symmetry

MDPI

Article

A Sparsified Densely Connected Network with Separable Convolution for Finger-Vein Recognition

Qiong Yao, Xiang Xu * and Wensheng Li

Artificial Intelligence and Computer Vision Laboratory, University of Electronic Science and Technology of China Zhongshan Institute, Zhongshan 528400, China
* Correspondence: xuxiang@zsc.edu.cn; Tel.: +86-0760-882-27202

Abstract: At present, ResNet and DenseNet have achieved significant performance gains in the field of finger-vein biometric recognition, which is partially attributed to the dominant design of cross-layer skip connection. In this manner, features from multiple layers can be effectively aggregated to provide sufficient discriminant representation. Nevertheless, an over-dense connection pattern may induce channel expansion of feature maps and excessive memory consumption. To address these issues, we proposed a low memory overhead and fairly lightweight network architecture for finger-vein recognition. The core components of the proposed network are a sequence of sparsified densely connected blocks with symmetric structure. In each block, a novel connection cropping strategy is adopted to balance the channel ratio of input/output feature maps. Beyond this, to facilitate smaller model volume and faster convergence, we substitute the standard convolutional kernels with separable convolutional kernels and introduce a robust loss metric that is defined on the geodesic distance of angular space. Our proposed sparsified densely connected network with separable convolution (hereinafter dubbed 'SC-SDCN') has been tested on two benchmark finger-vein datasets, including the Multimedia Lab of Chonbuk National University (MMCBNU)and Finger Vein of Universiti Sains Malaysia (FV-USM), and the advantages of our SC-SDCN can be evident from the experimental results. Specifically, an equal error rate (EER) of 0.01% and an accuracy of 99.98% are obtained on the MMCBNU dataset, and an EER of 0.45% and an accuracy of 99.74% are obtained on the FV-USM dataset.

Keywords: finger-vein recognition; densely connected; sparsified; separable convolution

check for updates

Citation: Yao, Q.; Xu, X.; Li, W. A Sparsified Densely Connected Network with Separable Convolution for Finger-Vein Recognition. *Symmetry* **2022**, *14*, 2686. https://doi.org/10.3390/sym14122686

Academic Editor: Nicola Mastronardi

Received: 17 November 2022
Accepted: 14 December 2022
Published: 19 December 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the growing demand for secure identity authentication, a variety of biometric traits, including facial, speech, iris, fingerprint, and vein, to name a few, have been emerging in recent decades [1] and gradually substituting traditional means of identity authentication such as Tokens, Smart Cards, PINs, etc. Among them, the finger-vein (FV) trait is a promising biometric recognition technology, which utilizes vein characteristics in subcutaneous tissues to identify individuals. Compared with some other biometric traits, the FV trait is more secure and stable and only relies on live detection [2]. Intuitively, the application scenarios of finger-vein recognition (FVR) are generally open-set, which means only a small number of categories are known in the training phase, while many unknown category samples appeared in the testing phase. In this light, how to obtain more robust and discriminative feature representation is particularly crucial for a FVR system [3,4]. Moreover, due to the variances of illumination, temperature, and finger position during image acquisition, inter-class similarity and intra-class variability are prevalent, leaving room for the improvement of FVR technologies.

Traditional handcrafted FV feature extraction approaches can be roughly categorized as either 'vein-level' or 'image-level' [5]. Vein-level approaches are devoted to characterizing topological structures of the pure vein network while ignoring the impact of

surrounding background information. Common vein structural morphologies covered minutia points [6,7], repeat lines [8] and wide lines [9], curvatures [10–12], as well as anatomies [13,14] and pulsation patterns [15], etc. Image-level approaches mainly focus on feature extraction from the whole image while not distinguishing vein and non-vein regions. It is based on the observation that optical characteristics such as absorption and scattering in non-vein regions are also conducive to recognition [16]. These approaches are devoted to extracting FV features from local or global perspectives [17–21]. Table 1 illustrates a rough summary of the handcrafted approaches employed for FVR tasks, along with corresponding test datasets and recognition results.

In a nutshell, handcrafted FV features have shown diversified research lines, but they have some common limitations. On the one hand, the design of handcrafted features mainly relies on expert knowledge, usually task-specific and weak generalization, and on the other hand, most of the handcrafted features belong to shallow feature representation, which is generally sensitive to noise, illumination, finger position, etc.

Table 1. A brief summary of handcrafted finger vein feature extraction approaches.

	Feature Type	Method (Ref)	Dataset	Performance Criteria	
				EER (%)	ACC (%)
Vein-Level	Points	SVD-based Minutiae Matching [6]	HKPU [22] [a] SDUMLA [23] [b]	5.01 2.46	95.71 98.63
		Zone-based Minutia Matching [7]	HKPU [22] SDUMLA [23]	0.36 2.61	99.67 96.27
	Lines	Repeated Lines [8]	own [g]	0.145	–
		Wide Line Detector (WLD) [9]	own	0.87	–
	Curvatures	Max Curvature [10]	own	0.0009	–
		Mean Curvature [11]	own	0.25	–
		Enhanced Maximum Curvature [12]	SDUMLA [23] PKU [24] [c]	0.14 0.33	– –
		Radon-like Features (RLF) [5]	HKPU [22] MMCBNU [25] [d] FV-USM [26] [e]	5.47 3.33 0.93	– – –
	Anatomies	Anatomy Structure Analysis based Vein Extraction (ASAVE) [14]	HKPU [22] SDUMLA [23]	0.38 1.39	– –
		Pulsation of Veins [15]	Video data [15] [f]	0.8	–
Image-Level	Local Pattern	Local Directional Code (LDC) [17]	MMCBNU [25]	1.03	–
		Discriminative Binary Descriptor [18]	HKPU [22] SDUMLA [23]	0.69 1.89	– –
	Filtering	Guided Gabor Filter [27]	SDUMLA [23]	2.24	–
		Gabor Filters and Morphological [22]	HKPU [22]	0.65	–
	Statistical	Principal Component Analysis (PCA) [20]	own	–	99.0
		$(2D)^2$PCA [28]	own	–	99.17
		Histogram of Competitive Orientations and Magnitudes (HCOM) [29]	MMCBNU [25]	0.36	–
		Histogram of Salient Edge Orientation Map (HSEOM) [30]	MMCBNU [25]	0.9	–
		Partial Least Squares Discriminant Analysis (PLS-DA) [21]	HKPU [22] SDUMLA [23] MMCBNU [25] FV-USM [26]	1.17 2.15 0.63 0.15	– 97.52 – 99.86

[a] Finger Image Database from Hong Kong Polytechnic University (HKPU). http://www4.comp.polyu.edu.hk/~csajaykr/fvdatabase.htm, (accessed on 1 January 2021). [b] Homologous Multi-modal Traits Database (SDUMLA). http://mla.sdu.edu.cn/sdumla-hmt.html, (accessed on 1 January 2021). [c] Finger Vein Database from Peking University (PKU). http://rate.pku.edu.cn/, (accessed on 1 March 2019). [d] Finger Vein Database from Multimedia Lab of Chonbuk National University (MMCBNU). http://multilab.jbnu.ac.kr/MMCBNU_6000, (accessed on 1 January 2021). [e] Finger Vein Database from Universiti Sains Malaysia (FV-USM). http://drfendi.com/fv_usm_database/, (accessed on 1 January 2021). [f] Finger Vein Video Database (FV_IIITMK_VideoData). https://duk.ac.in/crictr/arya/sampledatapage.html, (accessed on 1 June 2022). [g] Some Self Built Finger Vein Databases.

With the rapid penetration of deep learning into biometrics, handcrafted feature extraction modalities are swiftly altering to more generalized learning-based modalities. Among these, deep convolutional neural networks (DCNNs) equipped with various architectures have been migrated for FV biometrics and have delivered commendable success. In [31], a VGGNet-based DCNN (namely 'DeepVein') was constructed for FV verification. In [32], AlexNet was directly transferred for FV identification. Moreover, Capsule network [33], Convolutional Auto-Encoder (CAE) [34], Fully Convolutional network (FCN) [35], Generative Adversarial network (GAN) [36], Long Short-term Memory network (LSTM) [37], and Joint Attention network [38], etc., were applied to FV recognition. The classical deep learning models above generally adopt a data-driven learning process and rely on sufficient training samples to a great extent; thereby, some pre-training [39] and data augmentation strategies [40] were introduced to make up for the shortage of samples.

An alternative research line is advocated to design relatively lightweight and medium-depth semantic representation models [41,42]. It is attributed to the fact that FV images mainly contain some low-level and middle-level characteristics, such as textures and shape structures. In [43], a lightweight deep-learning model with two channels was exploited. In [44], a fusion loss function was introduced into a lightweight network to pursue highly discriminative features. In [45], a two-pathway lightweight network was built to extract multi-scale features. Corresponding to those classical DCNNs, these lightweight deep networks greatly diminish training costs while ensuring accuracy and are thus more suitable for real application scenarios.

In order to enhance feature reusability, a skip connection strategy is attempted in many network architectures, which allows the network to pass on features unimpededly from earlier layers to later layers. Among these, ResNet [46] introduced shortcuts to sum up a layer with multiple preceding layers, while DenseNet [47] concatenated each layer to every other layer in a feed-forward fashion. These two advanced network architectures alleviated gradient-vanishing, encouraged feature reuse, and gained rising attention in the field of FVR. In [48], multimodal biometrics, including finger vein and finger shape, were extracted from ResNet, respectively, and then fused for individual identity authentication. In [49], a Siamese framework embedded with two ResNet-50 branch sub-networks was used for FV verification. In [50], a pre-trained Xception network equipped with residual skip connection was built for FV classification. In [51], two FV images were synthesized and input into a DenseNet. In [52], vein shape features and texture features were successively input into a DenseNet and then fused for FV recognition. In [53], a densely-connected convolutional auto-encoder was built to learn discriminative hand vein features. Table 2 roughly summarizes deep feature learning approaches employed for FVR, along with corresponding backbones, test datasets and recognition results.

As noted earlier, resorting to the skip connection strategy, ResNet and DenseNet can effectively aggregate features from multiple preceding layers and promote the gradient flow throughout the network. However, skip connections that are too dense easily lead to channel expansion of feature maps and excessive memory overhead. It is imperative to seek more economical connection patterns to meet the demands of lightweight models. With this consideration, we proposed a low memory overhead and fairly lightweight network architecture. The core components of the proposed network are a sequence of sparsified densely connected blocks with symmetric structures. In each block, a novel connection cropping strategy is adopted to balance the channel ratio of input/output feature maps as much as possible, while not sacrificing the retention of effective information. In this manner, the proposed sparsified densely connected network with separable convolution (hereinafter dubbed 'SC-SDCN') model not only obtained surprising recognition performance but also exhibited smaller model size, faster convergence rate, as well as less training sample requirements. To sum up, the main innovative contributions of our work are three-fold:

- First, we proposed a low-memory overhead and sufficiently lightweight network model based on a group of sparsified dense connection (SDC) blocks. In each SDC

block, the number of output feature channels is compressed as much as possible under the premise of retaining efficient information of the aggregated features.

