Distillation of a CNN for a high accuracy mobile face recognition system

Francesco Guzzi, Luca De Bortoli, Stefano Marsi, Sergio Carrato and Giovanni Ramponi

University of Trieste, Department of Engineering and Architecture, Image Processing Laboratory (IPL),
Via A. Valerio, 10, 34127, Trieste, Italy
francesco.guzzi@phd.units.it

***Abstract* - In face recognition systems, the use of convolutional neural networks (CNNs) permits to achieve good accuracy performances, which derive largely from a huge number of well-trained parameters. While using online services any mobile device can suffice for an accurate identification, in the offline scenario, implemented on a wearable mobile hardware, it is difficult to achieve both real-time responsiveness and high accuracy. In this paper we present a solution to replace a large open source face recognizer network (provided as part of the dlib libraries), distilling its learned knowledge into a less demanding CNN. The former is used as an expert oracle that provides the targets, while the latter is trained on the same input image, following a regression approach. In addition to lightness, our CNN is trained to use smaller input images, naturally allowing the recognition of identities in a wider distance range and with a reduced amount of computation. This eventually permits the porting of the network into a dedicated mobile accelerating hardware. The hypothesis we want to demonstrate is that since the feature space topology has been deeply explored during the training of the expert network, and due to the fact that no information is created during the up sampling of a tiny face to the input size of the expert oracle, the smaller network can provide the same accuracy at a reduced computational cost.**

Keywords* - *face recognition, convolutional neural networks, deep learning, mobile, assistive technologies, model compression, knowledge distillation

I. INTRODUCTION

Face recognition system have been made ubiquitous and really accurate by the use of Deep Convolutional Neural Networks (DCNN) that, used as features extractor and classifiers, surpassed the performances of previous hand-crafted algorithm. The feature extractor is the most critical component of the entire signal chain, because it is responsible of generating discriminative and robust descriptors for each identity. Unfortunately, reliable *DCNNs* require *complex* (deep) *structure* that use both an high amount of memory and lots of processing power, making the use of this accurate but cumbersome models practically impossible on mobile hardware. The solution that many big companies offer is to provide an online service, but the inner privacy problem is highly evident.

It has been said that there is the need of a complex structure in order to achieve the complex task of face recognition, but it has to be clarified that *what* really *requires complexity is* the *extraction of* general *characteristics* from the provided samples (during the supervised learning process), *not their* actual *representation*. This means that when this knowledge has been inferred, it can be represented by a simpler structure, that can be deployed to mobile hardware.

In a previous work [1], we identified a framework and a technique for a reliable portable face recognition application, evaluating the performances obtainable using different types of classifiers, trained on a limited number of images for any such identity. The problem can be considered as a *multiclass recognition* in an *open set* scenario. We highlighted the crucial need of a good features extractor which, when paired with a dedicated classifier, allows to get a high *True Positive Rate* (TPR) while maintaining the *False Positive Rate* (FPR) and the incorrect classification rate (*misclassification*) close to zero.

In order to achieve this result, we needed a pre-trained feature extractor whose training typically requires weeks on a high end GPU accelerated hardware and on a huge dataset. Fortunately, the open source project "Dlib" [2] provides many advanced image processing routines, and even a face recognition model (*dlib-resnet-v1*) pretrained on a dataset composed of roughly 3 M images. This Deep CNN is an *embedding network* because, for any given input image (of a face), the model provides a 128-dimensional features vector which virtually belongs to the embedding of that particular identity, meaning that all the possible images of that person would lie in a hyper-sphere of radius lower than 0.6. This value represents the threshold (imposed during the training) which defines a particular identity in a Euclidean space metric. In spite of the undiscussed high accuracy, *dlib-resnet-v1* cannot be deployed satisfactorily on an embedded device with a mid-end ARM CPU, because the time of inference becomes unaffordable without a CUDA compatible GPU.

In this paper we present the results obtained with some new CNNs, with a lower number of parameters with respect to the original Dlib model; these networks are obtained by *distilling* (in a regression approach) the ability of *dlib-resnet-v1* to extract the features descriptor, i.e. its *knowledge*. This in turn will allow the use of such network on a mobile hardware. In order to reduce even more the processing power needed, the image input size has been even reduced by half and despite this, the accuracy of the network is comparable.

