GRU-based Neural architecture search for finger-Vein identification

Shaojiang Deng, Chao Fan, Yantao Li, Huafeng Qin, Mounim El Yacoubi, Gang Zhou

GRU-based Neural Architecture Search for Finger-Vein Identification

Shaojiang Deng
College of Computer Science
Chongqing University
Chongqing, China
sj_deng@cqu.edu.cn

Chao Fan
College of Computer Science
Chongqing University
Chongqing, China
202014021017@cqu.edu.cn

Yantao Li*
College of Computer Science
Chongqing University
Chongqing, China
yantaoli@cqu.edu.cn

Huafeng Qin
School of CS and Information Engineering
Chongqing Technology and Business University
Chongqing, China
qinhuafengfeng@163.com

Mounim A. El-Yacoubi
Télécom SudParis
Institut Polytechnique de Paris
Palaiseau, France
mounim.el_yacoubi@telecom-sudparis.eu

Gang Zhou
Computer Science Department
William & Mary
Williamsburg, USA
gzhou@cs.wm.edu

***Abstract*—In recent years, finger-vein biometrics has attracted extensive attention due to its potential for accurate and efficient identification. Deep neural networks (DNNs) have proven effective in automatically extracting discriminative features from large collections of finger-vein images, resulting in improving the accuracy and efficiency of finger-vein recognition. However, the DNN-based finger-vein recognition faces challenges. For example, designing network structures and tuning parameters rely on human prior knowledge, which can reduce both the effectiveness and efficiency of the recognition. To address these challenges, we propose GNAS-FV, a Gated recurrent unit-based Neural Architecture Search for Finger-Vein recognition, which automatically searches for the optimal network structure, reducing manual intervention and experience dependency, so as to improve the network performance and generalization ability. Specifically, we first provide a publicly available finger-vein database using commercial sensors to promote the development of finger-vein recognition. Then, a gated recurrent unit (GRU)-based neural architecture search approach is designed to automatically generate the network structure for the finger-vein image database. Next, we design a parameter-sharing supernet policy, which significantly reduces the search space and computation costs. Finally, the rigorous experiments are conducted on our finger-vein database and a public finger-vein database, and the experimental results demonstrate that the proposed GNAS-FV outperforms state-of-the-art methods in terms of recognition accuracy and equal error rate.**

***Index Terms*—Finger-vein identification, Deep learning, Neural architecture search (NAS), Gated recurrent unit (GRU), Accuracy**

I. Introduction

With the development and widespread applications of Internet technology, the information security has received more and more attention. Traditional identification methods, such as magnetic cards, keys, passwords, or personal identification numbers (PINs), have proven to be insufficiently secure because of their vulnerability to steal, copy and forge, as well as to various attack methods, such as smudge attacks [1] and shoulder-surfing attacks [2]. To address these shortcomings, biometric identification has been widely studied and has shown two advantages: 1) Convenience. As an individual's inherent modality, biometric traits can be automatically verified in less than one second; 2) Security. Biometric traits are difficult to copy and the users are required to participate in the image collection during the identification process. Biometric traits can be classified into physiological biometric traits and behavioral biometric traits. The behavioral biometric traits require highly-configured devices to capture motion patterns. By contrast, the physiological biometric traits are more easy to be collected and have been widely used in practical scenarios, such as door access and mobile payments, for identity or verification. Biometric identification mechanism has shown a potential candidate to replace traditional identification methods.

Physiological biometric traits can be divided into two categories: 1) extrinsic biometric traits, such as face [3], fingerprint [4] and iris [5], and 2) intrinsic biometric traits, such as palm vein [6] and finger vein [7]–[9]. Extrinsic biometric traits have been successfully applied in different scenarios, such as immigration clearance, financial payments, access control systems, and consumer electronic products. However, they are vulnerable to attack and may be copied without users' permission, which has raised concerns about security and privacy. In contrast, intrinsic biometric traits hidden in the human body are not easy to be observed by human eyes, and thus have two advantages [10], [11]: 1) They provide liveness verification. Vein patterns are only collected from living individuals, ensuring effective and efficient verification; 2) They offer high security and privacy protection. Blood vessels are hidden beneath the skin from birth, making them difficult to copy or forge in visible light. Therefore, finger-vein based recognition is receiving increasing attentions. Similar to facial recognition and fingerprint systems, most vein recognition systems follow the same mechanism: the physiological data is collected from individuals, a feature set is extracted from

*Yantao Li is the corresponding author.

the data, and a testing sample is matched with a template set for recognition. Currently, numerous researchers have explored image processing and classification approaches to improve performance of vein recognition.

