Multilayer Online Self-Acquired Knowledge Distillation

Maria Tzelepi, Charalampos Symeonidis, Nikos Nikolaidis and Anastasios Tefas
Department of Informatics
Aristotle University of Thessaloniki
Thessaloniki, Greece
Email: {mtzelepi, charsyme, nnik, tefas}@csd.auth.gr

***Abstract*—Online knowledge distillation has been proposed as an auspicious approach for circumventing the flaws of the conventional offline distillation (i.e., complex, and computationally and memory demanding process). In this work, a novel online self-distillation method, named *Multilayer Online Self-Acquired Knowledge Distillation* (MOSAKD), is proposed, aiming to develop fast-to-execute and effective models that can comply with applications with memory and computational restrictions, e.g., robotics applications. The MOSAKD method is able to mine additional knowledge both from the intermediate and the output layers of a deep neural model in an online fashion. To achieve this goal, k-nn non-parametric density estimation for estimating the unknown probability distributions of the data samples in the feature space generated by any neural layer is used. This enables us to compute the soft labels that explicitly express the similarities of the data with the classes, by directly estimating the posterior class probabilities of the data samples. The experimental evaluation on four datasets, including a dataset of synthetic images, indicates the effectiveness of the MOSAKD method and the superiority over existing online distillation methods.**

I. Introduction

Deploying state-of-the-art Deep Learning (DL) models on embedded systems is accompanied by specific computation requirements. Knowledge Distillation (KD), [1] has been recognized as a promising approach to address this issue. The typical KD (also known as offline distillation) refers to the process where the knowledge acquired in a -usually- powerful teacher model is transferred to a faster and lightweight student model. To do so, the student model is trained to mimic the so-called soft labels produced by the teacher. These soft labels reveal the similarities of the data with the classes. Among KD methods, we distinguish a subcategory, the so-called self-distillation methods, where the knowledge is transferred to a model of same capacity as the teacher [2], [3].

Even though, KD achieves to provide a solution for effectively training lightweight and fast models, it still suffers from some shortcomings. That is, even in its simpler approach, KD, due to the multiple stages of the training procedure (i.e., training first a complex model and transferring then the knowledge to a faster one) is a time-consuming, and computationally and memory demanding process. Thus, another line of research attempts to circumvent theses flaws by developing distillation methods that simplify the training pipeline in a single stage. That is, *online* KD describes the procedure where the teacher and the student models are trained concurrently, i.e., without the stage of pre-training the teacher model.

Several online KD methods have been proposed in the recent literature [4], [5]. For example, multiple student models teach each other throughout the training process, in [6]. In this work, we propose a single-stage online self-distillation method, named *Multilayer Online Self-Acquired Knowledge Distillation* (MOSAKD), for ameliorating the performance of any deep neural model, regardless of their capacity, for classification tasks. The overarching motivation of this work is to effectively train fast and lightweight models with an extra supervision, apart from the regular supervised loss, that reveals further knowledge beyond the hard labels, from the model itself and also in an online fashion, so as to mitigate the inherent flaws of offline KD caused by the multi-step training pipeline. The idea of this work is also based on the two following remarks: First, it has been shown that useful information can also be mined by transferring the knowledge from a teacher of identical capacity with the student [2], that is, not necessarily more powerful teacher. Second, it has been shown that compact models usually have the same representation capacity as their heavier counterparts, being however harder to be trained [7].

Thus, to accomplish our goal, based on [8], we propose to use the k-nn non-parametric density estimation in order to estimate the unknown probability distributions of the data samples in the feature space generated by any neural layer, that is either by intermediate layers or by the output layer. In this fashion, we are capable of directly estimating the posterior class probabilities of the data samples, based on the neighborhood of each sample, and use them as soft labels. The produced soft labels explicitly express the similarity of each sample with all the classes. It should be emphasized that the proposed method is able to acquire additional knowledge from any layer, and, as shown, useful information can be distilled even from the shallower layers.

The experimental evaluation on four datasets validates the effectiveness of the MOSAKD method. Among them, a synthetic dataset for discriminating between humans and non humans has been created. Synthetic data has become an increasingly useful tool in training DL models, accompanied by a series of benefits [9] with a wide range of robotics applications, e.g., [10]–[12]. In this work, since we are focusing on robotics applications, we use a fully convolutional model which is capable of operating in real-time (25 Frames Per Second - FPS) utilizing a low-power GPU [13] for high resolution input. The target is to provide semantic heatmaps of human presence on real data. That is, we train the real-time model using the proposed online distillation method on the synthetic data (only 100 real human images were used), and we test the model on unseen images that contain real humans, providing semantic heatmaps, as explained in [13].