- Second, to facilitate a smaller model size and faster convergence rate, we substituted the standard convolutional kernels with separable convolutional kernels, namely, a sequence assembly of 1×1 point-wise convolution and depth-wise convolution was adopted.
- Finally, considering intra-class variation and inter-class similarity, we introduced a more robust margin penalty to strengthen the discriminative power of a softmax loss framework, which was defined on the geodesic distance of angular space.

The experiments are carried out on two benchmark finger-vein datasets, including MMCBNU [25] and FV-USM [26]. The experimental results reveal the superior recognition performance of the proposed SC-SDCN, especially when the training samples are limited. Beyond that, our SC-SDCN model exhibits small model size and fast convergence rate, thus more suitable for real-time FVR scenarios.

Table 2. A brief summary of deep-learning-based finger vein recognition approaches.

	Category	Method (Ref)	Dataset	Performance Criteria	
				EER (%)	ACC (%)
Without Skip Connection	Classical	LeNet-5 CNN [54]	UTM [55] [a]	–	99.0
		DeepVein (VGGNet-16) [31]	FVRC2016 [24] [b]	2.14	–
		VGGNet-16 [56]	SDUMLA [23]	0.804	–
		FV-Net (VGGFace-Net) [57]	SDUMLA [23]	1.20	–
			MMCBNU [25]	0.30	–
			FV-USM [26]	0.76	–
		AlexNet [32]	Unknown	–	91.67
		CNN Competitive Order (CNN-CO) [58]	SDUMLA [23]	2.37	–
			MMCBNU [25]	0.74	–
		Capsule Network [33]	HKPU [22]	–	88.0
			SDUMLA [23]	–	100.0
			MMCBNU [25]	–	100.0
			UTFVP [59] [c]	–	94.0
		Convolutional Auto-Encoder (CAE) [34]	SDUMLA [23]	0.21	99.78
			FV-USM [26]	0.12	99.95
		Fully Convolutional Network (FCN) [60]	SDUMLA [23]	3.88	–
			UTFVP [59]	1.80	–
		Fully Convolutional Network+ Conditional Random Field (FCN+CRF) [35]	HKPU [22]	2.37	–
			SDUMLA [23]	5.83	–
			MMCBNU [25]	0.36	–
		Generative Adversarial Networks (FV-GAN) [61]	SDUMLA [23]	0.94	–
		Conditional GAN [36]	HKPU [22]	1.81	–
			SDUMLA [23]	3.934	–
		Triplet-classifier GAN [62]	HKPU [22]	0.40	–
			SDUMLA [23]	1.33	–
			FV-USM [26]	0.14	–
		Long Short-Term Memory Network (CNN-LSTM) [37]	own	–	99.10
	Self Build	Das's Model [63]	HKPU [22]	–	95.32
			SDUMLA [23]	–	97.48
			FV-USM [26]	–	97.53
			UTFVP [59]	–	95.56
	Lightweight	Two-Stream CNN [43]	SDUMLA [23]	0.47	–
			MMCBNU [25]	0.10	–
		Light CNN (LCNN) [64]	HKPU [22]	0.13	–
		Lightweight CNN Combining Center Loss and Dynamic Regularization [44]	MMCBNU [25]	0.503	99.05
			FV-USM [26]	1.07	97.95
		Lightweight Deep CNN [45]	SDUMAL [23]	1.13	99.30
			PKU [24]	0.67	99.60

Table 2. *Cont.*

	Category	Method [Ref]	Dataset	Performance Criteria	
				EER (%)	ACC (%)
With Skip Connection	ResNet	Multimodal with ResNet-101 [48]	HKPU [22] SDUMLA [23]	0.83 2.43	– –
		ResNet+Siamese CNN [49]	SDUMLA [23] MMCBNU [25] FV-USM [26]	0.66 0.12 0.30	– – –
		Efficient Channel Attention Residual Network (ECA-Resnet) [65]	HKPU [22] SDUMLA [23] FV-USM [26]	1.82 2.14 0.89	99.01 98.91 99.42
		ResNeXt-101 [66]	FV-USM [26]	–	98.10
		Xception Model with Depth-wise Separable CNN [50]	SDUMLA [23] THU-FVFDT2 [67] [d]	– –	98.50 90.0
	DenseNet	DenseNet-161+Composite Image [51]	HKPU [22] SDUMLA [23]	0.33 2.35	– –
		DenseNet-161+Score-level Fusion [52]	HKPU [22] SDUMLA [23]	0.05 1.65	– –
		Densely-Connected Convolutional Autoencoder [53]	HKPU [22] SDUMLA [23]	0.228 0.025	99.67 99.98

[a] Finger Vein Database from VeCAD Laboratory, University Technology Malaysia (UTM). [b] 2nd Competition on Finger Vein Recognition Competition (FVRC2016). http://rate.pku.edu.cn, (accessed on 1 January 2019). [c] University of Twente Finger Vascular Pattern (UTFVP). http://www.sas.el.utwente.nl/home/datasets, (accessed on 1 January 2019). [d] Tsinghua University Finger Vein and Finger Dorsal Texture Database (THU-FVFDT). https://www.sigs.tsinghua.edu.cn/labs/vipl/thu-fvfdt.html, (accessed on 1 January 2021).

The remainder of this paper is organized as follows. Section 2 provides a brief overview of related works. Section 3 details the architecture of SC-SDCN, as well as the design of core components. Section 4 discusses the experimental results, along with the comparison with some commonly used FVR approaches. Section 5 concludes the paper with some remarks and hints at plausible future research lines.

2. Related Works

In this section, the pioneering work related to our proposed SC-SDCN framework was briefly discussed. First, an overview of the skip connection and sparsified dense connection strategies was presented. Then, the basic idea of depth-wise separable convolution was briefly reviewed.

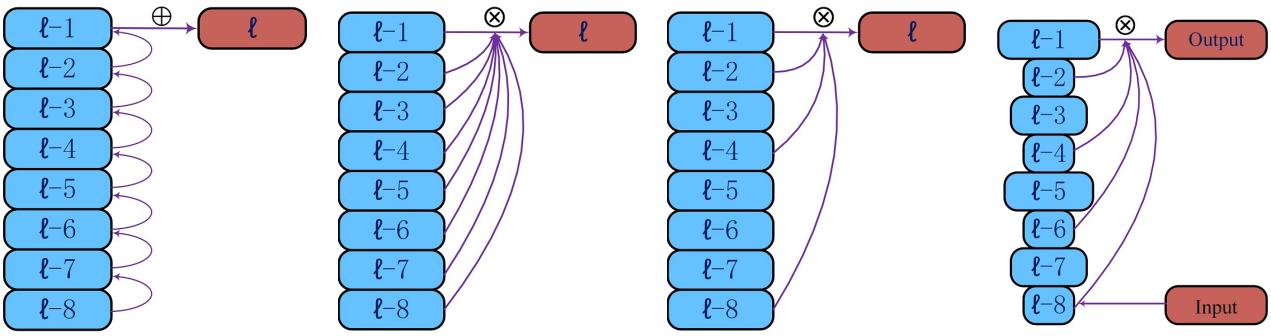
2.1. Skip Connection and Sparsification

During the learning of a DCNN, skip connection provides an internal features aggregation capability, allowing early assembly features to be easily accessible. To the best of our knowledge, this architecture was first adopted in the Highway network [68], in which a parameterized skip connection (called 'gating units') was designed to control the information flows unobstructed from early layers to later layers. Then, ResNet [46] adopted more pure identity mapping for skip connection, as shown in Figure 1a. Formally, let $f_\ell(\cdot)$ represent a typical convolutional transformation from layer $\ell - 1$ to ℓ, the output of the ℓ^{th} layer with the skip connection is calculated by using Equation (1).

$$H_\ell = \text{ReLU}(f_\ell(H_{\ell-1}) + \text{identity}(H_{\ell-1})), \quad (1)$$

where identity$(\cdot)$ denotes the identity mapping ReLU represents a nonlinear activation function. The propagation rule of Equation (1) allows the gradients and features to be transferred back and forth between layers through an identity transformation. Further on, a Stochastic depth network [69] was proposed to transform the constant training depth of ResNet into an expected lower-down training depth by randomly skipping a subset of layers entirely. Compared to ResNet, the back-propagation performance and behavior

of gradients further enhanced by stochastic depth, especially in the earlier layers, can be interpreted as training an ensemble of ResNets with varying depth implicitly.



(**a**) ResNet (**b**) DenseNet (**c**) Log-DenseNet (**d**) HarDNet

Figure 1. Illustration of different skip connection architectures.

When the density of skip connections reached the extreme, DenseNet [47] appeared. As shown in Figure 1b, each layer of DenseNet is concatenated with all preceding layers. Consequently, the ℓ^{th} layer receives the feature maps of all preceding layers as input, and the corresponding output of the ℓ^{th} layer can be calculated by Equation (2).

$$H_\ell = \text{ReLU}(f_\ell(H_{\ell-1}) + \text{identity}(concat(\{H_{\ell-k\,:\,k=1,\ldots,\ell}\}))). \tag{2}$$

DenseNet naturally integrates attributes of identity mapping, diversified depth, as well as implicit supervision, and it also allows feature reuse throughout the network. However, since not all connections can deliver informative flows, such excessive connectivity patterns may bring about potentially large amounts of redundancy and hinder scalability on deeper networks. Considering this, some sparsified DenseNet variants are presented, which only aggregate a sparse set of previous outputs at any given depth. In [70], a Log-DenseNet architecture was designed to aggregate the preceding layers with only exponential offsets, as shown in Figure 1c and defined in Equation (3). A similar sparsified aggregation structure was also applied in SparseNet [71].

$$H_\ell = \text{ReLU}\Big(f_\ell(H_{\ell-1}) + \text{identity}\Big(concat\Big(\{H_{\ell-2^k\,:\,k=0,\ldots,\lfloor\log(\ell)\rfloor}\}\Big)\Big)\Big). \tag{3}$$

Assuming that the network contains L layers, at this time, both Log-DenseNet and SparseNet have a total of $\mathcal{O}(L\log_2 L)$ skip connections, which is significantly lower than the $\mathcal{O}(L^2)$ of DenseNet, and its adverse impact is that the maximum back-propagation distance between layers increases from 1 to $(1+\log_2 L)$. As a compromise, they have sought to use more convolutional kernels to compensate for the attenuation of accuracy. The question of how to address the issues of effective feature aggregation and retention after connection cropping is still in its infancy.

Furthermore, a novel harmonized sparse, dense connection network, called HarDNet, was proposed in [72]. As shown in Figure 1d, the proposed HarDNet model not only sparsifies the number of skip connections to $\mathcal{O}(L\log_2 L)$ but also keeps the output feature maps of each layer in a harmonic manner. Specifically, in an even-indexed layer, the channel size is controlled by a given growth rate k, while in an odd-indexed layer, a constant channel size is maintained, and the final output of one block will only concatenate feature maps of odd-indexed layers. Compared with Log-DenseNet and SparseNet, HarDNet can achieve a smaller output channel size of feature maps, as well as lower memory consumption.

