Received 9 September 2023, accepted 19 September 2023, date of publication 18 October 2023,
date of current version 2 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3325134

RESEARCH ARTICLE

Deep Learning-Based Eye Gaze Estimation for Automotive Applications Using Knowledge Distillation

**IOAN LUCAN ORĂŞAN, ADRIAN-IOAN BUBLEA,
AND CĂTĂLIN DANIEL CĂLEANU, (Member, IEEE)**
Department of Applied Electronics, Faculty of Electronics, Telecommunications and Information Technologies, Politehnica University Timisoara, 300006 Timişoara, Romania

Corresponding author: Cătălin Daniel Căleanu (catalin.caleanu@upt.ro)

ABSTRACT Deep neural networks are currently applied in multiple domains, especially in the automotive industry. The main reason for this is related to the more complex challenges found in the field of signal processing, especially when the tasks involve image and video data types. Using conventional/statistical algorithms to deal with these high-complexity challenges is no longer a viable approach. Therefore, the involvement of artificial intelligence solutions like deep neural networks has significantly increased. In recent years, numerous architectures have been developed with the aim of maximizing performance. However, their size and computation requirements have increased at the same time. For this reason, special attention is currently being paid to the optimization of deep neural networks while trying to maintain (almost) the same performance. In this work, we aim to tackle the problem of eye gaze estimation considered within the automotive framework. Our proposal uses a knowledge distillation concept applied to a custom CNN architecture, called the teacher model. Based on this, several CNN student models are derived using layerwise and widthwise compression techniques. Furthermore, they are evaluated with respect to certain performance metrics, e.g. neural network size and inference time. In the experimental results, we propose certain compression methods which can address specific user requirements like model size, accuracy, and inference time. Finally, the student models are evaluated using an EdgeAI embedded device (STM32H747I-DISCO) in terms of accuracy, memory utilization, MACC complexity, and inference time. The combination of layerwise and widthwise compression results as the optimal method to derive student models with a good trade-off between the above-mentioned metrics. Using knowledge distillation, the accuracy can be improved by up to 9.5% over the conventional training procedure.

INDEX TERMS ARM Cortex-M, convolutional neural networks, eye gaze estimation, knowledge distillation, microcontrollers.

I. INTRODUCTION

Deep neural networks (DNNs) have provided state-of-the-art performance in various domains, such as image classification and recognition [1], object detection [2], video processing [3], and segmentation [4].

The domain of gaze estimation holds significant importance within the context of human-computer interaction, and it can find multiple applications across diverse fields, including the automotive industry, market research, and medical domain.

Currently, driver inattention continues to be a significant contributing factor in automobile collisions. The implementation of Advanced Driver Assistance Systems (ADAS) has been proposed as a potential solution to mitigate the incidence of car accidents caused by driver inattention. The driver's gaze can serve as an indicator to detect fatigue or lack of attention while driving. In such situations, it is possible to transmit warnings to the driver and, if necessary, take appropriate measures to avoid a collision [5].

The associate editor coordinating the review of this manuscript and approving it for publication was Prakasam Periasamy.

The application of gaze estimation in market research involves the evaluation of customer attention toward specific brands or products. The data obtained can provide information on customer search patterns, purchasing preferences, and their navigation within the store [6].

Gaze estimation has potential applications in the medical domain [7], including the diagnosis of autism spectrum disorder (ASD) and the aid of individuals with disabilities in their daily activities, by utilizing their gaze to operate electronic devices such as smartphones, computers, and televisions. Through the integration of various visual cues, such as deliberate blinking and eye movements, people with disabilities may be able to effectively operate the aforementioned devices.

To obtain a lightweight convolutional neural network (CNN) the following high-level processes are commonly used: (1) optimizing and compressing a pre-trained model, and (2) designing and training from scratch a light model architecture.

To optimize and compress a pre-trained model is a more convenient process, as many pre-trained CNN models are available for a wide range of applications. Different methods applicable to pre-trained models are presented in a recent survey paper [8]. The authors show a comparative study on structure-preserving compression techniques with the aim of providing insights into the most appropriate method for a given task. These techniques can be summarized in the following categories: connection pruning, weight quantization, and low-rank matrix / tensor (weight) decomposition. Connection pruning is used to remove connections from a network that have a small contribution to overall performance. To identify the connections, a commonly used strategy is magnitude-based pruning. The basic concept of this strategy is to define a threshold that can be global or layer-specific as a condition to eliminate a connection [9]. Weight quantization is applied with the aim of decreasing the memory footprint by reducing the numerical precision from 32-bit floating point representation to 16-bit floating point half-precision or 8-bit integer arithmetic. Furthermore, representations with less than 8 bits have already been proposed, and even binarization [10]. I. Orasan et al. [11] investigated several post-training quantization solutions using the TensorFlow Lite deep learning framework on CNN models of different sizes. The obtained compression ratio is up to 4 times, and the worst-case accuracy degradation is only 0.43%. The low-rank matrix and tensor decomposition is a method applied to the weight matrices by decomposing them into a lower-rank approximation. Singular Value Decomposition (SVD) [12] is the widely used algorithm for this purpose. However, the compression of a pre-trained model is subject to performance degradation. For example, when using magnitude-based connection pruning, a fine-tuning training step might be necessary.

To design from scratch a light model architecture involves applying certain optimizations to avoid a computationally expensive model. The SqueezeNet [13] architecture is a common example that has been developed in this direction with the aim of obtaining a smaller neural network with fewer parameters. The application of Neural Architecture Search (NAS) methodologies has also been widely exploited in recent years. For example, E-DNAS [14] is a differentiable architecture search method for embedded systems that computes the optimal kernel size that captures input data patterns at different resolutions. To reduce the number of operations, the authors of [14] make use of the additive property of convolution operations to merge kernels with different compatible sizes into one. However, the use of NAS methodologies is typically a long and time-consuming procedure.

Another algorithm that involves changes in architecture and training from scratch is knowledge distillation [15]. The basic concept using this method is to transfer the knowledge from a high-performance model, called the teacher, to a smaller one, the student. This transfer is performed during the training process using a specific knowledge distillation algorithm. Compared to the teacher model, the student can use a different architecture or can be a modified version of the teacher model. The most common approach found in the existing literature is to use a different architecture for the student network. However, using the modified teacher model could lead to several improvements such as: (1) the conventional training efficiency can be similar to the teacher model, as the student architecture is based on the teacher, (2) the training with knowledge distillation can have better results, especially when the student model is not significantly different than the teacher one. These improvements are more visible when the student model is not very different compared to the teacher model.

In this work, we use the knowledge distillation algorithm proposed in [15] to train a CNN model to solve the problem of estimating eye gaze for automotive application. Student models are derived from the teacher model by using the layerwise and widthwise compression methods. To find the optimal trade-off between accuracy and model size, additional student models are defined using a combination of layerwise and widthwise compression. The aim of this CNN model optimization is the neural network deployment on a resource-constrained device, such as low-cost microcontrollers based on ARM Cortex M cores. Motivation is justified by the availability of some neural models that have already been developed and tested in other applications using ARM Cortex-M-based processors [16], [17]. Furthermore, we tested and validated student models using the X-CUBE-AI STM32Cube expansion package [18] on the STM32H747I-DISCO development board. The STM32H747I-DISCO hardware is built around a dual core microcontroller based on ARM Cortex-M7 and -M4.

In summary, this work makes the following contributions.

1) We propose a custom CNN architecture as a teacher model for eye gaze estimation that has only 6.14MB size with 77.48% accuracy.

2) We apply knowledge distillation methodology with different specific parameters to train various student models obtained using layerwise and widthwise compression methods.