II. COMPRESSION TECHNIQUES

Distillation and transfer learning are somehow related but are actually two completely different concepts. ML practitioners apply *transfer learning* in order to train a model in a short period of time on a specific task, using a small not-so-diversified dataset. In the Computer Vision community, we typically use models pre-trained on a huge dataset called *ImageNet;* this procedure has been proven to generate a good set of weights for the convolutional kernels and represents the best available starting point for the training of the same model for a different task. We want to stress that in order to take advance of this sweet spot in the Loss-Parameters space, we need to use the network in exactly the same configuration as the training; this means, for example, that we have to replicate the number of channels of a grayscale image, or scale our image in order to fit the input size.

Model Compression [7] is the first step for reducing model complexity: in a Deep CNN, the majority of the memory requirements is in the fully connected layers, while both the computation time (number of MAC operations) and the buffer memory size are highly related to the convolutional layers. A large speed-up can be obtained with an aggressive quantization [7, 13, 14] and pruning of the weights [7, 15, 16], as well as the decomposition of each fully connected layer weight matrix; in any case the structure of the net remains unchanged.

The form of compression we used [4, 5, 6] takes a completely different approach, that decorrelates the accuracy in a task from the learned weights: what is really important to transfer to a new model is the I/O relationship of the model itself, or the capacity to reveal the latent distribution $p(T|X)$ that relates the inputs X and the outputs T. This capacity is called "*dark knowledge*" [5] and the act of transferring it from a cumbersome but well trained and powerful model is called "*knowledge distillation*" [5]. The ground truth (for example the one-hot encoded target of a classification task) can be seen as a "fact" and no explanation or verification are provided. The well-trained network dark knowledge is its vision of the world, a sort of metric learned while solving the task; this is the precious component that the "*teacher network*" has to teach to a "*student network"* [5] speaking their own "language" (error minimization in the weight/loss space). Said in other words, up to a certain limit, it is possible to disentangle the structure needed for learning to the one used during the inference, transferring the soul (the knowledge) from the teacher network to the student one.

Initially, knowledge distillation was used to train multiclass classifiers, investigating the role of outputs and logits vector [4]; later, the concept of regression of some particular quality of the teacher network by the student network emerged [5]. A recent and exhaustive review of many compression techniques is provided in [7]. With this work we experimentally demonstrate distillation in a metric framework, contrasting its supposedly limited applicability to the classification task.

III. DISTILLATION FRAMEWORK

Distillation is the best way to gather the dark knowledge of a feature extractor network like *dlib-resnet-v1.* That's why we casted the problem as a regression of the features produced by the teacher network. A similar situation (even in the applicability to face recognition) has been described in a 2016 paper [6], where the research team used a regression approach to learn the features generated by a deep network (Deep ID2/3) to a shallow network. The novelty of our work is threefold: 1) we do not use a subset of the features, but the entire vector, permitting a blind swap of the feature extractor network; 2) the entire framework is minimal, since it only requires the regression of the output target; 3) we reduce by half the image size at the input network, permitting the recognition of smaller faces by design, and avoiding the use of expensive upscaling operations for far people recognition. From the best of our knowledge, this type of distillation has never been tried before. While facing a more difficult task, our framework is simpler than the one used in [6]. One possible explanation for this outcome can be imputable to the teacher embedding characteristic: *dlib-resnet-v1* is an embedding network trained using a contrastive/siamese loss, without constraints on the vector norm. A big part of the training time is devoted to centroid positioning of the many identities in the feature multidimensional space. Only when the centroids are stable and far apart, the network actually starts to reduce the intraclass variance of the feature distribution of each ID. When distilling the knowledge into a student network, the centroid positions have been already chosen and the only task for the lighter network is to approximate that distribution. This behaviour is somehow similar to the concept of model reuse in [9].

The train set for the distillation framework, carried out as a supervised learning, is composed of the tuple *(X, T), input* and *corresponding target.* It is important to have the highest possible number of centroids, that are the representation of different individuals. In order to obtain a good cluster, we experimentally saw that is not the number of samples for each identity that counts, but the number of distinct individuals. This would make the student model to infer the latent embedding between faces. In order to synthetically increase the number of these individuals, one idea could be to generate random images and the corresponding targets, but the problem is that those points cover mainly uninteresting parts of the multidimensional space [3].