2.2. Separable Convolution

For an input feature map with B number of channels, the corresponding standard convolutional kernels also need B channels apart from spatial width and height, making the number of parameters to be learned inevitably huge. In this light, depth-separable convolution alleviates

computational costs, which factorizes the standard convolutional kernel into two separate convolutional kernels, namely depth-wise convolution and point-wise convolution. As shown in Figure 2, in the part of depth-wise convolution, each input feature map is split into B number of channel-level images. Then, each channel map is convolved with corresponding depth-wise convolutional kernels of the shape $k_1 \times k_2 \times 1$, resulting in a single-channel feature map. Finally, all channel maps are superimposed together to form the output feature maps. While in the part of point-wise convolution, a series of convolutional kernels with a size of $1 \times 1 \times B$ are iteratively performed so as to form the final convolved feature maps.

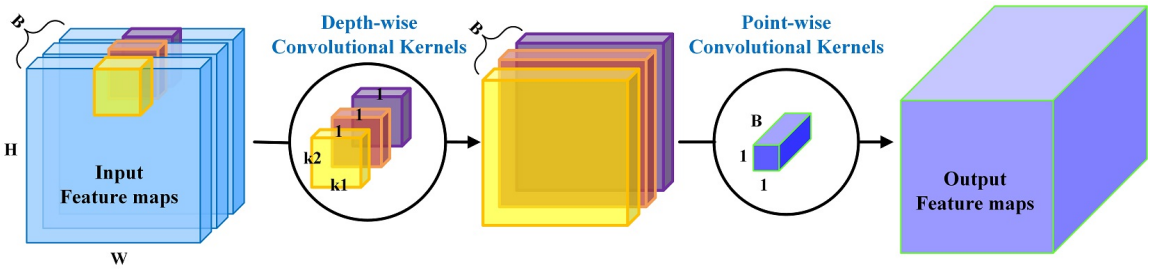


Figure 2. Schematic of the depth-separable convolution.

Depth-wise separable convolutions have been widely used in DCNNs. Nonetheless, the accuracy gains are trivial, and this architecture breaks the interaction between the channel dimension and the spatial dimension of the kernel and drastically reduces model size and accelerates convergence. In GoogLeNet's Inception modules, depth-wise separable convolutions were used to substitute standard convolutions at higher layers [73]. Later, MobileNet [74] further gave play to the superiority of depth-wise separable convolutions in mobile vision applications. In [75], an efficient implementation of depth-wise separable convolutions in the TensorFlow framework was presented.

3. Methodology

As indicated previously, DCNNs combined with skip connections have yielded significant performance gains in the FVR tasks. However, they still have room for improvement, especially in the aspects of efficient feature representation and reuse, as well as the lightweight aspect of network architecture. Motivated by these issues, we developed a low memory overhead and fairly lightweight convolutional network for FVR. In the following, the overall framework of our proposed SC-SDCN model and its processing flow when applied to FVR is firstly elaborated. Then, a detailed design of the SDC block is presented, which stands as the backbone of our SC-SDCN model. After, depth-wise separable convolutions are integrated into each backbone SDC block to substitute for standard convolutions. Finally, to further enhance the discrimination of features, a more robust margin penalty-based loss metric is introduced into the softmax loss framework.

3.1. Framework of SC-SDCN Model

The overall framework of the SC-SDCN model is shown in Figure 3. It is known as an end-to-end feature learning and matching network, which mainly consists of two parts; one is to use the SC-SDCN module for feature learning, and the other is to use the softmax classifier module for recognition. It should be noted that in the presented network configuration,'Input ROI' represents the ROI of each input sample image, each 'Conv' layer comprises a composition of convolution, batch normalization (BN), and ReLU activation steps, the 'DW-Conv' layer denotes a composition of depth-wise separable convolution followed by a batch normalization, and each 'SDC block' represents the core component of each sparsified densely connected block; their detailed architecture will be presented in Section 3.2. In addition, $\mathbf{x}_i$ denotes the final output feature vector of the SC-SDCN module, and $\mathbf{W}$ is the weight matrix in which $\mathbf{W}_{\mathbf{y}_i}$ denotes the ith column of matrix $\mathbf{W}$, which can be regarded as the ith class center vector.

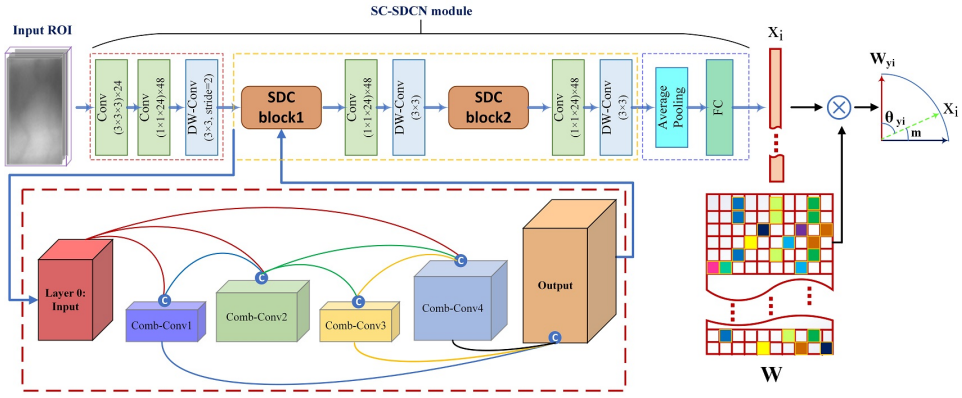


Figure 3. Overall framework of the proposed SC-SDCN model.

In the SC-SDCN part, a batch of FV sample images sized $224 \times 224 \times 3$ are fed into the first Conv layer, and each input image is convolved with 24 standard 3D convolutional kernels of size $3 \times 3 \times 3$ and stride 2, resulting in an output volume of size $112 \times 112 \times 24$. Then, a $112 \times 112 \times 24$-sized volume is sent to the next Conv layer and convolved with 48 kernels of size $1 \times 1 \times 24$ and stride 1, thus expanding the output channel to 48-dimensional. Subsequently, these 48-dimensional feature maps are sent to a pooling layer to further compress spatial resolution. Here, we adopted more efficient depth-wise separable convolution with a kernel size of 3×3 and stride 2 to approximate the pooling effect.

The output of the first Conv+Pooling is $56 \times 56 \times 48$, which will be fed into a sequence of SDC blocks to learn abstract semantic feature representations. Between two successive SDC blocks, the output feature maps of previous SDC blocks will flow to a 1×1 Conv layer for channel expansion and then flow to a DW-Conv layer for spatial pooling and down-sampling.

Once the feature learning of sequential SDC blocks is finished, the output feature volumes are fed into an adaptive average pooling layer to flatten into one-dimensional feature vectors. Here, the adaptive average pooling is used to average features along the spatial dimension and channel, thus forming an output with a fixed vector length. Finally, the fixed encoding feature vector is fed into fully connected layers, and category scores are calculated via the loss function. In order to enhance the discrimination of feature representations, a more robust angular margin metric is introduced to substitute a traditional Euclidean distance-based metric in a softmax cross-entropy loss function, and a detailed design of the adopted loss function will be presented in Section 3.3. To elaborate, Table 3 shows configurations of the proposed SC-SDCN model.

Table 3. Configurations of the proposed SC-SDCN model.

Layer	Kernel Size	Stride	Input Size	Output Size
Conv	3×3	2	$224 \times 224 \times 3$	$112 \times 112 \times 24$
Conv	1×1	1	$112 \times 112 \times 24$	$112 \times 112 \times 48$
DW-Conv	3×3	2	$112 \times 112 \times 48$	$56 \times 56 \times 48$
SDC block	$\begin{pmatrix} 1 \times 1 & Conv \\ 3 \times 3 & DW-Conv \end{pmatrix} \times 4$	1	$56 \times 56 \times 48$	$56 \times 56 \times 72$
Conv	1×1	1	$56 \times 56 \times 72$	$56 \times 56 \times 96$
DW-Conv	3×3	2	$56 \times 56 \times 96$	$28 \times 28 \times 96$
SDC block	$\begin{pmatrix} 1 \times 1 & Conv \\ 3 \times 3 & DW-Conv \end{pmatrix} \times 16$	1	$28 \times 28 \times 96$	$28 \times 28 \times 292$
Conv	1×1	1	$28 \times 28 \times 292$	$28 \times 28 \times 320$
DW-Conv	3×3	2	$28 \times 28 \times 320$	$14 \times 14 \times 320$
AvgPool			$14 \times 14 \times 320$	$1 \times 1 \times 320$
Flatten			$1 \times 1 \times 320$	320
Dropout			320	320
Linear			320	1000

3.2. Details of SDC Blocks

The design concept of the SDC block comes from two aspects, one of which refers to the feature aggregation and reuse on a macro-architectural level. As indicated by ResNet and DenseNet, skip connection provides an aggregation ability to transfer features of earlier layers directly to far deeper layers. However, one intuitive argument is that too-dense connections are prohibitively expensive, a naive implementation of DenseNet occupies $\mathcal{O}(L^2)$ memory, especially in the medium-depth output layers, and channel explosion and information redundancy occur most of the time. Actually, at each down-sampling, DenseNet halves the number of channels of previous layers to combat the scaling issue. Nevertheless, compressing previous features and newly appended features in a completely equal way arouses a balance issue involved in feature aggregation, which means more effort is spent on the previously seen features, while neglecting the balanced aggregation between current features and previous features. As a result, the contribution of newly added features becomes smaller and smaller. How to design more efficient skip connection patterns as well as feature aggregation strategies so as to not only alleviate over-constraining and over-burdening, but also balance previous and current features, still remains to be addressed. Except for the macro-architectural design, considering a micro design is also beneficial. As noted in Section 2.2, depth-wise separable convolution is an alternative to the standard convolutional layer that is supposed to be more efficient in computation costs.

Owing to the above factors, we introduced a specially designed sparsified dense connection architecture called 'SDC'. The main idea of SDC is derived from [72], in which a novel harmonized sparse, dense connection strategy was proposed (as shown in Figure 1d). Meanwhile, by substituting the standard convolutional kernels with depth-wise separable convolutional kernels, we ultimately obtained a sparsified skip connection structure with intelligent cropping and low computational complexity.

Concretely, each SDC block contains the number of layers to be an exponential power of two. Then, all internal layers in an SDC block are marked as odd-indexed layers and even-indexed layers, and the output features of one SDC block will aggregate all previous odd layers, as well as the last internal layer. In order to preserve a harmonic sparsification skip connection, we only let the ℓ^{th} layer connect with those layers of $\ell - 2^n$, if and only if 2^n can be divided by ℓ. Thus, for each odd layer, only its nearest even layer is aggregated, while for each even layer, not only the nearest odd layer but also parts of the previous even layers are aggregated, leading to a considerable number of feature channels in each even layer. The above strategy brings the first merit of SDC in that once the 2^n layer has been processed, all previous layers from 1 to $2^n - 1$ can be dumped out from the memory. With this regard, though SDC maintains a connection density of $L \log_2(L)$, it can obtain less concatenation cost than Log-DenseNet.

Furthermore, in order to balance the input/output channel ratio during each skip connection, the channel size of each layer is adjusted to an adaptive value of $\boldsymbol{k} \times \boldsymbol{m}^n$, in which $\boldsymbol{k}$ is a fixed value to denote the growth rate of channel numbers in each layer, $\boldsymbol{m}$ serves as a low-dimensional compression factor, and n is set to the maximum value that satisfies ℓ divisible by 2^n. With these settings, each odd-indexed layer has a constant channel size because of $n = 0$, which is derived from the feature compression of its previously nearest even layer, while for each even-indexed layer, the channel size is increased because of $n > 0$.

