PocketNet: Extreme Lightweight Face Recognition Network using Neural Architecture Search and Multi-Step Knowledge Distillation

Fadi Boutros[1,2], Patrick Siebke [1], Marcel Klemt [1],
Naser Damer[1,2], Florian Kirchbuchner[1], Arjan Kuijper[1,2]
[1]Fraunhofer Institute for Computer Graphics Research IGD, Darmstadt, Germany
[2]Mathematical and Applied Visual Computing, TU Darmstadt, Darmstadt, Germany
Email: fadi.boutros@igd.fraunhofer.de

Abstract

Deep neural networks have rapidly become the mainstream method for face recognition (FR). However, this limits the deployment of such models that contain an extremely large number of parameters to embedded and low-end devices. In this work, we present an extremely lightweight and accurate FR solution, namely PocketNet. We utilize neural architecture search to develop a new family of lightweight face-specific architectures. We additionally propose a novel training paradigm based on knowledge distillation (KD), the multi-step KD, where the knowledge is distilled from the teacher model to the student model at different stages of the training maturity. We conduct a detailed ablation study proving both, the sanity of using NAS for the specific task of FR rather than general object classification, and the benefits of our proposed multi-step KD. We present an extensive experimental evaluation and comparisons with the state-of-the-art (SOTA) compact FR models on nine different benchmarks including large-scale evaluation benchmarks such as IJB-B, IJB-C, and MegaFace. PocketNets have consistently advanced the SOTA FR performance on nine mainstream benchmarks when considering the same level of model compactness. With 0.92M parameters, our smallest network PocketNetS-128 achieved very competitive results to recent SOTA compacted models that contain up to 4M parameters. Training codes and pre-trained models are made public. [1]

1. Introduction

Face recognition is an active research field, and it has benefited from the recent advancements in machine learning, especially the advancements in deep learning [9] and the novelty of margin-based Softmax losses [5, 31], achieving a notable recognition accuracy. SOTA FR solutions rely on a deep learning models with an extremely large number of parameters [5, 24]. Deploying such models on embed-

---
[1] https://github.com/fdbtrs/PocketNet

ded devices or in applications with limited memory specifications is a major challenge [22, 6]. This challenge has received increased attention in the literature in the last few years [22, 6].

Recently, several compact FR models have been proposed in the literature. MobileFaceNet [3] proposed an efficient FR model based on MobileNetV2 [28] with around 1M parameters. ShuffleFaceNet [21] and VarGFaceNet [35] model architectures adopted ShuffleNetV2 [19] and VarGNet [38], respectively, for the FR task. VarGFaceNet contains 5M parameters. ShuffleFaceNet presented three architectures with different width scales (0.5, 1.5 and 2) containing 0.5, 2.6, and 4.5M parameters, respectively. MixFaceNets [1] use MixNets [30] as a baseline network structure to develop a new family of FR models. The smallest MixFaceNet architecture contains 1.04M parameters and the largest one contains 3.95M parameters. Martinez-Diaz et al. [22] evaluated the computational requirements and the verification performance of five compact model architectures including MobileFaceNet (2.0M parameters), VarGFaceNet [35] (5M parameters), ShufeFaceNet [21] (2.6m parameters), MobileFaceNetV1 (3.4m parameters), and ProxylessFaceNAS (3.2m parameters). The reported results by Martinez-Diaz et al. [22] demonstrated that compact FR models can still achieve high accuracies for FR.

However, none of these works [1, 3, 35, 21, 22] designed a network specifically for the FR task, rather than adopting existing architectures designed for common computer vision tasks. With the developments in AutoML, Neural Architecture Search (NAS) has shown SOTA performances in many computer vision tasks [16, 34]. NAS aims at automating the neural network architecture design achieving higher performances than the handcraft-designed network architectures. One of the early works of NAS was introduced by Zoph *et al*. [42]. That work [42] proposed that the architecture of a neural network can be described as a variable-length string. Thus, a Recurrent neural network (RNN) can be used as a controller to generate such a string. While this method showed competitive results in compar-

ison to SOTA models, it requires a very long search time (22,400 GPU days [42]). NASNet [43] points out that a convolutional neural network (CNN) such as ResNet [9] is a repetition of modules that consist of combinations of convolution operations. Based on that, they introduced a new search space, called NASNet. NASNet proposed to learn the network building block (cell), rather than learning the whole architecture. The network architecture, in this case, is constructed by stacking these cells together. NASNet was able to reduce the search time to 2,000 GPU days, in comparison to 22,400 GPU days needed by NAS [42]. ProxylessNAS [2] directly learned the architectures for the target task. It trained an over-parameterized network by gradient optimization that contained all candidate paths and pruned redundant paths to achieve a compact architecture. The architecture search needed 200 GPU hours, which was much faster than NASNet [43]. Differentiable Architecture Search (DARTS) [16] relaxed the discrete search space in a continuous manner. DARTS proposed to use gradient optimization to optimize the architecture search space. Similar to NASNet, DARTS proposed to learn the main building block (cell) of the network architecture rather than learning the entire network architecture. DARTS search algorithm requires around 1.5 GPU days, which was orders of magnitude faster than NASNet [43]. To reduce the NAS search time, all mentioned NAS algorithms proposed to learn from small training datasets such as CIFAR-10 [15] and then utilized the discovered architecture to train on larger datasets such as ImageNet [4]. This advancement in NAS solutions has only recently captured the attention of biometric recognition solutions [41, 32], however, with no deployments towards lightweight or embedded architectures.

In this work, we successfully aim at intelligently designing and training a family of lightweight FR models, namely the PocketNets, that offer the SOTA trade-off between model compactness and performance. To achieve that, we focus on two aspects, the first is the use of a NAS algorithm to learn an FR-specific lightweight network architecture, and the second is to design a novel knowledge distillation (KD) paradigm to relax training difficulties raised by the substantial discrepancy between teacher and student models. We use CASIA-WebFace (500K images) [36] to learn the optimal architecture using DARTS [16]. We additionally propose a novel training paradigm based on KD, namely multi-step KD, to enable transferring the knowledge of the teacher network at different stages of the training process, and thus enhance the verification performance of the compact student model. We prove our face-specific NAS-based architecture and the proposed multi-step KD in two detailed ablation studies. First, we experimentally evaluate the impact of the NAS training dataset source (face vs. general image classes) on the FR performance of the learned architecture. Second, we experimentally proved and analyzed

the competence of our proposed multi-step KD on improving FR performance in comparison to the baseline KD solutions, as well as training without KD. To experimentally demonstrate the competence of our proposed PocketNets, we report their FR performance on nine different benchmarks, in comparison to the recent SOTA compact models, in terms of FR performance and model compactness. In a detailed comparison, different versions of our PocketNets scored SOTA performances in both, under 1M parameters and under 2M parameters, FR model categories. Moreover, PocketNets achieved very competitive results to much larger FR models, and even outperformed them in many cases.