Further investigation has to be carried out before using generative models for synthesizing new subjects, because the metric learning for those fictitious ids can be tricky without a study on the particular face quality represented by each element of the features vector. This last sentence has not to be viewed as a negligence: we wanted to generalize as much as possible our procedure, without applying custom solutions for the particular *dlib-resnet-v1* network. That's why for the moment we decided to exclude samples generation, and we experimentally verified that its use is not even mandatory for having good results.

The last component of the distillation process is the loss function. Forcing the student network to provide the same descriptor generated by the teacher can be directly described in a distance metric framework: a distance greater than the hypersphere radius of each cluster automatically means a bad learning. In a typical regression problem, the mean square error (MSE) between each target element t_i and the corresponding predicted descriptor y_i

$$L_{MSE} = \frac{1}{N_{dim}} \sum_{i \in N_{dim}} ||t_i - y_i||^2 \qquad (1)$$

is the first choice, but in our study, we have seen that the use of the Euclidean distance L_d

$$L_d = \sqrt{\sum_{i \in N_{dim}} ||t_i - y_i||^2} \qquad (2)$$

as a loss provides the best result, even if it is not critical to use it. Comparing the gradients of each loss (calculated with respect to each element of the generated features vector y_i of the aforementioned losses)

$$\frac{\partial L_{MSE}}{\partial y_i} = -\frac{2}{N_{dim}} ||t_i - y_i|| \qquad (3)$$

$$\frac{\partial L_d}{\partial y_i} = -\frac{2}{\sqrt{L_d}} ||t_i - y_i|| \qquad (4)$$

it can be shown that the coefficient that multiplies the common factor is higher in the case of the distance, thus resulting in a faster convergence (for distances less than ${N_{dim}}^2$). The gradient of the Maximum Absolute Error (MAE) is a sign function, and it is insensitive to different shades of error; we have experimentally verified that in our scenario this is the worst choice.

IV. DISTILLATION EXPERIMENTS

The Dlib network *dlib-resnet-v1* is based on a ResNet-34 structure [17] with few layers removed and the number of filters per layer reduced by half [2]: it has a 150x150 pixel input size, 29 convolutional layers and one fully-connected output layer for a total of 5.58 M parameters.

For the new CNN, we decided to use a more recent architecture such as DenseNet [8] which, for its structure, requires fewer parameters and FLOPs, compared with the ResNet structure, and is consequently less prone to overfitting. We propose different “cuts” of the DenseNet-121 structure, obtaining CNNs with different number of parameters. In analogy with *dlib-resnet-v1*, we set as the output a 128-dimensional fully-connected layer.

In our application, the faces identified at a certain distance in the video stream typically do not exceed 100x100 pixels: for this reason, we fixed the dimension of the input of the new network at 80x80 pixels. We implemented out models using Keras, in order to maximise the compatibility with future implementations on hardware accelerators, e.g. Intel Movidius.

At first we take a *DenseNet-121* model pre-trained on ImageNet, with an 80x80x3 input size: the first columns of Table 1 describe the structure of the reference model where the dense blocks consist of a sequence of bottlenecks and compression layers (DenseNet-BC) as proposed by G.Huang et al. [8]. In the following columns of Table 1 four possible cuts of the *DenseNet-121* are shown on which the proposed tests will follow: for simplicity we will denote the four variants as *‘Net2.5’*, *‘Net2.0’*, *‘Net1.0’* and *‘Net0.5’* with reference to the level of cut. At the selected output layer, we apply an average pooling layer followed by a fully connected one with 128-D output; only for *Net2.5* and *Net2.0* we applied a bottleneck layer in order to adjust the output dimension at 256 before the pooling layer. The described networks have a number of parameters equal to 3.94 M, 1.48 M, 381 k and 123 k respectively.