3) We present and discuss the experimental results to find the most appropriate compression method considering user requirements (model size, accuracy, and inference time).

4) We validate the student models using the AI validation application of the X-CUBE-AI package on the STM32H747I-DISCO hardware device, providing details regarding accuracy, memory utilization, MACC complexity and inference time.

This paper is organized as follows. Section II refers to representative approaches in the domain of eye gaze estimation. It also emphasizes the need for neural network compression and optimization. In addition, it includes an in-depth analysis of the most recent knowledge distillation methods along with their different applications. Section III shows the knowledge distillation algorithm in more detail and the approach used for the definition of the teacher and student models. Section IV presents the experimental results related to knowledge distillation parametrization, inference time computation, hardware and software configuration, knowledge distillation results obtained on the host platform, and validation on the target device, respectively. The section also provides information regarding knowledge distillation efficiency on both host and target platforms. Section V concludes the paper.

II. RELATED WORK

Next, we briefly describe the eye gaze estimation problem along with several state-of-the-art deep learning solutions, then review a few recent representative papers that prove the efficiency of the knowledge distillation algorithms for different applications. We end the section by referring to the analysis paper [19] that follows a similar approach to our work.

A. EYE GAZE ESTIMATION

The estimation of a person's gaze through the use of DNN is an intensively researched topic in the field of biometrics today. Gaze estimation techniques have evolved over time from single-user applications operating within constrained environments to multiuser applications in complex unconstrained environments that involve lots of variations.

The current gaze estimation approaches are classified into two main categories: model-based and appearance-based approaches.

The first category implies the usage of a geometric model of the eye that includes ocular components such as the corneal reflection or the pupil. This technique requires high-resolution images or calibration from person to person to accommodate the differences in the shape of the eye, thus making it more time consuming; it also gives a more accurate result.

The appearance-based approach involves a direct mapping from images or eye features to gaze directions and is believed to work well in real-world scenarios due to a model's generalization capability. This kind of approach usually does not involve person-to-person calibration as in the previous method. This appearance-based approach can use low-resolution images and middistance scenarios.

In [20] Zhuang et al. propose a simplified gaze estimation network model based on a combination of LeNet and Xception neural network architectures, named SLeNet. To improve the network model's computational performance and cut down the number of parameters utilized in the convolution stage, the Xception network makes use of a technique known as deep separable convolution. The method of splicing the features of head posture is kept, and another branch neural network was designed to learn the head posture based on the image of the information given by the eye and the corners of the mouth. The experiment was carried out on the MPIIGaze dataset, and this method shows that SLeNet uses fewer parameters (923.989) and produces a lower MSE loss of 2.6971 accuracy and a better training and inference time compared to the traditional LeNet and VGG-16 network architectures.

A paper describing a gaze estimation system based on cooperative network CI-Net that makes use of a consistency (C-Net) and inconsistency (I-Net) estimation network is proposed in [21] by Luo et al. The consistency estimation network is used to estimate the main gaze (a measure calculated based on the monocular directions of the left and right eye) of the true gaze. Weight assignment between the eyes and face features is done adaptively by an attention mechanism. The residuals based on the true gaze are estimated by the inconsistency estimation network. To obtain more accurate eye directions, I-Net is able to get information from C-Net in a selective manner using a cross-attention module. The experiments were conducted using MPIIGaze (angular error: 3.8 degrees), EyeDiap (angular error: 5.4 deg) and RT-Gene (angular error: 7.9 deg) datasests and it shows that the CI-Net yields lower angle errors than current CNN mainstream methods.

Z. Zichen et al. researched a monocular gaze estimation method [22]. The paper addresses the issue that most of the current appearance-based gaze estimation papers use both eyes to predict and estimate the gaze direction. The proposed method is based on mixed attention (MGE-Net). The estimation of each eye gaze point is done through CNNs and involves using information about the monocular feature, monocular position, and face feature. The method was tested on MPIIFaceGaze (5.06 cm error) and GazeCapture (1.95 cm on phone, 2.75 cm on tablet) datasets.

A paper tackling the estimation of the gaze zone of a driver is described by Bi et al. in [23]. The article uses the Columbia Gaze dataset (CAVE dataset), which was preprocessed and then reclassified and divided into 8 gaze areas that correspond to a driver's gaze zone. The method used in this paper for gaze estimation is transfer learning combined with a structure of a VGG16 convolutional neural network. By removing the top 3 layers that are fully connected and adding other custom fully connected layers, together with a fully connected layer at the end of the model, they achieved the multiclassification task. The resulting model yielded an accuracy rate of 78.7% on the validation set.

Liu et al. describe a differential convolutional neural network approach for gaze estimation in [24], where the differential convolutional neural network predicts the differences between two eye images of the same person. By using the inferred differences between a set of subject-specific calibration photos, the gaze direction of a novel eye sample can be predicted. A better prediction can be achieved by reducing annoyance factors such as illumination perturbations, alignment, or eyelid closing. This proposed method can be fine-tuned to make the predictions consistent. This method outperforms current state-of-the-art method results on MPIIGaze, EyeDiap, and UT multiview databases even when using only one calibration sample. The experimental results for MPIIGaze and EyeDiap are 3.8- and 3.23-degree angular error, respectively.

Joo et al. describe a ''one-stage trainable 2D Gaze estimation network'' in [25]. Most of the proposed methods use two different steps for gaze estimation. The first step is the localization of the eye landmarks and the second step regresses the gaze direction. Furthermore, a deep neural network is proposed that combines the two stages mentioned above into one single stage. Its backbone network is a Stacked Hourglass Network, where both the coordinates of the eye landmarks and the normalized gaze direction vector are estimated simultaneously. The datasets used were UnityEyes for training and MPIIGaze for testing, and the best performance for this method was a mean square error of 0.039.

A gaze estimation method that found its inspiration in the two-eye asymmetry was described by Cheng et al. [22] in his paper using FARE-Net. The system is based on the observation that the two eyes of the same person may appear uneven; therefore, the differences between them are used to optimize the result of gaze estimation using a face-based asymmetric regression evaluation network (FARE-Net). It consists of two networks, one for the regression task and another for the evaluation, finally combined into a single network. It uses an asymmetric mechanism for training that yields asymmetric weights. The authors evaluated 3 datasets, namely MPIIGaze (4.3 deg ang err), EyeDiap (5.71 deg ang err), and RT-Gene (8.4 deg ang err). This method predicts 3D gaze directions for both eyes and achieves leading performances in the mentioned datasets.

B. NEURAL MODEL OPTIMIZATION AND COMPRESSION TECHNIQUES

To overcome the expensive resource demand, e.g., the high computational cost and model size, in recent years, different optimization methods have been developed and evaluated. Examples of the most common methods are quantization of the model parameters [26], neural network pruning [9], or knowledge distillation [15].

Quantization is applied to reduce the numerical representation of the neural network parameters with the aim of decreasing the memory footprint and consequently the model size. Since neural network models are usually highly overparameterized, the precision could be maintained at a high level [11].

Neural network pruning is a method used to remove certain structure elements from a network (e.g., convolutional layers, connections, or neurons) that are redundant and have a small contribution to the network performance. Unlike quantization, the accuracy of the model can be more affected, and a fine-tuning training step may be necessary for recovery.

Knowledge distillation is a method that involves training to transfer the knowledge from a pre-trained teacher model to a smaller one that is called the student model. The aim of using this method is to train a lightweight model (student) that makes use of the knowledge learned by a large model (teacher). This approach increases performance over conventional training. Unlike previous methods, this method can be considered as a first step of the optimization process, as the student model can be further optimized using quantization or pruning.

