DiVA
http://www.diva-portal.org

Preprint

This is the submitted version of a paper presented at *VIII International Workshop on Artificial Intelligence and Pattern Recognition, IWAIPR, Varadero, Cuba, September 27-29, 2023.*

Citation for the original published paper:

Alonso-Fernandez, F., Hernandez-Diaz, K., Buades Rubio, J M., Bigun, J. (2023)
SqueezerFaceNet: Reducing a Small Face Recognition CNN Even More Via Filter Pruning
In:
https://doi.org/10.48550/arXiv.2307.10697

N.B. When citing this work, cite the original published paper.

Permanent link to this version:
http://urn.kb.se/resolve?urn=urn:nbn:se:hh:diva-51299

SqueezerFaceNet: Reducing a Small Face Recognition CNN Even More Via Filter Pruning

Fernando Alonso-Fernandez[1], Kevin Hernandez-Diaz[1], Jose Maria Buades Rubio[2], and Josef Bigun[1]

[1] School of Information Technology, Halmstad University, Sweden
feralo@hh.se, kevin.hernandez-diaz@hh.se, josef.bigun@hh.se
[2] Computer Graphics and Vision and AI Group, University of Balearic Islands, Spain
josemaria.buades@uib.es

Abstract. The widespread use of mobile devices for various digital services has created a need for reliable and real-time person authentication. In this context, facial recognition technologies have emerged as a dependable method for verifying users due to the prevalence of cameras in mobile devices and their integration into everyday applications. The rapid advancement of deep Convolutional Neural Networks (CNNs) has led to numerous face verification architectures. However, these models are often large and impractical for mobile applications, reaching sizes of hundreds of megabytes with millions of parameters. We address this issue by developing SqueezerFaceNet, a light face recognition network which less than 1M parameters. This is achieved by applying a network pruning method based on Taylor scores, where filters with small importance scores are removed iteratively. Starting from an already small network (of 1.24M) based on SqueezeNet, we show that it can be further reduced (up to 40%) without an appreciable loss in performance. To the best of our knowledge, we are the first to evaluate network pruning methods for the task of face recognition.

Keywords: Face recognition · Mobile Biometrics · CNN pruning · Taylor scores.

1 Introduction

The widespread use of smartphones as all-in-one platforms has led to more people relying on them for accessing online services such as e-commerce and banking. This makes it crucial to implement robust user authentication mechanisms to ensure secure device unlocking and protected transactions. Here, we address face recognition (FR) for mobile applications, where biometric verification is increasingly employed for the mentioned purposes. As with many other vision tasks, Convolutional Neural Networks (CNNs) have become a very popular tool for biometrics, including FR [19]. Nevertheless, the high-performing models proposed in the literature, e.g. [7], usually entail extensive storage and computational resources due to their millions of parameters. This poses a significant challenge for deploying them on resource-limited devices.

Table 1: Proposed lightweight models in the literature for face recognition.

Network	Input size	Para-meters	Vector Size	Base Architecture
LightCNN (18) [21]	128×128	12.6M	256	
MobileFaceNets (18) [6]	112×112	0.99M	256	MobileNetv2
MobiFace (19) [9]	112×112	n/a	512	MobileNetv2
ShuffleFaceNet (19) [15]	112×112	0.5-4.5M	128	ShuffleNet
SeesawFaceNets (19) [23]	112×112	1.3M	512	
VarGFaceNet (19) [22]	112×112	5M	512	VarGNet
SqueezeFacePoseNet [2]	113×113	0.86-1.24M	1000	SqueezeNet
PocketNet (21) [4]	112×112	0.92-0.99 M	128-256	PocketNet
MixFaceNets (21) [3]	112×112	1.04-3.95M	512	MixNets
SqueezerFaceNet (ours)	113×113	0.65-0.94M	1000	SqueezeNet

Across the years, several light CNNs have been presented, mainly for common visual tasks in the context of the ImageNet challenge [17]. Examples include SqueezeNet [11] (1.24M parameters), MobileNetV2 [18] (3.5M), ShuffleNet [25] (1.4M), MixNets [20] (5M), or VarGNet [24] (13.23M). They employ different techniques to achieve fewer parameters and faster processing, such as point-wise convolution, depth-wise separable convolution, variable group convolution, mixed convolution, channel shuffle, and bottleneck layers. Some works (Table 1) have adapted these networks for FR purposes [6, 9, 15, 22, 2, 3]. Instead of adapting existing common architectures, the work [4] suggested applying Neural Architecture Search (NAS) to design a family of light FR models, named PocketNets.