Finally, in the output of one SDC block, only the output of odd-indexed layers is concatenated and passed on to the next layer, while the outputs of even-indexed layers have been omitted. In this manner, we can obtain the second merit of SDC; namely, the current layer will show more of an effect than previous layers as it occupies a larger proportion of the aggregated features. In the meantime, maintaining a relatively appropriate value of $\boldsymbol{m}$ yields a higher compression ratio of output feature channels.

Admittedly, each SDC block essentially contains a combination of multiple network layers, and sparsified skip connections are used to aggregate features from different internal layers. Considering that skip connections only exist within each block while just a single

sequential connection is maintained between blocks, we define this structure as a sparsified dense connection (SDC) block.

To be specific, we provided detailed parameters setting of the first SDC block in Table 4, which contains four 'DW-Conv' layers, and each combined layer contains a point-wise convolution layer followed by a depth-wise separable convolution layer. The 1×1 point-wise convolutions are used to compress channels, while the depth-wise separable convolutions are used to learn the newly featured representations. Here, the output channel size of the corresponding combined layer is taking an integer result of $\boldsymbol{k} \times \boldsymbol{m}^n$. Likewise, Table 5 presented detailed parameter settings of the second SDC block, which contains sixteen 'DW-Conv' layers. The reason why different numbers of internal layers are used in SDC blocks of different positions is mainly due to the requirements of feature extraction at different depths. In this sense, the first SDC block is set to retain more low-level feature information, while the second SDC block is set to facilitate extracting high-level abstract semantic features.

Table 4. Parameters of the first SDC block, which includes 4 combined layers, and each combined layer contains a point-wise convolution layer followed by a depth-wise separable convolution layer. Here, the output channel size of the corresponding combined layer is taking an integer result of $\boldsymbol{k} \times \boldsymbol{m}^n$, and $\otimes$ in the 'Linked Layers' column represents channel-level concatenation.

Layers	Linked Layers	Input Size	Kernels	Output Size	n	m	k
0 (Input)	–	–	–	$56 \times 56 \times 48$	–	–	–
1	0	$56 \times 56 \times 48$	PW: $16 \times 1 \times 1 \times 48$ DW: 3×3	$56 \times 56 \times 16$	0	**1.6**	**16**
2	1⊗0	$56 \times 56 \times 64$	PW: $26 \times 1 \times 1 \times 64$ DW: 3×3	$56 \times 56 \times 26$	1		
3	2	$56 \times 56 \times 26$	PW: $16 \times 1 \times 1 \times 26$ DW: 3×3	$56 \times 56 \times 16$	0		
4	3⊗2⊗0	$56 \times 56 \times 90$	PW: $40 \times 1 \times 1 \times 90$ DW: 3×3	$56 \times 56 \times 40$	2		
5 (Output)	4⊗3⊗1	$56 \times 56 \times 72$	–	–	–	–	–

Table 5. Parameters of the second SDC blocks, which have 16 combined layers, and each combined layer contains a point-wise convolution layer followed by a depth-wise separable convolution layer. Here the output channel size of the corresponding combined layer is taking an integer result of $\boldsymbol{k} \times \boldsymbol{m}^n$.

Layers	Linked Layers	Input Size	Kernels	Output Size	n	m	k
0 (Input)	–	–	–	$28 \times 28 \times 96$	–	–	–
1	0	$28 \times 28 \times 96$	PW: $20 \times 1 \times 1 \times 96$ DW: 3×3	$28 \times 28 \times 20$	0	**1.6**	**20**
2	1⊗0	$28 \times 28 \times 116$	PW: $32 \times 1 \times 1 \times 116$ DW: 3×3	$28 \times 28 \times 32$	1		
3	2	$28 \times 28 \times 32$	PW: $20 \times 1 \times 1 \times 32$ DW: 3×3	$28 \times 28 \times 20$	0		
4	3⊗2⊗0	$28 \times 28 \times 148$	PW: $52 \times 1 \times 1 \times 148$ DW: 3×3	$28 \times 28 \times 52$	2		
5	4	$28 \times 28 \times 52$	PW: $20 \times 1 \times 1 \times 52$ DW: 3×3	$28 \times 28 \times 20$	0		
6	5⊗4	$28 \times 28 \times 72$	PW: $32 \times 1 \times 1 \times 72$ DW: 3×3	$28 \times 28 \times 32$	1		
7	6	$28 \times 28 \times 32$	PW: $20 \times 1 \times 1 \times 32$ DW: 3×3	$28 \times 28 \times 20$	0		
8	7⊗6⊗4⊗0	$28 \times 28 \times 200$	PW: $82 \times 1 \times 1 \times 200$ DW: 3×3	$28 \times 28 \times 82$	3		
9	8	$28 \times 28 \times 82$	PW: $20 \times 1 \times 1 \times 82$ DW: 3×3	$28 \times 28 \times 20$	0		

Table 5. *Cont.*

Layers	Linked Layers	Input Size	Kernels	Output Size	n	m	k
10	9⊗8	28 × 28 × 102	PW: 32 × 1 × 1 × 102 DW: 3 × 3	28 × 28 × 32	1		
11	10	20 × 28 × 32	PW: 20 × 1 × 1 × 32 DW: 3 × 3	28 × 28 × 20	0		
12	11⊗10⊗8	28 × 28 × 134	PW: 52 × 1 × 1 × 134 DW: 3 × 3	28 × 28 × 52	2		
13	12	28 × 28 × 52	PW: 20 × 1 × 1 × 52 DW: 3 × 3	28 × 28 × 20	0		
14	13⊗12	28 × 28 × 72	PW: 32 × 1 × 1 × 72 DW: 3 × 3	28 × 28 × 32	1		
15	14	28 × 28 × 32	PW: 20 × 1 × 1 × 32 DW: 3 × 3	28 × 28 × 20	0		
16	15⊗14⊗ 12⊗8⊗0	28 × 28 × 282	PW: 132 × 1 × 1 × 282 DW: 3 × 3	28 × 28 × 132	4		
17 (Output)	16⊗15⊗13 ⊗11⊗9⊗7 ⊗5⊗3⊗1	28 × 28 × 292	–	–	–	–	–

3.3. Loss Function and Training Strategy

By means of SDC block architectures, the maximum distance of gradient back-propagation in SC-SDCN can be controlled within the range of log L, which will shorten the convergence time to a certain extent. However, we also need to choose or design an appropriate loss function, as it is responsible for measuring the discrepancies between the predicted value and ground truth and then employed for error back-propagation and network parameters estimation.

In most FVR deep networks, a softmax cross-entropy loss function is usually adopted. It should be noted that traditional softmax only learns a feature embedding with overlapped decision boundaries between different classes, which leads to unreliable generalization to unknown classes in the open-set scenarios. Considering that FV images are generally captured with different acquisition devices, inter-class similarity and inner-class variability are ubiquitous, and the obtained feature embeddings are sensitive to scale issues. To overcome these drawbacks, some softmax loss variants have been designed. For the impact of scale, L_2 normalization is enforced on the feature and weight vectors, respectively, so that the network optimization is transformed into a cosine similarity to facilitate angular discrimination. Admittedly, transforming from Euclidean space to angular space does not intrinsically enhance the discrimination of classes, and some margin strategies are imposed on the angular value or its cosine value to further maximize the decision margin in the angular space. Furthermore, in [76], an additive angular margin penalty (dubbed 'AAMP') loss function was proposed for deep face recognition. Firstly, the angle between the output feature vector and the corresponding weight vector is calculated by using an arc-cosine function. Afterward, a meticulously designed margin penalty is directly added to the angle so as to expand the angular distance among different classes. Finally, the logit value is obtained again by the cosine function. Corresponding to the margin strategies that were imposed on the cosine of the angle or multiplied with the angle, AAMP has simper differential computation and explicit geometric interpretation and thus significantly enhances the discriminative feature embedding. Therefore, we introduce the AAMP into our proposed SC-SDCN to substitute the Euclidean metric in the softmax loss framework.

Concretely, let $\boldsymbol{x}_i \in \mathbb{R}^d$ denote the output feature vector corresponding to the i^{th} sample image, and its class label belongs to the ${\boldsymbol{y}_i}^{th}$ class. Meanwhile, let $\mathbf{W} \in \mathbb{R}^{d \times U}$ represent the target category weight matrix, and its j^{th} column is represented by $\mathbf{W}_j \in \mathbb{R}^d$, where d and U are the feature size and number of classes, respectively. When computing the logit, the

bias term is ignored for simplicity, and the activation of the final fully connected layer $\mathbf{F}_j$ can be represented as follows:

$$\mathbf{F}_j = \mathbf{W}_j^T \boldsymbol{x}_i = \|\mathbf{W}_j\| \|\boldsymbol{x}_i\| \cos \boldsymbol{\theta}_j, \tag{4}$$

where $\boldsymbol{\theta}_j$ denotes the angle between weight $\mathbf{W}_j$ and vein feature $\boldsymbol{x}_i$; thus, the softmax loss function can be formulated as:

$$L_{\text{softmax}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\mathbf{F}_{y_i})}{\sum_{j=1}^{U} \exp(\mathbf{F}_j)}, \tag{5}$$

In formula (5), B is the batch size. When the L_2 normalization is performed on the feature and weight vectors, the activation term in Formula (4) becomes $\mathbf{F}_j = \boldsymbol{s} \cos \boldsymbol{\theta}_j$, being $\|\mathbf{W}_j\| = 1$ and $\|\boldsymbol{x}_i\| = \boldsymbol{s}$. Finally, the AAMP loss function can be formulated as

$$L_{\text{AAMP}} = -\frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\boldsymbol{s} \cos(\boldsymbol{\theta}_{y_i} + \boldsymbol{m}))}{\exp(\boldsymbol{s} \cos(\boldsymbol{\theta}_{y_i} + \boldsymbol{m})) + \sum_{j=1, j \neq y_i}^{U} \exp(\boldsymbol{s} \cos \boldsymbol{\theta}_j)}, \tag{6}$$

In Formula (6), the hyper-parameters $\boldsymbol{s}$ and $\boldsymbol{m}$ are the scale factor and penalty margin, respectively. When the AAMP loss function is employed for our SC-SDCN model, the penalty margin $\boldsymbol{m}$ is selected in the range of $[0.3, 0.7]$ with a step size of 0.05, and the scale parameter $\boldsymbol{s}$ is selected in the range of $[16, 96]$ with a step size of 16.

After the SC-SDCN model has been well-trained, it will be used to predict the recognition results of each input test image. Given that the output of the softmax classifier is a sample-to-class probability value, the final prediction result will be derived from the maximum class confidence score.