Pourramezan Fard et al. [27] used the concept of knowledge distillation to create a new architecture for the facial landmark detection task. EfficientNet-B3 architecture was used for the teacher network and MobileNetV2 for the student network. The main contribution is to design a specific knowledge distillation loss function and a teacher-student architecture. The architecture is evaluated using three different datasets. The student network accuracy is significantly improved than using the original MobileNetV2 and in the same range with state-of-the-art methods, e.g., using COFW dataset the area-under-the-curve (AUC) has a difference of only 0.01 between teacher and student model.

Liu et al. [28] proposed a transformer-to-transformer knowledge distillation framework for semantic segmentation using transformer-specific patch embedding as the primary knowledge source. The framework combines feature map and patch embedding distillation. Two fundamental modules are developed, Cross Selective Fusion and Patch Embedding Alignment, and two optimization modules, Global-Local Context Mixer and Embedding Assistant. The framework is validated using Cityscapes, ACDC, and BYUv2 datasets showing improvements over existing knowledge distillation methods. To present the results, they used a common evaluation metric for semantic segmentation tasks such as mean Intersection over Union (mIoU). The best result was obtained using an optimized version of the proposed framework that achieves a +13.12% improvement in mIoU against a student model trained without knowledge distillation.

Yang et al. [29] use knowledge distillation to obtain efficient deep learning models for a 3D object detection application. A study has been performed to achieve student detectors with acceptable performance from the perspective of model compression and input resolution reduction. In the end, six pairs of teachers and students are used based on *voxel* and *pillar* architectures. Seven existing knowledge distillation methods are employed to benchmark the models. Efficient knowledge transfer from teacher to student is implemented using Teacher Guided Initialization which uses pre-trained teacher parameters to initialize the student model. The results showed that efficient model design along with knowledge distillation provides superior performance for 3D detectors based on *pillar* and *voxel* architectures. A voxel-based model outperforms its teacher model, while FLOPS decreased by 2.4x. The most efficient detector is 2.2x faster than previous voxel/pillar-based detectors.

Xiao et al. [30] use knowledge distillation to design a DNN to estimate depth information for a 3D real-time pedestrian tracking with monocular camera. This is effective in measuring the level of occlusion by monitoring the distance between the target and the camera. For the teacher network, they used an open-source monocular depth estimation method, and the student network is a 5-layer CNN. Tracking performance is tested before and after knowledge distillation. Using distillation, the depth prediction accuracy is slightly improved, and the inference speed is significantly accelerated by 2.08x, from 17.1 to 52.6 Hz.

Hong et al. [19] present a model compression analysis for CNNs applying layerwise and widthwise compression. Training is performed using the CIFAR-10 dataset with knowledge distillation to improve the accuracy of compressed models. The teacher is a simple model with 15 convolutional layers that achieves 89.56% accuracy, while the student architecture is MobileNetV1. The Deep Taylor decomposition is used to visualize the features learned by the student model and demonstrate a better representation compared to conventional training. The results show that layerwise compression is more suitable when the inference speed must be improved, while widthwise compression is recommended for memory requirement and computational cost reduction. For widthwise compression, a compression rate of 42.27% is obtained, while for layerwise compression is 32.42%. In terms of accuracy improvement using knowledge distillation, their suggestion is to use widthwise compression with increasing accuracy above 4.71% against conventional training. However, the student model has a different architecture compared to the teacher model, and their analysis does not include the combination of layerwise and widthwise compression. Details about the inference time measurement or calculation are also not provided. For instance, the widthwise compression does not considerably modify the inference time, while the number of parameters and FLOPs are significantly reduced.

III. METHODOLOGY

A. KNOWLEDGE DISTILLATION

The compression of a DNN is common practice for obtaining a lightweight network for low-cost and resource-scarce hardware devices. Knowledge distillation is a particular method that involves training to transfer knowledge from a larger network to a different one that has a significantly smaller size. Bucilua et al. [31] in their important paper successfully demonstrated for the first time that the knowledge acquired by a large ensemble of models can be transferred to a single small model. The aim of using this method is to train the student network so that it can reproduce the performance of the teacher network, but with fewer resources.

The process of applying knowledge distillation consists of the following main stages: (1) definition of teacher and student networks, (2) training of the teacher network, and (3) training of the student network with knowledge transfer from the teacher network. They are detailed below.

1) DEFINITION OF THE TEACHER AND STUDENT NETWORKS

The teacher network can be a high-performance standard model with high number of parameters (e.g., ResNet, DenseNet, EfficientNet), while the student network can be a standard model as well, but with a smaller number of parameters. Depending on the task, a custom architecture of the teacher and student networks can be used.

2) TRAINING THE TEACHER NETWORK

The teacher network is trained using a standard training procedure with the aim of achieving the most appropriate accuracy.

3) TRAINING THE STUDENT NETWORK WITH KNOWLEDGE TRANSFER FROM THE TEACHER NETWORK

In this stage, the knowledge distillation algorithm is used. Forward propagation is performed for the teacher and student networks, while backpropagation is applied only to the student network. The main loss function is defined using two different loss functions: student loss and distillation loss.

Knowledge from a model is categorized into three different types: Response-based knowledge, Feature-based knowledge, and Relation-based knowledge [32]. Response-based knowledge focuses on the final output layer, where the student model will learn the predictions of the teacher model. Feature-based knowledge leverages the knowledge of the data from teacher intermediate layers to train a student model. Relation-based knowledge focuses on the correlation between feature maps, graphs, similarity matrix, feature embeddings, or probabilistic distributions. In this paper, response-based knowledge is used because it showed the best results in different tasks and applications.

The block diagram for the knowledge distillation process is shown in Fig. 1.

The knowledge transfer from teacher to student is performed by minimizing the main loss function with the target to yield the same probabilities as using the teacher model. Basically, it refers to the output of the *softmax* function applied to non-normalized predictions. These predictions of the teacher model are usually found under the name of logits. Most often, the correct class of the probability distribution has a higher level compared to the other class probabilities that are close to zero. Therefore, the outcome

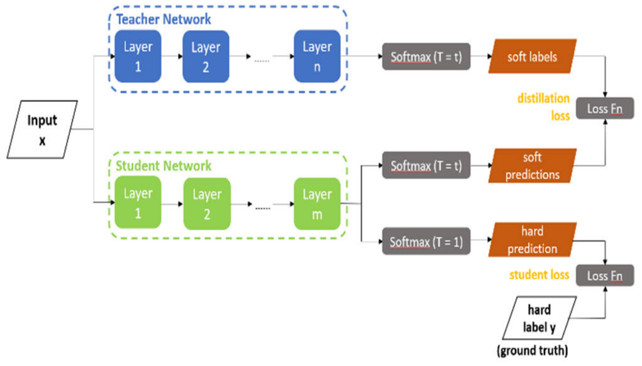

FIGURE 1. Knowledge distillation block diagram.

provided by this probability distribution is very similar to the ground truth labels of the dataset. Regarding this behavior, Hinton et al. [15] introduced the *softmax* parameter temperature. By denoting this parameter with T, the probability p_i of class i from the logit z_i is computed as in (1).

$$p_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{1}$$

When the parameter T is set to 1, the equation becomes a standard *softmax* function. Setting a value higher than 1 for T, the probability distribution will provide more information about the classes where the teacher model reported a prediction close to the correct class. This is the knowledge of the teacher model that is transferred to the student model using the distillation algorithm. When the distillation loss function is calculated using the soft labels, the same value of T is used to calculate the *softmax* on the student's logits.

