ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Real-time On-edge Classification: an Application to Domestic Acoustic Event Recognition

Lode Vuegen and Peter Karsmakers *

KU Leuven - Department of Computer Science
Declarative Languages and Artificial Intelligence (DTAI)
Kleinhoefstraat 4, B-2440 Geel - Belgium

Abstract. In this paper two different convolutional neural network (CNN) architectures are investigated for the purpose of real-time on-edge domestic acoustic event classification. For training and evaluation of the models, a real-life acoustical dataset was recorded in 72 different home environments. A quantization-aware training scheme was applied that takes into account that the models need to run on 8-bit fixed-point processing hardware. Once trained, the models were successfully deployed on an ARM cortex-M7 microcontroller unit (i.MX RT1064). This study indicates that the used procedure can lead to an efficient and real-time embedded on-edge implementation of a domestic sound event classifier that does not sacrifice classification performance compared to its floating-point counterpart.

1 Introduction

Shifting machine learning (ML) algorithms from the cloud to the sensor gained a lot of interest in recent years. At sensor side there are often stringent constraints regarding computational resources and energy consumption resulting in the need of small and low-power ML algorithms. On-edge processing comes with the advantage that no data must be transmitted (wirelessly) and thereby offering an improved user privacy, lower latency and reduced energy consumption. This in combination with the increased computing performance in today's off-the-shelf microcontroller units (MCU's) makes that a wide range of new applications became available. For instance, real-time predictive maintenance in factories from inertial and motion sensor data [1], long-term patient monitoring at home from vital sensor data [2], autonomous controlled vehicles and robots from environmental data [3], and so on.

Although the use of acoustic sensors for indoor monitoring applications is already widely examined in the scientific community, most of the proposed solutions are focusing to the use of complex and computationally demanding deep learning frameworks [4]. Hence, in this paper we will specifically focus to the use of a compact convolutional neural network (CNN) architecture that can be deployed on a standard off-the-shelf MCU in a real-time setting during inference.

The remainder of this paper is organized as follows: Section 2 discusses the investigated on-edge convolutional classifier architectures. Section 3 describes

*This research received funding from the Flemish Government (AI Research Program) and Belgian Government (project *"ISAAC: Intelligent Sensors for Anomaly Detection In Harsh Environments"*), and from Vlaams Agentschap voor Innoveren en Ondernemen (project *"WATCHDOG: Bringing in Artificial Intelligence to Context Aware Systems to Enable the Cognitive Home"*).

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

the details of the performed experiments. The obtained results in terms of classification performance, memory requirements and on-edge inference time are discussed in Section 4, and is followed by the final concluding remarks which are given in Section 5.

2 Pooling vs. no-pooling model architectures

The acoustic event classifiers examined in this work are all based on a CNN architecture. The trade-off between model complexity and classification accuracy will be investigated for the purpose of a real-time on-edge deployment. The complexity of a CNN model can be controlled on both the architectural level (i.e. number of layers, filters, etc.) and the arithmetic level (i.e. data type precision). Hence, in this research two different CNN architectures will be investigated both for a 32-bit floating-point and a 8-bit fixed-point implementation.

Traditional CNN models typically use a pooling layer after each convolutional layer to add local translation invariance and to reduce the dimensionality of the feature maps[1]. The lower the feature map dimensions the less processing is required. The feature map dimensionality of the network can also be reduced without the need of pooling layers by increasing the stride length of the filters. This removes the local translation invariance property of the model but comes with a significantly lower number of computational operations which is benificial for an embedded implementation.

This paper will empirically examine if the local invariance is really needed in case when working on a time-frequency representations of short-term acoustic signals. The latter is done by comparing the obtained classification results of a traditional *'pooling'* model architecture with a *'no-pooling'* model architecture when learned from acoustical data.

3 Experimental setup

3.1 Dataset

The on-edge acoustic event models will be learned from a real-life dataset collected at 72 different home environments. Each participant of the measurement campaign was asked to record eight different domestic sound events by following a predefined recording protocol. All recordings were labelled on-the-fly and were directly saved to an SD-card with a sampling frequency of 32 KHz and 16 bits per sample. In total 47.7 hours of data and 1519 recordings were collected during the measurement campaign. An overview of the dataset is given in Table 1.

3.2 Feature extraction

The most commonly used features in the domain of acoustic scene and event classification in combination with a deep learning strategy are the so-called log-Mel features [4]. In this work, the log-Mel features are computed from short

[1]Moreover, to some extent adding the pooling layer reduces the risk of overfitting.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

Class	Hours	Recordings
Background	10.5	205
Door & window	5.3	141
Faucet & shower	9.3	386
Footstep	4.2	220
Kitchen hood	4.0	140
Speech	4.9	217
Toilet	5.5	136
Radio & television	4.0	74

Table 1: Overview of the recorded domestic sound event dataset.

overlapping frames with a frame size of 32 ms and a frame shift of 10 ms. Next, a 1024-point fast-Fourier transform (FFT) is applied to each frame and is followed by a log Mel-frequency warping operation using a Mel-spaced filterbank spanning 64 filters.