3.4. Evaluation Criteria

For quantitative evaluation, FAR (false acceptance rate), FRR (false rejection rate), and EER (equal error rate) are adopted as the evaluation metrics of experimental results. Among, FAR is the ratio of the number of accepted imposter claims divided by the number of identification attempts, as shown in Equation (7), where FA_NUM is the number of false acceptances, and IA_NUM is the number of impostor recognition attempts.

$$\text{FAR} = \frac{\text{FA\_NUM}}{\text{IA\_NUM}} \times 100\%, \tag{7}$$

FRR is the ratio of the number of rejected genuine claims divided by the number of identification attempts, as shown in Equation (8), where FR_NUM is the number of false rejections, and GRA_NUM is the number of genuine recognition attempts.

$$\text{FRR} = \frac{\text{FR\_NUM}}{\text{GRA\_NUM}} \times 100\%, \tag{8}$$

Finally, EER is the rate at which the FAR is equal to the FRR, and a lower EER exhibits better performance in FVR. In addition, we also apply recognition accuracy (ACC) for performance evaluation, which is the ratio of the number of correct recognition divided by the number of total recognition.

4. Experimental Results and Discussion

In this section, to ascertain the effectiveness of our SC-SDCN model, we carried out a comprehensive experimental analysis by utilizing two available benchmark FV datasets, including MMCBNU [25] and FV-USM [26]. First, Section 4.1 provided a brief description of the two FV datasets. Then, the relevant experimental settings and training procedures were presented in Section 4.2. Next, the recognition accuracies under different splitting ratios of training/test set, as well as the corresponding computational costs, were analyzed in Sections 4.3 and 4.4, respectively. After, in Section 4.5, the discriminative power of AAMP

loss function was quantitatively evaluated under two scenarios of self-learning and transfer learning. Finally, Section 4.6 reported the recognition results of the SC-SDCN model as compared with several mainstream FVR methods.

4.1. Finger Vein Datasets

In our experiments, two publicly available benchmark FV datasets are selected to test the performance of the SC-SDCN model, one of which is MMCBNU [25] (available at: http://multilab.jbnu.ac.kr/MMCBNU_6000) (accessed on 1 January 2021), and the other is FV-USM [26] (available at: http://drfendi.com/fv_usm_database/) (accessed on 1 January 2021), their specific descriptions are shown as follows. It should be noted that both datasets have provided well-tailored ROI images. Moreover, plenty of published FVR approaches have provided their recognition results on these two datasets, which facilitates the subsequent analysis and comparison.

The MMCBNU [25] database was published by the Multimedia Lab, Division of Electronic and Information Engineering, Chonbuk National University. It collected 100 volunteers from 20 countries, and each volunteer provided the index finger, middle finger and ring finger of two hands, thus forming a total of 600 classes with 10 images for each class. The provided ROI images are in grayscale with a size of 128×60, and the fingertips are horizontal to the right.

The FV-USM [26] dataset was published by the University of Sains Malaysia. It collected finger vein images from 123 subjects in two different sessions. Each subject provided the index finger and middle finger of two hands. Thus, there are a total of 492 classes with 12 sample images per finger class. In our experiments, we only used images from a single session, namely, a total of 2952 images from 492 classes with six samples per class. The corresponding ROI images are also in grayscale and with a size of 50×150, and the fingertips are upward.

More detailed descriptions of above two FV datasets are shown in Table 6. As can be seen, the number of samples provided by each class is very small, while the number of corresponding classes is relatively large. Moreover, Figure 4 shows the respective pair of images in both datasets, including the original acquired image and the corresponding ROI image, and we will use ROI images directly in the following experiments.

(**a**) Raw of MMCBNU (**b**) ROI of MMCBNU (**c**) Raw of FV-USM (**d**) ROI of FV-USM

Figure 4. Illustration of the respective pair of sample images (original acquired image and corresponding ROI image) in both finger vein datasets.

Table 6. Descriptions of the adopted finger vein datasets.

Name	Subjects	Fingers	Classes	Samples/Class	Total Samples	Sessions	Orientations
MMCBNU	100	index middle ring	600	10	6000	1	right
FV-USM	123	index middle	492	12	5904	2	down

4.2. Experimental Settings

4.2.1. Splitting of Training/Test Set

For finger vein recognition scenarios, it is usually performed in two different configurations: open-set and closed-set. In a closed-set experiment, all available classes should be used in the training process. However, this is an unrealistic situation as samples from all classes are hard to hold in advance. In this regard, it is more reasonable to carry out experiments in an open-set scenario, namely, training with parts of available classes and then testing on other remaining classes or even new classes. With this consideration, we adopted an open-set configuration in subsequent experiments and kept non-overlapping of the training and test sets. In addition, to verify the model performance in the case of a small training sample size, we randomly split the above FV datasets into two parts according to different ratios. For example, for the MMCBNU dataset, the ratio of 9:1 denotes 90% is used for training, and the remaining 10% is used for testing. Therefore, 540 classes with a total of 5400 sample images are randomly picked out for training, while the remaining 60 classes with a total of 600 images are used for testing. In the following experiments, five different ratios are conducted, including 9:1, 8:2, 7:3, 6:4 and 5:5. For each case, we always split the whole sample image set into 10 equal parts; then, we randomly select a non-overlapped training set and test set according to the split ratios, and the reported experimental results are derived from the average of multiple experiments. For example, when the ratio is 9:1, the number of experiments is 10, with each group selected as the test set and the remaining 9 groups as the training set. When the ratio is 5:5, only two experiments are conducted, in which five groups are randomly selected as the training set and the remaining five groups as the test set, and then cross-validated. Finally, all sample images are re-scaled to a size of 224×224, with a normalization of zero mean and unit variance, and then fed into the SC-SDCN.

4.2.2. Network Initialization and Optimization

As mentioned earlier, the considered network is trained with the identification mode, while at the testing stage, the outputs of the last fully connected layer are taken as feature templates for verification purposes. For the network initialization, the initial weights of convolutional kernels are set to a normal distribution with zero-mean and a standard deviation of 0.01, and the biases are initialized from a normal distribution with a mean value of 0.5 and a standard deviation of 0.01. In the fully connected layer, the weights are drawn from a normal distribution with zero-mean and standard deviation of 0.2, and the biases are initialized in the same way as the convolutional layers. In addition, the stochastic gradient descent (SGD) optimizer with a batch size of 32, a learning rate of 0.01, a weight decay of 0.01, and a momentum of 0.9 is employed for speeding up the convergence of gradient vectors. The network is trained up to 200 epochs, and the dimension of the output feature embedding is set to 1000 dimensions.

Finally, we would like to emphasize that all experiments were conducted by using Python 3.8 with the PyTorch 1.8.0 framework and run on a desktop PC equipped with the configurations of 32Gb RAM, an NVIDIA GeForce GTX 1080 Ti GPU, and an Intel Core i7 CPU (at 3.6 GHz).

4.3. Analysis of Different Splitting Ratios of Training/Test Set

In this experiment, we analyzed the performance of the SC-SDCN model on different splitting ratios of the training/test set. Here, five ratios are tested, and the corresponding ACC and EER results are presented in Tables 7 and 8. As observed on both datasets, more training data and less testing data lead to higher ACC and lower EER results; namely, 9:1 results are better than those under the other ratios. However, we can observe that even with less training data, such as in the case of 5:5, satisfactory results can still be obtained. For the MMCBNU dataset, under the optimal network configuration, the ACC of the 9:1 ratio is 0.9998, while the ACC of the 5:5 ratio is 0.9968, just 0.3% lower. Likewise, for the FV-USM dataset, the ratio was just 0.12% lower under the optimal configuration.

The experimental results show that our SC-SDCN model can achieve promising recognition accuracy even when the training samples are insufficient. It is because the design of the SDC block advocates for the efficient use of compositional skip connections to shorten the distances among feature layers during back-propagation; such cropping optimization not only significantly reduces the number of model parameters but also synchronously maintains a balance between the newly aggregated features and the previously incoming features. As a result, the saliency of the newly aggregated features is enhanced. Corresponding to the DenseNet or Log-DenseNet, such connection cropping strategy is not a post-processing pruning operation, nor does it involve cutting connections arbitrarily but advocates for a network design principle to place skip connections intelligently to pursue the efficiency of feature aggregation. In addition, by introducing depth-wise separable convolution, the number of channels in each layer is further compressed so that the overall number of network parameters is further reduced.

Table 7. ACC and EER results on the MMCBNU dataset with a different number of SDC blocks in the SC-SDCN architecture, in which 'SL' denotes a fully self-training procedure, and 'TL' denotes the pre-training on the ImageNet dataset and then transfers the pre-trained parameters to the target FV dataset for fine-tuning.

Architectures	Criteria	Splitting Ratio of Training/Test Set					Model Size (M)
		9:1	8:2	7:3	6:4	5:5	
2 SDC+SL	ACC	0.9974	0.9968	0.9967	0.9953	0.9926	4.22
	EER	0.0034	0.0059	0.0059	0.0094	0.0155	
2 SDC+TL	ACC	**0.9998**	0.9982	0.9983	0.9981	0.9968	
	EER	**0.0001**	0.0032	0.0038	0.0031	0.0058	
3 SDC+SL	ACC	0.9985	0.9961	0.9961	0.9937	0.9932	12.1
	EER	0.0016	0.0066	0.0076	0.0113	0.0137	
3 SDC+TL	ACC	0.9992	0.9982	0.9981	0.9963	0.9971	
	EER	0.0009	0.0035	0.0036	0.0067	0.0061	
4 SDC+SL	ACC	0.9934	0.9926	0.9931	0.9886	0.9912	30.7
	EER	0.0115	0.0125	0.0135	0.0206	0.0149	
4 SDC+TL	ACC	0.9942	0.9951	0.9929	0.9896	0.9901	
	EER	0.0095	0.0121	0.0126	0.0194	0.0184	

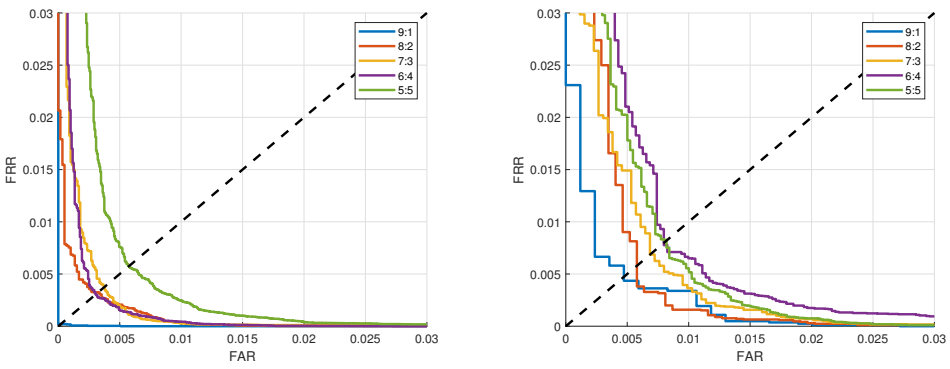
Table 8. ACC and EER results on the FV-USM dataset with a different number of SDC blocks in the SC-SDCN architecture.