During the process of finger-vein image acquisition, various factors, such as lighting [12], temperature [13], light scattering [14] and user behavior [13], can result in noise and irregular shadow regions in the captured vein images, ultimately degrading the accuracy of finger-vein verification. Therefore, how to achieve accurate and reliable finger-vein identification is still a challenging task. Currently, researchers have proposed various algorithms based on machine learning, deep learning and other technologies to improve the accuracy and robustness of finger-vein recognition, promoting the development of finger-vein recognition technology. Recently, neural architecture search (NAS) [15] can automatically search for an optimal neural network architecture by using the reinforcement learning technique, which has been widely applied in various fields, such as computer vision, natural language processing, and speech recognition. As the neural network architecture is determined by automatic search policy instead of prior knowledge, they can significantly improve the classification performance and efficiency. To improve the performance of finger-vein recognition, we present GNAS-FV, a Gated recurrent unit-based Neural Architecture Search for Finger-Vein image identification system. GNAS-FV employs the neural architecture search model to automatically generate the most suitable network structure for the current finger-vein classification task with a given dataset. To achieve this, GNAS-FV utilizes a parameter-sharing supernet structure that generates subnets for searching, which effectively reduces the search space and computation costs, resulting in the improvement of the search efficiency.

The main contributions of this work can be summarized as follows:

- We provide a publicly available finger-vein image database captured by commercial sensors, rather than experimental prototypes, to promote the development of finger-vein recognition. Using vein image captured by commercial sensors, which are more representative of daily usage, can facilitate more valuable results for finger-vein recognition.
- We present GNAS-FV, a gated recurrent unit-based neural structure search method for finger-vein recognition. The controller in GNAS-FV uses a gated recurrent unit-based structure to automatically search for the optimal neural network structure, which can generate the next candidate structure based on the current selection.
- We design a parameter-sharing supernet for finger-vein identification, where GNAS-FV combines the controller's structural descriptor output with a weight-sharing supernet to generate subnets, which significantly reduces the search space and computation costs. We then calculate the reward based on the subnet's performance on the validation set and update the parameters of the GNAS-FV controller iteratively, producing the optimal deep neural network architecture for finger-vein recognition.
- We conduct rigorous experiments on our finger-vein database and a public finger-vein database to evaluate the search efficiency of GNAS-FV, and the identification performance of the deep neural network generated by GNAS-FV. The experimental results on two finger-vein databases demonstrate that deep neural network generated by our GNAS-FV significantly outperforms existing deep learning based finger-vein recognition approaches in terms of the recognition accuracy and equal error rate.

The remainder of this paper is organized as follows. In Section II, we introduce the related work on finger-vein classification tasks. In Section III, we detail the proposed GNAS-FV, and the performance of GNAS-FV is evaluated in Section IV. Finally, Section V concludes this work.

II. Related Work

Finger-vein recognition algorithms refer to verifying or identifying individual identities through the acquired finger-vein patterns. Due to the unique and stable vein texture information contained in finger-vein images, finger-vein recognition shows the advantages of high accuracy, security, and robustness, and has been widely used in real-life scenarios, such as financial payments, border inspections, and access control systems. To improve the accuracy of finger-vein recognition, various methods have been proposed to extract strong and robust features for verification. These methods can be divided into three categories:

A. Local binary descriptors

Local binary descriptor-based approaches for finger-vein recognition include local statistical information based approaches and local invariance based approaches. Representative local statistical information based methods consist of Local Binary Pattern (LBP) [16] and Discriminative Binary Codes (DBC) [17]. The typical local invariance based methods contain the Scale-Invariant Feature Transform (SIFT) [18], which has been used to detect robust points for verification.

B. Traditional machine learning methods

Traditional machine learning models, such as k-means clustering [19], SVM [20], [21], PCA [22], 2D-PCA [23], and Sparse Representation (SR) [24], have been proposed to learn vein feature representation. To improve their performance, LRR (Low Rank Representation) [25] is used for discriminative feature extraction by adding a regularization term to constrain the low-rank coefficients and enhance the discriminative power of LRR.

C. Deep learning-based methods

Recently, deep learning-based methods, such as convolutional neural networks (CNN) and Transformer, have shown robust feature representation capability and have been successfully applied for various computer vision tasks. Inspired by their success, some researchers have introduced the deep learning to the vein verification tasks. ResNet [26], VGGNet [27], and GoogLeNet [28] as an typical examples of neural

network structures designed manually have performed well on many tasks. Specifically, the ResNet uses residual functions to alleviate the problems of gradient vanishing and expression degradation [26], VGGNet increases network depth to improve performance [27], and GoogLeNet employs Inception modules to enable the network to select convolution filters of different sizes [28]. In past years, the deep learning based models have been proposed for vein recognition. For example, FV-CNN [29] is a specially-designed CNN for finger-vein recognition. PV-CNN [30] proposes to use MSMD-GAN to generate augmented data for single-sample palm vein recognition. To improve the computation efficacy, LightweightDeepCNN [31] proposes a lightweight vein recognition algorithm that uses a triplet loss function to train the model. To exract the robust feature, the Arcvein [32] proposes a loss function called the cosine center loss to learn both inter-class and intra-class information, so as to improve the discriminating ability of CNNs for finger-vein verification. Recently, FVRASNet [33] proposes a lightweight CNN model that integrates recognition tasks and anti-spoofing tasks into a unified CNN model, achieving high security and strong real-time performance through the use of multi-task learning methods.