3.3 Model architectures and quantization

As already briefly mentioned in Section 2, two different CNN model architectures will be investigated in this work. Both model CNN architectures will be made up of three convolutional layers (feature learning part) and two fully connected layers (classification part), and will be evaluated with two different input sizes and three different numbers of filters per convolutional layer. The feature learning parts of the two model architectures are configured as follows:

- *'pooling'* architecture: three convolutional layers with a filter size of (4×4), filter stride of (1×1), padding, relu activation, and a pool size of (1×4).
- *'no-pooling'* architecture: three convolutional layers with a filter size of (4×4), filter stride of (1×4), no-padding, relu activation, and no-pooling.

The classification part of both models is made up of a fully-connected layer with 64 output neurons and a relu activation, and a fully-connected layer with 8 output neurons (nr. of classes) and a softmax activation for classification. The two examined input shape sizes are (50×64) (T=50) and (100×64) (T=100) corresponding to an audio input size of 0.5 and 1.0 sec. respectively given a frame shift of 10 ms. The examined number of filters per convolutional layer are 16 (model small), 32 (model medium) and 64 (model large).

All models are learned in TensorFlow with a 32-bit floating-point precision. The learned models are then converted to an 8-bit fixed-point precision for an embedded deployment using the TensorFlow Lite quantized-aware retraining procedure [5]. This retraining operation simulates the effects of quantization during inference and is done as follows:

- Feed-forward pass: the effects of quantization are simulated by quantizing the weights and activations to 8-bit integers and converting them back

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

to 32-bit floats. This operation mimics the effects of a less precise data representation although all computations are still performed with a 32-bit floating-point representation.

- Feed-backward pass: the model weights will be updated using backpropagation. The gradients to update the weights are calculated from the outputs obtained in the feed-forward pass.

Both steps are repeated until convergence or the maximum number of epochs (iterations) is reached. Once the quantized-aware retraining procedure is completed, the model weights can be converted to an 8-bit fixed-point precision and all computations can be performed with integers only during inference.

3.4 Training, validation and test set generation

The training, validation and test sets are obtained by applying a 4-fold cross-validation on the randomly shuffled log-Mel input segments of size T. This operation results in to four independent training and test sets with a ratio of 75/25% respectively. The validation sets are then obtained by randomly drawing 30% of the segments from the corresponding training sets.

From Table 1 it can be clearly seen that there is also a significant class imbalance. Therefore, the minority classes in both the training and validations sets are up-sampled to have as much segments as the majority class. This results in a balanced training and validation set per fold which prevents the learned model to focus to the largest classes while disregarding the smaller classes.

Next, per cross-validation iteration the dataset is also normalized to zero mean and unit standard deviation in each log-Mel dimension. The normalisation parameters are learned from the training set and are kept fixed for the validation and test set. After normalisation, each log-Mel dimension has a zero mean and unit standard deviation thereby preventing the model to focus to some specific log-Mel features.

4 Results

All obtained results are listed in Table 2. By analyzing the classification scores for both the unquantized and quantized models, i.e. u/q-rows respectively, it can be clearly seen that the used quantization scheme has practically no impact on the classification performance ($< 1\%$). This implies that these models can be deployed on an embedded platform with an integer-only inference framework without the need of sacrificing classification accuracy. The best and least accurate scores are $88.6 \pm 0.2\%$ and $71.6 \pm 0.3\%$, and are obtained with the settings *'no-pooling, model large, T=100'* and *'pooling, model small, T=50'* respectively.

By analyzing the quantized weights and buffer sizes, and the corresponding inference times, i.e. s/t-rows respectively, it can also be seen that using a more complex model architecture (i.e. a higher number of filters per convolutional layer), or using a larger input size, has a significant impact on the required

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

memory and inference time as well. Doubling the number of filters per convolutional layer, or doubling the input size of the network, significantly boosts the classification performance but comes with the cost of an increased memory usage and inference time. The reason for the increased performance is simply due to the fact that a more complex model architecture is able to learn more details from the data, and that a larger input size provides more data to the CNN model to rely on for the prediction.

Another important observation that can be made is that the *'no-pooling'* model architecture yields a higher classification performance, a three to four times faster inference speed, and a four times smaller memory footprint compared to its *'pooling'* counterpart. The reason for the smaller memory footprint is that the larger filter stride in the convolutional layers (i.e. (1×4) in this work) directly produce smaller feature maps compared to a unit filter stride. In addition, using a larger filter stride also yields less arithmetic operations during the convolution operation, and in combination with the fact that no pooling layers are required to reduce the feature maps makes that the inference time of the network can be significantly reduced.