In this paper, we follow another strategy, consisting of applying network compression to existing architectures. Common techniques include knowledge distillation, quantization, or pruning. A number of them have been used to reduce the size of general image classification models, and some works recently started to apply them for face detection [13] or ocular recognition [1]. Here, we use a pruning method based on importance scores of network filters [16] to reduce an already small FR network of 1.24M parameters that uses a modified SqueezeNet architecture [11, 2]. Thus, we call our network SqueezerFaceNet. The importance score of a filter is obtained considering its effect on the error if it is removed. This is computed by first-order Taylor approximation, which only requires the elements of the gradient computed during training via backpropagation.

To the best of our knowledge, we are the first to evaluate network pruning methods for the task of FR. We test SqueezerFaceNet on a face verification scenario over VGGFace2-Pose, a subset of the VGGFace2 database [5] with 11040 images from 368 subjects on three poses (frontal, three-quarter, and profile). We show that the number of filters of the network can be reduced up to 15% without a significant loss of accuracy in one-to-one comparisons for any given pose. If we allow five images per user to build an identity template, accuracy is not significantly affected until a 30-40% reduction in the number of filters.

2 Network Pruning Method

We apply the method of [16], which iteratively estimates the importance scores of individual elements based on their effect on the network loss. Then, elements with the lowest scores are pruned, leading to a more compact network.

Given a network with parameters $\mathbf{W} = \{w_0, w_1, ..., w_M\}$ and a training set $\mathcal{D}$ of input (x_i) and output (y_i) pairs $\mathcal{D} = \{(x_0, y_0), (x_1, y_1), ..., (x_K, y_K)\}$, the aim of network training is to minimize the classification error E by solving $\min_{\mathbf{W}} E(\mathcal{D}, \mathbf{W}) = \min_{\mathbf{W}} E(y|x, \mathbf{W})$. The importance of a parameter w_m can be defined by its impact on the error if it is removed. Under an *i.i.d.* assumption, the induced error can be quantified as the squared difference of the prediction error E with and without the parameter:

$$\mathcal{I}_m = \Big(E(\mathcal{D}, \mathbf{W}) - E(\mathcal{D}, \mathbf{W}|w_m = 0) \Big)^2 \tag{1}$$

However, computing $\mathcal{I}_m$ for each parameter using Eq. 1 would demand to evaluate M versions of the network, one for each removed parameter, making the process expensive computationally. This is avoided by approximating $\mathcal{I}_m$ in the vicinity of $\mathbf{W}$ by its first-order Taylor expansion $\mathcal{I}_m^1(\mathbf{W}) = (g_m w_m)^2$, where $g_m = \frac{\partial E}{\partial w_m}$ are the elements of the gradient g. A second-order expansion is also proposed [16], but it demands computing the Hessian of E, so we employ the first-order approximation for a more compact and fast computation. The gradient g is available from backpropagation, so $\mathcal{I}_m$ can be easily computed. To compute the joint importance of a set of parameters $\mathbf{W}_S$ (e.g. a filter), we apply:

$$\mathcal{I}_S^1(\mathbf{W}) \triangleq \sum_{s \in S} (g_s w_s)^2 \tag{2}$$

The algorithm starts with a trained network, which is pruned iteratively over the same training set. Given a mini-batch, the gradients are computed, and the network weights are updated by gradient descent. Simultaneously, the importance of each filter is computed via Eq. 2. At the end of each epoch, the importance scores of each filter are averaged over the mini-batches, and the filters with the smallest importance scores are removed. The pruning process is then stopped after a certain number of epochs. The resulting network can be then fine-tuned again over the training set to regain potential accuracy losses due to filter removal.

3 SqueezerFaceNet Architecture and Database

As the backbone for SqueezerFaceNet, we employ SqueezeNet [11]. This is among the smallest generic CNNs proposed in the context of the ImageNet challenge, and one of the early networks designed to reduce the number of parameters and size. It has only 18 convolutional layers, 1.24M parameters, and 4.6 MB in its uncompressed version. To reduce the network size, it uses fire modules, which first reduce the input channel dimensionality via 1×1 point-wise filters (*squeeze*

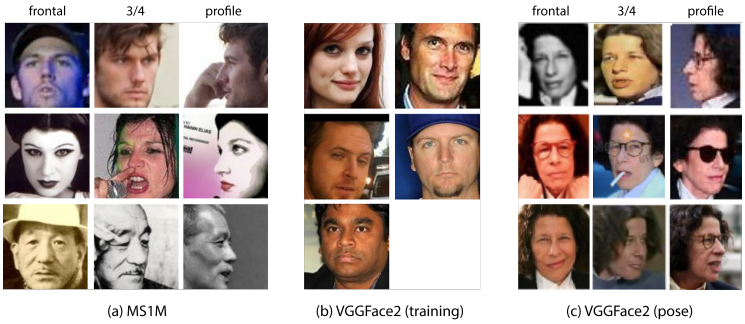

Fig. 1: Example images of the databases used. (a) MS1M from three users (by row) and three viewpoints (column). (b) VGGFace2 training images with a random crop. (c) VGGFace2 pose templates from three viewpoints (by column).

