INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Neural Network Distillation on IoT Platforms for Sound Event Detection

Gianmarco Cerutti[1], *Rahul Prasad*[1,2], *Alessio Brutti*[1], *Elisabetta Farella*[1]

[1]ICT-irst, Fondazione Bruno Kessler, Trento, Italy
[2]School of Information Science, Manipal Academy of Higher Education, Manipal, Karnataka, India
{gcerutti,rprasad,brutti,efarella}@fbk.eu

Abstract

In most classification tasks, wide and deep neural networks perform and generalize better than their smaller counterparts, in particular when they are exposed to large and heterogeneous training sets. However, in the emerging field of Internet of Things memory footprint and energy budget pose severe limits on the size and complexity of the neural models that can be implemented on embedded devices. The Student-Teacher approach is an attractive strategy to distill knowledge from a large network into smaller ones, that can fit on low-energy low-complexity embedded IoT platforms. In this paper, we consider the outdoor sound event detection task as a use case. Building upon the VGGish network, we investigate different distillation strategies to substantially reduce the classifier's size and computational cost with minimal performance losses. Experiments on the *UrbanSound8K* dataset show that extreme compression factors (up to $4.2 \cdot 10^{-4}$ for parameters and $1.2 \cdot 10^{-3}$ for operations with respect to VGGish) can be achieved, limiting the accuracy degradation from 75% to 70%. Finally, we compare different embedded platforms to analyze the trade-off between available resources and achievable accuracy.

Index Terms: Sound Event Detection, Knowledge Distillation, Internet of Things, Deep Learning.

1. Introduction

A key aspect in developing services tailored to individual citizens in a smart city is access to information about behaviors and contexts. This combination is critical to understand events and trigger consequent actions. A clear trend facilitating this vision is the adoption of Internet of Things (IoT) technologies that connect everything everywhere providing new services for citizens and administrators. This is enabled by the growing functionalities of mobile devices and sensor nodes at low-cost. Physical spaces, people and even objects can be sensorized and wirelessly connected to become producers of data afterwards processed by artificial intelligence, typically in the cloud. However, this vision implies a data deluge. And, furthermore, demanding energy requirements to support both pervasive communication and energy autonomy of all the data-producer devices. In addition, privacy issues increase with data sources and their distribution, when information control is delegated to the cloud [1]. The collection of all these data is largely unattainable since the cost of communication is high in terms of network infrastructure, but also in terms of reliability, latency and, finally, energy available in mobile devices. [2].

Thanks to the improvements in embedded technology, microcontrollers with consumption in the range of mW enable artificial intelligence directly on board, without the need to access the cloud. Therefore, a viable approach is to analyze information near the sensor and to communicate only condensed highly informative data, many orders of magnitude smaller than the original ones, to the cloud or to a close gateway [3] [4].

These concepts apply to the scenarios of acoustic scene recognition and Sound Event Detection (SED) on which we focus in this work due to their applicability to smart city: traffic monitoring [5], crowd monitoring [6], analysis of occupancy level for energy efficiency in buildings [7], and so on. The benefit of understanding events locally where they happen would be high in terms of privacy [8], enables reaction time in the range of ms and guarantees device lifetime up to several years of operation, when energy harvesting is applied and the transmission is limited to few bytes. However SED, especially in outdoor contexts, is particularly challenging since, after some pioneering efforts [9], an established solution is not available yet. Recent progresses in deep learning and the release of sound event datasets and challenges like *UrbanSound8K* [10], AudioSet [11] and DCASE [12] have reawakened interest in these applications. Nevertheless, advances in terms of accuracy and robustness of current SED algorithms are achieved by using large neural network, which are increasingly demanding in terms of computational cost and memory. This fact prevents the development of applications suitable for cheap low power low complexity platforms, which would be required when targeting a pervasive network of energy neutral devices monitoring public spaces. Table 1 shows a non-exhaustive list of processing platforms potentially adequate to be integrated in an end-device, with their power consumption and memory footprint.