Architectures	Criteria	Splitting Ratio of Training/Test Set					Model Size (M)
		9:1	8:2	7:3	6:4	5:5	
2 SDC+SL	ACC	0.9938	0.9945	0.9924	0.9902	0.9932	4.22
	EER	0.0110	0.0112	0.0165	0.0234	0.0167	
2 SDC+TL	ACC	**0.9974**	0.9972	0.9963	0.9959	0.9962	
	EER	**0.0045**	0.0048	0.0069	0.0086	0.0082	
3 SDC+SL	ACC	0.9880	0.9894	0.9892	0.9883	0.9894	12.1
	EER	0.0201	0.0229	0.0217	0.0285	0.0269	
3 SDC+TL	ACC	0.9968	0.9956	0.9923	0.9946	0.9951	
	EER	0.0045	0.0093	0.0108	0.0122	0.0092	
4 SDC+SL	ACC	0.9885	0.9881	0.9876	0.9855	0.9857	30.7
	EER	0.0203	0.0231	0.0225	0.0282	0.0279	
4 SDC+TL	ACC	0.9840	0.9907	0.9898	0.9912	0.9895	
	EER	0.0225	0.0162	0.0142	0.0171	0.0181	

To further assess the robustness and generalization of our SC-SDCN model, we then carried out a series of related experiments from the perspective of model architecture design and transfer learning. In the aspect of structural design, we constructed a network model by using a different number of SDC blocks so as to assess the impact of different network depths. The number of SDC blocks with 2, 3, 4 are constructed and compared. Here, the number of layers within each SDC block is set to $\{4, 16, 8, 4\}$ in order. Meanwhile, by following a 1×1 channel-wise convolutional layer, the final number of output channels of each block is set to $\{96, 320, 640, 1024\}$, respectively, in order. In this case, the fixed channel growth rate of $\boldsymbol{k}$ in each SDC block is set to $\{16, 20, 64, 160\}$ in order, while the compression factor of $\boldsymbol{m}$ is set to the same value of 1.6 in all SDC blocks. In the aspect of network pre-training, we adopted transfer learning to overcome the lack of samples so as to evaluate the issues of limited datasets for model training as it can be mitigated by employing stable models that have been trained on large-scale datasets. Here, we carried out model pre-training on ImageNet and then transferred the pre-trained weight parameters to the target FV dataset for fine-tuning.

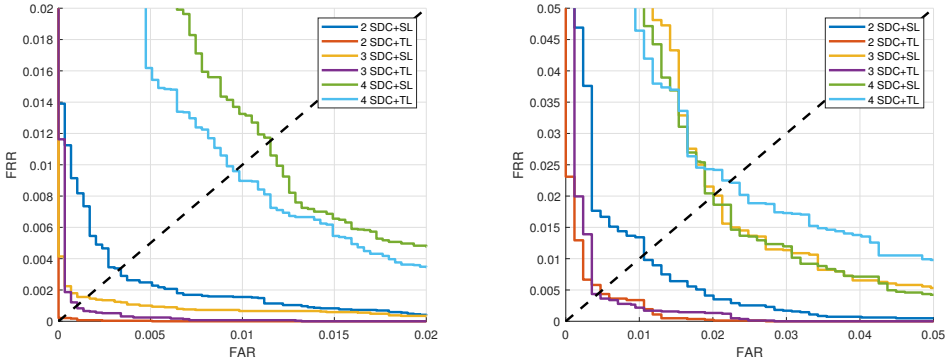
Tables 7 and 8 also presented ACC and EER results under different compositions of SDC block structures and with/without transfer learning. Furthermore, 'SL' means self-learning on the FV dataset only, and 'TL' means pre-training on the ImageNet dataset and then transferring to the FV dataset for fine-tuning. The experimental results on both FV datasets show that the pre-trained model with the two SDC blocks structure has better performance gains, which is also in line with the previous statements that the recognition of finger vein images mainly relies on mid-level features, and too-abstract semantic features can easily lead to performance degradation. Considering that the model with two SDC blocks has a smaller parameter quantity and model size, we adopted a configuration of two SDC blocks with transfer learning in the subsequent experiments.

Finally, for visual representation, we also provided Detection Error Tradeoff (DET) curves of different splitting ratios of training/test set in Figure 5 and DET curves of various compositions of SDC block structures and training strategies in Figure 6. In all DET curves, the x-axis and y-axis are FAR and FRR, respectively. Generally, the larger the threshold, the smaller the FAR value, and the larger the FRR value. When FAR is equal to FRR, it is defined as EER; at this time, the smaller the EER value, the closer the DET curve is to the x-axis and y-axis, which indicates a better model performance. As can be observed from Figure 5, though the higher the ratio of training/test set, the better the recognition performance, the difference between different ratios is not obvious and thus further supports the excellent model performance in the case of insufficient samples. From Figure 6, we can intuitively find that two SDCs are better than four SDCs, transfer learning ('TL') is better than self-learning ('SL') on both FV datasets and even two SDC+SL is better than four SDC+TL. This is because the FVR task does not require an overly semantic feature representation. In addition, a very deep network often requires more training samples; otherwise. it is more likely to overfit.



(**a**) MMCBNU (**b**) FV-USM

Figure 5. DET curves of different splitting ratios of training/test set on two-finger vein datasets.



(**a**) MMCBNU (**b**) FV-USM

Figure 6. DET curves of a different number of SDC blocks in the SC-SDCN architecture.

4.4. Analysis of Computational Cost

In this experiment, we analyzed the computational cost of the structural design of the SC-SDCN model. Tables 9 and 10 have shown the model size, training time, and feature extraction time of the SC-SDCN model on two FV datasets. Here, the training cost is represented by the training time for each epoch. After the model has been well-trained, we also give the feature extraction time of each image. Since the size of the MMCBNU dataset is larger than the FV-USM dataset, the corresponding epoch training time is also longer. Relatively speaking, since each sample image is normalized to the same size of 224×224 before being fed into the network, the feature extraction takes almost the same time.

Obviously, fewer SDC blocks means smaller model size, in which the model size of three SDCs has 7.88M more than two SDCs, and the model size of four SDCs has 18.6M more than three SDCs. The model size seems to grow exponentially with the increase in the number of SDC blocks. This is because as the number of SDC blocks increases, the number of skip connections increases at a rate of $L \log_2 L$, resulting in an approximately exponential growth in model size. Finally, as the number of SDC blocks increases, so does the training time and feature extraction time. From the comprehensive analysis based on Tables 7 and 8, the recognition results have not become better as the number of blocks increases, so we suggest using a model containing two SDC blocks for the FVR task; in this case, the model is more conducive to deployment and installation on embedded hardware devices.

Table 9. Computational cost of MMCBNU.

Architectures	Model Size (M)	Training (s/epoch)	Feature Extraction (ms/image)
2 SDC+TL	4.22	4.4077	0.0651
3 SDC+TL	12.1	5.8126	0.0959
4 SDC+TL	30.7	6.6173	0.1122

Table 10. Computational cost of FV-USM data.

Architectures	Model Size (M)	Training (s/epoch)	Feature Extraction (ms/image)
2 SDC+TL	4.22	2.0135	0.0656
3 SDC+TL	12.1	2.6914	0.0958
4 SDC+TL	30.7	3.0614	0.1118

4.5. Analysis of Loss Function

The purpose of this experiment is to verify that the AAMP loss function can enhance the discrimination of features. As declared previously, AAMP introduced an additive margin that was directly applied to the angle between the feature vector and the weight

vector, thus expanding the inter-class distance as well as narrowing the intra-class distance. Here, we choose traditional softmax loss for comparison. In the whole experimental procedure, we kept the consistency of the feature extraction process, except for replacing different loss functions. In addition, considering that the per-training process of transfer learning has a positive effect on the network performance, we also provided the results of different loss functions under the pre-training scenario, as shown in Tables 11 and 12. Among these tables, the second and fourth columns are the results of AAMP and traditional softmax that are equipped with pre-training model parameters. It shows that AAMP is better than traditional softmax. While for the third and fifth columns, there are no pre-training parameters to be used, and they are worse than the counterpart loss function equipped with pre-training parameters.

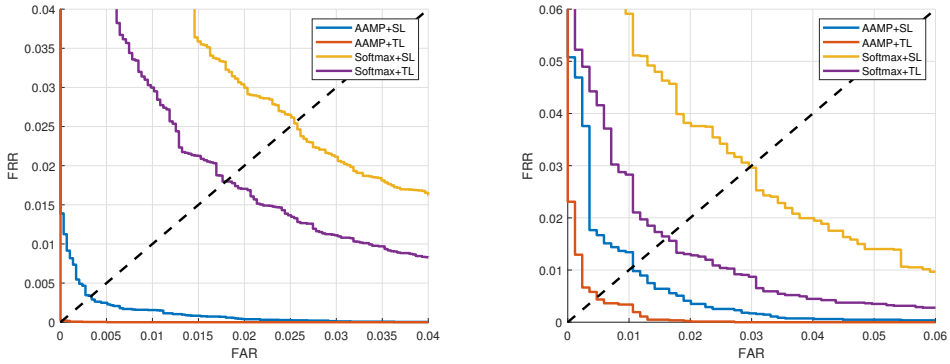
Table 11. AAMP loss function compared with traditional softmax loss on the MMCBNU dataset. The ACC and EER are obtained after 200 epochs.

	AAMP		**Traditional Softmax**	
	TL	**SL**	**TL**	**SL**
ACC	**0.9998**	0.9974	0.9867	0.9828
EER	**0.0001**	0.0034	0.0180	0.0254

Table 12. AAMP loss function compared with traditional softmax loss on the FV-USM dataset. The ACC and EER are obtained after 200 epochs.

	AAMP		**Traditional Softmax**	
	TL	**SL**	**TL**	**SL**
ACC	**0.9974**	0.9938	0.9897	0.9826
EER	**0.0045**	0.0110	0.0165	0.0296

For visual representation, we also presented the DET curves by using AAMP or traditional softmax loss functions on two FV datasets. As can be seen in Figure 7, the AAMP loss function obtained significantly lower EER results than those achieved with traditional softmax. This is because traditional softmax only recognizes samples to classes while not considering the discrimination between classes. To solve this issue, AAMP introduces an angular additive margin to intuitively and efficiently penalize the overlap of the decision boundaries between different classes in an angular space.



(**a**) MMCBNU (**b**) FV-USM

Figure 7. DET curves of a different number of SDC blocks in the SC-SDCN architecture.

4.6. Comparison with State-of-the-Art

In the last experiment, we compared our proposed SC-SDCN model with some mainstream FVR methods. These compared methods can be divided into two categories. One

category covers the handcrafted FV feature extraction methods, and we chose six methods for comparison, including local directional code (LDC) [17], histogram of salient edge orientation map (HSEOM) [30], 2D principal component analysis (2D-PCA) [77], and histogram of competitive orientations and magnitudes (HCOM) [29]; moreover, two recently published methods, Radon-like features (RLFs) [5] and partial least squares discriminant analysis (PLS-DA) [21] were also selected for comparison. The second category covers the deep learning models, five representative networks are selected and compared with our SC-SDCN model, including fully convolutional network (FCN) [78], two-stream CNN [43], CNN competitive order (CNN-CO) [58], convolutional auto-encoder (CAE) [34], and a lightweight CNN combining center loss and dynamic regularization (lightweight CNN) [44]. For those handcrafted methods, after finishing the feature extraction, there will be a template match on a pair of finger feature maps. If they are from the same finger, they will output genuine matches; otherwise, imposter matches will be output if they are from different fingers. For those deep network models, most of them have an end-to-end learning procedure and directly output the category to which the sample belongs.