phase), to be then processed with a larger amount of (more costly) 3×3 and 1×1 filters in a lower dimensional space (*expand* phase). Another strategy is late downsampling, so convolution layers are presented maps as large as possible. According to its authors, it should lead to higher accuracy.

In the present paper, we adopt the SqueezeNet implementation previously proposed for FR using light CNNs in [2], referred to as SqueezeFacePoseNet. In particular, the network employs an input of 113×113, instead of the original 227×227 of SqueezeNet. This is achieved by changing the stride of the first convolutional layer from 2 to 1, while keeping the rest of the network unchanged, which allows to reuse ImageNet parameters as starting model. Such transfer learning strategy from ImageNet has been shown to provide equal or better performance than if initialized from scratch, while converging faster [14]. In the present paper, we have also added batch normalization between convolutions and ReLU layers. This is missing in the original SqueezeNet and in [2], but batch normalization is commonly used before non-linearities to aid in the training of deep networks [12]. Compared to [2], we observe that it also leads to increased recognition accuracy, with a small overhead of parameters.

The database for training and evaluation is VGGFace2, with 3.31M images of 9131 celebrities (363.6 images/person on average) [5]. The images, downloaded from the Internet, show significant variations in pose, age, ethnicity, lightning and background. The protocol contemplates 8631 training classes (3.14M images) and the remaining 500 classes for testing. For cross-pose experiments, a subset of 368 subjects from the test set is defined (called VGGFace2-Pose), having 10 images per pose (frontal, three-quarter, and profile) and a total of 11040 images.

To further improve recognition performance, we also pre-train SqueezerFaceNet in the RetinaFace cleaned set of the MS-Celeb-1M database [10] (MS1M for short), with 5.1M images of 93.4K identities. The release contains 113×113 images of MS1M cropped with the five facial landmarks provided by RetinaFace [8]. While MS1M has a more significant number of images, its intra-identity variation is limited due to an average of 81 images/person. Following previous research [5, 2], we first pre-train SqueezerFaceNet on a dataset with a large

number of images (MS1M) and then fine-tune it with more intra-class diversity (VGGFace2). This has been shown to provide better performance than training the models only with VGGFace2. Some example images are shown in Figure 1.

4 Experimental Protocol

SqueezerFaceNet is trained for biometric identification using the soft-max function and ImageNet as initialization. We follow the training/evaluation protocol of VGGFace2 [5]. For training, the bounding boxes of VGGFace2 images are resized, so the shorter side has 129 pixels, and a random crop of 113×113 is taken. A random crop is not possible with MS1M, since images are directly at 113×113. We also apply horizontal random flip to both databases. The optimizer is SGDM with a mini-batch of 128. The initial learning rate is 0.01, which is decreased to 0.005, 0.001, and 0.0001 when the validation loss plateaus. Two percent of images per user in the training set are set aside for validation. Users in MS1M with fewer than 70 images are removed to reduce the parameters of the fully connected layer dedicated to under-represented classes and ensure at least one image per user in the validation set. This results in 35016 users and 3.16M images. We train with Matlab r2022b and use the ImageNet pre-trained model that comes with such release.

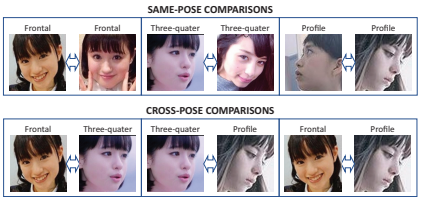

Fig. 2: Same-pose (top) and cross-pose comparisons (bottom).

Verification experiments are done with VGGFace2-Pose following the protocol of [5]. A center crop of 113×113 is taken after the shortest image side is resized to 129 pixels. Identity templates per user are created by combining five faces with the same pose, resulting in two templates available per user and pose. To test the robustness of the network and the pruning method in more adverse conditions, we also do experiments using only one image as template. A template vector is created by averaging the descriptors of the faces in the template set, which are obtained from the layer adjacent to the classification layer (i.e., the Global Average Pooling). With SqueezeNet, this corresponds to a descriptor of 1000 elements. To further improve performance against pose variation, we also average the descriptor of an image and its horizontally flipped counterpart, which is hypothesized to help to minimize the effect of pose variation [9]. The cosine similarity is then used to compare two given templates.