The rest of the manuscript is organized as follows. Section II discusses relevant online distillation works. Section III presents in detail the proposed MOSAKD method. Subsequently, in Section IV the experiments conducted to evaluate the MOSAKD method are provided, and finally the conclusions are drawn in Section V.

II. Prior Work

In the recent literature, several works proposing online distillation have been proposed. Firstly, co-distillation ameliorates the classification accuracy by training multiple copies of a given model in parallel, by introducing a distillation term to the loss function of one model in order to mimic the average prediction of the rest models in [14]. Next, multiple student models teach each other during the training procedure in [6]. More specifically, every student, apart from the supervised loss, is trained with a distillation loss that matches each student's posterior class probabilities with the class probabilities of the rest students.

Subsequently, in [15] identical branches of a given network create a multibranch version of it. Each branch constites an independent classification model which shares shallow layers. In this way, a strong teacher model is built employing a gated logit ensemble of the aforementioned branches. A supervised loss and a distillation loss are used to trained each branch. Next, [16] proposed a framework of collaborative learning where several classifier heads of the same network are trained on the same training data, concurrently. A population of classifier heads is generated throughout the training procedure, where each head, apart from the hard labels, learns from the soft labels which were generated by the whole population.

Additionally, an online distillation method which combines the previous works [6] and [15] is proposed in [17]. The method proposes an online mutual knowledge distillation method for ameliorating the performance of both the fusion module and the sub-networks. More specifically, when identical sub-networks are used, the low level layers are shared, and a multi-branch architecture similar to [15] is used, while when different sub-networks are used, the sub-networks are trained similar to [6].

Next, a method that aims at recovering the similarities inside classes performs online subclass self-distillation in [8]. In addition, [18] proposes a method that considers all models of various capacities as students and trains them in a single-stage. Finally, [19] proposes Online Knowledge Distillation with Diverse peers, a two-level distillation strategy, where two types of students are involved, i.e., multiple auxiliary peers and one group leader. Distillation is performed among auxiliary peers with a strategy for preserving diversity, and then an ensemble of predictions of these peers is further distilled to the group leader.

III. Proposed Method

In this paper, a novel self-distillation method which allows for developing fast-to-execute and effective models that can comply with applications with memory and computational restrictions is proposed. The proposed method is capable of acquiring additional knowledge about the similarities of the samples with the classes both from the output and the intermediate layers of the model and also in an online fashion.

More specifically, a C-class classification problem, and the labeled data $\{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, where $\mathbf{x}_i \in \Re^D$ is an input vector and D its dimensionality, while $\mathbf{y}_i \in \mathcal{Z}^C$ corresponds to its C-dimensional one-hot class label vector are considered. We also consider, for an input space $\mathcal{X} \subseteq \Re^D$ and an output space $\mathcal{F} \subseteq \Re^C$, as $\Phi(\cdot\,; \mathcal{W}) : \mathcal{X} \to \mathcal{F}$ a deep neural network with $N_L \in \mathbb{N}$ layers, and set of parameters $\mathcal{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_{N_L}\}$, where $\mathbf{W}_L$ are the weights of a specific layer L, which transforms its input vector to a C-dimensional probability vector. That is, for an input vector $\mathbf{x}_i \in \mathcal{X}$, $\Phi(\mathbf{x}_i\,; \mathcal{W}) \in \mathcal{F}$ is the output vector given by the network Φ with parameters $\mathcal{W}$.

Thus, considering a classification problem, our goal is to mine additional knowledge from the model itself in an online fashion. To accomplish this goal, we propose to use k-nn non-parametric density estimation [20] in order to estimate the unknown probability distributions of the data samples in the feature space generated by any neural layer, that is either any intermediate layer or the output layer. The idea of non-parametric density estimation is based, in general, on the fact that the probability P that a vector $\mathbf{x}$ will fall in a region R is given by: $P = \int_R p(\mathbf{x}')d\mathbf{x}'$. Thus, P is a smoothed version of the density function $p(\mathbf{x})$ and this smoothed value of p can be estimated by estimating the probability P. If we consider that N samples $\{\mathbf{x}_i\}_{i=1}^{N}$ are drawn independently and identically distributed according to $p(\mathbf{x})$, then the probability that k of these N lie inside in R is given by the binomial law: $P_k = \binom{N}{k} P^k (1-P)^{N-k}$, and the expected value for k is given by: $\mathbb{E}[k] = NP$.