2. Methodology

This section presents the methodology leading to our proposed PocketNets solution, both the architecture design and the training paradigm. We first present the NAS process leading to the architecture of our proposed PocketNets. Then, we present our proposed multi-step knowledge distillation training paradigm.

2.1. Towards PocketNet Architecture

Neural architecture search (NAS) automates the network design by learning the network architecture that achieves the best performance for a specific task. NAS has proved to be a robust method in discovering and optimizing neural network architecture. Previous works [2, 16] demonstrate that the discovered network architectures by NAS do outperform handcraft-designed network architectures for different computer vision tasks. For our PocketNets, we opt to use differential architecture search (DARTS) algorithm [16] to search for two types of building blocks (cell) i.e. normal cell and reduce cell, which can be stacked to form the final architecture. Our choice for DARTS is based on: a) it achieved a competitive result to the SOTA NAS solutions on different image classification tasks [16], and b) the search time for DARTS is feasible in comparison to other search methods [42, 43] and thus, it can be adapted to a large-scale dataset. Unlike common NAS algorithms that are applied on a small image size of a small dataset, our NAS will be learned on a large-scale face image dataset with relatively high resolution. In the following, we briefly present the DARTS algorithm. Our goal here is not only to build an optimal architecture, but also to analyze the FR performance implications when optimizing such an architecture on a different learning task, as will be clarified later in this work.

DARTS aims at learning two types of cells: normal cell and reduce cell. Each cell is a direct acyclic graph (DAG) that consist of N nodes. Each node x_i is a latent representation, where $i \in [0, N]$. The operation space O is a set of candidate operation e.g. convolutional layer, skip-connection, pooling layer etc. Each edge (i, j) be-

2

tween node x_i and x_j is a candidate operation $o^{(i,j)} \in O$ that applies a particular transformation on x_i. Each candidate operation o is weighted by the architecture parameter $a(i,j)$. An intermediate node x_j is calculated as $x_j = \sum_{i<j, i \in [0,N]} o^{(i,j)}(x_i)$. Each cell (DAG) has two input nodes and a single output node. The two input nodes are the output of the previous two cells of the network. The output of the last node x_{N-1} i.e. the cell output, is a concatenation of all nodes in the DAG excluding the input nodes. The candidate operation applied to $x^{(i)}$ is represented as a function $o(.)$. The choice of a candidate operation is formulated by applying a Softmax function over the weights of all possible operations O:

$$\bar{o}^{(i,j)}(x) = \sum_{o \in O} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o' \in O} \exp(\alpha_{o'^{(i,j)}})} o(x), \qquad (1)$$

where $\alpha_o^{(i,j)}$ is a network architecture weight parameter of a candidate operation o. Therefore, the architecture search becomes a task of learning a set of parameters $\alpha = \{\alpha^{(i,j)}\}$. The learning procedure of DARTS is based on jointly learning the network architecture represented by α and the network weights w. Given L_{train} and L_{val} as the train and validation loss, respectively. The learning objective of DARTS is to find the optimal architecture represented by α^* that minimizes the validation loss $L_{val}(w^*, \alpha^*)$ with $w^* = \arg\min_w L_{train}(w, \alpha^*)$ as the best performing network weights on the training set. The architecture parameters are learned using a bi-level optimization problem with α as the upper-level and w the lower level variable:

$$\min_\alpha L_{val}(w^*(\alpha), \alpha)$$
$$s.t. w^*(\alpha) = \arg\min_w L_{train}(w, \alpha). \qquad (2)$$

The final discrete architecture is derived by setting $o^{(i,j)} = argmax_{o \in O} \alpha_o^{(i,j)}$. Given an input of the shape $w \times h \times c$, the output of the reduction cell is $w/2 \times h/2 \times 2c$ and the output of the normal cell is $w \times h \times c$. The first two nodes of cell k represent the output of the two previous cells $k-1$ and $k-2$.

Search space: PocketNet search space includes the following operations: 1) $3 \times 3, 5 \times 5, 7 \times 7$ depthwise separable convolutions [11] with kernel size of $\{3 \times 3, 5 \times 5, 7 \times 7\}$, padding of $\{1, 2, 3\}$ to preserve the spatial resolution, and they have a stride of one (if applicable). 2) 1×1 Conv, a convolution layer with kernel size of 1×1 and zero padding. 3) max pooling layer with kernel size of 3×3. 4) average pooling layer with a kernel size of 3×3. 5) identity. 6) zero. A zero operation indicates that there is no connection between nodes. The max and average pooling layers are followed by batch noramlization (BN) [13]. We use Parametric Rectified Linear Unit (PReLU) [8] as the non-linearity in all convolutional operation.

Operation	Output size	R	Param.
Conv2d(k=3,s=2,p=1),BN	[64 x 56 x 56]	1	1856
Normal-Cell 1-6	[64 x 56 x 56]	6	33,792
Reduction-Cell 1	[128 x 28 x 28]	1	10,688
Normal-Cell 7-11	[128 x 28 x 28]	5	92608
Reduction-Cell 2	[256 x 14 x 14]	1	35,712
Normal-Cell 12-15	[256 x 14 x 14]	4	60493,824
Reduction-Cell 3	[512 x 7 x 7]	1	128,768
PReLU, Conv2d(k=1), BN, PReLU	[512 x 7 x 7]	1	264192
Conv2d(k=7,g=512), BN	[512 x 1 x 1]	1	26112
Conv2d(k=1), BN	[128 x 1 x 1]	1	65792

Table 1: Architecture of PocketNetS-128. Normal and reduction cells are the cells learned by DARTS on CASIA-WebFace. The table shows the number of parameters for each operation. If the operation contains a set of sub-operations (e.g. Conv2d, BN), the number of parameters is presented as the sum of parameters for all these sub-operations and multiplied by R. Column R indicates how many times the operation is repeated. The k of the convolution layer (Conv2d) refers to the kernel size, s is the stride, p is the padding, and g is the group parameter.