It was noticed as a benefit to train the distilled model also using the ground truth labels of the dataset. Therefore, a second loss function is computed with the parameter T set to 1. This is a standard loss function called student loss. By denoting the distillation loss function with H_d (2), and the student loss function with H_s (3), the main loss function is calculated using (4).

$$H_d = H\left(\sigma\left(z_t; T = \tau\right), \sigma\left(z_s; T = \tau\right)\right) \tag{2}$$

$$H_s = H(y, \sigma(z_s; T = 1)) \tag{3}$$

$$L\left(x; W\right) = \alpha * H_s + \beta * H_d \tag{4}$$

where H is the cross-entropy loss function, σ is the *softmax* function, z_t and z_s are the logits of the teacher and student models, y is the ground truth label, x is the input, W are the student model parameters, while α and β are coefficients.

B. DEFINITION OF TEACHER AND STUDENT MODELS

1) DEFINITION OF THE TEACHER

To maximize accuracy while keeping the model size small, the teacher model is based on a custom architecture. A tiny CNN model that consists of five convolutional layers is proposed with the aim to derive small-size student models that can be deployed on the target device, a STM32H747I-DISCO hardware platform.

To construct the data pipeline and generate batches of tensor image data, we use the *ImageDataGenerator* class. Additionally, some pre-processing layers are built into the model such as rescaling, resizing, and random contrast. These layers are depicted in the teacher model architecture shown in Fig. 2 [33]. It consists of five 2D convolutional layers followed by *BatchNormalization*, *ReLU* activation function, *MaxPooling2D* and *Dropout* layers. *ZeroPadding2D* is used with the *same* padding type for all layers. At the end, a *Flatten* layer is used, followed by two *Dense* layers. Using this definition, the model has 505.017 parameters and a size of 6.14MB.

2) DEFINITION OF THE STUDENT

The teacher model is used as a baseline for the student models. To obtain a smaller student model, two different techniques are employed: layerwise and widthwise compression. Layerwise compression is applied by reducing the number of convolutional layers, while widthwise compression is applied by reducing the number of filters. The optimal student model is the one with the smaller size without a significant accuracy loss. To increase the search space for the optimal student model, different combinations of layerwise and widthwise compression are considered.

For the layerwise compression, the next student models are defined:

1. *Student_cut5* with the last convolutional layer removed
2. *Student_cut4* where the last two convolutional layers are removed

For the widthwise compression, the next student models are defined:

1. *Student_width10,*a model with 10% fewer filters
2. *Student_width30,*a model with 30% fewer filters
3. *Student_width50,*a model with 50% fewer filters

The combinations of layerwise and widthwise compression are defined by applying widthwise compression to the models resulting from the layerwise compression. We use the name *Student_cutX&widthY*, where X is the layerwise model, and Y is the widthwise model. In total, 11 student models are used to find the optimal configuration.

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

1) HARDWARE AND SOFTWARE CONFIGURATION

The experiments in this paper use a PC station with the following configuration: GeForce GTX 1070 graphic card, 8G RAM memory, and Ubuntu 20.04.3 LTS operating system. The main prerequisites for the virtual environment are: TensorFlow 2.7.0, Python 3.8.5, and Keras 2.7.0.

2) THE DATASET

The performance of the proposed models is evaluated using the Columbia Gaze dataset [34] that is used in several recent articles for the estimation of eye gaze [35], [36]. This is a publicly available dataset that consists of 5880 images

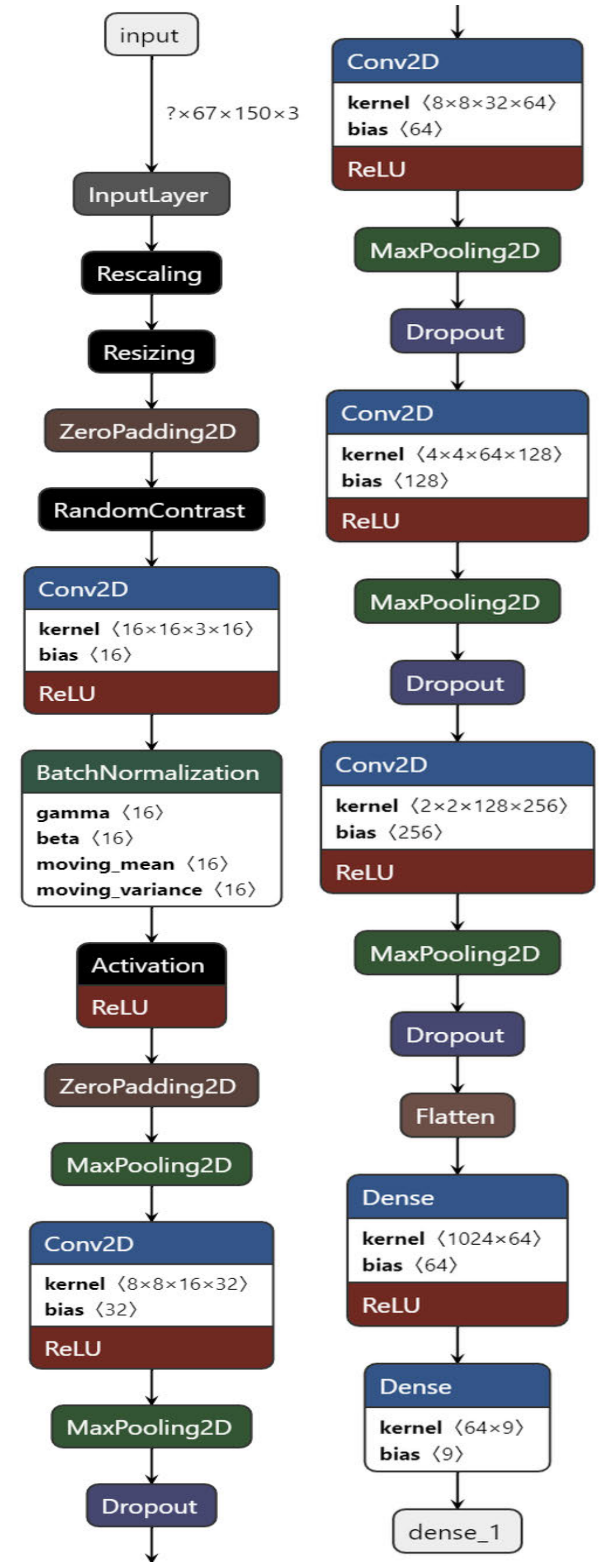

FIGURE 2. Architecture of the teacher model [33].

of 56 people performing various gaze orientation and head poses. Samples from the original dataset are shown in Fig. 3.

FIGURE 3. Samples from the columbia gaze dataset [34].

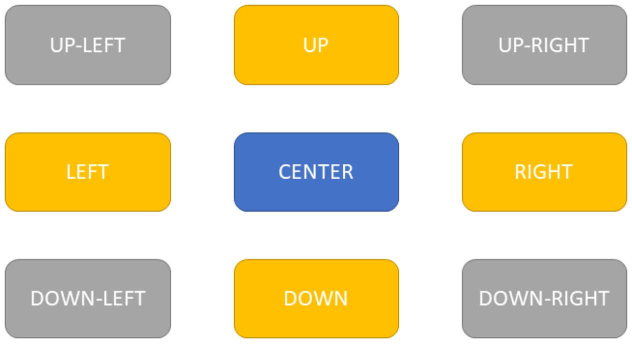

FIGURE 4. Eye gaze directions.

A dataset preparation is necessary since we reformulate the problem from a regression (image to numerical angle values) to a classification one (image to zone labels). More precisely, the images must be classified into nine gaze directions, as depicted in Fig. 4.