Layers	DenseNet121 [8]	Net 2.5	Net 2.0	Net 1.0	Net 0.5
Input	Shape (80x80x3)				
Convolution Output shape	Conv 7x7, stride 2, 64 kernels (40x40x64)				
Pooling Output shape	Max pool 3x3, stride 2 (20x20x64)				
Dense Block 1 Output shape	6x basic block (20x20x256)				**2x** basic block (20x20x128)
Transition Output shape	Conv 1x1, 128 kernel + Avg pool 2x2, stride 2 (10x10x128)				
Dense Block 2 Output shape	12x basic block (10x10x512)				
Transition Output shape	Conv 1x1, 256 kernels + Avg pool 2x2, stride 2 (5x5x256)		**Conv 1x1, 256 kernels** (10x10x256)		
Dense Block 3 Output shape	24x basic block (5x5x1024)	**20x** basic block (5x5x896)			
Transition Output shape	Conv 1x1, 512 kernels + Avg pool 2x2, stride 2 (2x2x512)	**Conv 1x1, 256 kernels** (5x5x256)			
Dense Block 4 Output shape	16x basic block (2x2x1024)				
Pooling Output shape	Avg pool 2x2 (1x1024)	Avg pool 5x5 (1x256)	Avg pool 10x10 (1x256)	Avg pool 20x20 (1x256)	Avg pool 20x20 (1x128)
Fully Connected	Output Shape (1x128)				
N° of Parameter	7,17 M	3,94 M	1,48 M	381 K	123 K

Layers	Basic Block
Input	Shape (NxNxK)
Bottleneck Output shape	Conv 1x1, 128 kernel (NxNx128)
Compression Output shape	Conv 3x3, 128 kernel (NxNx32)
Concatenate Output shape	ConCat(Input, Compression) (NxNxK+32)

Table 1. *Densenet121* [8] and our proposed variation. Each "conv" layer corresponds to the sequence BN-ReLU-Conv.

To train the new network we collected a training dataset of 250000 images coming from the Casia [10] and the Vgg [11] dataset. For the following tests we used the Facescub [12] dataset removing numerous disseminated errors in each identity. For each image, the face ROI were extracted using Dlib's HoG face detector, aligned and resizing the corresponding area to 80x80 pixel; at the same time the corresponding features extracted from *dlib-resnet-v1* were collected. Having built this set of (image, target) tuples, we then began to train the proposed network by regression at the features level.

We tried two different normalization schemes both for the images and the features: subtracting their average value, or using the standard scaling; experimentally we have noticed that the best choice is to subtract the respective average values, calculated on our training dataset.

While as said in section 3 we excluded exotic data synthesis procedures, we checked the effect of a simple

data augmentation scheme on the dataset: for a given target (features) of the original image, we tried to rotate (few degrees), to scale and shift (few pixels) the original image, following the concept that a slightly distorted image of the same subject should be mapped theoretically onto the "same" point. This idea, while obviously working during a normal inference, has not been confirmed during this type of training, as it yielded an evident clustering degradation during the tests: a possible explanation for this phenomenon could be related to the concept of "low temperature" target presented by Hinton [4]. Forcing the same label for many augmented images would represent a hard target for the network, that is not able to extract the subtle dark knowledge from the relationship between different features of slightly different images. Following the same idea we would have to find a different strategy for the target generation, but this would imply an *ad hoc* solution for *dlib-resnet-v1,* solution that we want to avoid.

In conclusion, we have trained the four network variants with the described input and output normalization, distance-based loss function, for 30 epochs with 250000 images with 128-size batches, using Adam optimization.

Finally, using the features of the distilled student model, shallow MLP classifiers were trained, for each network, on a variable number of classes to perform tests on the specific problem. The proposed MLP classifiers are composed of three fully connected layers: the first two layers have 100 nodes, while the last one obviously has a number of nodes equal to the number of classes.

V. Simulation Results

The main target of the tests is to verify if the performance obtainable using the features extracted from the proposed simplified networks are comparable to the ones given by *Dlib*, addressing a classification problem similar to the one proposed in [1]. In particular the target is to correctly identify known people, in an environment were also unknown people are present, avoiding as far as possible to exchange a stranger for a known person. An application example could be a blind person who uses the system to correctly recognize friends in a crowded location where several guests are present, avoiding in particular to exchange a stranger for a friend.

To solve this problem, we have developed some multiclass classifiers that after a training based on samples of the known friends would provide, during the inference phase, both the most probable class to which the input sample belongs and a "confidence index", i.e. an estimation of the confidence that the classifier has in identifying that particular class.

Adopting such an index it is possible to discriminate the people who have not been employed in the training set, since for them the confidence index is naturally low. In addition, it is also possible to refine the identification of known people, keeping as true only those values that have a relatively high index and rejecting all the others.