In addition, this binomial distribution for k peaks very sharply around the mean [20], hence we expect that the ratio $\frac{k}{N}$ will be a good estimate for the probability P, and hence for the smoothed density function. The estimate becomes more accurate as N increases. Making also the assumption that $p(\mathbf{x})$ is continuous and the region R is so small that p does not significantly vary within it, we arrive at the equation: $\int_R p(\mathbf{x}')d\mathbf{x}' \simeq p(\mathbf{x})V$, where $\mathbf{x}$ is a point within R and V is the volume enclosed by R. By combining previous equations the following estimate for $p(\mathbf{x})$ results: $p(\mathbf{x}) = \frac{\frac{k}{N}}{V}$.

This equation can be, in general, exploited in two different manners: either to define the number of nearest neighbors, k, and to adjust the volume V from the data, which stands for the k-nn density estimation, or to define the volume V and to observe how many points k fall into the region, which gives rise to the parzen windows. The essential advantage of the first approach is that it allows for directly estimating the posterior probabilities $P(c_m|\mathbf{x})$ of the class being c_m, $m = \{1, \cdots, C\}$, from a set of N labeled data by using the samples to estimate the densities involved. To do this, we apply the k-nn density estimation to each class separately and then we make use of the Bayes rule. More specifically, assuming that we have a dataset consisting of N_m samples that belong to class c_m, with N samples in total, so that $\sum_m N_m = N$, and we place a sphere of volume V around $\mathbf{x}$ and capture k samples, k_m of which belong to the class c_m. Then, the estimate for the probability $p(\mathbf{x}|c_m)$ is: $p(\mathbf{x}|c_m) = \frac{k_m}{N_m V}$, and similarly the unconditional density is given by: $p(\mathbf{x}) = \frac{k}{NV}$, while the priors can be approximated by: $P(c_m) = \frac{N_m}{N}$.

Therefore, we use the Bayes rule to compute the posterior class probabilities:

$$P(c_m|\mathbf{x}) = \frac{p(\mathbf{x}|c_m)P(c_m)}{p(\mathbf{x})} = \frac{\frac{k_m}{N_m V}\frac{N_m}{N}}{\frac{k}{NV}} = \frac{k_m}{k}. \tag{1}$$

Thus, considering the output of a specific layer, L, and for a specific number of nearest neighbors in terms of Euclidean distance, k, at the feature space generated by the layer L, the posterior probabilities can be estimated and used as soft labels. These soft labels explicitly encode information about the similarities of the samples with the classes. That is, the soft label for a sample i is given by:

$$\mathbf{s}_i^L = [\frac{k_1^L}{k}, \frac{k_2^L}{k}, \ldots, \frac{k_C^L}{k}], \tag{2}$$

where k_m^L, $m = \{1, \cdots, C\}$ is the number of nearest neighbors that belong to the class c_m, $m = \{1, \cdots, C\}$ at the layer L.

Therefore, considering the regular classification task, in the MOSAKD training procedure, the goal is to find the parameters $\mathcal{W}^*$ that minimize the overall loss of cross entropy, ℓ_{ce}, between the output vector $\phi(\mathbf{x}_i\,; \mathcal{W})$ and the one-hot class label vector $\mathbf{y}_i$, and self-distillation, ℓ_{sd} between the output vector $\phi(\mathbf{x}_i\,; \mathcal{W})$ and the soft-label vector $\mathbf{s}_i^L$ (Mean Squared Error is used as self-distillation loss in our experiments). That is:

$$\mathcal{W}^* = \arg\min_{\mathcal{W}} \sum_{i=1}^{N} [\ell_{ce}\big(\mathbf{y}_i, \Phi(\mathbf{x}_i\,; \mathcal{W})\big) + \lambda \ell_{sd}\big(\mathbf{s}_i^L, \Phi(\mathbf{x}_i\,; \mathcal{W})\big)], \tag{3}$$

where λ controls the relative importance between the two losses. Note that this online distillation strategy can be applied to any layer separately, but also to two (or more) different layers at the same time. In the latter case, two (or more) distillation losses are added to the overall loss.

Thus, to recap the proposed MOSAKD training pipeline: the input images are introduced to the network and for each image the predictions for belonging to each of the classes are generated. Concurrently, based on the neighborhood of each image, the soft labels are computed considering the feature space generated by a specific neural layer, according to the process described above. Then, the network is trained using the cross entropy loss with the hard labels, and concurrently using the distillation loss to regress the produced soft labels, encouraging it to regard the similarity of each sample with the classes. In this way, we expect that the model will learn to better generalize, improving its classification accuracy. The MOSAKD training process is also illustrated in Fig. 1.