Every image in the CAVE dataset has a descriptive name that includes information about the camera rotation and gaze direction. For example, one instance of an image identifier is 0001_2m_15P_-10V_15H.jpg. The identifier comprises five components:

1. The participant's number (0001)
2. The camera's distance from the subject (2m)
3. The camera's rotation from the subject's frontal position (15P)
4. The subject's gaze direction on the OY axis (-10V)
5. The subject's gaze direction on the OX axis (15H)

Using this knowledge, we define a mapping scheme to automatically assign the appropriate class to each image.

Depending on the direction, the last three values of the name might take either positive or negative values. A positive numerical value for *P* denotes a leftward rotation, and the horizontal values *H* adhere to the same notation. Vertical values *V* are denoted by positive and negative numbers, where the positive values indicate an upward direction, and the negative values indicate a downward direction.

Considering that the photographs encompass the complete facial structure of the subject, supplementary preprocessing measures are necessary to exclusively isolate the subject's eyes from the images.

The preparation of the dataset involves multiple procedures such as eye cropping, image scaling, grayscale conversion, and channel reduction (RGB to gray scale). To retrieve the eyes from the photos in the database, a face detector was used. Specifically, a Haar cascade-type feature from the OpenCV library was utilized for this purpose. When employing this method, the face contained in the image will be located.

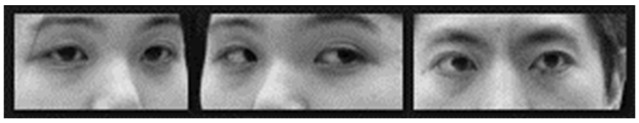

FIGURE 5. The dataset following the preprocessing stage.

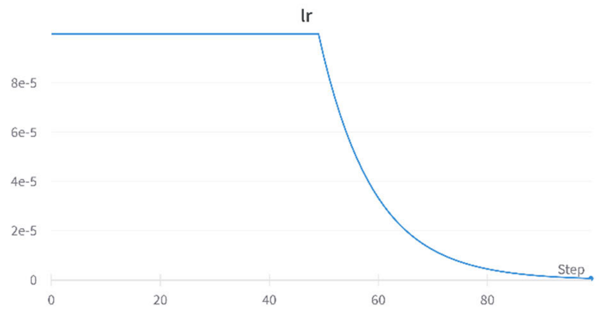

FIGURE 6. Exponential decay learning rate schedule [38].

The OpenCV package utilizes a specific module as an eye detector, which produces an image mainly containing the eye region for the given task. Consequently, the complete photographs depict solely a monochromatic region of the subjects' ocular region (refer to Fig. 5).

Finally, the 60-20-20 split is performed with respect to training, testing, and validation sets [37].

3) TRAINING CONFIGURATION

The learning policy is set using a batch size of 16, a Categorical Crossentropy loss function, and Adam optimizer with a learning rate of 1e-4. The number of epochs is set to 100. An exponential decay learning rate schedule is used that keeps the initial learning rate for the first 50 epochs and decreases it exponentially after that, as depicted in Fig. 6 [38]. The experimental results show that using this learning rate schedule improves accuracy. Training of both student and teacher models is performed using the same parameters and configuration.

To use the model with the highest accuracy that is obtained during training, we make use of *ModelCheckpoint* callback. Using this callback, the accuracy of the validation is monitored to save the model with the maximum value.

4) KNOWLEDGE DISTILLATION PARAMETRIZATION

The student models are trained with knowledge distillation using different values for the T parameter and α coefficient. The β coefficient is calculated as $1 - \alpha$. To find the optimal result, the parameter T is used in the range 2 - 16 with the next set points: 2, 4, 8, 10, 12, 14, and 16. For each T value, three different values for α coefficient are used: 0, 0.5, and 0.8. When the α coefficient is set to 0, the student loss function H_s from (4) is discarded. When the α coefficient is set to 0.5, it is equal to β coefficient and, therefore, the student loss function H_s and the distillation loss function H_d have the same weight in the main loss function (4). When the α coefficient is set to 0.8, the distillation loss function H_d has a small contribution in the main loss function (4).

5) INFERENCE TIME COMPUTATION

The inference time is calculated using the total number of computations that the model will have to perform. This parameter is denoted by FLOPs. Next, the floating-point operations per second, which is hardware-specific, are denoted by FLOPS. For the PC station used in this work, the FLOPS number is 6.463 TFLOPS. To compute the FLOPs for a specific model, we make use of the *keras-flops* library [39]. Therefore, the inference time is computed using the relation provided in (5).

$$Inferencetime = \frac{FLOPs}{FLOPS}[s] \tag{5}$$

6) STM32 HARDWARE CONFIGURATION

We chose as the target embedded platform the ARM Cortex-M7 core STM32H747XIH6 microcontroller with enabled data and instruction cache memory, which equips the STM32H747I-DISCO development board. The operating frequency can be configured up to 480MHz. To ensure the proper functionality of the CPU and to make use of the maximum performance, we configure the clock management unit to generate a system clock of 475MHz. In the graphical user interface for the X-CUBE-AI configuration, we set the *Compression* setting to *None*and *Optimization*to *Balanced.*For the other peripherals, the default configuration is used.

B. KNOWLEDGE DISTILLATION RESULTS ON THE HOST PLATFORM

In the first stage, the student model is trained without knowledge distillation. This creates the baseline model that has been useful to emphasize the improvement brought by the knowledge distillation algorithm. The test accuracy obtained from this conventional training is referred to in Table 1 and denoted as *Standard accuracy*.

Table 1 shows the results that do not depend on α or T for all student models: model size, number of parameters, compression ratio, standard accuracy, and inference time. Tables 2 - 5 show the test accuracy results for all the student models depending on α and T values. The highest accuracy obtained for each model is highlighted in bold. To present all of these results, we follow the next approach: (1) we present the results for each compression method in terms of maximum accuracy, knowledge distillation efficiency, compression ratio, and inference time along with a summary for each method, (2) we discuss the compression method that can be used depending on user requirements in terms of performance, size, and inference time, and (3) we summarize the main conclusions based on the presented results.

1) LAYERWISE COMPRESSION

When the last convolutional layer is removed, i.e., the *Student_cut5*model, the accuracy is higher than the teacher

TABLE 1. Teacher and student models.

Model	Size	Parameters	Compression ratio	Standard accuracy	Inference time
Teacher	6.14MB	505.017	N/A	77.48%	60.77µs
Student_width10	4.91MB	402.183	1.25	75.94%	49.47µs
Student_cut5	4.17MB	340.921	1.47	74.57%	59.74µs
Student_cut5&width10	3.31MB	269.933	1.85	73.37%	48.64µs
Student_width30	3.02MB	244.610	2.03	72.08%	33.75µs
Student_cut4	2.39MB	193.337	2.56	70.20%	55.68µs
Student_cut5&width30	2.05MB	164.867	2.98	72.78%	33.25µs
Student_cut4&width10	1.89MB	151.714	3.24	70.37%	45.39µs
Student_width50	1.64MB	129.633	3.74	70.86%	21µs
Student_cut4&width30	1.20MB	94.202	5.09	63.09%	31.30µs
Student_cut5&width50	1.14MB	88.545	5.37	67.35%	20.74µs
Student_cut4&width50	693KB	51.617	8.86	62.53%	19.72µs

TABLE 2. Layerwise compression results.