PocketNet architecture: We followed [16] by setting the number of nodes in all cells to $N = 7$. We apply fast downsampling in the beginning of the network using 3×3 convolution (stride=2) followed by BN [13]. To obtain the feature embedding of the input face image, we use global depthwise convolution [11] rather than using average pooling or fully connected layer directly before the classification layer. Our choice of using the global depthwise convolution for the embedding stage is based on: a) it contains fewer parameters than a fully connected layer, b) convolutional neural network (CNN) with global depth-wise convolution is more accurate than the one with average pooling for FR, as reported in previous works [3, 1]. The rest of the network architecture is constructed by stacking M normal cells and 3 reduction cells at 1/3 and 2/3 of the network depth, and after the last normal cell. We trained the NAS to optimize α_{normal} and $\alpha_{reduction}$ used to construct the normal and reduction cells, respectively.

We trained the search algorithm to learn from the CASIA-WebFace dataset [36]. Training details are presented later in Section 3.1. The best discovered normal and reduction cells by DARTS are shown in Figures 1a and 1b, respectively. In this work, we present four architectures based on the learned cells: PocketNetS-128, PocketNetS-256, PocketNetM-128, and PocketNetM-256. The architecture of PocketNetS-128 and PocketNetS-256 (PocketNet small) are identical. Each of them contains 18 cells i.e 15 normal cells and 3 reduction cells. The number of feature maps (out channel) of the first layer is 64. The only difference is the embedding size, where the embedding in PocketNetS-128 is of size 128-D and in PocketNetS-256 is of size 256-D. Table 1 presents the overall architecture of PocketNetS-128. PocketNetS-128 contains in total 925,632 trainable parameters and setting the embed-

Operation	Output size	R	Param
Conv2d(k=3,s=2,p=1),BN	[128 x 56 x 56]	1	3712
Normal-Cell1-6	[128 x 56 x 56]	3	56832
Reduction-Cell 1	[256 x 28 x 28]	1	35,712
Normal-Cell 7-11	[256 x 28 x 28]	2	128896
Reduction-Cell 2	[512 x 14 x 14]	1	128,768
Normal-Cell 12-15	[512 x 14 x 14]	1	227,072
Reduction-Cell 3	[1024 x 7 x 7]	1	486,912
PReLU, Conv2d(k=1), BN, PReLU	[512 x 7 x 7]	1	526848
Conv2d(k=7,g=512), BN	[512 x 1 x 1]	1	26112
Conv2d(k=1), BN	[128 x 1 x 1]	1	65792

Table 2: Architecture of PocketNetM-128. Normal and reduction cells are the cells learned by DARTS on CASIA-WebFace. The table shows the number of parameters for each operation. If the operation contains a set of sub-operations (e.g. Conv2d, BN), the number of parameters is presented as the sum of parameters for all these sub-operations and multiplied by R. Column R indicates how many times the operation is repeated. The k of the convolution layer (Conv2d) refers to the kernel size, s is the stride, p is the padding, and g is the group parameter.

ding size to 256 increases the number of parameters in PocketNetS-256 to 991,424. All networks use floating-point 32 and the required memory footprints are 3.7 and 3.9 MB by PocketNetS-128 and PocketNetS-256, respectively. The main motivation for using different embedding sizes is to evaluate the effect of embedding size on the network performance and memory footprint. We also investigate a wider architecture of PocketNet by doubling the number of feature maps of the network and reducing the number of cells from 18 to 9. This result in two networks: PocketNetM-128 and PocketNetM-256 (PocketNet medium) with embedding size of 128-D and 256-D, respectively. The architecture of PocketNetM-128 is presented in Table 2. PocketNetM-128 contains 1,686,656 parameters and PocketNetM-256 contains 1,752,448 parameters.

2.2. PocketNet Training Paradigm

Towards the PocketNet training paradigm that incorporates our proposed multi-Step KD, we start by formulating the margin-based Softmax loss and knowledge distillation concept. Margin-Based Softmax loss has been widely deployed in recent FR solutions [5, 31, 24]. It achieved SOTA accuracy on major benchmarks [5, 22, 24]. In this work, we utilize the ArcFace loss [5] to train our PocketNets. ArcFace loss extends over the softmax loss by manipulating the decision boundary between the classes by deploying an additive angular margin penalty on the angle between the weights of the last fully connected layer and the feature representation. Formally, ArcFace loss is defined as follow:

$$L_{Arc} = \frac{1}{M} \sum_{i \in M} -log \frac{e^{s(cos(\theta_{y_i}+m))}}{e^{s(cos(\theta_{y_i}+m))} + \sum_{j=1, j \neq y_i}^{C} e^{s(cos(\theta_j))}}, \quad (3)$$

where θ_{y_i} is the angle between the feature f_i and $i-th$ class center, $y_i \in [1, C]$ (C is the number of classes), M is batch size, m is the margin penalty value and s is scale parameter.

(a) Normal cell learned on CASIA-WebFace.

(b) Reduction cell learned on CASIA-WebFace.

(c) Normal cell learned on CIFAR-10.

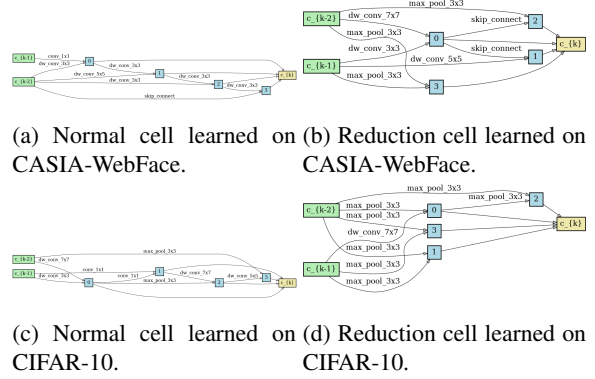(d) Reduction cell learned on CIFAR-10.

Figure 1: Normal and reduction cells learned by DARTS on CASIA-WebFace and CIFAR-10 datasets.