III. METHOD

In this section, we first introduce our GNAS-FV. Then, we describe how to generate a deep neural network by GNAS-FV. Next, we detail the subnet generation, controller design, and supernet design, respectively. Furthermore, we explain three stages of the GNAS-FV algorithm. Finally, we discuss the training parameters of GNAS-FV to ensure a comprehensive understanding of the algorithm.

A. GNAS-FV design

Deep neural networks (DNNs) have the capability to automatically learn effective features from a large number of finger-vein images for finger-vein recognition. However, applying DNNs to finger-vein recognition faces several challenges, such as how to design appropriate network structures and how to select and tune hyperparameters. Neural architecture search (NAS) as an effective method for exploring and optimizing network structures can automatically generate optimal DNNs to improve the finger-vein recognition accuracy. However, due to the characteristics of the finger-vein dataset, such as a small number of sample per user, multiple users, and finger-vein features that prioritize texture over common image classification, a supernet must be specifically designed for each finger vein dataset. This supernet is activated by the structural descriptor generated by the controller, and each branch inside the supernet is activated, generating a subnet. Moreover, this subnet shares the weights of the supernet, avoiding the need to train the subnet from scratch every time, which can greatly reduce the consumption of the time and memory.

The framework of the proposed GNAS-FV is illustrated in Fig. 1. As depicted in Fig. 1, GNAS-FV employs the controller to generate structural descriptors, and then uses such structural descriptors to activate the subnets in the supernet.

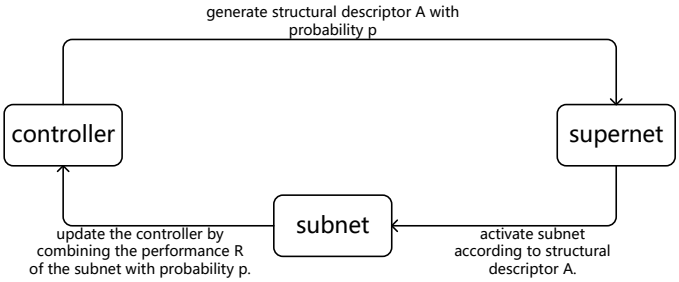

Fig. 1. Framework of GNAS-FV.

The accuracy of the subnet on the validation set is used as a reward signal to optimize the controller's parameters. During the training process, GNAS-FV uses the trained controller to generate structural descriptor to determine the DNN architecture for finger-vein recognition. Specifically, during the training phase of the supernet, the output of the controller is used as the probability p_1, and a set of structural descriptors A_1 are sampled with probability p_1. The supernet employs this structural descriptor A_1 to specify one of its subnets as the active network, and then trains itself based on the finger-vein data from the training set to update its parameters. During the training phase of the controller, the output of the controller is utilized as the probability p_2, and a set of structural descriptors A_2 are sampled with probability p_2. The supernet uses this structural descriptor A_2 to determine one of its subnets as the active network. Then, the accuracy of the current active network on the validation set is used as the reward R, so as to compute the gradient of p_2 and optimize the parameters of the controller. After training, the controller outputs the structural descriptor A' with the highest probability for each layer. The supernet uses such structural descriptor A' to extract a subnet as the generated deep neural network for finger-vein recognition.

B. Subnet Generation

In our approach, selecting an activated subnet from a supernet is treated as the process of choosing a path from the root node to a leaf node on a full N-ary tree. Each node on the path is associated with a specific structure of one layer. The GNAS-FV controller generates a structural descriptor that specifies the process of selecting one of its child nodes as the active node for each node on the path. The supernet structure requires each structural descriptor to make three decisions:

1) Decide whether to choose the direct path, which uses the output of the previous layer as the input for the next layer instead of using the output of the current layer;
2) When not choosing the direct path, decide on the type and parameters of the network structure in the current layer;
3) When not choosing the direct path, decide whether to use a shortcut path.

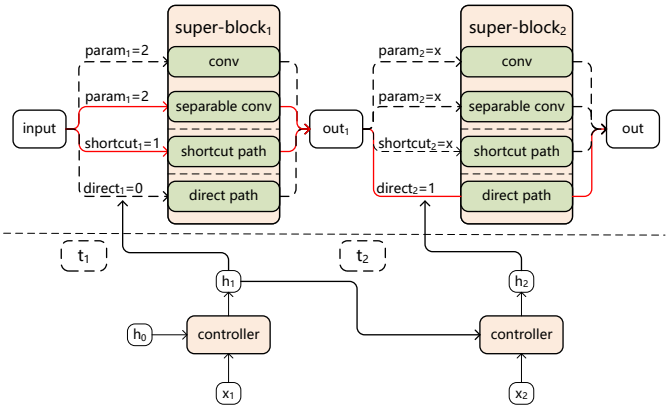

Fig. 2. Using GNAS-FV to generate a subnet from the supernet.