Table 1: *Examples of embedded platforms and their hardware capabilities.*

Board Name	Flash[KB]	RAM[KB]	Power [mW]	MIPS
Arduino	32	2	60	20
ChipKit uc32	512	32	181	124.8
STM32L476RG	1024	128	26	100
TI MSP432P4111	2048	256	23	58.56
BeagleBone Black	Ext	524288	2300	1607
Raspberry Pi 3 B+	Ext	1048576	5500	2800

The last two platforms in Table 1 have enough RAM and Million Instructions Per Second (MIPS) to handle state-of-the-art models for SED. However, their power consumption would require recharging the battery periodically or a wired power supply. The other platforms are more energy efficient, but this comes at the costs of very limited hardware resources: shallow networks fitting on board would perform considerably worse than the state-of-the-art counterparts if trained from scratch [13]. Therefore, network complexity reduction while preserving the generalization capability is of major interest towards porting deep and complex architectures on heavily constrained IoT node. Ones way to achieve this goal is Knowledge Distillation (KD) [14] that, taking advantage of the redundancy that characterizes large networks [15], allows training small networks mimicking the big ones.

http://dx.doi.org/10.21437/Interspeech.2019-2394

In this paper, starting from a SED classifier for *UrbanSound8K* [16] composed of the publicly available VGGish [17] feature extractor and a recurrent classifier, we apply KD to obtain extreme compression factors while limiting the performance degradation. The resulting reduced networks fit in a large variety of low-power microcontrollers, such as most of those in Table 1, thus enabling deep neural networks for audio event detection on embedded platforms. Note that this high parameter reduction is a peculiarity of our study, whereas knowledge distillation is often used in literature to slightly reduce networks to obtain performance improvements.

The following sections will demonstrate the possibility of classifying sound event outdoor, with approximately state-of-the-art performance, using very thin neural networks on embedded platforms.

2. Related Works

In literature, several approaches exist to reduce the number of parameters of a neural network: network pruning [18], parameter sharing [19] and matrix decomposition [20]. The last two methods reduce the number of network parameters (e.g. sharing the weights), but they keep the same architecture, therefore they require the same buffers (RAM) and throughput (MIPS). Network pruning requires manual setup of sensitivity for each layer and fine-tuning of the parameters. In addition, the network depth cannot change.

An attractive approach is KD that transfers the knowledge of a complex model into a simpler one [21]. This approach is also referred to as Student-Teacher, because the smaller network (student) is trained to mimic the output of the larger one (teacher) [22]. The underlying idea is that the output of the neural network (soft labels) is richer in information than the hard labels and makes the training easier [14]. For example, the logits of a pre-trained network can give information about the similarity of two samples of different classes.This paradigm was first proposed in [21], where a shallow network was trained to mimic an ensemble of strong classifiers, without decreasing performance. Being in the early stages of deep learning, the work was limited to shallow models. KD was further investigated in [14] by merging two losses: one from comparison between a softened version of the teacher output and the other based on ground-truth labels. Unfortunately, the distilled models are also cumbersome. More recently, higher compression factors, with students up to 36 times smaller than the teacher, are achieved in [23], by using also intermediate output of the teacher network in the loss definition. However, the reduction obtained is not sufficient in our application context.

3. Knowledge Distillation using a Compound Loss

Considering a generic neural architecture where a feature extractor is followed by classifier, several strategies can be adopted to train the student network. Figure 1 graphically shows the distillation process, where the upper part represents the teacher network and the lower part is the student.

The first training strategy uses the hard (one-hot) labels, which corresponds to training the network from scratch without any other knowledge. Since we are dealing with a multi-class classification problem, the loss is based on cross-entropy:

$$\mathcal{L}_{\mathrm{h}}(\mathbf{X}) = -\sum_{n=1}^{N}\sum_{c=1}^{C} \mathbf{y}_n \log(p_c(\mathbf{x}_n)), \tag{1}$$

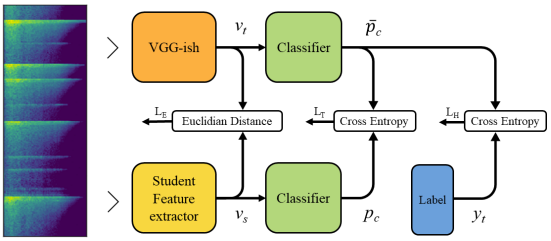

Figure 1: *Block diagram of KD using teacher intermediate features (L_E), soft teacher output (L_T) and dataset labels (L_H)*

where $\mathbf{X} = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$ is the set of the network input features, N is the number of samples, C is the number of classes, $\mathbf{Y} = \{\mathbf{y}_1, \dots \mathbf{y}_N\}$ are the one-hot labels, $\mathbf{y}_n$ is a C-dimensional binary vector and $p_c(\mathbf{x}_n)$ is the network output for class $c = \{1, \dots, C\}$ given the input vector $\mathbf{x}_n$. Denoting as $\overline{p_c}(\mathbf{x}_n)$ the output of the teacher network, the soft loss $\mathcal{L}_{\mathrm{s}}(\mathbf{X})$ can be formulated by replacing the hard labels with the teacher output:

$$\mathcal{L}_{\mathrm{s}}(\mathbf{X}) = -\sum_{n=1}^{N}\sum_{c=1}^{C} \overline{p_c}(\mathbf{x}_n) \log(p_c(\mathbf{x}_n)). \tag{2}$$