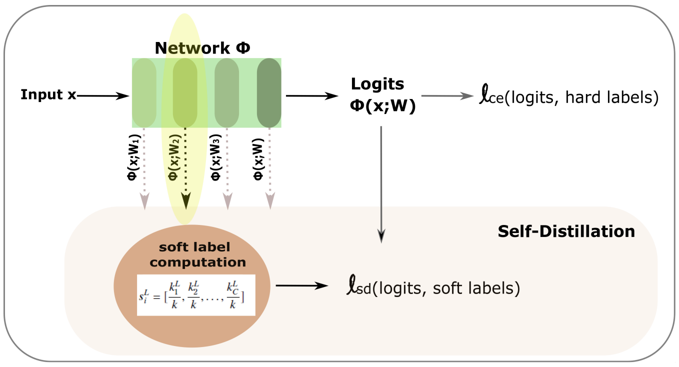

Fig. 1: MOSAKD Training Process: The input images are propagated to the network, and for each sample the predictions for belonging to each of the classes are produced. Concurrently, considering a specific layer the soft labels are computed based on the neighborhood of each sample in the feature space generated by the specific layer. Then, the network is trained at the same time using the cross entropy loss with the hard labels and using the distillation loss with the generated soft labels.

IV. Experiments

A. Datasets

Four datasets are utilized in order to evaluate the effectiveness of the MOSAKD method: *Cifar-10* [21], Street View House Numbers (SVHN) [22], Tiny ImageNet [1], and Synthetic Human datasets. We use the experimental setup that is followed in each utilized dataset, regarding the train-test split: in the case of Cifar-10, 50,000 images are used as train set and 10,000 images as test set, in the case of SVHN, we use 73,257 train images and 26,032 test images, while Tiny ImageNet consists of a training set of 200 classes, each containing 500 images, and a validation set consisting of 50 images.

The Synthetic Human dataset consists of real background images populated with 3D human models in various poses. The human models were generated using PIFu [23], which is a state-of-the-art method for generating realistic 3D human models from single-view images. Overall, the dataset contains 133 human models, generated using full-body images of people from the Clothing Co-Parsing (CCP) [24] dataset as PIFu's input. The background images were taken from the Cityscapes [25] dataset that contains video sequences depicting street scenes in various cities. To accomplish a higher level of realism, the 3D human models are placed on potential 2D image locations (e.g., roads, pavements), based on

[1]https://tiny-imagenet.herokuapp.com/

coarse annotations for semantic image segmentation provided by Cityscapes.

Since, the target is to train models that can run in real-time on high-resolution input for producing heatmaps of human presence [13], we crop the generated images, and we create a train set of 19,900 synthetic cropped images containing humans and 100 real images depicting humans, which are derived from the CUHK Person Re-identification datasets [26], [27]. Furthermore, the train set contains 20,000 non human images, cropped from images of the Cityscapes dataset. The test set consists of 5,000 real images containing humans and 5,000 real images without humans, cropped from video frames that were gathered by querying YouTube video search engine with random keywords. The cropped images are of size 64×64.

B. Model Architectures and Utilized Layers

The objective of this work is to assess the effect of the proposed MOSAKD method on training lightweight models that can be effectively deployed on embedded devices. Towards this end, we first utilize mainly lightweight models (apart from the case of Tiny ImageNet dataset, since it is more challenging). More specifically, in the case of Cifar-10 and SVHN datasets, a lightweight five-layer CNN model is utilized, comprising of 63K parameters. The first two layers are convolutional with 6 filters of size 5×5 and 16 filters of size 5×5 respectively, while the last three layers are fully connected ($128 \times 64 \times 10$). A 2×2 max-pooling layer with a stride of 2 follows the convolutional layers. In the performed experiments we use all the layers for computing the soft labels. The first convolutional layer is denoted as *Layer 1* in the experimental results, and correspondingly the last fully connected layer is denoted as *Layer 5*.

In the case of Tiny ImageNet dataset, we use a more powerful model, i.e., ResNet-50 [28] (without utilizing any pre-trained model). This also validates our claim that the proposed MOSAKD method is model agnostic. In our experiments, we use the last fully connected layer denoted as *Layer 5*, and the output of the four convolutional blocks before the aforementioned last fully connected layer, where the last convolutional block is denoted as *Layer 4*, and so on.