Table 2: Face verification results on the VGGFace2-Pose database (EER %) without pruning. F=Frontal View. 3/4= Three-Quarter. P=Profile.

	One face image per template (1-1)							Five face images per template (5-5)						
	Same-Pose			Cross-Pose			Over-	Same-Pose			Cross-Pose			Over-
Network	**F-F**	**3/4-3/4**	**P-P**	**F-3/4**	**F-P**	**3/4-P**	**all**	**F-F**	**3/4-3/4**	**P-P**	**F-3/4**	**F-P**	**3/4-P**	**all**
SqueezerFaceNet (ours)	5.32	4.87	7.36	5.09	7.32	6.47	6.07	0.27	0.3	0.85	0.23	0.74	0.75	0.52
SqueezeFacePoseNet [2]	6.39	5.47	7.88	6.09	8.15	7.02	6.34	0.27	0.06	0.54	0.2	1.23	0.88	0.52

Table 3: Number of biometric verification scores.

	SAME-POSE		CROSS-POSE	
Template	**Genuine**	**Impostor**	**Genuine**	**Impostor**
1 image (1-1)	$368 \times (9+8+...+1) = 16560$	$368 \times 100 = 36800$	$368 \times 10 \times 10 = 36800$	$368 \times 100 = 36800$
5 images (5-5)	$368 \times 1 = 368$	$368 \times 100 = 36800$	$368 \times 2 \times 2 = 1472$	$368 \times 100 = 36800$

5 Results

We first report the verification accuracy of SqueezerFaceNet without any pruning in Table 2. This will be the baseline to which we will compare after pruning. We also give the results of SqueezeFacePoseNet from [2]. We detail the results of both same- and cross-pose experiments (Figure 2), as well as the overall performance across all poses. Same-pose comparisons are made with only templates generated with images of the same pose, while cross-pose experiments are done between templates of different poses. Genuine (mated) scores are obtained by comparing each template of a user to the remaining templates of the same user, avoiding symmetric comparisons. For impostor (non-mated) scores, the first template of a user is used as the enrolment template and compared with the second template of the next 100 users. Table 3 shows the total number of scores.

One observation from Table 2 is that SqueezerFaceNet improves the results of [2]. The main differences of the present paper are that we have added batch normalization to the network, we apply random horizontal flip to the training images, we use cosine similarity instead of χ^2 distance to compare vectors, and

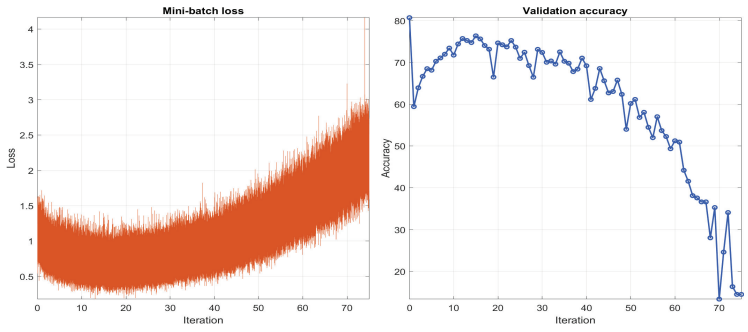

Fig. 3: Mini-batch loss and validation accuracy during the pruning of SqueezerFaceNet. One iteration removes 1% of the filters with the lowest importance scores.

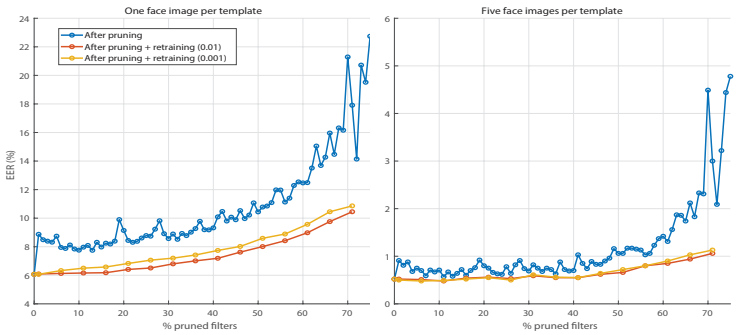

Fig. 4: Face verification results on the VGGFace2-Pose database (EER %) during the pruning of SqueezerFaceNet (overall accuracy across all pose comparison types). One iteration removes 1% of the filters with the lowest importance scores.

we compute an image descriptor by averaging the descriptor of the original image and its horizontally flipped counterpart [9]. These modifications seem to have an overall positive effect. Regarding pose comparison types, it can be seen that the worst performance is given by the most difficult ones, either when the image is only visible from one side (Profile vs. Profile) or when there is a maximum difference between query and test templates (Frontal vs. Profile). It is also worth noting the substantial improvement observed when five images are used to generate user's templates (5-5) in comparison to using one (1-1).