Combining equations 1 and 2 leads to the approach in [14]:

$$\mathcal{L}_{\mathrm{h,s}}(\mathbf{X}) = \alpha_h \mathcal{L}_{\mathrm{h}}(\mathbf{X}) + \alpha_s \mathcal{L}_{\mathrm{s}}(\mathbf{X}), \tag{3}$$

where α_h and α_s are combination weights. A further strategy, in line with [23], is to make the student learn how to replicate the features produced by the teacher, leaving the classification independent. The embedding loss $\mathcal{L}_{\mathrm{e}}(\mathbf{X})$ is thus defined as:

$$\mathcal{L}_{\mathrm{e}}(\mathbf{X}) = \sum_{n=1}^{N} ||v_t(\mathbf{x}_n) - v_s(\mathbf{x}_n)||^2, \tag{4}$$

where $v_t(\cdot)$ and $v_s(\cdot)$ are the feature vectors produced by the teacher and student networks respectively. Finally, we consider a compound loss obtained as linear combination of the three losses introduced above:

$$\mathcal{L}(\mathbf{X}) = \alpha_h \mathcal{L}_{\mathrm{h}}(\mathbf{X}) + \alpha_s \mathcal{L}_{\mathrm{s}}(\mathbf{X}) + \alpha_e \mathcal{L}_{\mathrm{e}}(\mathbf{X}) \tag{5}$$

4. Experimental Analysis

Considering SED in outdoor urban environments, two datasets are often used in literature: *UrbanSound8K* and TUT Sound events 2017. The latter is particularly attractive because it features real recordings and several comparative methods are available thanks to the related challenge DCASE2017 [24]. Unfortunately, the task is very challenging and classes are highly unbalanced, preventing a fair evaluation of KD methods.

Therefore, we used the *UrbanSound8K* dataset [10]. It includes 8732 audio samples related to the city environment, with different sampling rates, number of channels and maximum length of 4 seconds. Each recording has a unique label among ten possible classes: air conditioner, car horn, children playing, dog bark, drilling, engine idling, gun shot, jackhammer, siren, and street music. All clips are taken from Freesound[1], a huge collaborative database of audio samples.

Following the recipe reported in [10], we use 10-fold cross validation and average scores. However, we took one additional

[1]"http://www.freesound.org"

fold for validation: 8 folds are used as training data, one is used as validation and the remaining one for test (training-validation-test ratio is 0.8-0.1-0.1). We choose randomly the validation one, one index less than the test one (for example test fold 9, validation fold 8). Performance is measured in terms of classification accuracy. In some experiments, the dataset is augmented through pitch shifting [16], with both positive (up) and negative (down) semitones with values -2,-1,1,2.

4.1. Teacher

State-of-the-art solutions for SED are mostly Convolutional Neural Network (CNN) fed with mel-spectrogram [25][26][27]. However, to fully exploit the potential of the KD approach, rather than training from scratch our big CNN, we employed the publicly available VGGish[2] feature extractor [17], followed by a classification stage tailored on *UrbanSound8K*. VGGish is trained on the Youtube-8M Dataset [28] and it is expected to generalize well to other application contexts. Note that this fact introduces a further novelty in our work since distillation is performed on a different dataset than that used in the original training. VGGish converts $960\,\mathrm{ms}$ audio input mel spectrogram into a 128 dimensional embedding that can be used as input for a further classification model. The classifier can be shallow as the VGGish embeddings are more semantically compact (than raw audio features). The VGGish architecture is described in Table 3. For all the convolutional layers the kernel size is 3, the stride is 1 and the activation function is ReLu. Max pooling layers are implemented with a 2x2 kernel and a stride of 2.

The classifier, attached after the fully connected layer with 128 units, consists of a Gated Recurrent Unit (GRU) followed by a fully connected layers and maps the VGGish embeddings into the 10 classes of *UrbanSound8K*. A BatchNormalization layer is inserted between the feature extractor and the classifier to accelerate training.