Knowledge distillation (KD): KD is a technique to improve the performance and generalizability of smaller models by transferring the knowledge learned by a cumbersome model (teacher) to a single small model (student) [10]. The idea is to guide the student model to learn the relationship between different classes discovered by the teacher model that contains more complex information beyond the ground truth labels [10]. The KD is originally proposed to improve the performance of a small backbone trained with SoftMax loss for a classification task [10]. However, the learning objective of the FR model is to optimize feature representations needed for face verification. In this work, as a step towards our proposed multi-step KD, we train our PocketNet model to learn feature representations that are similar to the ones learned by the teacher model. We achieve that by introducing an additional loss function (Mean squared error (MSE)) to ArcFace loss operated on the embedding layer. Formally, the l_{mse} loss is defined as follows:

$$l_{mse} = \frac{1}{M} \sum_{i \in M} 1 - \frac{1}{D} \Sigma_{h=1}^{D} \left(\Phi_t^S(x)_h - \Phi_t^T(x)_h \right)^2, \quad (4)$$

where Φ_t^S and Φ_t^T are the feature representations obtained from the last fully connected layer of student and teacher models, respectively, and D is the size of the feature representation. The final training loss function is defined as follow:

$$l_{mse} = l_{Arc} + \lambda l_{mse}, \quad (5)$$

where λ is a weight parameter. The feature representations learned by the ArcFace loss are normalized. Thus, the value range of l_{mse} is much small i.e. ≤ 0.007. This value is very small in comparison to the ArcFace loss value (around 60 at the beginning of the training phase.) We set the λ value to 100. Thus, the l_{mse} contributes to the model training.

Multi-Step Knowledge Distillation: Previous works [25, 35] observed that transforming the knowledge from a very deep teacher model to a small student model is difficult when the gap in terms of network size between the teacher and the student model is large.

In this work, we present a novel concept by relaxing this difficulty of a substantial discrepancy between teacher model and student by synchronizing the student and the teacher model during the training, without the need for transforming the knowledge to intermediate networks [25, 35]. Our solution is designed to transfer the knowledge learned by a teacher model in a step-wise manner after each x number of iterations, i.e. Multi-Step KD. The key idea is that the information learned by a teacher at different steps of the training phase is different from the one learned when the teacher is fully converged. Thus, transferring the knowledge learned by a teacher at an early stage of training is easier for a student to learn. Thus, at a later point when the student is converged to some degree, it can learn more complex patterns from the teacher. To achieve that, we first train the teacher for I iterations. This teacher model is noted as $T1$. Then, we train the student model for the same number of iteration I with the assistance of the teacher $T1$. In this case, Φ^T (Equation 4) is $T1$ obtained after the first I iterations. We choose to train the teacher for one epoch each time. This will give the teacher a chance to learn from the whole training dataset. We repeated these two steps until the teacher and student models are converged. To simplify the implementation, we train first the teacher model until it is converged and save the model weights after each epoch. Then, we train the student model with the assistance of the teacher models. During the student training, we load the teacher weights that correspond to the same training epoch.

3. Experimental Setups

3.1. Neural Architecture Search

We train the DARTS to learn the normal and reduction cells on the CASIA-Webface dataset [36]. CASIA-Webface consists of 494,141 face images from 10,757 different identities. We split the dataset equally into two parts used for training and validation. The images are pre-aligned and cropped to 120×120 for the training subset and to 112×112 for the validation subset using the Multi-task Cascaded Convolutional Networks (MTCNN) solution [37]. During the training phase, the training images are randomly cropped to have a fixed size of 112×112 and then randomly horizontally flipped to make the search more robust, following common practice in FR research [5, 24]. All the training and validation images are normalized to have pixel values between -1 and 1. We followed DARTS training setup [16] by using Stochastic Gradient Descent with the momentum of 0.9 and weight decay of $3e - 4$ to optimize the DARTS weight w. We utilize a cosine anneal-
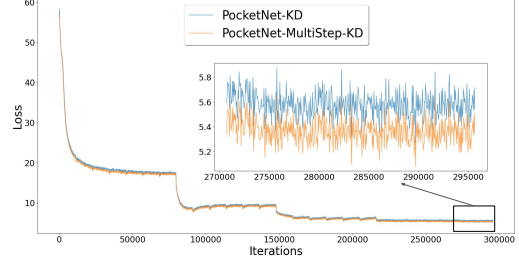
(a) ArcFace loss value of the model trained with KD vs. the model trained with multi-step KD over training iterations.
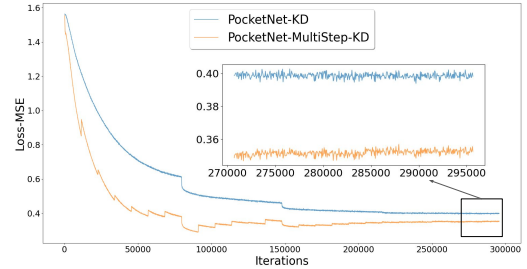
(b) KD vs. multi-step KD loss values over training iterations.

Figure 2: Effect of multi-step KD on the student model convergence. It can be noticed that multi-step KD enables the model trained with ArcFace and multi-step KD losses to better converges in comparison to the case where the model is trained with ArcFace and KD losses (Figure 2a). Also, it can be observed that training with multi-step KD guides the model to learn feature representations that are more similar (in comparison to KD) to the teacher ones (Figure 2b). These figures are based on training the PocketNetS-128 network.

ing strategy [18] to decrease the learning rate after each epoch with a minimum learning rate of 0.004. We set the batch size to 128 and the initial learning rate to 0.1. For α optimization, we use similar setup to DARTS [16] by using Adam optimizer with momentum $\beta = (0.5, 0.999)$ and weight decay of $1e - 3$. We set the initial learning rate for Adam optimizer to 0.0012. The initial channel size is set to 64 and the number of nodes in each cell is set to 8. We use a batch size of 128 and train DARTS for 50 epochs. These configurations are chosen to enable DARTS training on available GPUs. All training codes are implemented in Pytorch [27] and trained on 6 NVIDIA GeForce RTX 2080 Ti (11GB) GPUs. The training lasted 2274 hours. We additionally conducted an additional experiment on CIFAR-10 [15] as a NAS domain ablation study for this work. The CIFAR-10 is a commonly used dataset for object detection and image classification tasks consisting of 60000 images (of the size 32×32) of 10 classes. We split CIFAR-10

equally into two parts: training and validation subsets. We run the DARTS search using the exact configurations described previously in this section to learn on the CIFAR-10 dataset. The training lasted around 30 hours on 6 NVIDIA GeForce RTX 2080 Ti (11GB) GPUs.

3.2. Face Recognition models and training

Based on the normal and reduction cells learned by DARTS on CASIA-WebFace [36], we trained three instances of PocketNetS-128. The first instance (noted as PocketNetS-128 (no KD)) is only trained with ArcFace loss described in Section 2.2. The second instance (noted as PocketNetS-128 (KD)) is trained with ArcFace loss with KD. The third instance is trained with ArcFace loss along with our proposed multi-step KD (noted as PocketNetS-128 (multi-step KD)). These three instances are used in our ablation study towards the proposed multi-step KD. On the other hand, based on the normal and reduction cells learned on CIFAR-10 [15] (object classification domain), we train another model based on these cells, noted as DartFaceNet-128 (no KD). This training is used as an ablation study to analyze the effect of training dataset sources on the neural architecture search algorithm by comparing its FR performance to its direct counterpart PocketNetS-128 (no KD).