			Model S	**Model M**	**Model L**
			16 filters/layer	*32 filters/layer*	*64 filters/layer*
Pooling	T=50	u	$71.6 \pm 0.3\%$	$77.9 \pm 0.1\%$	$84.0 \pm 0.3\%$
		q	$70.8 \pm 0.1\%$	$77.2 \pm 0.2\%$	$83.5 \pm 0.1\%$
		s	61/64 kb	137/128 kb	339/256 kb
		t	77 ms	158 ms	443 ms
	T=100	u	$75.7 \pm 0.6\%$	$81.5 \pm 0.4\%$	$86.2 \pm 0.2\%$
		q	$75.2 \pm 0.6\%$	$80.9 \pm 0.1\%$	$85.9 \pm 0.3\%$
		s	112/128 kb	239/256 kb	543/512 kb
		t	137 ms	315 ms	877 ms
No-pooling	T=50	u	$72.8 \pm 0.2\%$	$79.8 \pm 0.1\%$	$86.1 \pm 0.1\%$
		q	$72.9 \pm 0.2\%$	$79.6 \pm 0.3\%$	$85.9 \pm 0.1\%$
		s	61/16 kb	137/32 kb	339/64 kb
		t	24 ms	44 ms	114 ms
	T=100	u	$77.9 \pm 0.1\%$	$83.8 \pm 0.1\%$	$88.6 \pm 0.2\%$
		q	$77.6 \pm 0.3\%$	$83.7 \pm 0.2\%$	$88.2 \pm 0.1\%$
		s	112/32 kb	239/64 kb	543/128 kb
		t	38 ms	88 ms	223 ms

Table 2: Obtained results for the two model architectures. The u/q-rows are the classification scores for the unquantized and quantized models, the s-rows are the quantized model weights and buffers sizes, and the t-rows are the inference times to classify one input segment on the i.MX RT1064 development board.

Note 1: The log-Mel feature extraction part requires 70 kb of memory and takes about 29.75 ms and 59.50 ms to compute 50 and 100 input frames respectively.
Note 2: The model weights and buffer sizes are indicative only. For an embedded deployment larger memory banks are required to store intermediate feature maps and additional overhead.
Note 3: The classification latency is dependent of the used input size. More specifically, a classification can only occur when a complete input segment is processed from the audio data.

ESANN 2021 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Online event, 6-8 October 2021, i6doc.com publ., ISBN 978287587082-7.
Available from http://www.i6doc.com/en/.

5 Conclusions

This paper investigates a convolutional neural network (CNN) based architecture for a real-time on-edge classification of domestic sound events. Two different CNN model architectures are examined, i.e. a *'pooling'* and a *'no-pooling'* approach to reduce the feature maps and the number of arithmetic operations throughout the network, and are evaluated with respect to classification performance, memory footprint and inference time. The proposed model architectures are evaluated on a self-recorded acoustic event dataset collected from 72 different home environments. The obtained results indicate that the no-pooling model architecture settings achieve higher classification scores compared to their pooling counterparts, and that they also come with a significantly reduced memory footprint (factor four) and a faster inference time (factor three to four). Furthermore, this work also shows that the used quantized-aware retraining procedure to convert the learned models to an 8-bit fixed-point representation for an embedded deployment yields no degradation in classification performance. The best obtained embedded classification accuracy is $88.2 \pm 0.1\%$ and is obtained by the no-pooling model architecture using an input size of 100 frames and 64 filters per convolutional layer. This model architecture requires 543 kb for the model weights and 128 kb to store the intermediate feature maps, and takes only 223 ms to classify 1.0 second of audio data. Future research will mainly focus to the development of an on-edge model adaptation strategy using a fixed-point learning scheme. This allows the model to adapt its model parameters and to learn new sound events on-the-fly in a real-time setting from newly collected samples. Furthermore, future research will also focus to reduce the energy consumption of the proposed system architecture in order to enable battery powered solutions. Possible techniques to lower the power consumption can be done on both the hardware level (e.g. optimized hardware) and software level (e.g. standby mode when no sound is detected).

References

[1] V. De Leon, Y. Alcazar and J. L. Villa, Use of Edge Computing for Predictive Maintenance of Industrial Electric Motors. In *Applied Computer Sciences in Engineering*, 2019, pages 523-533.

[2] D. Lin and Y. Tang,. Edge Computing-Based Mobile Health System: Network Architecture and Resource Allocation. In Proceedings of the *IEEE Systems Journal, vol. 14, no. 2*, 2020, pages 1716-1727.

[3] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang and W. Shi. Edge Computing for Autonomous Driving: Opportunities and Challenges. In Proceedings of the *IEEE vol. 107, no. 8*, 2019, pages 1697-1716.

[4] A. Politis, A. Mesaros, S. Adavanne, T. Heittola and T. Virtanen, Overview and Evaluation of Sound Event Localization and Detection in DCASE 2019. In Transactions on *Audio, Speech, and Language Processing (TASLP)*, 2020, Pages 1-14.

[5] 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. In Proceedings of the *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pages 2704-2713.