition via distilled hashing coding," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 714–723.

[26] K. Luo and D. Zhong, "Robust and adaptive region of interest extraction for unconstrained palmprint recognition," *Journal of Electronic Imaging*, vol. 30, no. 3, pp. 1 – 18, 2021. Online Available at: https://doi.org/10.1117/1.JEI.30.3.033005

[27] W. Wu, S. J. Elliott, S. Lin, S. Sun, and Y. Tang, "Review of palm vein recognition," *IET Biometrics*, vol. 9, no. 1, pp. 1–10, 2020.

[28] X. Ma, X. Jing, H. Huang, Y. Cui, and J. Mu, "Palm vein recognition scheme based on an adaptive gabor filter," *IET Biom.*, vol. 6, pp. 325–333, 2017.

[29] O. Nikisins, T. Eglitis, M. Pudzs, and M. Greitans, "Algorithms for a novel touchless bimodal palm biometric system," in *2015 International Conference on Biometrics (ICB)*, 2015, pp. 436–443.

[30] X. Bao and Z. Guo, "Extracting region of interest for palmprint by convolutional neural networks," in *2016 Sixth International Conference on Image Processing Theory, Tools and Applications (IPTA)*, 2016, pp. 1–6.

[31] M. Izadpanahkakhk, S. M. Razavi, M. Taghipour-Gorjikolaie, S. H. Zahiri, and A. Uncini, "Deep region of interest and feature extraction models for palmprint verification using convolutional neural networks transfer learning," *Applied Sciences*, vol. 8, no. 7, 2018. Online Available at: https://www.mdpi.com/2076-3417/8/7/1210

[32] L. Leng, "Palmprint recognition system with double-assistant-point on ios mobile devices," in *BMVC*, 2018.

[33] L. Leng, F. Gao, Q. Chen, and C. Kim, "Palmprint recognition system on mobile devices with double-line-single-point assistance," *Personal Ubiquitous Comput.*, vol. 22, no. 1, p. 93–104, feb 2018. Online Available at: https://doi.org/10.1007/s00779-017-1105-2

[34] M. Afifi, "11k hands: gender recognition and biometric identification using a large dataset of hand images," *Multimedia Tools and Applications*, 2019. Online Available at: https://doi.org/10.1007/s11042-019-7424-8

[35] R. S. Kuzu, E. Maiorana, and P. Campisi, "Vein-based biometric verification using transfer learning," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020, pp. 403–409.

[36] W. Jia, D.-S. Huang, and D. Zhang, "Palmprint verification based on robust line orientation code," *Pattern Recognition*, vol. 41, no. 5, p. 1504–1513, may 2008.

[37] Z. Guo, D. Zhang, L. Zhang, and W. Zuo, "Palmprint verification using binary orientation co-occurrence vector," *Pattern Recognition Letters*, vol. 30, no. 13, pp. 1219–1227, 2009. Online Available at: https://www.sciencedirect.com/science/article/pii/S0167865509001196

doi:10.6342/NTU202203861

[38] Z. Sun, T. Tan, Y. Wang, and S. Li, "Ordinal palmprint represention for personal identification [represention read representation]," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 279–284 vol. 1.

[39] D. Zhang, W.-K. Kong, J. You, and M. Wong, "Online palmprint identification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1041–1050, 2003.

[40] A.-K. Kong and D. Zhang, "Competitive coding scheme for palmprint verification," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 1, 2004, pp. 520–523 Vol.1.

[41] J. Chen and Z. Guo, "Palmprint matching by minutiae and ridge distance," in *Cloud Computing and Security*, X. Sun, A. Liu, H.-C. Chao, and E. Bertino, Eds. Cham: Springer International Publishing, 2016, pp. 371–382.

[42] Y. Wang, Q. Ruan, and X. Pan, "Palmprint recognition method using dual-tree complex wavelet transform and local binary pattern histogram," in *2007 International Symposium on Intelligent Signal Processing and Communication Systems*, 2007, pp. 646–649.

[43] X. Bai, N. Gao, Z. Zhang, and D. Zhang, "3d palmprint identification combining blocked st and pca," *Pattern Recogn. Lett.*, vol. 100, no. C, p. 89–95, dec 2017. Online Available at: https://doi.org/10.1016/j.patrec.2017.10.008

[44] X. Liang, J. Yang, G. Lu, and D. Zhang, "Compnet: Competitive neural network for palmprint recognition using learnable gabor kernels," *IEEE Signal Processing Letters*, vol. 28, pp. 1739–1743, 2021.

[45] Z. Cheng, X. Zhu, and S. Gong, "Face re-identification challenge: Are face recognition models good enough?" *Pattern Recognition*, vol. 107, p. 107422, 2020. Online Available at: https://www.sciencedirect.com/science/article/pii/S0031320320302259

[46] H. Cui, L. Zhu, J. Li, Y. Yang, and L. Nie, "Scalable deep hashing for large-scale social image retrieval," *IEEE Transactions on Image Processing*, vol. 29, pp. 1271–1284, 2020.

[47] C. Kha Vu. (2021) Deep metric learning: A (long) survey. Online Available at: https://hav4ik.github.io/articles/deep-metric-learning-survey

[48] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, pp. 539–546 vol. 1.