We then apply the pruning of Sect. 2 to SqueezerFaceNet. On each iteration, we use a random 25% of the VGGFace2 training set to compute the importance score of each convolution filter. After each iteration, we remove 1% of the filters with the lowest scores. The optimizer is SGDM with a mini-batch of 128 and a learning rate of 0.01. Figure 3 shows the mini-batch loss and validation accuracy across different iterations. An interesting observation is that the loss decreases a bit until ∼15% of the filters have been pruned and then increases again (the validation loss shows the opposite behavior, as expected). However, after removing just 1% of the filters (first iteration), the validation accuracy decreases sharply from ∼80% to ∼60%, and then it is regained again as the network is pruned up to ∼15% of the filters. Figure 4 (blue curves) shows the overall verification accuracy of the pruned network on the VGGFace2-Pose database. The origin of the x-axis (x=0) corresponds to SqueezerFaceNet without pruning. As can be seen also here, after removing just 1% of the filters, there is a jump towards a worse performance, after which performance is regained a bit until the network is pruned approximately by 10-15%. In five-to-five comparisons (right plot), performance is kept more stable until 30-40% of the network has been pruned, suggesting that combining several face images to create a user template can be a method to counteract the effect of eliminating convolution filters. In one-to-one comparisons, however, accuracy decreases quicker.

After pruning the network with different percentages, we retrain it over VGGFace2 according to the same protocol of the original unpruned network (Sect. 4)

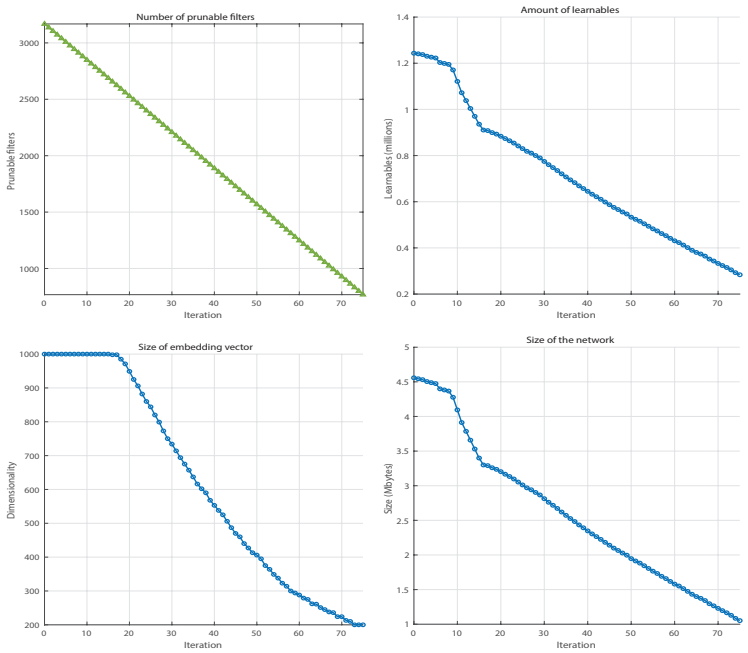

Fig. 5: Effect of pruning in: number of filters (top left), learnables (top right), embedding size (bottom left) and size (bottom right) of SqueezerFaceNet.

in order to regain the accuracy lost during pruning. Given the time that it takes to train the network over the entire VGGFace2, we do the retraining only every 5 iterations of the pruning algorithm (starting at 1%). The results are given in Figure 4 as well. The network is retrained either with a starting learning rate of 0.01 (red curve) or 0.001 (orange). The rationale between these two options is that even if the network is pruned, it has already been trained once over the same database, so starting with a high learning rate may be counterproductive. However, as seen in Figure 4, this is not the case. Indeed, in one-to-one comparisons, the best accuracy is given by starting with 0.01. Regarding the accuracy lost after pruning, it can be seen that training the pruned network again is able to recover the original accuracy up to a certain percentage of pruned filters. In one-to-one comparisons, performance remains stable until ∼15% of the filters have been eliminated. Then, accuracy worsens exponentially. In five-to-five comparisons, on the other hand, performance remains at the same level as the unpruned network until about 30-40%. A remarkable result, in any case, is that after 70% of the filters have been removed, the EER is less than double, so a certain reduction in the number of filters does not translate to accuracy in the same proportion. In five-to-five comparisons, the EER goes from 0.52% (unpruned network) to 1.06% (network pruned at 71%).