Additionally, as detailed earlier, we trained four instances of PocketNets: PocketNetS-128, PocketNetS-256, PocketNetM-128, and PocketNetM-256 to compare our proposed PocketNets with the recent compact FR models proposed in the literature on different levels of compactness. All these models are trained with ArcFace loss along with our proposed multi-step KD. To enable KD multi-step solutions, we trained two instances of the ResNet-100 model with embedding sizes of $128 - D$ and $256 - D$. The ResNet-100(128) is used as a teacher for PocketNetS-128 and PocketNetM-128, while ResNet-100(256) is used as a teacher for PocketNetS-256 and PocketNetM-256.

We use the MS1MV2 dataset [5] to train all the investigated FR models in this work. The MS1MV2 is a refined version [5] of the MS-Celeb-1M [7] containing 5.8M images of 85K identities. We follow the common setting [5] to set the scale parameter s to 64 and margin value of ArcFace loss to 0.5. We set the mini-batch size to 512 and train our models on a single Linux machine (Ubuntu 20.04.2 LTS) with Intel(R) Xeon(R) Gold 5218 CPU 2.30GHz, 512 G RAM, and 4 Nvidia GeForce RTX 6000 GPUs. The proposed models in this paper are implemented using Pytorch [27]. All FR models are trained with Stochastic Gradient Descent (SGD) optimizer with an initial learning rate of 1e-1. We set the momentum to 0.9 and the weight decay to 5e-4. The learning rate is divided by 10 at 80k, 140k, 210k, and 280k training iterations. The total number of training iteration is 295K. During the training, we use random horizontal flipping with a probability of 0.5 for data augmenta-

tion. The networks are trained (and evaluated) on images of the size $112 \times 112 \times 3$, with pixel values between -1 and 1. These images are aligned and cropped using the Multi-task Cascaded Convolutional Networks (MTCNN) [37], following [5].

3.3. Evaluation benchmarks and metrics

We evaluate our PocketNets and build a comparison to SOTA based on 9 benchmarks detailed in this section. The considered evaluation benchmarks are Labeled Faces in the Wild (LFW) [12], Cross-age LFW (CA-LFW) [40], Cross-Pose LFW (CP-LFW) [39], Celebrities in Frontal-Profile in the Wild (CFP-FP) [29], AgeDB-30 [26], IARPA Janus Benchmark-B (IJB-B) [33], IARPA Janus Benchmark–C (IJB-C) [23], MegaFace [14], and MegaFace (R) [5, 14].

We acknowledge the evaluation metrics in the ISO/IEC 19795-1 [20] standard, however, for comparability, we follow the evaluation metrics defined in the utilized benchmarks as follows: LFW (accuracy), CA-LFW (accuracy), CP-LFW (accuracy), CFP-FP (accuracy), AgeDB-30 (accuracy), MegaFace (Rank-1 identification rate and true acceptance rates (TAR) at false acceptance rates (FAR) of 1e-6), IJB-B (TAR at FAR1e-4), IJB-C (TAR at FAR1e-4) and MegaFace (R), (Rank-1 identification rate and TAR at FAR1e-6).

4. Ablation Study

This section presents two ablation studies addressing the two main aspects of our design of the PocketNets solution.

Ablation Study on NAS training dataset source: We trained two different instances of DARTS search algorithm to learn from CASIA-WebFace [36] (face images) and CIFAR-10 [15] (animals, cars, etc.), respectively. Figure 1 presents the normal and reduction cells learned on CASIA-WebFace and CIFAR-10, used to build our PocketNetS-128 (no KD) and the DartFaceNetS-129 (no KD), respectively. These networks share the same structure including the embedding stage and the number of cells. These networks are trained using the exact training setup described in Section 3.2. DartFaceNetS-128 (no KD) contains 885,184 parameters with 620.9286 MFLOPs. PocketNetS-128 (no KD) contains 925,632 parameters with 587.11 MFLOPs. Table 3 presents the achieved performance by PocketNetS-128 (no KD) and DartFaceNetS-128 (no KD) on nine different benchmarks. It can be clearly noticed that PocketNetS-128 (no KD) outperformed DartFaceNetS-128 (no KD) with an obvious margin on all considered benchmarks. The demonstrates that utilizing neural network architecture designed for common computer vision tasks leads to sub-optimal performance when it is used for the FR. It also supports our choice for training NAS to learn from a face image dataset and points out that FR does require face-specific architecture design.

| (a) LFW | (b) CA-LFW | (c) CP-LFW | (d) CFP-FP | (e) AgeDB-30 |

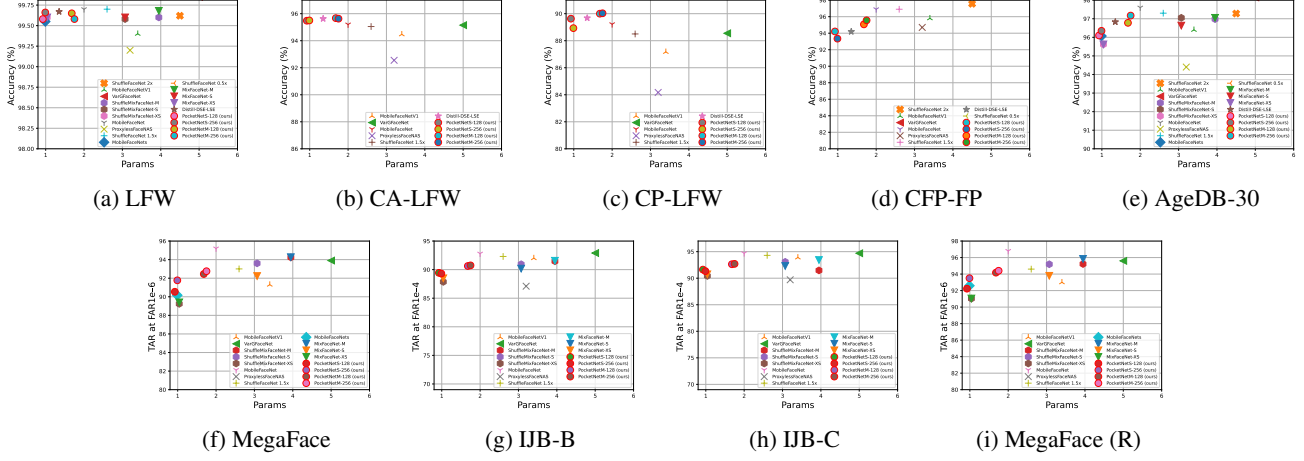| (f) MegaFace | (g) IJB-B | (h) IJB-C | (i) MegaFace (R) |

Figure 3: Number of parameters (in millions) vs. performance on LFW (accuracy), CA-LFW (accuracy), CP-LFW (accuracy), CFP-FP (accuracy), AgeDB-30 (accuracy), MegaFace (TAR at FAR1e-6), IJB-B (TAR at FAR1e-4), IJB-C (TAR at FAR1e-4) and MegaFace (R), (TAR at FAR1e-6). Our PocketNets are marked with circle marker and red edge color and are placed repeatedly in the top left corner, proving a SOTA trade-off between FR performance and compactness.