In the case of Synthetic Human dataset, we use the fully convolutional lightweight VGG-1080p model [13] comprising of 11K parameters. The model, which consists of five convolutional layers, runs in real-time for input 1080p on a low-power Jetson TX-2. We use all the convolutional layers for the soft labels computation, where the first convolutional layer is denoted as *Layer 1* in the experimental results, and correspondingly the last convolutional layer is denoted as *Layer 5*.

Finally, we use Wide ResNet 20-8 (WRN-20-8) model [29] in order to compare the MOSAKD method with state-of-the-art online KD approaches on Cifar-10 dataset. In the performed experiments, we use the last fully connected layer denoted as *Layer 4*, and the output of the four convolutional blocks before the last fully connected layer, where the last convolutional block is denoted as *Layer 3*, and so on.

C. Experimental Setup and Evaluation Metrics

In this work, four set of experiments are performed. First, the performance of the MOSAKD method using different layers of the utilized models, and different numbers of nearest neighbors, k, in eq. (2) (i.e., 8, 12, and 16) for computing the soft labels, using mainly lightweight models is evaluated. Classification accuracy is used as evaluation metric (each experiment is repeated five times, and the mean value of classification accuracy and the standard deviation are reported). In the second set of experiments, we compare the performance of MOSAKD with state-of-the-art online distillation methods, using a more complex and powerful model. Next, in the third set of experiments, the efficiency of the proposed method is evaluated utilizing the sum of floating point operations (FLOPs), and the memory requirements for training. Finally, in the fourth set of experiments, some qualitative results are provided using the real-time model. The trained model is used to generate heatmaps of human presence on real high-resolution test images.

D. Implementation Details

In this paper, we train our models using the mini-batch gradient descent with mini-batch of 64 samples, and we set the momentum to 0.9 and the learning rate to 10^{-3}. Additionally, we set the parameter λ in eq. (3) for controlling the relative importance between the classification and the distillation losses to 0.1 for all the datasets except for the Tiny ImageNet where it is set to 0.01. The method is implemented in Pytorch, and the models are trained for 100 epochs on an NVIDIA 2080 Ti with 11GB of GPU memory.

E. Experimental Results

In the following the four sets of experiments are presented.

a) Evaluation using lightweight models: The experimental results for evaluating the performance of the MOSAKD method using the lightweight models on the Cifar-10, SVHN, Tiny ImageNet, and Synthetic Human datasets are presented in Tables I-IV respectively. The best results, considering the different utilized layers, for each specific number of nearest neighbors are printed in bold, while the best performance in each dataset is also underlined.

From the demonstrated results, it is obvious that MOSAKD method remarkably ameliorates the baseline performance of training without distillation on all the utilized datasets, using different layers and numbers of nearest neighbors for computing the soft labels. Interestingly, considerably good performance can be accomplished using the first layers for the soft label computation, apart from the output layer which is mainly used in the KD approaches. However, we can not draw any conclusion on the optimal layer for computing the soft labels, since it is evident that, depending on the dataset, both first and output layers convey useful information that allow for producing reliable soft labels.

In addition, it can be observed that the proposed method considerably ameliorates the performance on all utilized datasets regardless of the number of classes. That is, it ameliorates the performance of 2-class, 10-class, and 200-class problems. In the case of the latter ones, inherently rich information about the similarities with the classes can be conveyed, leading to improved performance. However, it is evident that the proposed method can acquire useful information in the case of binary problems too, considerably improving the performance. This can be attributed to the contextual information that the soft label reveals.

TABLE I: Cifar-10: Classification accuracy of the proposed MOSAKD method utilizing different layers and number of nearest neighbors. Baseline: $64.734\% \pm 0.654\%$.

NN / Layer	8NN	12NN	16NN
Layer 1	66.076% ± 0.241%	65.818% ± 0.560%	65.994% ± 0.659%
Layer 2	65.660% ± 0.710%	**66.756% ± 0.953%**	66.423% ± 1.100%
Layer 3	65.752% ± 0.534%	65.318% ± 1.035%	65.856% ± 0.638%
Layer 4	**66.176% ± 0.979%**	66.561% ± 0.635%	66.468% ± 0.587%
Layer 5	65.729% ± 0.424%	65.962% ± 0.841%	**66.570% ± 0.563%**

TABLE II: SVHN: Classification accuracy of the proposed MOSAKD method utilizing different layers and number of nearest neighbors. Baseline: $88.706\% \pm 0.306\%$.