[49] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015. Online Available at: https://doi.org/10.1109%2Fcvpr.2015.7298682

doi:10.6342/NTU202203861

[50] 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," 2018. Online Available at: https://arxiv.org/abs/1801.09414

[51] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," 2017. Online Available at: https://arxiv.org/abs/1704.08063

[52] Y. Liu and A. Kumar, "Contactless palmprint identification using deeply learned residual features," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 2, pp. 172–181, 2020.

[53] J. Zhu, D. Zhong, and K. Luo, "Boosting unconstrained palmprint recognition with adversarial metric learning," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 2, no. 4, pp. 388–398, 2020.

[54] H. Shao and D. Zhong, "Learning with partners to improve the multi-source cross-dataset palmprint recognition," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5182–5194, 2021.

[55] X. Du, D. Zhong, and H. Shao, "Cross-domain palmprint recognition via regularized adversarial domain adaptive hashing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 6, pp. 2372–2385, 2021.

[56] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell, "Adversarial discriminative domain adaptation," 2017. Online Available at: https://arxiv.org/abs/1702.05464

[57] H. Shao and D. Zhong, "One-shot cross-dataset palmprint recognition via adversarial domain adaptation," *Neurocomputing*, vol. 432, pp. 288–299, 2021. Online Available at: https://www.sciencedirect.com/science/article/pii/S092523122031972X

[58] H. Shao, D. Zhong, and Y. Li, "Palmgan for cross-domain palmprint recognition," in *2019 IEEE International Conference on Multimedia and Expo (ICME)*, 2019, pp. 1390–1395.

[59] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2242–2251.

[60] H. Shao and D. Zhong, "Towards cross-dataset palmprint recognition via joint pixel and feature alignment," *IEEE Transactions on Image Processing*, vol. 30, pp. 3764–3777, 2021.

[61] AngeloUNIMI. Keras implementation of arcface, cosface, and sphereface. Online Available at: https://github.com/4uiiurz1/keras-arcface

[62] K. Ito, T. Sato, S. Aoyama, S. Sakai, S. Yusa, and T. Aoki, "Palm region extraction for contactless palmprint recognition," in *2015 International Conference on Biometrics (ICB)*, 2015, pp. 334–340.

doi:10.6342/NTU202203861

[63] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[64] R. A. Kirsch, "Computer determination of the constituent structure of biological images," *Computers and Biomedical Research*, vol. 4, no. 3, pp. 315–328, 1971. Online Available at: https://www.sciencedirect.com/science/article/pii/0010480971900346

[65] G. K. Ong Michael, T. Connie, and A. B. Jin Teoh, "Touch-less palm print biometrics: Novel design and implementation," *Image and Vision Computing*, vol. 26, no. 12, pp. 1551–1560, 2008. Online Available at: https://www.sciencedirect.com/science/article/pii/S0262885608001406

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. Online Available at: https://arxiv.org/abs/1512.03385

[67] R. Ranjan, C. D. Castillo, and R. Chellappa, "L2-constrained softmax loss for discriminative face verification," 2017. Online Available at: https://arxiv.org/abs/1703.09507

[68] F. Wang, X. Xiang, J. Cheng, and A. L. Yuille, "NormFace," in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, oct 2017. Online Available at: https://doi.org/10.1145%2F3123266.3123359

[69] M. Lin, Q. Chen, and S. Yan, "Network in network," 2013. Online Available at: https://arxiv.org/abs/1312.4400

[70] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, p. 60, 2019. Online Available at: https://doi.org/10.1186/s40537-019-0197-0

[71] 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," 2020. Online Available at: https://arxiv.org/abs/2004.00288

[72] L. McInnes, J. Healy, and J. Melville, "Umap: Uniform manifold approximation and projection for dimension reduction," 2018. Online Available at: https://arxiv.org/abs/1802.03426

[73] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019. Online Available at: https://arxiv.org/abs/1907.10902

[74] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. Red Hook, NY, USA: Curran Associates Inc., 2011, p. 2546–2554.

[75] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *J. of Global Optimization*, vol. 21, no. 4, p. 345–383, dec 2001. Online Available at: https://doi.org/10.1023/A:1012771025575

doi:10.6342/NTU202203861

[76] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, E. P. Xing and T. Jebara, Eds., vol. 32, no. 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 754–762. Online Available at: https://proceedings.mlr.press/v32/hutter14.html

[77] P. J. Huber, "Robust Estimation of a Location Parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73 – 101, 1964. Online Available at: https://doi.org/10.1214/aoms/1177703732

[78] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2014. Online Available at: https://arxiv.org/abs/1409.4842

[79] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *arXiv preprint arXiv:1611.05431*, 2016.

[80] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," 2019. Online Available at: https://arxiv.org/abs/1903.06586

[81] A. Chowdhury, M. Jiang, S. Chaudhuri, and C. Jermaine, "Few-shot image classification: Just use a library of pre-trained feature extractors and a simple classifier," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9425–9434.

[82] K. He, R. Girshick, and P. Dollár, "Rethinking imagenet pre-training," 2018. Online Available at: https://arxiv.org/abs/1811.08883

[83] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," 2019. Online Available at: https://arxiv.org/abs/1901.09960

[84] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. Red Hook, NY, USA: Curran Associates Inc., 2012, p. 1097–1105.

doi:10.6342/NTU202203861