VGGish expects as input $960\,\mathrm{ms}$ of audio signal sampled at $16\,\mathrm{kHz}$. Each clip in *UrbanSound8K* is resampled and padded or cropped to get a length of 960*4=$3840\,\mathrm{ms}$. The resulting signal is divided into 4 non-overlapping $960\,\mathrm{ms}$ frames. For each frame, the short-time Fourier transform is computed on $25\,\mathrm{ms}$ windows every $10\,\mathrm{ms}$. The resulting spectrogram is integrated into 64 mel-spaced frequency bins, covering the range 125-7500 Hz, and log-transformed. This gives 4 patches of 96 x 64 bins that form the input of the VGGish.

4.2. Student

Although our goal is to fit the classifier on an IoT device, we consider 4 different degrees of compression, to better assess the potential and limits of the proposed approach. Note that the student networks reduce drastically the feature extractor only. The classifier is re-trained, but it keeps the same architecture, i.e. a recurrent layer and 10-unit fully connected layers. We evaluate 4 training strategy based on the 4 losses in section 3. Weights are listed in Table 2. The T_e training procedure is composed by two independent training stages: the student is trained to replicate the VGGish features, minimizing the cost function in eq.4; then the classifier is trained with the hard loss in equation 1). For each training strategy we train 5 different models.

Table 3 shows different networks, built changing layers and number of filters in an heuristic way. The model subscript is the approximate number of parameter of the upstream part. Table 4 reports the computational requirements of each network: number of parameter to store, operations and buffer size. This approximation refers to an implementation using 8-bit quantized weights and buffers size as specified in the CMSIS-NN library, an embedded-C framework for neural network developed for Cortex M4-M7 based microcontroller[29]. During training it is important to have floating-point 32-bit representation, because it allows fine tuning. However, such high precision is not required during inference [30]. With 8 bit quantization assumption, each parameter takes one byte, thus the number of parameters is equal to the non-volatile memory needed.

The number of operations (both multiplications and sums) depends on the layer type: in convolution and maxpooling layers, each output pixel comes from a filter application, then each filter application requires, given the kernel size k and the number of input channels c:

$$Ops_{filter} = 2 \cdot c \cdot k^2 \tag{6}$$

$$Ops_{Conv} = Ops_{filter} \cdot out_H \cdot out_W \cdot out_C \tag{7}$$

where out refers to output shapes for height, width and channels

For dense and GRU layers the number of operations is the number of matrix multiplications:

$$Ops_{DotProduct} = 2 \cdot in_{shape} \cdot out_{shape} \tag{8}$$

Gated recurrent unit requires also small element-wise product.

Concerning the buffer size, one buffer must contain the biggest input of all layers and another one must contain the

Table 2: *Combination weights for different training strategies*

	α_h	α_s	α_e
T_h	1	0	0
$T_{h,s}$	0.5	1	0
$T_{h,s,e}$	1	1	1
T_e	0	0	1

Table 3: *Architecture of the models under analysis. "-" means that the layer is not active, "x" means that the layer is active.*

Layer	VGGish/M_{70M}	M_{20M}	M_{2M}	M_{200k}	M_{20k}
Conv1	64	64	32	8	4
Pool1	x	x	x	x	x
Conv2	128	128	64	16	8
Pool2	x	x	x	x	x
Conv3	256	256	128	32	16
Conv4	256	-	-	-	-
Pool3	x	x	x	x	x
Conv5	512	256	128	64	16
Conv6	512	-	-	-	-
Pool4	x	x	x	x	x
Conv7	-	-	-	64	32
Pool5	-	-	-	x	x
FC1	4096	2048	512	256	64
FC2	4096	2048	-	-	-
FC3	128	128	128	128	128
BatchNorm	x	x	x	x	x
GRU	20	20	20	20	20
FC4	10	10	10	10	10

Table 4: *Computational and memory requirements for each network*

	VGGish	M_{20M}	M_{2M}	M_{200k}	M_{20k}
#Param	~72.1M	~18.0M	~1.88M	~202k	~30.6k
#Operations	~1.72G	~608M	~148M	~13.6M	~2.11M
#Buffer [B]	~614k	~602k	~301k	~76.0k	~34.3k

[2]"https://github.com/tensorflow/models/tree/master/research/audioset"