As shown in Table 13, the EERs obtained by deep-learning-based methods are generally better than those handcrafted-based feature extraction methods. This is because handcrafted methods mainly extract shallow features, and these shallow features are easily affected by noise, as well as image rotation and translation. On the contrary, deep-learning-based methods can extract higher-level features, which are more conducive to discrimination. In addition, of these compared deep learning methods, only lightweight CNN [44] belongs to the lightweight network. Our SC-SDCN model obtained the smallest EER value of 0.01% on the MMCBNU dataset, while on the FV-USM dataset, the third lowest EER result was obtained by our model, and the lowest result of CAE [34] was achieved in a closed-set scenario. Moreover, compared to the Lightweight CNN [44], we obtained a better EER of 0.45%, and our model size is only 4.22M, which is also relatively smaller than the lightweight CNN [44].

Table 13. EER (%) results compared with some finger vein recognition methods.

Category	Method	MMCBNU	FV-USM
Handcrafted	Local Directional Code (LDC) [17]	1.03	–
	Histogram of Salient Edge Orientation Map (HSEOM) [30]	0.9	–
	2D-PCA [77]	–	2.32
	Histogram of Competitive Orientations and Magnitudes (HCOM) [29]	0.36	–
	Radon-like Features (RLF) [5]	3.33	0.93
	Partial Least Squares Discriminant Analysis (PLS-DA) [21]	0.63	0.15
Deep learning	FCN+Segmentation [78]	–	1.42
	Two-stream CNN [43]	0.1	–
	CNN Competitive Order (CNN-CO) [58]	0.74	–
	Convolutional Auto-Encoder (CAE) [34]	–	**0.12**
	Lightweight CNN Combining Center Loss and Dynamic Regularization [44]	0.503	1.07
	Our Proposed SC-SDCN	**0.01**	0.45

On the whole, our SC-SDCN has shown superior recognition performance on both FV datasets, especially when the training samples were limited. Additionally, our SC-SDCN model has a smaller parameter quantity and model size and is thus more suitable for real-time FVR scenarios.

5. Conclusions and Future Research

In this paper, we have proposed a novel CNN pipeline for finger-vein recognition called SC-SDCN. By introducing a sparsified dense connection architecture and making full use of the pre-trained network weights, the proposed SC-SDCN achieved EERs of 0.01% on the MMCBNU dataset and 0.45% on the FVUSM dataset. Moreover, thanks to the AAMP loss function to replace the traditional softmax loss, the SC-SDCN outperforms some state-of-the-art FVR methods on two benchmark FV datasets.

In the experiments, though an open-set environment is maintained, we directly adopted extracted ROI images for network training. In this regard, the training quality of the network model depends on the quality of the input ROI image samples. In the future, we will explore the impact of the ROI image quality on the network learning so as to further improve the generalization ability of the SC-SDCN model. Recent research has revealed that the FVR system is vulnerable to presentation attacks. In this sense, whether our SC-SDCN model has the ability to detect spoof attacks still deserves further exploration. Furthermore, the fusion of multimodal biometrics has shown significant improvement in individual traits. In the future, we will explore the capability of our SC-SDCN model in feature-level fusion of palm vein and finger vein traits and expect better performance in different biometric traits. Furthermore, since lightweight models are used to facilitate deployment and application, and some novel lightweight design methods have been proposed recently [79,80], we will explore these new lightweight methods and try to apply them to finger vein recognition scenarios in the future. Finally, we will carry out experiments on the other publicly available FV datasets to assess the efficiency of the pre-training weight parameters based on transfer learning more fully.

Author Contributions: Methodology, Q.Y. and X.X.; writing—original draft preparation, Q.Y.; writing—review and editing, X.X. and W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This study was supported by the National Natural Science Foundation of China under Grant 62271130 and 62002053. Science and Technology Foundation of Guangdong Province under Grant 2021A0101180005. Education and Research Foundation of Guangdong Province under Grant 2020KTSCX182.

Data Availability Statement: Not applicable.

Acknowledgments: The authors would like to take this opportunity to thank the editors and the anonymous reviewers for their detailed comments and suggestions, which greatly helped us to improve the clarity and presentation of our manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Khan, S.A.; Naaz, S. Comparative Analysis of Finger Vein, Iris and Human Body Odor as Biometric Approach in Cyber Security System. In Proceedings of the 2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), Bangalore, India, 5–7 March 2020; pp. 525–530.
2. Shaheed, K.; Liu, H.; Yang, G.; Qureshi, I.; Gou, J.; Yin, Y. A Systematic Review of Finger Vein Recognition Techniques. *Information* **2018**, *9*, 213. [CrossRef]
3. Kumar, R.; Bharti, V. A Critical Review of Finger Vein Recognition Techniques for Human Identification. In Proceedings of the 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 2–4 September 2021; pp. 1–9.
4. Shaheed, K.; Mao, A.; Qureshi, I.; Kumar, M.; Hussain, S.; Zhang, X. Recent advancements in finger vein recognition technology: Methodology, challenges and opportunities. *Inf. Fusion* **2022**, *79*, 84–109. [CrossRef]
5. Yao, Q.; Song, D.; Xu, X.; Zou, K. A Novel Finger Vein Recognition Method Based on Aggregation of Radon-Like Features. *Sensors* **2021**, *21*, 1885. [CrossRef]
6. Liu, F.; Yang, G.; Yin, Y.; Wang, S. Singular value decomposition based minutiae matching method for finger vein recognition. *Neurocomputing* **2014**, *145*, 75–89. [CrossRef]
7. Meng, X.; Zheng, J.; Xi, X.; Zhang, Q.; Yin, Y. Finger vein recognition based on zone-based minutia matching. *Neurocomputing* **2021**, *423*, 110–123. [CrossRef]

8. Miura, N.; Nagasaka, A.; Miyatake, T. Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification. *Mach. Vis. Appl.* **2004**, *15*, 194–203. [CrossRef]
9. Huang, B.; Dai, Y.; Li, R.; Tang, D.; Li, W. Finger-Vein Authentication Based on Wide Line Detector and Pattern Normalization. In Proceedings of the 2010 20th International Conference on Pattern Recognition, Istanbul, Turkey, 23–26 August 2010; pp. 1269–1272.
10. Miura, N.; Nagasaka, A.; Miyatake, T. Extraction Of Finger-vein Patterns Using Maximum Curvature Points In Image Profiles. *Ieice Trans. Inf. Syst.* **2007**, *e90-d*, 1185–1194. [CrossRef]
11. Song, W.; Kim, T.; Kim, H.C.; Choi, J.H.; Kong, H.J.; Lee, S.R. A finger-vein verification system using mean curvature. *Pattern Recognit. Lett.* **2011**, *32*, 1541–1547. [CrossRef]
12. Syarif, M.A.; Ong, T.S.; Teoh, A.B.J.; Tee, C. Enhanced maximum curvature descriptors for finger vein verification. *Multimed. Tools Appl.* **2017**, *76*, 6859–6887. [CrossRef]
13. Yang, L.; Yang, G.; Xi, X.; Meng, X.; Zhang, C.; Yin, Y. Tri-Branch Vein Structure Assisted Finger Vein Recognition. *IEEE Access* **2017**, *5*, 21020–21028. [CrossRef]
14. Yang, L.; Yang, G.; Yin, Y.; Xi, X. Finger Vein Recognition With Anatomy Structure Analysis. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 1892–1905. [CrossRef]
15. Krishnan, A.; Thomas, T.; Mishra, D. Finger Vein Pulsation-Based Biometric Recognition. *IEEE Trans. Inf. Forensics Secur.* **2021**, *16*, 5034–5044. [CrossRef]
16. Huang, D.; Tang, Y.; Wang, Y.; Chen, L.; Wang, Y. Hand-Dorsa Vein Recognition by Matching Local Features of Multisource Keypoints. *IEEE Trans. Cybern.* **2015**, *45*, 1823–1837. [CrossRef]
17. Meng, X.; Yang, G.; Yin, Y.; Xiao, R. Finger Vein Recognition Based on Local Directional Code. *Sensors* **2012**, *12*, 14937–14952. [CrossRef]
18. Liu, H.; Yang, L.; Yang, G.; Yin, Y. Discriminative Binary Descriptor for Finger Vein Recognition. *IEEE Access* **2018**, *6*, 5795–5804. [CrossRef]
19. Hu, N.; Ma, H.; Zhan, T. Finger vein biometric verification using block multi-scale uniform local binary pattern features and block two-directional two-dimension principal component analysis. *Optik* **2020**, *208*, 163664. [CrossRef]
20. Wu, J.D.; Liu, C.T. Finger-vein pattern identification using principal component analysis and the neural network technique. *Expert Syst. Appl.* **2011**, *38*, 5423–5427. [CrossRef]
21. Zhang, L.; Sun, L.; Li, W.; Zhang, J.; Cai, W.; Cheng, C.; Ning, X. A Joint Bayesian Framework Based on Partial Least Squares Discriminant Analysis for Finger Vein Recognition. *IEEE Sens. J.* **2022**, *22*, 785–794. [CrossRef]
22. Kumar, A.; Zhou, Y. Human Identification Using Finger Images. *IEEE Trans. Image Process.* **2012**, *21*, 2228–2244. [CrossRef]
23. Yin, Y.; Liu, L.; Sun, X. SDUMLA-HMT: A Multimodal Biometric Database. In *Biometric Recognition*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 260–268.
24. Ye, Y.; Ni, L.; Zheng, H.; Liu, S.; Zhu, Y.; Zhang, D.; Xiang, W.; Li, W. FVRC2016: The 2nd Finger Vein Recognition Competition. In Proceedings of the 2016 International Conference on Biometrics (ICB), Halmstad, Sweden, 13–16 June 2016; pp. 1–6.
25. Lu, Y.; Xie, S.; Yoon, S.; Yang, J.; Park, D. Robust Finger Vein ROI Localization Based on Flexible Segmentation. *Sensors* **2013**, *13*, 14339–14366. [CrossRef]
26. Asaari, M.S.M.; Suandi, S.A.; Rosdi, B.A. Fusion of band limited phase Only correlation and width centroid contour distance for finger based biometrics. *Expert Syst. Appl.* **2014**, *41*, 3367–3382. [CrossRef]
27. Xie, S.J.; Yang, J.; Yoon, S.; Yu, L.; Park, D.S. Guided Gabor Filter for Finger Vein Pattern Extraction. In Proceedings of the 2012 Eighth International Conference on Signal Image Technology and Internet Based Systems, Naples, Italy, 25–29 November 2012; pp. 118–123.
28. Yang, G.; Xi, X.; Yin, Y. Finger Vein Recognition Based on $(2D)^2$ PCA and Metric Learning. *J. Biomed. Biotechnol.* **2012**, *2012*, 324249. [CrossRef]
29. Lu, Y.; Wu, S.; Fang, Z.; Xiong, N.; Yoon, S.; Park, D.S. Exploring finger vein based personal authentication for secure IoT. *Future Gener. Comput. Syst.* **2017**, *77*, 149–160. [CrossRef]
30. Lu, Y.; Yoon, S.; Xie, S.J.; Yang, J.; Wang, Z.; Park, D.S. Efficient descriptor of histogram of salient edge orientation map for finger vein recognition. *Appl. Opt.* **2014**, *53*, 4585–4593. [CrossRef]
31. Huang, H.; Liu, S.; Zheng, H.; Ni, L.; Zhang, Y.; Li, W. DeepVein: Novel finger vein verification methods based on Deep Convolutional Neural Networks. In Proceedings of the 2017 IEEE International Conference on Identity, Security and Behavior Analysis (ISBA), New Delhi, India, 22–24 February 2017; pp. 1–8.
32. Fairuz, S.; Habaebi, M.H.; Elsheikh, E.M.A. Finger Vein Identification Based On Transfer Learning of AlexNet. In Proceedings of the 2018 7th International Conference on Computer and Communication Engineering (ICCCE), Kuala Lumpur, Malaysia, 19–20 September 2018; pp. 465–469.
33. Gumusbas, D.; Yildirim, T.; Kocakulak, M.; Acir, N. Capsule Network for Finger-Vein-based Biometric Identification. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; pp. 437–441.
34. Hou, B.; Yan, R. Convolutional Autoencoder Model for Finger-Vein Verification. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 2067–2074. [CrossRef]