Ablation Study on Multi-Step KD: Here, we prove the benefit of introducing our Multi-step KD training process on the PocketNet FR performance. This step-wise ablation study first looks into the advances provided by the KD training in comparison to training with no KD, proving the advancement achieved by our multi-step KD in comparison to KD. Introducing KD to the PocketNet training phase improved the verification performances on all evaluation benchmarks by comparing PocketNetS-128 (no KD) to PocketNetS-128 (KD), ass observed in Table 3. PocketNetS-128 (no KD) is trained only with ArcFace loss, while PocketNetS-128 (KD) is trained with ArcFace along with KD from the ResNet-100 model. When PocketNetS-128 is trained with ArcFace along with our multi-step KD (i.e. PocketNetS-128 (multi-step KD)), the achieved verification performance improved in eight out of nine different benchmarks in comparison to PocketNetS-128 (KD) (Table 3), empirically proving the benefit of our multi-step KD. We also investigated the competence of our proposed multi-step KD on improving the model convergence. Figure 2a presents a comparison between ArcFace loss values of PocketNetS-128 (KD) and PocketNetS-128 (multi-step KD). It can be noticed that multi-step KD improved the model convergence. Also, our multi-step KD enhanced the similarity between the feature representation of the teacher model and the student model. This observation is seen in Figure 2b where the MSE values of PocketNetS-128 (multi-step KD) is smaller than the one of PocketNetS-128 (KD).

5. Experimental results

Table 4 presents the achieved FR results by our Pocket-Nets on all evaluation benchmarks. It also presents a comparison between our proposed PocksetNets and the recent compact models proposed in the literature. The presented models are ordered in groups based on the number of parameters (compactness). The first part of Table 4 presents the achieved result by the models that have between 2 and 5M trainable parameters, while the second and third parts present the results for the models with less than 2M and less than 1M trainable parameters, respectively.

Our PocketNetS-128 (0.92M parameters) and PocketNetS-256 (0.99M parameters) outperformed all models that have less than 1M parameters. With 10% less parameter than MobileFaceNets [3], PocketNetS-128 outperformed MobileFaceNets on all considered benchmarks. Also, PocketNetS-128 and PocketNetS-256 achieved competitive results to other deeper models that contain 4 or 5 times more parameters than PocketNets. For example, PocketNetS-128 outperformed VarGFaceNet (5M parameters) on the challenging CA-LFW and CP-LFW benchmarks where the achieved accuracies by PocketNetS-128 are 95.48% on CA-LFW and 89.63% on CP-LFW in comparison to 95.15% on CA-LFW and 88.55% CP-LFW achieved by VarGFaceNet [35].

Our PocketNetM-128 (1.68M parameters) and PocketNetM-256 (1.75M parameters) outperformed all models proposed in the literature that have less than 2M parameters. They also achieved competitive results to the models that have between 2 and 5M parameters, even outperforming them in many cases. For example, our PocketNetM-128 achieved SOTA accuracies on the challenging CA-LFW and CP-LFW among all models that have less than 5M of trainable parameters. On the large-scale evaluation benchmarks, IJB-B and IJB-C, our PocketNetM

Model	Param. (M)	MFLOPs	LFW (%)	CA-LFW (%)	CP-LFW (%)	CFP-FP (%)	AgeDB-30 (%)	IJB-B (%)	IJB-C (%)	MegaFace Rank-1(%)	MegaFace Ver.(%)	MegaFace (R) Rank-1(%)	MegaFace (R) Ver.(%)
ResNet100-128 - Teacher	55.52	24192.51	99.83	96.16	93.1	98.64	98.3	94.72	96.08	80.55	97.13	98.36	98.66
DartFaceNetS-128 (no KD)	0.89	620.9	99.26	94.98	88.5	93.18	95.23	87.89	90.5	73.44	87.65	87.99	89.42
PocketNetS-128 (no KD)	0.925	587.11	99.5	95.01	88.93	93.78	95.88	88.29	90.79	74.42	88.99	89.46	90.67
PocketNetS-128 - KD	0.925	587.11	99.55	95.15	89.13	93.82	**96.50**	89.23	91.47	75.22	90.21	90.72	92.04
PocketNetS-128 - multi-step KD	0.925	587.11	**99.58**	**95.48**	**89.63**	**94.21**	96.10	**89.44**	**91.62**	**75.81**	**90.54**	**91.22**	**92.23**

Table 3: Comparative evaluation results of ResNet100-128, DartFaceNetS-128 (no KD), PocketNetS-128 (no KD), PocketNetS-128 KD, and PocketNetS-128 multi-step KD on different evaluation benchmarks. The results are reported based on the evaluation metric described in Section 3.3. ResNet100-128, DartFaceNetS-128 (no KD) and PocketNetS-128 (no KD) are trained with ArcFace loss. PocketNetS-128 KD is trained with ArcFace loss with KD from teacher model (ResNet100-128). PocketNetS-128 multi-step KD is trained with ArcFace loss with multi-step KD from teacher model (ResNet100-128). PocketNetS-128 (no KD) performed better than the DartFaceNetS-128 (no KD), proving the sanity of designing FR-specific architecture. PocketNetS-128 multi-step KD performes better than PocketNetS-128 (no KD) and PocketNetS-128 KD, proving the benefits of the proposed multi-step KD.