The three decisions are crucial for both the performance of the supernet and the effectiveness of the subnets. First, the direct path enables the GNAS-FV to autonomously decide whether to reduce the number of structural decisions and hence reduce the search space. Second, the choice of network structure type and parameters can directly impact the subnets' performance and efficiency, such as selecting depth, width, convolution kernel size, and other parameters. Finally, using the shortcut path can improve the information propagation in the network, alleviating the problem of the information loss and gradient vanishing, and improving the performance of deep networks. Therefore, when generating structural descriptors, the GNAS-FV controller needs to consider the interaction and balance of these decisions.

Fig. 2 illustrates how to generate a DNN using the controller and the supernet. This DNN contains two hidden layers, each of which includes two alternative network structures, e.g. conventional convolutional block and separable convolutional block.

1) At time t_1, the controller takes a seed vector x_1 corresponding to the structural descriptor of the first layer and the initial state vector h_0 as the inputs, and then generates the state vector h_1. h_1 is then mapped to a probability tuple (a_{1_1}, a_{2_1}, a_{3_1}) of structural descriptors by a non-linear fully connected layer. Based on this probability tuple, the controller generates the structural descriptor ($direct_1$, $param_1$, $shortcut_1$) for the first layer by randomly sampling according to the probabilities. Here, $direct_1 = 0$ indicates that the direct path is not selected, $param_1 = 2$ indicates the selection of the second type of network structure, and $shortcut_1 = 1$ indicates the use of a shortcut path. Therefore, in this super-block, the separable convolution block and shortcut path of this layer are activated as the passageway for data transmission.
2) At time t_2, the controller takes the seed vector x_2 and the previous layer's state vector h_1 corresponding to the first layer's network structural descriptor as its inputs. It generates the state vector h_2, which is then mapped to a probability tuple (a_{1_2}, a_{2_2}, a_{3_2}) by a non-linear fully connected layer. Using this probability tuple, the controller generates the second layer's structural descriptor ($direct_2$, $param_2$, $shortcut_2$) in the same way as the first layer. In this case, $direct_2 = 1$ indicates that the direct path is chosen. As a result, the input of the first layer is directly used as the output of the second layer, and the values of $param_2$ and $shortcut_2$ are not used. Thus, in the supernet, the direct path of the second layer is activated as the path for data propagation.

Finally, after two iterations, the controller generates two sets of structural descriptors and selects different structures as data paths in the supernet, resulting in a DNN to be evaluated. This controller-supernet-based search method can flexibly generate various neural network structures, which are automatically optimized for different tasks and datasets.

C. Controller Design

In NAS, the controller plays a crucial role in generating candidate network structures and tuning its own parameters based on feedback reward signals. Typically, a recurrent neural network (RNN) is used as the controller, and the gated recurrent unit (GRU) is a common variant of the RNN. The GRU has two important gates: the update gate and the reset gate. The update gate controls how much information from the previous hidden state is retained in the current time step, while the reset gate determines how much information from the previous hidden state is used to compute the candidate hidden state at the current time step. The gate mechanism of GRU can effectively solve the problems of gradient vanishing and exploding, thereby enabling the network to effectively process long sequence data. Furthermore, compared to other RNN-based structures such as LSTM, GRU has fewer parameters and faster computation speed, so it has been widely employed in many practical applications.

The controller of GNAS-FV is a GRU structure that can generate the next candidate structure based on the current structure selection. Thus, the model can automatically search for the optimal neural network structure without human intervention. Specifically, the controller of GNAS-FV consists of two stacked GRUs and a classifier group, which can transform the hidden state into a tuple of probability vectors. The tuple contains three elements: $direct$, $param$, and $shortcut$, which represent the probability distribution of direct path, network structure type and parameters, and shortcut path, respectively. Due to the different dimensions of these three elements, the classifier group adopts different processing methods for each element. For $direct$ and $shortcut$, fully connected layers and sigmoid activation function are used, while for $param$, the fully connected layers and softmax layer are used.

In GNAS-FV, the controller takes a seed vector x_t and the previous hidden state h_{t-1} as its inputs at time t, and generates a new hidden state h_t. The classifier group then transforms h_t into a probability vector tuple for the network structure of the

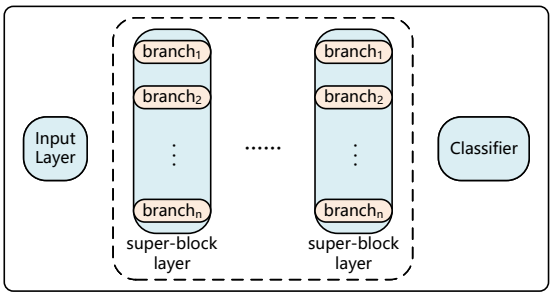

Fig. 3. Architecture of the supernet.

TABLE I
OPTIONAL STRUCTURE OF SUPER-BLOCK LAYER

Structure	Kernel size	Using shortcut
conv-relu-norm	(3,3)	✓
conv-relu-norm	(3,3)	×
conv-relu-norm	(5,5)	✓
conv-relu-norm	(5,5)	×
separable conv-relu-norm	(3,3)	✓
separable conv-relu-norm	(3,3)	×
separable conv-relu-norm	(5,5)	✓
separable conv-relu-norm	(5,5)	×
average pool	(2,2)	✓
average pool	(2,2)	×
max pool	(2,2)	✓
max pool	(2,2)	×
direct path	-	✓

tth layer. The process is repeated recursively by passing h_t as input to GNAS-FV for time step $t+1$.