Model	T	Accuracy		
		α = 0	α = 0.5	α = 0.8
Student_cut5	2	76.36%	76.61%	77.05%
	4	75.77%	77.39%	75.59%
	8	76.71%	75.94%	76.71%
	10	74.91%	77.14%	76.62%
	12	78.51%	77.99%	76.45%
	14	76.79%	**80.13%**	77.39%
	16	77.65%	75.59%	76.71%
Student_cut4	2	71.74%	s73.80%	72.51%
	4	**76.54%**	76.02%	73.63%
	8	74.14%	73.88%	75.94%
	10	74.22%	73.80%	75.34%
	12	75.42%	74.82%	75.08%
	14	76.19%	75.08%	74.74%
	16	75.17%	74.82%	74.05%

TABLE 3. Widthwise compression results.

Model	T	Accuracy		
		α = 0	α = 0.5	α = 0.8
Student_width10	2	76.19%	76.36%	74.31%
	4	74.40%	77.56%	76.45%
	8	72.60%	76.79%	77.56%
	10	76.02%	77.99%	**78.16%**
	12	78.08%	75.77%	76.45%
	14	76.96%	77.48%	76.88%
	16	76.79%	75.68%	76.45%
Student_width30	2	75.68%	76.79%	72.51%
	4	74.91%	74.82%	76.71%
	8	74.82%	**76.96%**	72.17%
	10	72.77%	76.88%	75.25%
	12	73.28%	**76.96%**	73.80%
	14	73.71%	76.02%	75.17%
	16	74.91%	73.03%	72.68%
Student_width50	2	69.60%	69.17%	71.23%
	4	65.32%	67.03%	63.35%
	8	65.66%	69.52%	67.03%
	10	65.92%	**72.85%**	71.57%
	12	70.29%	69.60%	69.77%
	14	70.71%	69.00%	70.71%
	16	69.00%	68.23%	70.54%

model and the compression ratio is 1.47. Since the number of computations for this layer is small, the inference time is less affected. When layer 4 is also removed, i.e. the *Student_cut4*model, the accuracy is lower than the one provided by the teacher model, but the knowledge distillation improvement with reference to the standard accuracy is the same as for the *Student_cut5*model. A higher compression ratio is obtained, but the difference in the inference time is small. Therefore, layerwise compression could lead to improvements in terms of accuracy, even higher than the teacher model accuracy, a modest compression ratio. The inference time has only a small decrease, since the number of computations is not significantly affected.

2) WIDTHWISE COMPRESSION

With the number of filters reduced by 10%, the accuracy using knowledge distillation is slightly higher than the standard accuracy. The compression ratio is also small, but the inference time is even shorter. Using the *Student_width30*model, the accuracy is higher than the standard accuracy, the compression ratio has increased, while the inference time has decreased significantly. When the number of filters is reduced by 50%, the accuracy improvement is similar to the *Student_width10*model accuracy, but the compression ratio has significantly increased, while the inference time is reduced by more than half compared to the teacher model.

The *Student_width30* model is most suitable to be used in practical applications because the accuracy is close to the accuracy of the teacher model, the knowledge distillation algorithm makes a significant improvement while the compression ratio and inference time can be acceptable.

3) LAYERWISE AND WIDTHWISE COMPRESSION

The results for the layerwise and widthwise compression are presented in Table 4 and Table 5. Table 4 shows the results for *Student_cut5* model in combination with all the widthwise compression models. Table 5 follows a similar approach, but uses the *Student_cut4*model.

TABLE 4. Layerwise and widthwise compression results.

Model	T	Accuracy		
		α = 0	α = 0.5	α = 0.8
Student_cut5 & width10	2	73.39%	75.39%	75.15%
	4	77.14%	76.19%	76.93%
	8	76.29%	75.32%	**77.71%**
	10	75.68%	75.60%	76.71%
	12	76.96%	76.89%	77.14%
	14	76.45%	76.62%	74.34%
	16	75.59%	76.11%	76.88%
Student_cut5 & width30	2	71.44%	74.23%	72.44%
	4	75.86%	73.97%	74.23%
	8	75.18%	74.48%	74.02%
	10	75.17%	76.11%	**76.62%**
	12	73.20%	74.74%	73.23%
	14	71.68%	73.40%	74.32%
	16	74.66%	75.34%	73.22%
Student_cut5 & width50	2	67.53%	68.12%	64.84%
	4	70.20%	69.07%	65.60%
	8	69.23%	69.55%	68.75%
	10	68.00%	70.71%	65.49%
	12	66.95%	70.03%	67.83%
	14	61.92%	64.68%	71.16%
	16	66.53%	67.89%	**71.40%**

TABLE 5. Layerwise and widthwise compression results.

Model	T	Accuracy		
		α = 0	α = 0.5	α = 0.8
Student_cut4 & width10	2	73.11%	71.12%	72.94%
	4	73.31%	74.33%	71.55%
	8	74.41%	74.15%	**76.14%**
	10	73.32%	72.46%	72.34%
	12	75.27%	73.06%	74.94%
	14	75.25%	73.54%	73.28%
	16	74.10%	73.33%	73.37%
Student_cut4 & width30	2	68.23%	67.72%	71.14%
	4	72.43%	68.63%	70.41%
	8	70.46%	70.93%	72.38%
	10	70.59%	69.85%	71.88%
	12	71.86%	70.33%	70.63%
	14	70.42%	**72.68%**	70.89%
	16	67.52%	70.01%	70.46%
Student_cut4 & width50	2	63.98%	59.94%	62.92%
	4	55.81%	65.83%	63.84%
	8	63.38%	68.01%	60.41%
	10	64.91%	67.07%	59.80%
	12	66.09%	67.20%	64.98%
	14	60.70%	**68.29%**	63.87%
	16	60.18%	61.86%	62.50%

Using the *Student_cut4*layerwise compression model, the knowledge distillation efficiency is higher than using the *Student_cut5*model. The compression ratio and inference time improve as the number of filters is reduced. The highest compression ratio and decrease in inference time are obtained using the *Student_cut4&width50*model, but the accuracy is more affected in this case.

C. VALIDATION ON THE TARGET PLATFORM

The student models are validated on the STM32H747I-DISCO hardware device using the X-CUBE-AI toolchain. We noticed that X-CUBE-AI currently does not support rescaling, resizing, and random contrast preprocessing layers built into the model. Therefore, these pre-processing layers were removed and implemented outside the model. To be consistent, this change was made for both the student models and the teacher model. For this reason, the teacher model is trained from scratch again and the student models are trained using the knowledge distillation configuration (parameters α and T) that achieved the highest accuracy in Section B.

The following configurations are available for the X-CUBE-AI software: system performance, validation, and template application. System performance is a standalone test application for performance assessment and validation. In this paper, we use the validation set-up from the available template. The results obtained are presented in Table 6 in terms of accuracy, RAM and ROM memory utilization, MACC complexity, and inference time. The MACC metric is directly related to the computational complexity of a certain neural model. Thus, it can be used to assess the real-time capabilities of the target platform. The memory footprint is important to find a suitable hardware device, i.e. the micro-controller, since the size of the model should not exceed the RAM size of the processor. All these results are automatically computed with the validation process and reported by the STM32CubeMX graphical tool.

D. DISCUSSION

1) KNOWLEDGE DISTILLATION RESULTS ON THE HOST PLATFORM

Depending on the user requirements, different results can be expected after compressing a model: (1) reduce the model size while maintaining the performance or even to improve the accuracy, if possible, (2) reduce the model size and inference time with a small accuracy degradation, (3) significantly reduce the model size and inference time but keeping the accuracy in acceptable limits.