We then analyze the effect on the network of the pruning process (Figure 5). Obviously, the number of filters decreases linearly on each iteration (by 1%),

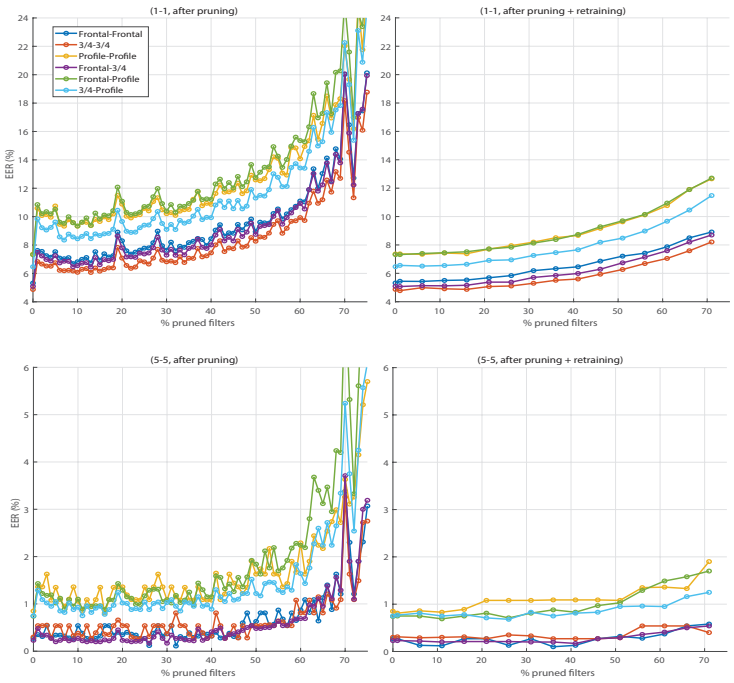

Fig. 6: Face verification on VGGFace2-Pose (EER %) during the pruning of SqueezerFaceNet per pose comparison type (retraining with a starting rate = 0.1). Each iteration removes 1% of the filters with the lowest importance scores.

since we have designed the experiments that way. However, the amount of learnables or the size of the network first decreases slowly until about 10% of the filters are removed. Between 10 and 15%, there is a significant drop in learnables, and then the decrease is stabilized again at a slower pace. This suggests that the filters that are removed first are not big and/or do not affect a high amount of channels, but then, the pruning algorithm removes filters having a larger amount of parameters. Regarding the size of the embedding vector, it is maintained constant until a pruning of about 18%, indicating that the filters that are removed first do not affect the last layer of the network. If we set 15% as the optimal pruning (from Figure 4), it translates to a reduction in parameters from 1.24M to 0.94M (by 24%) and in size from 4.6MB to 3.4MB (by 26%). This is without losing accuracy significantly. In five-to-five comparisons, we could go even higher and prune about 40% of the network, resulting in 0.65M parameters and 2.35MB (a reduction of 48% and 49%, respectively).

We finally give the verification accuracy per pose comparison type after network pruning (Figure 6), with and without retraining. Obviously, the same accuracy gains after retraining are also observed here, and how the performance is maintained until a certain percentage of the filters is removed. It is also more evident the oscillations per iteration when SqueezerFaceNet is pruned but not

retrained (left column), an effect that is alleviated after retraining (right column). Table 4 also details the exact per-pose EER values for different degrees of pruning. It can be observed that the combinations that do not involve profile (P) images result in better performance. Still, in five-to-five comparisons, even the difficult profile-profile (P-P) or frontal-profile (F-P) comparisons provide a very competitive EER of 1% or less. The table also shows the results with two variants of ResNet50 deployed by the authors of the VGGFace2 database [5] having a much higher amount of parameters. They use input images of 224×224 and produce a feature vector of 2048 elements. These two networks clearly stand out in comparison to our SqueezeNet model but at the cost of a larger number of parameters and size (∼150MB), which is infeasible for mobile applications.

Table 4: Face verification results of SqueezerFaceNet on the VGGFace2-Pose database (EER %) with different degrees of pruning (pruned networks retrained with a starting rate = 0.01). F=Frontal View. 3/4= Three-Quarter. P=Profile. Results with two large networks (ResNet50 variants [5]) are also shown.