D. Supernet Design

Traditional NAS methods usually use a controller to generate structural descriptors, followed by creating and training a new DNN. However, this approach has the drawbacks of consuming a significant amount of time and hardware resources for each search. As a result, traditional NAS methods often struggle to scale to large-scale network search tasks in practical applications. In contrast, GNAS-FV employs a supernet-based method to search for optimal subnet. In this method, the controller generates the structural descriptors to determine the structure of the subnet, and the supernet activates the sub-nodes layer by layer based on these descriptors to form a subnet. In this way, this method enables GNAS-FV to avoid creating and training a new DNN for each search, resulting in the reduction of time cost.

As depicted in Fig. 3, the supernet architecture in GNAS-FV includes three modules. The first module is the input layer, which uses a 3×3 convolutional structure and a batch normalization layer to map the input to a 32-channel high-dimensional space for subsequent processing. The second module consists of 10 super-block layers, each comprising N nodes representing N selectable network structures. At runtime, GNAS-FV activates only one node as the active node for the current layer. In this work, N is set to 13, indicating 13 selectable network structures. The selectable structures of the nodes in the super-block layer are listed in Table I. Finally, the third module includes a classifier layer, which takes the feature maps extracted from the super-block layer as the input and outputs the probability of the input belonging to each class.

As the first module of the entire network, the input layer aims to extract rough feature and reduce the dimension of the input. The super-block layers are the core of the supernet. Each node in the super-block layer represents a candidate network structure, and the architecture of the entire network is determined by selecting these nodes. The classifier layer takes the features extracted by the super-block layer as the input and predict its probability belonged to each class, achieving the final classification task. Overall, GNAS-FV uses the designed supernet to automatically determine a optimal network structure for a specific task by the node selection, which provides the flexibility and adaptability for different finger-vein recognition task.

E. Three Stages

GNAS-FV consists of two main blocks: the controller and the supernet, which associate with some trainable parameters, respectively. The parameters of the controller and the supernet are denoted as θ and ω, respectively, which are shared among the sub-networks. The training process of GNAS-FV can be divided into three stages:

1) Warm-up stage: In this stage, the parameters of the controller θ are fixed, while the parameters of the supernet ω are trained. During this stage, the controller generates a set of random network structural descriptors based on the parameter θ to specify the subnet in the supernet. Then, the activated subnet in the supernet are trained with the data in the training set.
2) Main training stage: The main training stage consists of two sub-phases, which alternate between the training of the parameters of the controller θ and the parameters of the supernet ω:
   - Train the parameters of the supernet ω, while keeping the parameters of the controller θ fixed. At each training step, the controller generates a set of architecture descriptors, selects and activates a subnet in the supernet. Then, the parameters of the supernet ω are updated using the data from the training set.
   - Train the parameter of the controller θ, while keeping the parameters of the supernet ω fixed. At each training step, the controller generates a set of structural descriptors to select and activate a subnet in the supernet. The selected subnet is then evaluated by taking the accuracy on the validation set as the reward R. After accumulating M evaluations, the mean gradient of the last M evaluations is computed and backpropagated to update the parameters of the controller θ.

3) Generation stage: This is the last stage of the training, indicating the parameters of the controller θ have converged. In this stage, the optimal structural descriptor generated by the controller is used to select a subnet from the supernet, and a DNN is created based on the parameters and structure of this subnet. Such network is trained on both the training and validation sets until convergence. After training, the resulting model is employed for finger-vein recognition.

F. GNAS-FV Training

Cross-entropy can be used to compute the loss and optimize the parameters of the supernet ω to generate weights that are suitable for the subnet. Specifically, the supernet can activate the subnet corresponding to its internal structure, which performs the finger-vein recognition task, given a structural descriptor and an input finger-vein image. Then, the subnet calculates the loss and gradient through forward and backward propagation on the target task, which can be propagated to the supernet through the chain rule, so to update ω.

The controller generates a sequence representing a candidate network structure $(a_1, a_2, ..., a_N)$. This candidate structure is trained and evaluated on a certain task, obtaining a reward signal, such as the accuracy on the validation set. Then, this reward signal is used to update the parameters of the controller, enabling it to generate better network structures. However, there is no functional relationship between the reward and the parameters of the controller θ, so the gradient cannot be propagated through the chain rule. To solve this problem, the REINFORCE [34], a policy gradient algorithm, can be used to maximize the expected reward of the controller generating network structures. Specifically, the formula for updating the controller using the REINFORCE algorithm is defined as Eq. (1):

$$\theta \leftarrow \theta + \alpha \frac{1}{M} \sum_{m=1}^{M} \sum_{n=1}^{N} \nabla_\theta log P(a_n | a_{(n-1):1}; \theta)(r_m - b), \quad (1)$$

where θ are the parameters of the controller, α is the learning rate, N is the number of layers in the generated network structure, and b is the moving average of the reward r_m over a certain period of time. The term $P(a_n | a_{(n-1):1}; \theta_c)$ represents the conditional probability of the controller selecting network structure a_n at the nth layer given the previous structures $a_{(n-1):1}$. M controls the training frequency of the controller, which indicates that the gradient is accumulated over a period of M runs and the controller's parameters are updated by averaging the accumulated gradients.