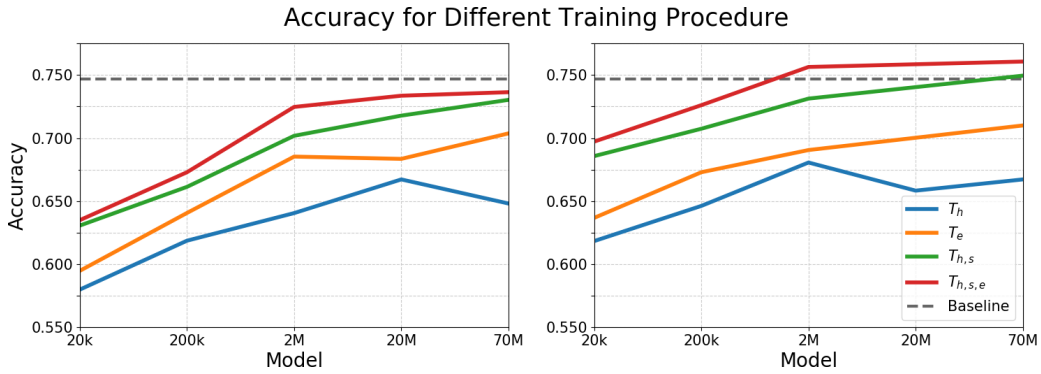

Figure 2: *Average accuracy for the 5 different models in table 3 with (right) and without (left) data augmentation. Each line represents a different training strategy. Dashed line is the teacher accuracy, here considered as a baseline.*

biggest output, because they can be reused in following layers. The CMSIS-NN framework requires also a smaller buffer for intermediate calculations, which can be reused in the implementation too.

5. Results

In this section, we analyze how different compression factors impact on the final classification accuracy, using the 4 losses described in Sec.3 and the 4 models in Table 3. Figure 2 reports the results in terms of classification accuracy without (left) and with (right) data augmentation. First of all, the training strategy $T_{h,s,e}$ is the best performing for all model sizes. As expected, the worst approach is training the student from scratch using the hard labels (T_h). Without data augmentation, all methods are below the teacher performance, the dashed line. This is quite unexpected as the larger model is, basically, a copy of the teacher. One of the possible reasons is that *UrbanSound8k* is not rich enough to learn how to mimic the VGGish model trained on a much larger dataset. This is partially confirmed in the right part of Figure 2, where data augmentation is employed: all models and training strategies substantially improve and the teacher is outperformed also by model M_{2M}. Note also that although the smallest network M_{20k} never achieves the baseline result, its accuracy is just 5 points lower than VGGish (74.7% vs 69.7%) using less than 0.0424% of parameters and 0.12% of operations.

5.1. Hardware Comparison

Since our goal is running SED on an IoT platform, this section explores achievable performance for the set of embedded devices in Table 1. For simplicity, we assume that the number of operations is equal to the number of instructions. Even if this is not true in general, the number of instructions per second is the closest number to operations per second available in microcontroller datasheets. Branch instructions are not taken into account, thus instructions are typically slightly more than operations. At the same time, most of 32 bit microcontrollers support Single Instruction Multiple Data (SIMD) (multiple sums and multiplications in a single instruction), thus operations can be less than instructions. Therefore, we can assume that these two effects are balanced and the number of operation is equal to the number of instruction. To ensure real time classification, we assume that the system must process each 960 ms audio frame (~1 second) before the next frame. Thus, classification time must be shorter than 1 second. Of course, we are not considering the time and resources required for other processing stages (e.g., mel-bins extraction) Given this two assumptions, the Instructions Per Second (IPS) available should be larger or equal than the operation needed in one forward propagation. Memory constraints are also relevant: the RAM on-board must contain the intermediate values and the non-volatile memory should contains all network parameters. However, non-volatile memory is not the main limit in our examples: whenever the network does not fit in Flash memory it does not respect one of the others two parameters (RAM or IPS). For this reason, Figure 3 reports only these two constraints.

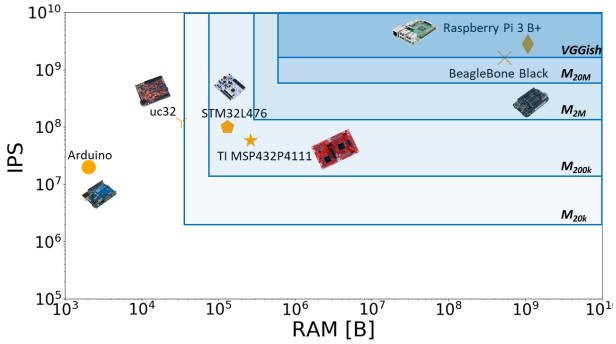

Figure 3: *Models requirements versus hardware capabilities. Rectangles define the network requirements in term of Instruction per Second (IPS) and volatile memory (RAM).*

None of the models proposed fits Arduino platform due to its limited RAM. Uc32 is faster than the STM32 and TI platforms, but it cannot run the smallest model M_{20k} because it would need ~2 kB of RAM more. BeagleBone Black has enough hardware capability to run most of the models, but VGGish fits only in Raspberrypi 3 B+. However, they both consumes too much. To conclude, M_{20k} and M_{200k} are the only suitable models for the target application.