		One face image per template (1-1)							**Five face images per template (5-5)**						
		Same-Pose			**Cross-Pose**			**Over-**	**Same-Pose**			**Cross-Pose**			**Over-**
Network	**Parameters**	**F-F**	**3/4-3/4**	**P-P**	**F-3/4**	**F-P**	**3/4-P**	**all**	**F-F**	**3/4-3/4**	**P-P**	**F-3/4**	**F-P**	**3/4-P**	**all**
No pruning	1.24M	5.32	4.87	7.36	5.09	7.32	6.47	6.07	0.27	0.3	0.85	0.23	0.74	0.75	0.52
Pruning 16%	0.91M	5.53	4.87	7.38	5.16	7.52	6.64	6.18	0.27	0.31	0.89	0.21	0.75	0.78	0.54
Pruning 31%	0.76M	6.19	5.3	8.2	5.71	8.13	7.26	6.80	0.27	0.33	1.08	0.2	0.81	0.83	0.59
Pruning 46%	0.58M	6.86	5.94	9.17	6.29	9.28	8.19	7.62	0.27	0.27	1.09	0.27	0.97	0.83	0.62
Pruning 51%	0.52M	7.21	6.27	9.63	6.74	9.7	8.48	8.01	0.32	0.29	1.08	0.29	1.03	0.95	0.66
ResNet50ft [5]	25.6M	4.14	3.13	5.16	3.68	4.99	4.25	4.23	0.01	0.02	0.27	0.07	0.14	0.14	0.11
SENet50ft [5]	28.1M	3.86	2.87	4.16	3.36	4.48	3.71	3.74	0.02	0.02	0.2	0.07	0.14	0.2	0.12

6 Conclusions

This paper deals with the task of developing SqueezerFaceNet, a lightweight deep network architecture for mobile face recognition. For such purpose, we apply a CNN pruning method based on Taylor scores which assigns an importance measure to each filter of a given network. Such importance metric is based on the impact on the error if the filter is removed, and it only requires the back-propagation gradient for its computation. The method starts with a network trained for the target task (here: face recognition). Then, it is iteratively pruned by removing filters with the smallest importance. To regain potential accuracy losses, the pruned network is finally retrained again for the target task. The method is applied to an already light model (1.24M parameters) based on a modified SqueezeNet architecture [11]. As training sets, we use the large-scale MS-Celeb-1M (3.16M images, 35K identities) [10] and VGGFace2 (3.31M images, 9.1K identities) [5] datasets. We evaluate two verification scenarios, consisting of using a different number of images to create a user template. In one case, a template consists of five face images with the same pose, following the evaluation protocol of [5]. In the second case, we consider the much more difficult case of

only one image to generate a user template. Different pose combinations between enrolment and query templates are tested too (Figure 2).

Our experiments show that the pruning method is able to further reduce the number of filters of SqueezerFaceNet without decreasing accuracy significantly. This is especially evident if we employ a sufficient number of images to create a user template (five in our experiments). In such case, the number of filters can be reduced up to 40% without an appreciable accuracy loss. In one-to-one comparisons, a more difficult case, a reduction of up to 15% is also feasible. The resulting network in each case has 0.65M and 0.94M parameters, respectively. As future work, we are looking into evaluating the employed pruning method in more powerful CNN architectures which are widely used in face recognition, such as ResNet [7]. If the same effects as in the present paper are observed, it would allow to lower error rates in comparison to the ones obtained in this paper (Table 4), for a fraction of the size of such large networks. We are also considering other alternatives for network compression to evaluate if they are capable of producing even more reductions in network size [1].

Acknowledgements. This work was partly done while F. A.-F. was a visiting researcher at the University of the Balearic Islands. F. A.-F., K. H.-D., and J. B. thank the Swedish Research Council (VR) and the Swedish Innovation Agency (VINNOVA) for funding their research. Author J. M. B. thanks the project EXPLAINING - "Project EXPLainable Artificial INtelligence systems for health and well-beING", under Spanish national projects funding (PID2019-104829RA-I00/AEI/10.13039/501100011033). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan V GPU used for this research. The data handling in Sweden was enabled by the National Academic Infrastructure for Supercomputing in Sweden (NAISS).