IV. Performance evaluation

To evaluate the performance of GNAS-FV, we conduct rigorous experiments on our database and a public finger-vein database, using the PyTorch deep learning framework. The experiments are performed on a high-performance computer with an Intel-10900X 3.7 GHz processor with 10 cores and 20 threads, 128GB memory, and NVIDIA 3090 graphics card. In the experiments, GNAS-FV is compared with classical neural network models, e.g. ResNet, VGGNet, and GoogLeNet, as well as the state-of-the-art models, e.g. FV-CNN, PV-CNN, LightweightDeepCNN, Arcvein, and FVRASNet, respectively.

TABLE II
Accuracy (%) Comparison with representative classifiers on Testing sets

Model	Database A	Database B
ResNet	96.21	93.76
VGGNet	91.21	92.57
GoogLeNet	96.21	95.84
FV-CNN	93.42	92.47
PV-CNN	96.25	96.53
LightweightDeepCNN	95.08	97.22
Arcvein	92.58	94.15
FVRASNet	94.13	95.24
GNAS-FV	**96.71**	**97.62**

A. Database

We evaluate the performance of GNAS-FV on our finger-vein database and a public finger-vein database [35]. In our experiments, we divide them into three subsets: a training set, a validation set, and a testing set, respectively.

1) Database A: Currently, most existing works are tested based on finger-vein images collected through device prototypes rather than commercial sensors, so it is difficult to fully evaluate the effectiveness of finger-vein recognition methods in practical applications. However, there is currently no publicly-available finger-vein database collected by the commercial sensors, which limits the development of the recognition methods. To address this issue, we construct a finger-vein database using commercial sensors, comprising 6,000 images from 100 subjects (66 males and 34 females). Each participant provides six fingers, e.g. the index, middle, and ring fingers of both hands. The database is split into a training set with 2,520 images, a validation set with 1,080 images, and a testing set with 2,400 images.

2) Database B: The Hong Kong Polytechnic University finger-vein image database (HKPU dataset) is collected from 156 participants using a non-contact imaging device. The first 105 participants provide 2,520 images (105×2 fingers $\times 6$ images $\times$ 2 sessions), obtained at two different times. Each participant provides six images for each of their index and middle fingers at each session, resulting in a total of 24 images (6 images $\times 2$ fingers $\times 2$ sessions) for two sessions. The remaining 51 participants contribute only 612 images, collected only in the first session. In our experiments, only the first 2,520 images captured at two sessions are used to evaluate the performance of GNAS-FV. The training set consists of 1,058 images, the validation set contains 453 images, and the testing set includes 1,009 images.

B. Experimental setup

We employ the stochastic gradient descent and the validation set to optimize the controller, with a learning rate of 0.05, momentum parameter of 0.9, and weight decay of 2.5×10^{-4}. For the supernet, we use the Adam optimizer and the training

set to optimize its parameters, with the learning rate of 0.001. In the training stage, the supernet is trained using 10 batches of training data for each epoch, and the controller is trained 50 times per epoch. During each training of the controller, 10 structural descriptors are generated and their performance is evaluated on the validation set to update the parameters of the controller. GNAS-FV produces a DNN after 200 training epochs in the training stage. Such DNN, along with other comparative models, is trained for 1,000 epochs on the training set with a batch size of 32. The model with best performance on the validation set is then evaluated on the testing set.

C. Search Efficiency

To train the supernet, we use the training sets of Database A and Database B, which contain 2,520 and 1,058 images, respectively. The supernet model is trained for 200 epochs, with 10 batches of data per epoch, and a batch size of 32 on the training set. Therefore, in the main training stage, the training set of Database A is used approximately 25 times, while the training set of Database B is used approximately 61 times.

In order to continually improve the generated subnet structures, the controller requires to update its parameters. Therefore, we evaluate the performance of 10 structures generated by the controller, and use their accuracy on the validation set as a reward to update the parameters of the controller. This process is repeated 50 times, resulting in the generation and evaluation of 10^5 structures, but the supernet does not require any additional training. However, each generated structure needs to build and train a new subnet without the supernet strategy. Even if we only use one complete training set data for the initial screening, and ignore the time and resource consumption for evaluating on the validation set, the entire process still requires building, training, and evaluating subnet models 10^5 times. This means that the training sets of both Database A and Database B are reused 10^5 times, which results in about 3,937 times and 1,653 times more than the supernet strategy, respectively.