NN / Layer	8NN	12NN	16NN
Layer 1	89.393% ± 0.243%	89.175% ± 0.552%	89.283% ± 0.303%
Layer 2	**89.619% ± 0.329%**	89.275% ± 0.219%	89.568% ± 0.460%
Layer 3	89.606% ± 0.104%	**89.644% ± 0.203%**	**89.736% ± 0.187%**
Layer 4	89.256% ± 0.388%	89.612% ± 0.234%	89.708% ± 0.279%
Layer 5	89.318% ± 0.204%	89.560% ± 0.198%	89.422% ± 0.330%

TABLE III: Tiny ImageNet: Classification accuracy of the proposed MOSAKD method utilizing different layers and number of nearest neighbors. Baseline: $31.050\% \pm 1.550\%$.

NN / Layer	8NN	12NN	16NN
Layer 1	31.131% ± 0.876%	31.270% ± 1.022%	31.695% ± 0.724%
Layer 2	31.101% ± 1.120%	31.236% ± 0.551%	**32.245% ± 0.527%**
Layer 3	31.215% ± 1.020%	31.800% ± 0.811%	31.381% ± 0.741%
Layer 4	**32.070% ± 0.600%**	**31.895% ± 0.635%**	31.588% ± 0.788%
Layer 5	31.631% ± 0.519%	31.357% ± 1.045%	31.556% ± 0.643%

TABLE IV: Synthetic Human: Classification accuracy of the proposed MOSAKD method utilizing different layers and number of nearest neighbors. Baseline: $97.048\% \pm 0.476\%$.

NN / Layer	8NN	12NN	16NN
Layer 1	**98.275% ± 0.550%**	98.192% ± 0.801%	98.223% ± 1.288%
Layer 2	98.096% ± 1.129%	**98.322% ± 0.738%**	97.914% ± 0.641%
Layer 3	96.814% ± 1.916%	97.084% ± 1.113%	98.046% ± 0.908%
Layer 4	97.264% ± 1.482%	98.192% ± 1.082%	97.404% ± 1.129%
Layer 5	97.946% ± 0.899%	98.268% ± 0.556%	**98.376% ± 0.611%**

Furthermore, we have performed experiments using different values of λ in eq. (3), in order to capture the trade-off between the main classification and the distillation loss. More specifically, we performed experiments on Cifar-10 dataset, utilizing 12NN. The experimental results are illustrated in Table V. As it is shown, the proposed online distillation method

TABLE V: Cifar-10 Dataset: Classification Accuracy for different values of parameter λ in eq. (10) using 12NN for the soft label computation.

Layer	$\lambda = 0.1$	$\lambda = 0.01$	$\lambda = 0.001$
1	65.818% ± 0.560%	65.797% ± 0.495%	65.477% ± 0.691%
2	66.756% ± 0.953%	65.202% ± 0.833%	65.472% ± 0.810%
3	65.318% ± 1.035%	65.789% ± 0.632%	65.240% ± 0.880%
4	66.561% ± 0.635%	65.520% ± 1.041%	65.057% ± 0.659%
5	65.962% ± 0.841%	65.840 %± 0.745%	65.729% ± 0.136%

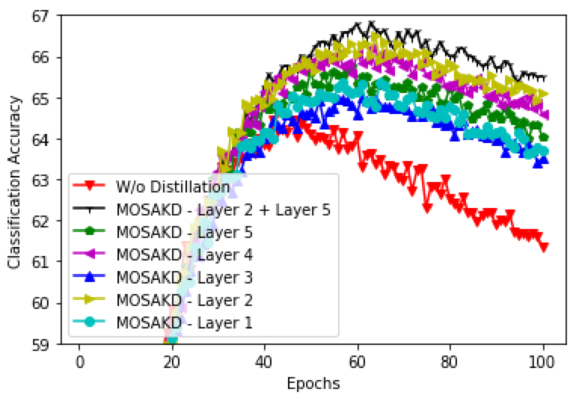

Fig. 2: Cifar-10: Evaluating the classification accuracy throughout the training epochs utilizing 12NN, and different layers for computing the soft labels.

provides generally better performance for the highest value of λ (the bigger the value of λ, the bigger improvement, too), however it should be emphasized that the MOSAKD method ameliorates the classification accuracy in all the considered cases.