35. Zeng, J.; Wang, F.; Deng, J.; Qin, C.; Zhai, Y.; Gan, J.; Piuri, V. Finger Vein Verification Algorithm Based on Fully Convolutional Neural Network and Conditional Random Field. *IEEE Access* **2020**, *8*, 65402–65419. [CrossRef]
36. Choi, J.; Noh, K.J.; Cho, S.W.; Nam, S.H.; Owais, M.; Park, K.R. Modified Conditional Generative Adversarial Network-Based Optical Blur Restoration for Finger-Vein Recognition. *IEEE Access* **2020**, *8*, 16281–16301. [CrossRef]
37. Kuzu, R.S.; Piciucco, E.; Maiorana, E.; Campisi, P. On-the-Fly Finger-Vein-Based Biometric Recognition Using Deep Neural Networks. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 2641–2654. [CrossRef]
38. Ren, H.; Sun, L.; Guo, J.; Han, C.; Cao, Y. A high compatibility finger vein image quality assessment system based on deep learning. *Expert Syst. Appl.* **2022**, *196*, 116603.1–116603.12. [CrossRef]
39. Kuzu, R.S.; Maiorana, E.; Campisi, P. Vein-based Biometric Verification using Transfer Learning. In Proceedings of the 2020 43rd International Conference on Telecommunications and Signal Processing (TSP), Milan, Italy, 7–9 July 2020; pp. 403–409.
40. Ou, W.F.; Po, L.M.; Zhou, C.; Rehman, Y.A.U.; Xian, P.F.; Zhang, Y.J. Fusion loss and inter-class data augmentation for deep finger vein feature learning. *Expert Syst. Appl.* **2021**, *171*, 114584. [CrossRef]
41. Zhang, Z.; Tang, Z.; Wang, Y.; Zhang, Z.; Yan, S.; Wang, M. Compressed DenseNet for Lightweight Character Recognition. *arXiv* **2019**, arXiv:1912.07016.
42. He, Y. A New Lightweight DenseNet Based on Mix-Structure Convolution. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *790*, 012113. [CrossRef]
43. Fang, Y.; Wu, Q.; Kang, W. A novel finger vein verification system based on two-stream convolutional network learning. *Neurocomputing* **2018**, *290*, 100–107. [CrossRef]
44. Zhao, D.; Ma, H.; Yang, Z.; Li, J.; Tian, W. Finger vein recognition based on lightweight CNN combining center loss and dynamic regularization. *Infrared Phys. Technol.* **2020**, *105*, 103221. [CrossRef]
45. Shen, J.; Liu, N.; Xu, C.; Sun, H.; Xiao, Y.; Li, D.; Zhang, Y. Finger Vein Recognition Algorithm Based on Lightweight Deep Convolutional Neural Network. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–13. [CrossRef]
46. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
47. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely Connected Convolutional Networks. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 2261–2269.
48. Wan, K.; Min, S.J.; Ryoung, P.K. Multimodal Biometric Recognition Based on Convolutional Neural Network by the Fusion of Finger-Vein and Finger Shape Using Near-Infrared (NIR) Camera Sensor. *Sensors* **2018**, *18*, 2296.
49. Tang, S.; Zhou, S.; Kang, W.; Wu, Q.; Deng, F. Finger vein verification using a Siamese CNN. *IET Biom.* **2019**, *8*, 306–315. [CrossRef]
50. Shaheed, K.; Mao, A.; Qureshi, I.; Kumar, M.; Hussain, S.; Ullah, I.; Zhang, X. DS-CNN: A pre-trained Xception model based on depth-wise separable convolutional neural network for finger vein recognition. *Expert Syst. Appl.* **2022**, *191*, 116288. [CrossRef]
51. Song, J.M.; Kim, W.; Park, K.R. Finger-Vein Recognition Based on Deep DenseNet Using Composite Image. *IEEE Access* **2019**, *7*, 66845–66863. [CrossRef]
52. Noh, K.J.; Choi, J.; Hong, J.S.; Park, K.R. Finger-Vein Recognition Based on Densely Connected Convolutional Network Using Score-Level Fusion With Shape and Texture Images. *IEEE Access* **2020**, *8*, 96748–96766. [CrossRef]
53. Kuzu, R.S.; Maiorana, E.; Campisi, P. Vein-Based Biometric Verification Using Densely-Connected Convolutional Autoencoder. *IEEE Signal Process. Lett.* **2020**, *27*, 1869–1873. [CrossRef]
54. Ahmad Radzi, S.; Khalil-Hani, M.; Bakhteri, R. Finger-vein biometric identification using convolutional neural network. *Turk. J. Electr. Eng. Comput. Sci.* **2016**, *24*, 1863–1878. [CrossRef]
55. Lee, Y.H.; Khalil-Hani, M.; Bakhteri, R. FPGA-based finger vein biometric system with adaptive illumination for better image acquisition. In Proceedings of the 2012 International Symposium on Computer Applications and Industrial Electronics (ISCAIE), Kota Kinabalu, Malaysia, 3–4 December 2012; pp. 107–112.
56. Hong, H.G.; Lee, M.B.; Park, K.R. Convolutional Neural Network-Based Finger-Vein Recognition Using NIR Image Sensors. *Sensors* **2017**, *17*, 1297. [CrossRef]
57. Hu, H.; Kang, W.; Lu, Y.; Fang, Y.; Liu, H.; Zhao, J.; Deng, F. FV-Net: Learning a finger-vein feature representation based on a CNN. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; pp. 3489–3494.
58. Lu, Y.; Xie, S.; Wu, S. Exploring Competitive Features Using Deep Convolutional Neural Network for Finger Vein Recognition. *IEEE Access* **2019**, *7*, 35113–35123. [CrossRef]
59. Ton, B.T.; Veldhuis, R.N.J. A high quality finger vascular pattern dataset collected using a custom designed capturing device. In Proceedings of the 2013 International Conference on Biometrics (ICB), Madrid, Spain, 4–7 June 2013; pp. 1–5.
60. Jalilian, E.; Uhl, A. Finger-vein recognition using deep fully convolutional neural semantic segmentation networks: The impact of training data. In Proceedings of the 2018 IEEE International Workshop on Information Forensics and Security (WIFS), Hong Kong, China, 11–13 December 2018; pp. 1–8.
61. Yang, W.; Hui, C.; Chen, Z.; Xue, J.; Liao, Q. FV-GAN: Finger Vein Representation Using Generative Adversarial Networks. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 2512–2524. [CrossRef]
62. Hou, B.; Yan, R. Triplet-Classifier GAN for Finger-Vein Verification. *IEEE Trans. Instrum. Meas.* **2022**, *71*, 1–12. [CrossRef]

63. Das, R.; Piciucco, E.; Maiorana, E.; Campisi, P. Convolutional Neural Network for Finger-Vein-Based Biometric Identification. *IEEE Trans. Inf. Forensics Secur.* **2019**, *14*, 360–373. [CrossRef]
64. Xie, C.; Kumar, A. Finger vein identification using Convolutional Neural Network and supervised discrete hashing. *Pattern Recognit. Lett.* **2019**, *119*, 148–156. [CrossRef]
65. Hou, B.; Yan, R. ArcVein-Arccosine Center Loss for Finger Vein Verification. *IEEE Trans. Instrum. Meas.* **2021**, *70*, 1–11. [CrossRef]
66. Yeh, J.; Chan, H.T.; Hsia, C.H. ResNeXt with Cutout for Finger Vein Analysis. In Proceedings of the 2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Hualien, Taiwan, 16–19 November 2021; pp. 1–2.
67. Yang, W.; Huang, X.; Zhou, F.; Liao, Q. Comparative competitive coding for personal identification by using finger vein and finger dorsal texture fusion. *Inf. Sci.* **2014**, *268*, 20–32. [CrossRef]
68. Srivastava, R.K.; Greff, K.; Schmidhuber, J. Highway Networks. *arXiv* **2015**, arXiv:1505.00387.
69. Huang, G.; Sun, Y.; Liu, Z.; Sedra, D.; Weinberger, K.Q. Deep Networks with Stochastic Depth. In Proceedings of the Computer Vision—ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016; pp. 646–661.
70. Hu, H.; Dey, D.; Giorno, A.D.; Hebert, M.; Bagnell, J.A. Log-DenseNet: How to Sparsify a DenseNet. *arXiv* **2017**, arXiv:1711.00002.
71. Zhu, L.; Deng, R.; Maire, M.; Deng, Z.; Mori, G.; Tan, P. Sparsely Aggregated Convolutional Networks. In Proceedings of the Computer Vision—ECCV 2018, Munich, Germany, 8–14 September 2018; pp. 192–208.
72. Chao, P.; Kao, C.Y.; Ruan, Y.; Huang, C.H.; Lin, Y.L. HarDNet: A Low Memory Traffic Network. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3551–3560.
73. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
74. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861.
75. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 1800–1807.
76. Deng, J.; Guo, J.; Yang, J.; Xue, N.; Kotsia, I.; Zafeiriou, S. ArcFace: Additive Angular Margin Loss for Deep Face Recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 5962–5979.
77. Qiu, S.; Liu, Y.; Zhou, Y.; Huang, J.; Nie, Y. Finger-vein recognition based on dual-sliding window localization and pseudo-elliptical transformer. *Expert Syst. Appl.* **2016**, *64*, 618–632. [CrossRef]
78. Qin, H.; El-Yacoubi, M.A. Deep Representation-Based Feature Extraction and Recovering for Finger-Vein Verification. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1816–1829. [CrossRef]
79. Lu, X.; Tao, M.; Fu, X.; Gui, G.; Ohtsuki, T.; Sari, H. Lightweight Network Design Based on ResNet Structure for Modulation Recognition. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Norman, OK, USA, 27–30 September 2021; pp. 1–5.
80. Li, X.; Duan, C.; Yin, P.; Wang, N. Pedestrian Re-identity Based on ResNet Lightweight Network. *J. Phys. Conf. Ser.* **2021**, *2083*, 032087. [CrossRef]