D. GNAS-FV Performance

To evaluate the performance of GNAS-FV, we repeat representative classifiers: the classical neural network models such as ResNet, VGGNet and GoogLeNet, and the state-of-the-art models including FV-CNN, PV-CNN, LightweightDeepCNN, Arcvein, and FVRASNet, respectively, for comparison.

We evaluate the performance of the proposed GNAS-FV on two finger-vein datasets. The accuracy of representative methods on testing set is shown in Table II. The experimental results demonstrate that GNAS-FV achieves the highest accuracy of 96.71% and 97.62% on Databases A and B, respectively. The corresponding EERs on testing set are illustrated in Table III, Fig. 4 and Fig. 5, respectively. The experimental results imply that GNAS-FV shows the lowest EERs, e.g. 1.67% and 0.89% on Databases A and B, respectively. These results indicate that GNAS-FV outperforms classical and advanced neural network models in terms of improving recognition performance, e.g. identification accuracy and equal error rate (EER).

TABLE III
EER (%) COMPARISON WITH REPRESENTATIVE CLASSIFIERS ON TESTING SETS

Model	Database A	Database B
ResNet	2.25	2.48
VGGNet	5.24	3.37
GoogleNet	1.83	1.98
FV-CNN	3.21	4.16
PV-CNN	1.75	1.78
LightweightDeepCNN	2.21	1.28
Arcvein	2.92	2.78
FVRASNet	2.83	1.98
GNAS-FV	**1.67**	**0.89**

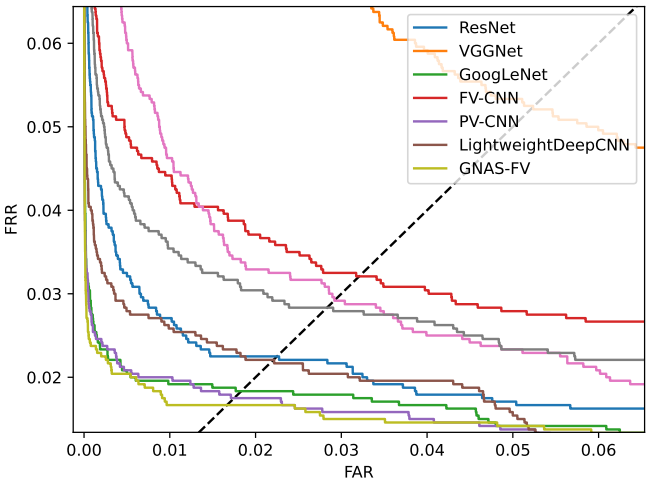

Fig. 4. ROC of Representative Classifiers on Testing Set of Database A.

V. CONCLUSION

We propose GNAS-FV, a novel GRU-based neural architecture search for finger-vein recognition in this paper. GNAS-FV uses a supernet approach to automatically search for the optimal network architecture and achieves the state-of-the-art performance on two finger-vein datasets. In experiments, we compare our GNAS-FV with several classic and advanced neural network models, and the experimental results demonstrate that our approach outperforms existing approaches in terms of improving the accuracy and EER.

ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 62072061 and 61976030), the Fellowship of China Postdoctoral Science Foundation (Grant No. 59676651E), the Funds for Creative Research Groups of Chongqing Municipal Education Commission (Grant No. CXQT21034), and the Scientific and Technological Research Program of Chongqing Municipal Education Commission (Grant Nos. KJQN201900848 and KJQN201500814).

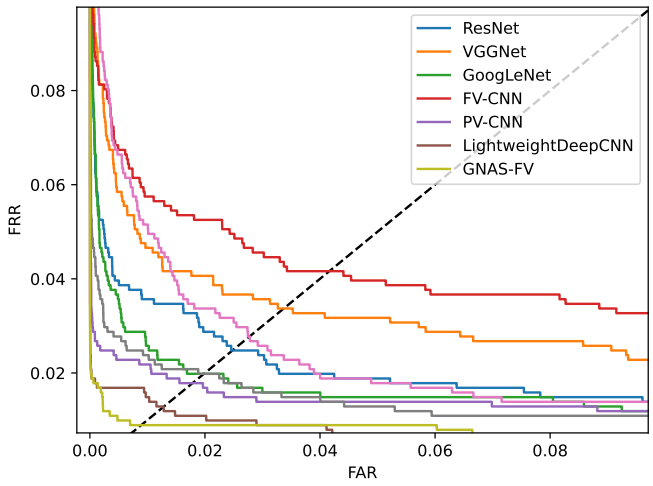

Fig. 5. ROC of Representative Classifiers on Testing Set of Database B.

REFERENCES

[1] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proceedings of the 4th USENIX Conference on Offensive Technologies (WOOT)*, 2010, pp. 1–7.

[2] S. Wiedenbeck, J. Waters, L. Sobrado, and J.-C. Birget, "Design and evaluation of a shoulder-surfing resistant graphical password scheme," in *Proceedings of the Working Conference on Advanced Visual Interfaces (AVI)*, 2006, pp. 177–184.

[3] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings of 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1991, pp. 586–591.