Finally, the proposed method, as previously mentioned, can also be applied on two (or more) different layers at the same time. In this case, two distillation losses are added to the overall loss. This strategy can in some cases further improve the performance accomplished by using each of the layers separately. For example, in the case of the Cifar-10 dataset, we can achieve classification accuracy 67.118±0.251 by combining the second and fifth layer, using 12NN, which is the best performance on Cifar-10 dataset. In Fig. 2 the curves of mean classification accuracy, utilizing 12NN, and various layers for computing the soft labels are illustrated.

b) Comparison with state-of-the-art: In the second set of experiments, we compare the proposed method with state-of-the-art online distillation methods. More specifically, we utilize the WRN-20-08 model and compare the performance of the MOSAKD method with the recent OKDDip [19] method, as well as with DML [6], ONE [15], and CL-ILR [16] methods. We perform experiments on Cifar-10 dataset. To ensure the fairness of comparisons, we use the same experimental settings as in [19]. That is, stochastic gradient descent is used with Nesterov momentum. The initial learning rate is set to 0.1 and is divided by 10 at 150 and 225 epochs, while the networks are trained for 300 epochs. Finally, we consider batch size of 128 samples. The experimental results are provided in

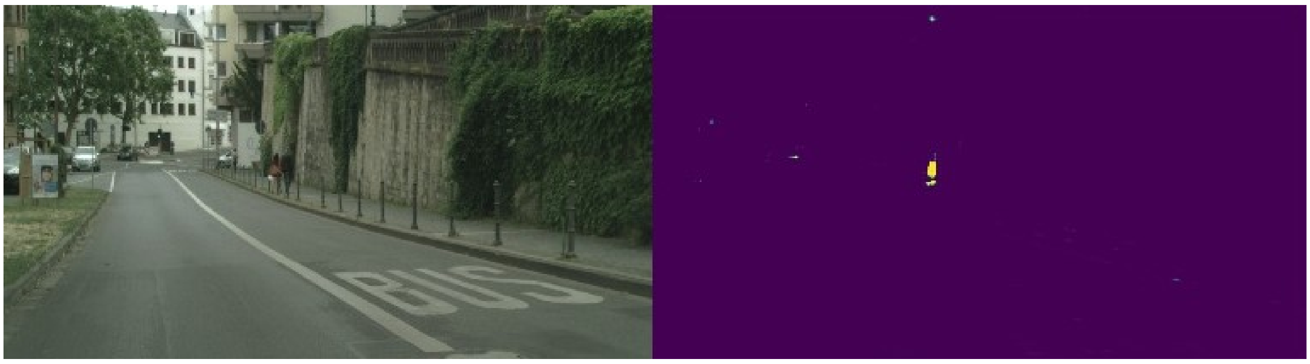
Fig. 3: Heatmap on real-image containing humans utilizing the human detection model trained on synthetic humans.

TABLE VI: Comparison against state-of-the-art online distillation methods on Cifar-10 utilizing the WRN-20-8 architecture.

Method	**Classification Accuracy**
WRN-20-8	94.73% ± 0.06%
DML [6]	94.96% ± 0.08%
ONE [15]	94.73% ± 0.02%
CL-ILR [16]	94.88% ± 0.16%
OKDDip [19]	95.16% ± 0.07%
MOSAKD - Layer 1	**96.00% ± 0.06%**
MOSAKD - Layer 2	**96.19% ± 0.10%**
MOSAKD - Layer 3	**96.09% ± 0.07%**
MOSAKD - Layer 4	**96.01% ± 0.11%**

Table VI. As it can be observed from the provided results, the MOSAKD method significantly improves the baseline, and achieves superior performance over the state-of-the-art online distillation methods.

c) Evaluation of efficiency: In the third set of experiments, the efficiency of the MOSAKD method is evaluated. First, we evaluate the training cost (complexity) utilizing the FLOPs considering one forward pass. The experiments are performed on the Cifar-10 dataset, using the WRN-20-8 model. The MOSAKD method requires 2.46 GFLOPs. To gain further insights on the efficiency of the proposed method, we compare the training cost with the most well-known offline KD methodology, [1]. For the offline KD, the more powerful WRN-40-8 is used as teacher to mine additional knowledge. The offline KD method requires 7.64 GFLOPs. We also emphasize, that apart from the expected superiority over the offline methodology, due to the fact the proposed method does not require utilizing a separate teacher model for realizing the distillation process, the MOSAKD method is also more cost-effective against existing online distillation methods, since they use at least two copies of the network to mine additional knowledge, and hence they require at least two times more FLOPs than the MOSAKD method.