The classifiers adopted in our tests are the Multi-Layer Perceptron (MLP) described in Section IV, that proved to have excellent performances [1]. The output neuron which presents the maximum value identifies the class to which the incoming sample belongs. As a confidence measure, we suggest to use a normalized distance between the two most likely classes according to Eq. 5:

$$C = \frac{d_1 - d_2}{d_1 - d_n} \qquad (5)$$

where d_1, d_2 and d_n are all "logit", i.e. the output of the latest but one layer of the MLP (before the SoftMax operator) respectively of the largest, the second-largest and the smallest value. The proposed measure estimates the confidence through the idea that the smaller is this distance between d_1 and d_2, the less accurate would be the classification and vice-versa. The denominator of Eq. 5 serves to guarantee that the value of C is always between 0 and 1.

An appropriate threshold value for *C* makes it possible to introduce a discriminant between known and unknown identities. For each confidence threshold, all the results under the threshold will be classified as unknown persons while the results above the threshold will be classified as belonging to the class identified by the largest value in the last layer.

To set up a simulation as similar as possible to the actual application of the system, we prepared the following testbench: taking advantage from the well-known FaceScrub [12] image dataset, we initially proceeded to eliminate all the images which present more than one face depicted inside, since in such cases the belonging class suggested in the original database is obviously misleading. Subsequently, all the images have been processed by the four networks under test and by *dlib-resnet-v1*, generating five distinct databases, each one containing the features generated by the corresponding network. All the following operations have been carried out in the same way for all the five databases: the features of each database have been first split into two subsets: the first one, *'K_{dBi}',* would represent features of known people, while a second one, *'U_{dBi}',* would be related to completely unknown people. The former has been created selecting the 165 persons for which FaceSbrub provides at least 130 samples, while U_{dBi} contains all the remaining features. Then all the K_{dBi} databases have been split, as usual, into three different sets for the training, the validation and the testing of the classifier. The testing set is composed by 70 samples for each class (i.e. for each known person), the validation set has 20 samples per class while the training set uses no more than 40 samples per class.

The first test we addressed aims at comparing the performances of the four networks under test with those obtained through Dlib. Performances were analysed with reference to three specific problems, i.e. the identification of a certain number (10, 20 or 50) of known persons, respectively. Classifiers have been trained, separately for each network, using 40 samples for each person while performances have been estimated using a testing set consisting of 70 samples, taken from the testing set, for each known individual plus an equal number of samples taken from U_{dBi}.

The True Positive Rate and the False Positive Rate have been estimated adopting the following equations:

$$TPR = \frac{N_p}{N} \quad (6)$$

$$FPR = \frac{N_f}{N+F} \quad (7)$$

where N_p is the number of correctly classified samples and N is the number of positive samples provided during the test, i.e. the number of the known people samples. N_f is the number of misclassified samples. They are the sum of the number of samples extracted from U_{dBi}, which are classified with a confidence index above the threshold (thus they would be erroneously classified as a known person), plus the number of samples from K_{dBi} which have been misclassified. Finally, F is the number of negative samples, i.e. the number of samples taken from U_{dBi} used in the testing phase.

Please note that these ratios represent in both cases the proportion between a given number of selected samples and the entire number of the provided samples. In the evaluation of the TPR we must consider the proportion between the correctly classified samples and the entire number of positive samples. Thus, in the best case in which all positive samples are correctly classified, the TPR must be equal to one. By contrast, in the second case the total number of cases in which the system could fail is the entire number of proposed samples (rather than just F). In the worst case, indeed, if all the proposed samples belonging both to U_{dBi} and to K_{dBi} fail the classification the value of FPR must be equal to one.

Each test was repeated 10 times by randomly extracting both the identities to be identified from the K_{dBi} database and the unknown ones to U_{dBi}. In Figure 1, the shadowed regions, represents the areas where 99% of the results obtained during the 10 tests are located while the bold line represent the mean curve. The measure has been repeated using all the different features extractors in the 3 cases considered (i.e. 10, 20 and 50 known identities).

It is clear from these figures that the performances of the networks under test can be very close to the ones obtained through the Dlib network. The networks with more parameters perform better than the one with smaller size but obviously when the network size becomes too small, as for instance in the case of *Net0.5,* the performances degrade significantly. Furthermore, it can be seen that classifiers trained on a larger number of known individuals have generally slightly better performances, regardless of the network used to extract the features. Regarding the specific problem addressed in this paper, we want to avoid a stranger to be wrongly classified as a known person, even accepting the cost that some known people could be missed in the identification; it can be noted that, except for *Net0.5*, it is possible to achieve an almost negligible FPR with a TPR largely above 50%. Obviously, other applications could privilege a higher value of TPR but with a much lower FPR.