For the first case, the following models can be used: *Student_cut5*, *Student_width10*, and *Student_cut5&width10*. For these models, the accuracy is higher than the accuracy of the teacher model. The highest compression ratio and reduced inference time are obtained for the *Student_cut5&width10*model. If the user requirement is not to achieve maximum accuracy, then *Student_cut5&width10*is the most appropriate to use.

For the second case, the following models have achieved a decrease in accuracy of less than 1% compared to the teacher model *Student_cut4*, *Student_width30* and *Student_cut5&width30*. The highest compression ratio and reduced inference time are obtained for the *Student_cut5&width30*model, which is the most suitable to use for this category.

When model size and inference time are considered the most important aspects, the following most appropriate solutions are *Student_width50, Student_cut5&width50, Student_cut4&width10, Student_cut4&width30,* and

TABLE 6. Validation on target platform results.

Model	Host platform accuracy	Target platform accuracy	RAM	ROM	MACC	Inference time
Student_width30	72.17%	72.39%	296.30 KB	981.00 KB	98,799,751	910.063ms
Student_cut4	73.71%	73.75%	397.11 KB	779.75 KB	164,364,985	1508.588ms
Student_cut5&width30	71.40%	71.28%	296.02 KB	669.04 KB	97,759,844	902.817ms
Student_cut4&width10	73.45%	73.32%	347.77 KB	617.18 KB	134,115,120	1268.091ms
Student_width50	65.83%	65.93%	288.21 KB	531.89 KB	61,721,697	563.455ms
Student_cut4&width30	73.63%	73.66%	291.56 KB	392.54 KB	92,665,890	870.506ms
Student_cut5&width50	68.66%	68.65%	287.93 KB	370.93 KB	61,186,017	561.296ms
Student_cut4&width50	70.80%	70.86%	287.65 KB	226.22 KB	58,519,905	541.118ms

Student_cut4&width50. The highest compression ratio and reduced inference time is obtained using the *Student_cut4 &width50*model, but the accuracy is only 68.29%. The model in this category that has the highest accuracy is *Student_cut4&width10*. It is close to the accuracy of the teacher model, while the compression ratio is 3.24.

Considering the above results, the following conclusions can be summarized.

1. Using knowledge distillation, accuracy can be improved by up to 9.5% compared to a conventional training procedure. This is achieved using the layerwise and widthwise compression model *Student_cut4&width30*.

2. The knowledge distillation algorithm is more efficient when the coefficient α is set to 0 or 0.5, and T is set to a higher value such as 14. Therefore, the efficiency is higher when the main loss function (4) is based only on the distillation loss function H_d or the student loss functionH_s and the distillation loss function H_d have the same weight.

3. Combining layerwise compression and widthwise compression is more efficient than using them independently.

4. Inference time is further reduced using widthwise compression than using layerwise compression. The combination of them comes with an improvement, but not significantly large.

2) VALIDATION ON THE TARGET PLATFORM

The accuracy results shown in Table 6 are in the same range as the accuracy results presented in Section B. They are not identical because the models undergo a different training process. The purpose of the validation on the STM32 hardware is to show that the accuracy achieved is similar to that obtained on the host platform, as no compression is applied that could introduce a drop. The small difference could be due to the accuracy being computed using different tools.

Regarding RAM and ROM memory utilization, the following considerations can be summarized depending on layerwise or widthwise compression: (1) the RAM memory utilization is reduced using widthwise compression. Using layerwise compression, the difference is not significant, (2) the ROM memory utilization is gradually reduced as the model size decreases. For this result, no difference is visible between layerwise and widthwise compression, and (3) combining layerwise compression and widthwise compression is more efficient because RAM memory is also reduced.

The MACC complexity is higher when only layerwise compression is used. It is significantly reduced when widthwise compression is applied. The most efficient choice is also to use a combination of layerwise and widthwise compression.

The inference time follows a distribution similar to that computed in Section B which presents the results on the host platform. Therefore, the inference time is even more reduced using widthwise compression, and the combination with layerwise compression comes with improvements. For the *Student_cut4&width30*model, the inference time is 870.5ms, while the accuracy is 73.66%. This means that one frame per second can be achieved with accuracy close to the maximum. This inference time can be accepted in practical applications, enabling us to implement a real-time gaze detection system on a low-power/low-cost embedded platform.

V. CONCLUSION

Today, deep neural networks and their associated deep learning paradigm are omnipresent in the automotive domain [40]. Our work refers to an optimization procedure aimed at the deployment of an eye gaze estimation neural model on automotive-specific hardware. The knowledge distillation algorithm can be successfully applied to significantly reduce the memory footprint of a convolutional neural network. In this work, the layerwise and widthwise compression methods have been used independently or in combination to define various student models. With knowledge distillation, accuracy can be improved by up to 9.5% compared to using the conventional training procedure. A compression ratio of up to 8.86 is achieved with a decrease of less than 10% in accuracy. The inference time is further reduced using widthwise compression. Combining layerwise and widthwise compression is more efficient and a good trade-off between model size, accuracy, and inference time. The results of the validation on the STM32 hardware show that to reduce the utilization of ROM and RAM memory, the combination of layerwise and widthwise compression represents the optimal solution. The same combination represents a viable solution to simultaneously optimize the MACC complexity and inference time. Still we could identify some limitations of the proposed methodology as it only transfers knowledge related to the initial model outputs and not capturing the internal representations learned by teacher model. Also, the student model can learn little from some teacher models if there is an important architecture gap between them.

Future work proposes to extend this paper by exploring more advanced knowledge distillation algorithms, e.g. data-free [41] or gaussian noised-based [42] knowledge distillation. Furthermore, combining knowledge distillation with other optimization methods like, e.g., quantization and pruning is expected to further reduce the model size and consequently the memory footprint.

REFERENCES

[1] A. A. M. Al-Saffar, H. Tao, and M. A. Talab, "Review of deep convolution neural network in image classification," in *Proc. Int. Conf. Radar, Antenna, Microw., Electron., Telecommun. (ICRAMET)*, 2017, pp. 26–31.

[2] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019.

[3] V. Sharma, M. Gupta, A. Kumar, and D. Mishra, "Video processing using deep learning techniques: A systematic literature review," *IEEE Access*, vol. 9, pp. 139489–139507, 2021.

[4] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 3523–3542, Jul. 2022.

[5] H. U. Draz, M. I. Ali, M. U. G. Khan, M. Ahmad, S. Mahmood, and M. A. Javaid, "An embedded solution of gaze estimation for driver assistance using computer vision," in *Proc. Int. Conf. Innov. Comput. (ICIC)*, 2021, pp. 1–8.

[6] M. Gao, F. Meng, and Y. Meng, "Analysis of consumer supermarket shopping behaviors based on eye movement information," in *Proc. 13th Int. Congr. Image Signal Process., BioMedical Eng. Inf. (CISP-BMEI)*, 2020, pp. 18–22.

[7] Y. Fang, H. Duan, F. Shi, X. Min, and G. Zhai, "Identifying children with autism spectrum disorder based on gaze-following," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Oct. 2020, pp. 423–427.

[8] G. C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca, "Deep neural networks compression: A comparative survey and choice recommendations," *Neurocomputing*, vol. 520, pp. 152–170, Feb. 2023.

[9] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" 2020, *arXiv:2003.03033*.

[10] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*.

[11] I. L. Orăşan, C. Seiculescu, and C. D. Caleanu, "Benchmarking TensorFlow lite quantization algorithms for deep neural networks," in *Proc. IEEE 16th Int. Symp. Appl. Comput. Intell. Inform. (SACI)*, Timisoara, Romania, 2022, pp. 000221–000226, doi: 10.1109/SACI55618.2022.9919465.