Model	Params.(M)	MFLOPs	LFW (%)	CA-LFW (%)	CP-LFW (%)	CFP-FP (%)	AgeDB-30 (%)	IJB-B (%)	IJB-C (%)	MegaFace Rank-1 (%)	MegaFace Ver. (%)	MegaFace(R) Rank-1 (%)	MegaFace(R) Ver. (%)
VarGFaceNet [35, 22]	5.0	1022	99.85	95.15	88.55	98.50	98.15	92.9	94.7	78.2	93.9	94.9	95.6
ShuffleFaceNet 2× [21]	4.5	1050	99.62	-	-	97.56	97.28	-	-	-	-	-	-
MixFaceNet-M [1]	3.95	626.1	99.68	-	-	-	97.05	91.55	93.42	78.20	94.26	94.95	95.83
ShuffleMixFaceNet-M [1]	3.95	626.1	99.60	-	-	-	96.98	91.47	91.47	78.13	94.24	94.64	95.22
MobileFaceNetV1 [22]	3.4	1100	99.4	94.47	87.17	95.8	96.4	92.0	93.9	76.0	91.3	91.7	93.0
ProxylessFaceNAS [22]	3.2	900	99.2	92.55	84.17	94.7	94.4	87.1	89.7	69.7	82.8	82.1	84.8
MixFaceNet-S [1]	3.07	451.7	99.6	-	-	-	96.63	90.17	92.30	76.49	92.23	92.67	93.79
ShuffleMixFaceNet-S [1]	3.07	451.7	99.58	-	-	-	97.05	90.94	93.08	77.41	93.60	94.07	95.19
ShuffleFaceNet 1.5x [21, 22]	2.6	577.5	99.7	95.05	88.50	96.9	97.3	92.3	94.3	77.4	93.0	94.1	94.6
MobileFaceNet [22]	2.0	933	99.7	95.2	89.22	96.9	97.6	92.8	94.7	79.3	95.2	95.8	96.8
PocketNetM-256 (Ours)	1.75	1099.15	99.58	95.63	90.03	95.66	97.17	90.74	92.70	78.23	92.75	94.13	94.40
PocketNetM-128 (Ours)	1.68	1099.02	99.65	95.67	90.00	95.07	96.78	90.63	92.63	76.49	92.45	92.77	94.17
Distill-DSE-LSE [17]	1.35	-	99.67	95.63	89.68	94.19	96.83	-	-	-	-	-	
MixFaceNet-XS [1]	1.04	161.9	99.60	-	-	-	95.85	88.48	90.73	74.18	89.40	89.35	91.04
ShuffleMixFaceNet-XS [1]	1.04	161.9	99.53	-	-	-	95.62	87.86	90.43	73.85	89.24	88.823	91.03
MobileFaceNets [3]	0.99	439.8	99.55	-	-	-	96.07	-	-	-	90.16	-	92.59
PocketNetS-256 (Ours)	0.99	587.24	99.66	95.50	88.93	93.34	96.35	89.31	91.33	76.53	91.77	92.29	93.5
PocketNetS-128 (Ours)	0.92	587.11	99.58	95.48	89.63	94.21	96.10	89.44	91.62	75.81	90.54	91.22	92.23
ShuffleFaceNet 0.5x [21]	0.5	66.9	99.23	-	-	92.59	93.22	-	-	-	-	-	-

Table 4: The achieved results on 9 benchmarks. The results are reported in % based on the evaluation metric described in Section 3.3. The models are ordered based on the number of parameters. Our PoacketNetS-128 and PocketNetS-256 consistently extend the SOTA performance on all evaluation benchmarks for the models that have less than 1M parameters. Our PoacketNetM-128 and PocketNetM-256 also achieved SOTA performances for models that have less than 2M parameters. Additionally, they achieved very competitive results to larger models that have between 2 and 5M parameters. All decimal points are provided as reported in the respective works.

achieved competitive performance to many of the larger models. For example, on IJB-C, our PocketNetM-128 (1.68M parameters) achieved verification performance of 92.63% TAR at FAR 1e-6 and the best verification performance is 94.7% achieved by MobileFaceNet [22] (2M parameters) and VarGFaceNet [35] (5M parameters). On MegaFace and the refined version of MegaFace, our PocketNetM outperfomred all the models than have less than 2M of trainable parameters and they achieved a competitive results in term of identification and verification accuracies to the models that have between 2 and 5M parameters. For example, our PocketNetM-258 (1.75M parameters) outperformed MixFaceNet-S [1] (3.07M parameters), ProxylessFaceNAS [22] (3.2M parameters) and MobileFaceNetV1 [22] (3.4M parameters) on MegaFace and MegaFace (R).

To visually illustrate the competence of our PocketNet, we plot the number of parameters vs. the achieved verification performance of our PocketNet and the recent compact models proposed in the literature (all numbers provided in Table 4). Figure 3 presents a trade-off between the number of parameters and the achieved verification performance. Each of the presented solutions is marked with a point(x,y) in the plot, where x is the number of parameters in millions and y is the achieved verification performance. The model that tends to be placed on the top-left corner (small x and large y) of the plot has the best trade-off between the model compactness and the achieved verification performance. It can be observed, in Figure 3, that our PocketNets are always in the top left corner in comparison to other methods, proving to achieve SOTA trade-off between model compactness and FR performance. It must be noted that all the reported PocketNets in this section are trained with our proposed multi-step KD.

6. Conclusion

We present in this paper a family of extremely lightweight FR models, namely PocketNets. This is one of the first efforts proposing to utilize NAS to learn to design a compact yet accurate FR model. We additionally presented a novel training paradigm based on knowledge distillation, namely mulit-step KD, where the knowledge distillation is performed at multiple stages of the teacher training maturity. Extensive step-wise ablation studies proved the benefits of both, designing a face-specific architecture, as well as, the enhanced performance of the lightweight model when trained with the proposed multi-step KD. Through extensive experimental evaluations on nine FR benchmarks, we demonstrated the high verification performance achieved by our compact PocketNet models and our proposed mulit-step KD. Under the same level of model compactness, our PocketNets consistently scored SOTA performances in comparison to the compact models proposed in the literature.

References

[1] Fadi Boutros, Naser Damer, Meiling Fang, Florian Kirchbuchner, and Arjan Kuijper. Mixfacenets: Extremely efficient face recognition networks. In *International IEEE Joint Conference on Biometrics, IJCB 2021, Shenzhen, China, August 4-7, 2021*, pages 1–8. IEEE, 2021.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[3] Sheng Chen, Yang Liu, Xiang Gao, and Zhen Han. Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. In Jie Zhou, Yunhong Wang, Zhenan Sun, Zhenhong Jia, Jianjiang Feng, Shiguang Shan, Kurban Ubul, and Zhenhua Guo, editors, *Biometric Recognition - 13th Chinese Conference, CCBR 2018, Urumqi, China, August 11-12, 2018, Proceedings*, volume 10996 of *Lecture Notes in Computer Science*, pages 428–438. Springer, 2018.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009.