[4] A. Jain, L. Hong, and R. Bolle, "On-line fingerprint verification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 4, pp. 302–314, 1997.

[5] J. Daugman, "How iris recognition works," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 1, pp. 21–30, 2004.

[6] H. Qin, M. A. El-Yacoubi, Y. Li, and C. Liu, "Multi-scale and multi-direction gan for cnn-based single palm-vein identification," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2652–2666, 2021.

[7] H. Qin and M. A. El-Yacoubi, "Deep representation-based feature extraction and recovering for finger-vein verification," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 8, pp. 1816–1829, 2017.

[8] J. Wang, G. Wang, and M. Zhou, "Bimodal vein data mining via cross-selected-domain knowledge transfer," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 3, pp. 733–74, 2017.

[9] I. S. Wang, H.-T. Chan, and C.-H. Hsia, "Finger-vein recognition using a nasnet with a cutout," in *Proceedings of 2021 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, 2021, pp. 1–2.

[10] T. Tanaka and N. Kubo, "Biometric authentication by hand vein patterns," in *SICE 2004 Annual Conference*, vol. 1, 2004, pp. 249–253.

[11] Y. Li, S. Ruan, H. Qin, S. Deng, and M. A. El-Yacoubi, "Transformer based defense gan against palm-vein adversarial attacks," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1509–1523, 2023.

[12] B. Huang, Y. Dai, R. Li, D. Tang, and W. Li, "Finger-vein authentication based on wide line detector and pattern normalization," in *Proceedings of 2010 International Conference on Pattern Recognition (ICPR)*, 2010, pp. 1269–1272.

[13] A. Kumar and Y. Zhou, "Human identification using finger images," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2228–2244, 2012.

[14] E. C. Lee and K. R. Park, "Image restoration of skin scattering and optical blurring for finger vein recognition," *Optics and Lasers in Engineering*, vol. 49, no. 7, pp. 816–828, 2011.

[15] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[16] B. Jun and D. Kim, "Robust face detection using local gradient patterns and evidence accumulation," *Pattern Recognition*, vol. 45, no. 9, pp. 3304–3316, 2012.

[17] X. Xi, L. Yang, and Y. Yin, "Learning discriminative binary codes for finger vein recognition," *Pattern Recognition*, vol. 66, pp. 26–33, 2017.

[18] B. Prommegger and A. Uhl, "Rotation invariant finger vein recognition," in *2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 2019, pp. 1–9.

[19] K. Su, G. Yang, L. Yang, and Y. Yin, "Finger vein image retrieval via affinity-preserving k-means hashing," in *2017 IEEE International Joint Conference on Biometrics (IJCB)*, 2017, pp. 375–382.

[20] K. Kapoor, S. Rani, M. Kumar, V. Chopra, and G. S. Brar, "Hybrid local phase quantization and grey wolf optimization based svm for finger vein recognition," *Multimedia Tools and Applications*, vol. 80, no. 10, pp. 15 233–15 271, 2021.

[21] H. Qin, C. Gong, Y. Li, X. Gao, and M. A. El-Yacoubi, "Label enhancement-based multiscale transformer for palm-vein recognition," *IEEE Transactions on Instrumentation and Measurement*, vol. 72, pp. 1–17, 2023.

[22] J.-D. Wu and C.-T. Liu, "Finger-vein pattern identification using principal component analysis and the neural network technique," *Expert Systems with Applications*, vol. 38, no. 5, pp. 5423–5427, 2011.

[23] G. Yang, X. Xi, and Y. Yin, "Finger vein recognition based on (2d) 2 pca and metric learning," *Journal of Biomedicine and Biotechnology*, vol. 2012, pp. 1–9, 2012.

[24] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, "Sparse representation for computer vision and pattern recognition," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.

[25] L. Yang, G. Yang, K. Wang, F. Hao, and Y. Yin, "Finger vein recognition via sparse reconstruction error constrained low-rank representation," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4869–4881, 2021.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[27] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *Proceedings of 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, pp. 730–734.

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[29] R. Das, E. Piciucco, E. Maiorana, and P. Campisi, "Convolutional neural network for finger-vein-based biometric identification," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 360–373, 2019.

[30] H. Qin, M. A. El-Yacoubi, Y. Li, and C. Liu, "Multi-scale and multi-direction gan for cnn-based single palm-vein identification," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 2652–2666, 2021.

[31] J. Shen, N. Liu, C. Xu, H. Sun, Y. Xiao, D. Li, and Y. Zhang, "Finger vein recognition algorithm based on lightweight deep convolutional neural network," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–13, 2022.

[32] B. Hou and R. Yan, "Arcvein-arccosine center loss for finger vein verification," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–11, 2021.

[33] W. Yang, W. Luo, W. Kang, Z. Huang, and Q. Wu, "Fvras-net: An embedded finger-vein recognition and antispoofing system using a unified cnn," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 11, pp. 8690–8701, 2020.

[34] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3, pp. 229–256, 1992.

[35] 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.