[12] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. Interspeech*, 2013, pp. 2365–2369, doi: 10.21437/Interspeech.2013-552.

[13] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1MB model size," 2016, *arXiv:1602.07360*.

[14] J. G. López, A. Agudo, and F. Moreno-Noguer, "E-DNAS: Differentiable neural architecture search for embedded systems," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Milan, Italy, Jan. 2021, pp. 4704–4711.

[15] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[16] I. L. Orăşan, C. Seiculescu, and C. D. Căleanu, "A brief review of deep neural network implementations for ARM cortex-M processor," *Electronics*, vol. 11, no. 16, p. 2545, Aug. 2022.

[17] I. L. Orăşan and C. D. Căleanu, "ARM embedded low cost solution for implementing deep learning paradigms," in *Proc. Int. Symp. Electron. Telecommun. (ISETC)*, Timisoara, Romania, 2020, pp. 1–4, doi: 10.1109/ISETC50328.2020.9301130.

[18] STMicroelectronics. *Data Brief—X-CUBE-AI—STMicroelectronics*. Accessed: Jun. 10, 2023. [Online]. Available: https://www.st.com/resource/en/data_brief/x-cube-ai.pdf

[19] Y.-W. Hong, J.-S. Leu, M. Faisal, and S. W. Prakosa, "Analysis of model compression using knowledge distillation," *IEEE Access*, vol. 10, pp. 85095–85105, 2022.

[20] Y. Zhuang, Y. Zhang, and H. Zhao, "Appearance-based gaze estimation using separable convolution neural networks," in *Proc. IEEE 5th Adv. Inf. Technol., Electron. Automat. Control Conf. (IAEAC)*, Chongqing, Mar. 2021, pp. 609–612.

[21] Y. Luo, J. Chen, and J. Chen, "CI-net: Appearance-based gaze estimation via cooperative network," *IEEE Access*, vol. 10, pp. 78739–78746, 2022.

[22] Y. Cheng, X. Zhang, F. Lu, and Y. Sato, "Gaze estimation by exploring two-eye asymmetry," *IEEE Trans. Image Process.*, vol. 29, pp. 5259–5272, 2020.

[23] Q. Bi, X. Ji, and Y. Sun, "Research on driver's gaze zone estimation based on transfer learning," in *Proc. IEEE Int. Conf. Inf. Technol., Big Data Artif. Intell. (ICIBA)*, Nov. 2020, pp. 1261–1264.

[24] G. Liu, Y. Yu, K. A. F. Mora, and J.-M. Odobez, "A differential approach for gaze estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 3, pp. 1092–1099, Mar. 2021.

[25] H. Joo, M. S. Ko, and H. Song, "OSTGazeNet: One-stage trainable 2D gaze estimation network," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Jeju Island, South Korea, 2021, pp. 1137–1139.

[26] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," 2017, *arXiv:1712.05877*.

[27] A. P. Fard and M. H. Mahoor, "Facial landmark points detection using knowledge distillation-based neural networks," 2021, *arXiv:2111.07047*.

[28] R. Liu, K. Yang, A. Roitberg, J. Zhang, K. Peng, H. Liu, and R. Stiefelhagen, "TransKD: Transformer knowledge distillation for efficient semantic segmentation," 2022, *arXiv:2202.13393*.

[29] J. Yang, S. Shi, R. Ding, Z. Wang, and X. Qi, "Towards efficient 3D object detection with knowledge distillation," 2022, *arXiv:2205.15156*.

[30] P. Xiao, F. Yan, J. Chi, and Z. Wang, "Real-time 3D pedestrian tracking with monocular camera," *Wireless Commun. Mobile Comput.*, vol. 2022, pp. 1–18, Feb. 2022.

[31] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proc. 12th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, New York, NY, USA, 2006, p. 296.

[32] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, Jun. 2021.

[33] L. Roeder. (Jan. 27, 2017). *Netron*. Accessed: Nov. 15, 2022. [Online]. Available: https://github.com/lutzroeder/netron

[34] B. A. Smith, Q. Yin, S. K. Feiner, and S. K. Nayar, "Gaze locking: Passive eye contact detection for human-object interaction," in *Proc. 26th Annu. ACM Symp. User Interface Softw. Technol.*, New York, NY, USA, Oct. 2013, pp. 1–9.

[35] A. Bublea and C. D. Căleanu, "Deep learning based eye gaze tracking for automotive applications: An auto-Keras approach," in *Proc. Int. Symp. Electron. Telecommun. (ISETC)*, Timisoara, 2020, pp. 1–4.

[36] A. Bublea and C. D. Căleanu, "AutoML and neural architecture search for gaze estimation," in *Proc. IEEE 16th Int. Symp. Appl. Comput. Intell. Inform. (SACI)*, Timisoara, 2022, pp. 143–148.

[37] J. Filter. (Feb. 3, 2022). *Split-Folders 0.5.1*. Accessed: Jun. 10, 2023. [Online]. Available: https://pypi.org/project/split-folders/

[38] CVP and L. Biewald. *Weights & Biases*. Accessed: Jun. 10, 2023. [Online]. Available: https://wandb.ai/site

[39] Tokusumi. (Aug. 17, 2020). *Keras-Flops 0.1.2*. Accessed: Jun. 10, 2023. [Online]. Available: https://pypi.org/project/keras-flops/

[40] H.-H. Jebamikyous and R. Kashef, "Autonomous vehicles perception (AVP) using deep learning: Modeling, assessment, and challenges," *IEEE Access*, vol. 10, pp. 10523–10535, 2022.

[41] S. Yu, J. Chen, H. Han, and S. Jiang, "Data-free knowledge distillation via feature exchange and activation region constraint," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2023, pp. 24266–24275.

[42] P. Raikwar and D. Mishra, "Discovering and overcoming limitations of noise-engineered data-free knowledge distillation," in *Proc. NeurIPS*, 2022, pp. 1–11.

IOAN LUCAN ORĂȘAN received the B.E. degree in electronics, telecommunications and information technologies from 1 December 1918 University, Alba Iulia, and the M.S. degree in electronics of intelligent systems from Politehnica University Timisoara, where he is currently pursuing the Ph.D. degree with the Department of Applied Electronics. From 2017 to 2023, he was a Software Engineer with Continental Automotive Romania and Vitesco Technologies, Timişoara. He is a Software Project Manager. His current research interests include the implementation of deep learning paradigms on embedded systems (microcontrollers), compression and optimization of deep neural networks, deep learning, image classification, and computer vision.

ADRIAN-IOAN BUBLEA received the B.E. degree in electronics, telecommunications and information technologies and the M.S. degree in electronics of intelligent systems from Politehnica University Timisoara, where he is currently pursuing the Ph.D. degree.

From 2018 to 2022, he was with Continental Automotive Romania as a System Tester and later as a Software Tester. He joined Porsche Engineering Romania, as a Software Engineer. His current research interest includes deep learning for biometrics.

CĂTĂLIN DANIEL CĂLEANU (Member, IEEE) received the master's degree in electronics of intelligent systems and the Ph.D. degree in electronics engineering from Politehnica University Timisoara (UPT), Timişoara, Romania. He is currently with the Faculty of Electronics, Telecommunications and Information Technologies, UPT, where he is also teaching deep neural networks, bioinspired systems, and embedded system courses. He has authored more than 100 technical journal articles and conference papers. He has coauthored ten books mainly in neural networks, fuzzy systems, evolutionary computation, computer vision, and embedded systems fields. His main research interests include deep learning applications for medical imaging, biometrics, video surveillance, and automotive.