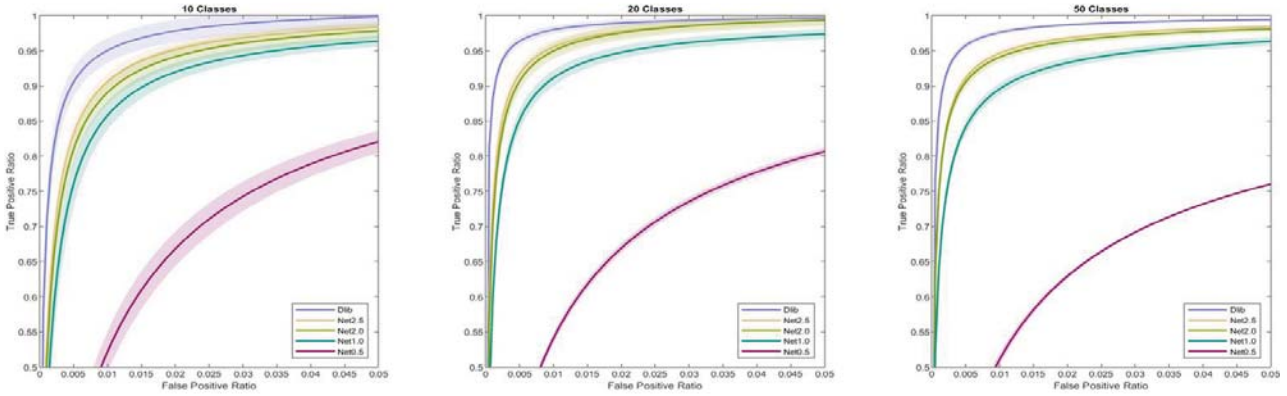

Figure 1. ROC curve: comparing distilled network with the teacher dlib network, in the case of 10, 20, 50 classes.

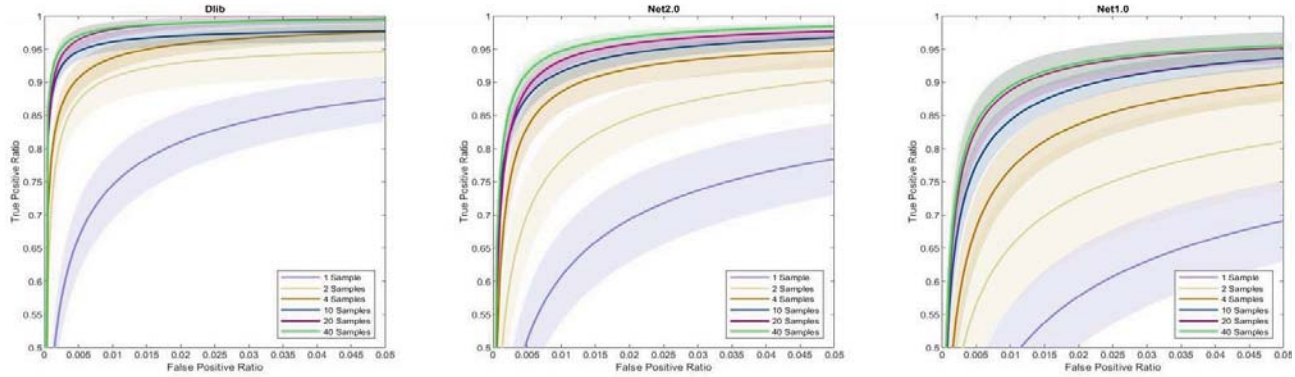

Figure 2. ROC curve: performance variation observed using different training set sizes for the two best distilled network and the teacher dlib.

In a second test, we wanted to determine the number of samples of each known individual that should be used to train the classifier to obtain an acceptable final result. This need arises from the practical constraint that, in real-world applications, the final user wanting to train his/her own classifier has typically very few sample images of each person he/she wants to recognize.