Finally, the utilized lightweight models are extremely low-memory demanding, considering the memory required for training them using the proposed training pipeline. More specifically, the required memory to train a lightweight model with the proposed method for example on CIfar-10 dataset is 920 MiB. To gain some more insights on the efficiency with respect to the memory requirements, we can compare the performance of the MOSAKD methodology with the conventional offline KD using the WRN-20-8 model. The MOSAKD method requires 2871 MiB, while for training first the more powerful WRN-40-8 model and transferring the knowledge to the WRN-20-8 model with the offline KD methodology are required 7340 MiB.

d) Qualitative Results: Finally, in the fourth set of experiments, we use the proposed trained model on the synthetic human dataset to produce heatmaps on unseen high-resolution images that contain real humans. That is, unseen images of size 1920×1080 are fed to the network, and for every window 64×64 we compute the output of the network at the output layer. Indicative evaluation results are illustrated in Fig. 3. As it shown, the model can successfully detect humans on real images.

V. Conclusions

In this paper, a novel online self-distillation approach was proposed, named Multilayer Online Self-Acquired Knowledge Distillation. The MOSAKD method is capable of acquiring additional knowledge either from the output or from the intermediate layers of the model, without modifying it, e.g., by introducing multiple copies of the model, and simultaneously in an one-stage training pipeline. Interestingly, it is evident that apart from the output layer of the model, which is mainly used in distillation, intermediate layers can also provide meaningful information. The effectiveness of the proposed method to ameliorate the classification accuracy of any model, regardless of their complexity, is experimentally validated on four datasets, including a synthetic dataset.

Acknowledgments

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871449 (OpenDR). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

References

[1] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015.

[2] T. Furlanello, Z. C. Lipton, M. Tschannen, L. Itti, and A. Anandkumar, "Born again neural networks," in *ICML*, 2018.

[3] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[4] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[5] M. Tzelepi, N. Passalis, and A. Tefas, "Probabilistic online self-distillation," *Neurocomputing*, 2022.

[6] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[7] J. Ba and R. Caruana, "Do deep nets really need to be deep?" in *Proceedings of the Advances in Neural Information Processing Systems 27*, 2014, pp. 2654–2662.

[8] M. Tzelepi, N. Passalis, and A. Tefas, "Efficient online subclass knowledge distillation for image classification," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 1007–1014.

[9] S. I. Nikolenko, "Synthetic data for deep learning," *arXiv preprint arXiv:1909.11512*, 2019.

[10] D. Ward, P. Moghadam, and N. Hudson, "Deep leaf segmentation using synthetic data," *arXiv preprint arXiv:1807.10931*, 2018.

[11] Z. Tang, M. Naphade, S. Birchfield, J. Tremblay, W. Hodge, R. Kumar, S. Wang, and X. Yang, "Pamtri: Pose-aware multi-task learning for vehicle re-identification using highly randomized synthetic data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 211–220.

[12] Y. Lin, C. Tang, F.-J. Chu, and P. A. Vela, "Using synthetic data and deep networks to recognize primitive shapes for object grasping," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 10 494–10 501.

[13] M. Tzelepi and A. Tefas, "Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization," *Pattern Recognition*, vol. 106, p. 107407, 2020.

[14] R. Anil, G. Pereyra, A. Passos, R. Ormandi, G. E. Dahl, and G. E. Hinton, "Large scale distributed neural network training through online distillation," 2018.

[15] x. lan, X. Zhu, and S. Gong, "Knowledge distillation by on-the-fly native ensemble," in *Proceedings of the Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 7517–7527.

[16] G. Song and W. Chai, "Collaborative learning for deep neural networks," *Advances in Neural Information Processing Systems*, vol. 31, pp. 1832–1841, 2018.

[17] J. Kim, M. Hyun, I. Chung, and N. Kwak, "Feature fusion for online mutual knowledge distillation," *arXiv preprint arXiv:1904.09058v1*, 2019.

[18] Q. Guo, X. Wang, Y. Wu, Z. Yu, D. Liang, X. Hu, and P. Luo, "Online knowledge distillation via collaborative learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 020–11 029.

[19] D. Chen, J.-P. Mei, C. Wang, Y. Feng, and C. Chen, "Online knowledge distillation with diverse peers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 3430–3437.

[20] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[21] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Citeseer, Tech. Rep., 2009.

[22] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[23] S. Saito, Z. Huang, R. Natsume, S. Morishima, A. Kanazawa, and H. Li, "Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019.

[24] W. Yang, P. Luo, and L. Lin, "Clothing co-parsing by joint image segmentation and labeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[25] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *PProceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[26] W. Li, R. Zhao, and X. Wang, "Human reidentification with transferred metric learning," in *ACCV*, 2012.

[27] W. Li and X. Wang, "Locally aligned feature transforms across views," in *CVPR*, 2013.

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

[29] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proceedings of the British Machine Vision Conference*, 2016, pp. 87.1–87.12.