References

1. Almadan, A., Rattani, A.: Benchmarking neural network compression techniques for ocular-based user authentication on smartphones. IEEE Access **11** (2023)
2. Alonso-Fernandez, F., Barrachina, J., Diaz, K.H., Bigun, J.: Squeezefaceposenet: Lightweight face verification across different poses for mobile platforms. In: Proc. IAPR TC4 Workshop on Mobile and Wearable Biometrics, WMWB, in conjunction with Intl Conf on Pattern Recognition, ICPR (2020)
3. Boutros, F., Damer, N., Fang, M., Kirchbuchner, F., Kuijper, A.: Mixfacenets: Extremely efficient face recognition networks. In: IEEE Intl Joint Conf on Biometrics, IJCB (2021)
4. Boutros, F., Siebke, P., Klemt, M., Damer, N., Kirchbuchner, F., Kuijper, A.: Pocketnet: Extreme lightweight face recognition network using neural architecture search and multistep knowledge distillation. IEEE Access **10** (2022)
5. Cao, Q., Shen, L., Xie, W., Parkhi, O.M., Zisserman, A.: Vggface2: A dataset for recognising faces across pose and age. In: 13th IEEE Intl Conf on Automatic Face and Gesture Recognition, FG (2018)
6. Chen, S., Liu, Y., Gao, X., Han, Z.: Mobilefacenets: Efficient cnns for accurate real-time face verification on mobile devices. CoRR **abs/1804.07573** (2018), http://arxiv.org/abs/1804.07573

7. Deng, J., Guo, J., Xue, N., Zafeiriou, S.: Arcface: Additive angular margin loss for deep face recognition. In: IEEE/CVF Conf on Computer Vision and Pattern Recognition, CVPR (2019)
8. Deng, J., Guo, J., Zhou, Y., Yu, J., Kotsia, I., Zafeiriou, S.: Retinaface: Single-stage dense face localisation in the wild. CoRR **abs/1905.00641** (2019), http://arxiv.org/abs/1905.00641
9. Duong, C.N., Quach, K.G., Jalata, I.K., Le, N., Luu, K.: Mobiface: A lightweight deep learning face recognition on mobile devices. In: IEEE 10th Intl Conf on Biometrics Theory, Applications and Systems, BTAS (2019)
10. Guo, Y., Zhang, L., Hu, Y., He, X., Gao, J.: Ms-celeb-1m: A dataset and benchmark for large-scale face recognition. In: 14th European Conf Comp Vis, ECCV (2016)
11. Iandola, F.N., Moskewicz, M.W., Ashraf, K., Han, S., Dally, W.J., Keutzer, K.: Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size. CoRR **abs/1602.07360** (2016), http://arxiv.org/abs/1602.07360
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: 32nd Intl Conf Machine Learn, ICML (2015)
13. Jiang, N., Xiong, Z., Tian, H., Zhao, X., Du, X., Zhao, C., Wang, J.: Prunefacedet: Pruning lightweight face detection network by sparsity training. Cognitive Computation and Systems **4**(4), 391–399 (2022)
14. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: Proc IEEE/CVF Conf Computer Vision and Pattern Recognition, CVPR (2019)
15. Martinez-Díaz, Y., Luevano, L.S., Mendez-Vazquez, H., Nicolas-Diaz, M., Chang, L., Gonzalez-Mendoza, M.: Shufflefacenet: A lightweight face architecture for efficient and highly-accurate face recognition. In: Proc IEEE/CVF Intl Conf Computer Vision Workshop, ICCVW (2019)
16. Molchanov, P., Mallya, A., Tyree, S., Frosio, I., Kautz, J.: Importance estimation for neural network pruning. In: IEEE/CVF Conf Computer Vision and Pattern Recognition, CVPR (2019)
17. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. Intl Journal Computer Vision **115**(3) (Dec 2015)
18. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: IEEE/CVF Conf Computer Vision and Pattern Recognition, CVPR (2018)
19. Sundararajan, K., Woodard, D.L.: Deep learning for biometrics: A survey. ACM Comput. Surv. **51**(3) (2018)
20. Tan, M., Le, Q.V.: Mixconv: Mixed depthwise convolutional kernels. In: 30th British Machine Vision Conf, BMVC (2019)
21. Wu, X., He, R., Sun, Z., Tan, T.: A light cnn for deep face representation with noisy labels. IEEE Trans Information Forensics and Security **13**(11) (2018)
22. Yan, M., Zhao, M., Xu, Z., Zhang, Q., Wang, G., Su, Z.: Vargfacenet: An efficient variable group convolutional neural network for lightweight face recognition. In: IEEE/CVF Intl Conf Computer Vision Workshop, ICCVW (2019)
23. Zhang, J.: Seesawfacenets: sparse and robust face verification model for mobile platform. CoRR **abs/1908.09124** (2019), https://arxiv.org/abs/1908.09124
24. Zhang, Q., Li, J., Yao, M., Song, L., Zhou, H., Li, Z., Meng, W., Zhang, X., Wang, G.: Vargnet: Variable group convolutional neural network for efficient embedded computing. CoRR **abs/1907.05653** (2020), https://arxiv.org/abs/1907.05653
25. Zhang, X., Zhou, X., Lin, M., Sun, J.: Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: IEEE/CVF Conf Computer Vision and Pattern Recognition, CVPR (2018)