Therefore, in our tests, classifiers for 50 subjects were trained on the features extracted respectively from *Net2.0*, *Net1.0* and *Dlib* networks using respectively 1, 2, 4 10, 20 and 40 samples for each individual. As before, the tests were replicated 10 times by randomly selecting different identities from K_{dBi} and unknown subjects from U_{dBi}. Both the region containing 99% of the results and the average ROC, are depicted in Figure 2.

The results obtained are of high interest as it was observed that, although a training performed with tens of samples obviously leads to better results, the classifier could --when necessary-- be trained using only very few samples for each person without incurring a dramatic degradation of the results. At the lowest level, training the classifier using just one sample for each person provides acceptable results in all the examined cases.

VI. Conclusion

This paper presented a novel knowledge distillation experiment. It was carried out on a large a face recognizer CNN which is provided as part of Dlib. The final products are faster and lighter models built upon many "cuts" of a DensNet121 CNN. The obtained accuracy is on par with that of the original network. The distillation of the dark knowledge has been reached in a teacher-student framework, optimizing the final model as regression problem, in which the former network is used as an expert oracle that provides the targets, while the latter is trained on the same input image. In addition to be lighter, our model is trained to use smaller input images, naturally allowing the recognition of identities in a wider distance range, with lesser computational requirements. The latter outcome is critical to pave the use of the network on a dedicated mobile accelerating hardware. The described procedure can eventually be used for many feature-embedding CNNs.

Acknowledgements

The support of the University of Trieste - FRA projects and of a fund in memory of Angelo Soranzo (1939-2012) is gratefully acknowledged.

References

[1] S. Marsi, L. De Bortoli, F. Guzzi, J. Bhattacharya, F. Cicala, S. Carrato, A. Canziani, G. Ramponi, "A Face Recognition System Using Off-the-Shelf Feature Extractors and an Ad-Hoc Classifier", S. Saponara and A. De Gloria (eds.), Applications in Electronics Pervading Industry, Environment and Society, Lecture Notes in Electrical Engineering 550

[2] "High Quality Face Recognition with Deep Metric Learning", http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html, seen on February 2019

[3] C. Bucilă, R. Caruana, A. Niculescu Mizil, "Model Compression", Knowledge Discovery and Data Mining conference, KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.

[4] G. Hinton, O. Vinyals, J. Dean, "Distilling the knowledge in a Neural Network", Conference on Neural Information Processing Systems, NIPS2014, 8-13 December 2014, Montreal, Canada.

[5] J. Ba, R. Caruana, "Do Deep Nets Really be Deep?", in Advances in Neural Information Processing Systems 27, 2014, pp. 2654-2662.

[6] P. Luo, Z. Zhu, Z. Liu, X. Wang, X. Tang, "Face Model Compression by Distilling Knowledge from Neurons", Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16), Hyatt Regency, Phoenix, Arizona (USA), 12 -17 February 2016.

[7] Y. Cheng, D. Wang, P. Zhou, T. Zhang, "A Survey of Model Compression and Acceleration for Deep Neural Networks", 2017, https://arxiv.org/abs/1710.09282

[8] G. Huang, Z. Liu, L. van der Maaten, K. Q. Weinberger, "Densely Connected Convolutional Networks", in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 4700-4708 .

[9] T. Chen, I. Goodfellow, J. Shlens, "Net2net: accelerating learning via knowledge transfer", International Conference on Learning Representations, ICLR2016, May 2 - 4, 2016 San Juan, Puerto Rico.

[10] D. Yi, Z. Lei, S. Liao, S. Z. Li, "Learning Face Representation from Scratch", 2014, https://arxiv.org/abs/1411.7923

[11] O. M. Parkhi, A. Vedaldi, A. Zisserman, "Deep Face Recognition", British Machine Vision Conference, 2015.

[12] H.-W. Ng, S. Winkler, "A data-driven approach to cleaning large face datasets", Proc. IEEE International Conference on Image Processing (ICIP), Paris, France, Oct. 27-30, 2014. http://vintage.winklerbros.net/facescrub.html

[13] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on cpus", in Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011, 2011.

[14] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision", in Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37, ser. ICML'15, 2015, pp. 1737–1746.

[15] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015, 2015, pp. 31.1–31.12.

[16] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks", in Proceedings of the 28th International Conference on Neural Information Processing Systems, ser. NIPS'15, 7-12 December 2015, Montreal, Canada

[17] K. He, X. Zhang, S. Ren, J. Sun, "Deep Residual Learning for Image Recognition", in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778.