[5] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.

[6] Jiankang Deng, Jia Guo, Debing Zhang, Yafeng Deng, Xiangju Lu, and Song Shi. Lightweight face recognition challenge. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Ko-*

rea (South), October 27-28, 2019, pages 2638–2646. IEEE, 2019.

[7] Yandong Guo, Lei Zhang, Yuxiao Hu, Xiaodong He, and Jianfeng Gao. Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III*, volume 9907 of *Lecture Notes in Computer Science*, pages 87–102. Springer, 2016.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016.

[10] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015.

[11] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[12] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, Uni. of Massachusetts, Amherst, October 2007.

[13] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[14] Ira Kemelmacher-Shlizerman, Steven M. Seitz, Daniel Miller, and Evan Brossard. The megaface benchmark: 1 million faces for recognition at scale. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4873–4882. IEEE Computer Society, 2016.

[15] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[16] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: differentiable architecture search. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[17] Wenting Liu, Li Zhou, and Jie Chen. Face recognition based on lightweight convolutional neural networks. *Inf.*, 12(5):191, 2021.

[18] Ilya Loshchilov and Frank Hutter. SGDR: stochastic gradient descent with warm restarts. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[19] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet V2: practical guidelines for efficient CNN architecture design. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, volume 11218 of *Lecture Notes in Computer Science*, pages 122–138. Springer, 2018.

[20] A Mansfield. Information technology–biometric performance testing and reporting–part 1: Principles and framework. *ISO/IEC*, pages 19795–1, 2006.

[21] Yoanna Martínez-Díaz, Luis S. Luevano, Heydi Mendez Vazquez, Miguel Nicolás-Díaz, Leonardo Chang, and Miguel González-Mendoza. Shufflefacenet: A lightweight face architecture for efficient and highly-accurate face recognition. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 2721–2728. IEEE, 2019.

[22] Yoanna Martínez-Díaz, Miguel Nicolás-Díaz, Heydi Méndez-Vázquez, Luis S Luevano, Leonardo Chang, Miguel Gonzalez-Mendoza, and Luis Enrique Sucar. Benchmarking lightweight face architectures on specific face recognition scenarios. *Artificial Intelligence Review*, pages 1–44, 2021.

[23] Brianna Maze, Jocelyn C. Adams, James A. Duncan, Nathan D. Kalka, Tim Miller, Charles Otto, Anil K. Jain, W. Tyler Niggel, Janet Anderson, Jordan Cheney, and Patrick Grother. IARPA janus benchmark - C: face dataset and protocol. In *2018 International Conference on Biometrics, ICB 2018, Gold Coast, Australia, February 20-23, 2018*, pages 158–165. IEEE, 2018.

[24] Qiang Meng, Shichao Zhao, Zhida Huang, and Feng Zhou. Magface: A universal representation for face recognition and quality assessment. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 14225–14234. Computer Vision Foundation / IEEE, 2021.

[25] Seyed-Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 5191–5198. AAAI Press, 2020.

[26] Stylianos Moschoglou, Athanasios Papaioannou, Christos Sagonas, Jiankang Deng, Irene Kotsia, and Stefanos Zafeiriou. Agedb: The first manually collected, in-the-wild age database. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1997–2005. IEEE Computer Society, 2017.

[27] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[28] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 4510–4520. Computer Vision Foundation / IEEE Computer Society, 2018.

[29] Soumyadip Sengupta, Jun-Cheng Chen, Carlos Domingo Castillo, Vishal M. Patel, Rama Chellappa, and David W. Jacobs. Frontal to profile face verification in the wild. In *2016 IEEE Winter Conference on Applications of Computer Vision, WACV 2016, Lake Placid, NY, USA, March 7-10, 2016*, pages 1–9. IEEE Computer Society, 2016.

[30] Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. In *30th British Machine Vision Conference 2019, BMVC 2019, Cardiff, UK, September 9-12, 2019*, page 74. BMVA Press, 2019.

[31] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu. Cosface: Large margin cosine loss for deep face recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5265–5274. IEEE Computer Society, 2018.

[32] Xiaobo Wang. Teacher guided neural architecture search for face recognition. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 2817–2825. AAAI Press, 2021.

[33] Cameron Whitelam, Emma Taborsky, Austin Blanton, Brianna Maze, Jocelyn C. Adams, Tim Miller, Nathan D. Kalka, Anil K. Jain, James A. Duncan, Kristen Allen, Jordan Cheney, and Patrick Grother. IARPA janus benchmark-b face dataset. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 592–600. IEEE Computer Society, 2017.

[34] Lumin Xu, Yingda Guan, Sheng Jin, Wentao Liu, Chen Qian, Ping Luo, Wanli Ouyang, and Xiaogang Wang. Vipnas: Efficient video pose estimation via neural architecture search. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 16072–16081. Computer Vision Foundation / IEEE, 2021.

[35] Mengjia Yan, Mengao Zhao, Zining Xu, Qian Zhang, Guoli Wang, and Zhizhong Su. Vargfacenet: An efficient variable group convolutional neural network for lightweight face recognition. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019,*

Seoul, Korea (South), October 27-28, 2019, pages 2647–2654. IEEE, 2019.

[36] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning face representation from scratch. *CoRR*, abs/1411.7923, 2014.

[37] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Process. Lett.*, 23(10):1499–1503, 2016.

[38] Qian Zhang, Jianjun Li, Meng Yao, Liangchen Song, Helong Zhou, Zhichao Li, Wenming Meng, Xuezhi Zhang, and Guoli Wang. Vargnet: Variable group convolutional neural network for efficient embedded computing. *CoRR*, abs/1907.05653, 2019.

[39] T. Zheng and W. Deng. Cross-pose lfw: A database for studying cross-pose face recognition in unconstrained environments. Technical Report 18-01, Beijing University of Posts and Telecommunications, February 2018.

[40] Tianyue Zheng, Weihong Deng, and Jiani Hu. Cross-age LFW: A database for studying cross-age face recognition in unconstrained environments. *CoRR*, abs/1708.08197, 2017.

[41] Ning Zhu, Zekuan Yu, and Caixia Kou. A new deep neural architecture search pipeline for face recognition. *IEEE Access*, 8:91303–91310, 2020.

[42] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[43] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. IEEE Computer Society, 2018.