6. Conclusions

In this work, we implemented different KD approaches for SED using CNN. We push the KD paradigm to extreme compression values, achieving models suitable for real time applications on IoT nodes. We compared different KD technique and boosted up the accuracy of the smallest model from 57.5% with standard training to 69.7%. Finally, we analyzed the hardware requirements of the 5 distilled neural networks. The model M_{200k}, with only ~202k parameters, would fit in many low-power embedded platforms achieving just 2.2 points less accuracy than the large baseline model (72.5% versus 74.7%).

7. References

[1] R. G. Baraniuk, "More is less: Signal processing and the data deluge," *Science*, vol. 331, no. 6018, pp. 717–719, 2011. [Online]. Available: http://science.sciencemag.org/content/331/6018/717

[2] P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel, A. Puschmann, A. Mitschele-Thiel, M. Muller, T. Elste, and M. Windisch, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70–78, February 2017.

[3] F. Conti, R. Schilling, P. D. Schiavone, A. Pullini, D. Rossi, F. K. Grkaynak, M. Muehlberghuber, M. Gautschi, I. Loi, G. Haugou, S. Mangard, and L. Benini, "An iot endpoint system-on-chip for secure and energy-efficient near-sensor analytics," *CAS*, vol. 64, no. 9, pp. 2481–2494, Sep. 2017.

[4] M. Rusci, D. Rossi, E. Farella, and L. Benini, "A sub-mw iot-endnode for always-on visual monitoring and smart triggering," *IoT-J*, vol. 4, no. 5, pp. 1284–1295, Oct 2017.

[5] Y. Na, Y. Guo, Q. Fu, and Y. Yan, "An acoustic traffic monitoring system: Design and implementation," in *UIC-ATC-ScalCom*. IEEE, 2015, pp. 119–126.

[6] Q. Meng and J. Kang, "The influence of crowd density on the sound environment of commercial pedestrian streets," *Science of the Total Environment*, vol. 511, pp. 249–258, 2015.

[7] S. Uziel, T. Elste, W. Kattanek, D. Hollosi, S. Gerlach, and S. Goetze, "Networked embedded acoustic processing system for smart building applications," in *DASIP*. IEEE, 2013, pp. 349–350.

[8] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," *SIGCOMM*, vol. 45, no. 5, pp. 37–42, Sep. 2015. [Online]. Available: http://doi.acm.org/10.1145/2831347.2831354

[9] A. Temko, R. Malkin, C. Zieger, D. Macho, C. Nadeu, and M. Omologo, "Clear evaluation of acoustic event detection and classification systems," in *CLEAR*, R. Stiefelhagen and J. Garofolo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 311–322.

[10] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *MM*. ACM, 2014, pp. 1041–1044.

[11] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *ICASSP 2017*, New Orleans, LA, 2017.

[12] A. Mesaros, T. Heittola, and T. Virtanen, "TUT database for acoustic scene classification and sound event detection," in *EUSIPCO*, Budapest, Hungary, 2016.

[13] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," *arXiv preprint arXiv:1605.07678*, 2016.

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

[15] Y. L. Cun, J. S. Denker, and S. A. Solla, "Advances in neural information processing systems 2," D. S. Touretzky, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, ch. Optimal Brain Damage, pp. 598–605.

[16] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *SIGNAL PROC LET. ISSN*, vol. 24, no. 3, pp. 279–283, 2017.

[17] S. Hershey, S. Chaudhuri, D. P. Ellis, J. F. Gemmeke, A. Jansen, R. C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold *et al.*, "Cnn architectures for large-scale audio classification," in *ICASSP*. IEEE, 2017, pp. 131–135.

[18] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *IJCNN*. IEEE, 1993, pp. 293–299.

[19] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *ICML*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 4095–4104.

[20] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing neural networks," in *NIPS*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 442–450.

[21] C. Bucilu, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *SIGKDD*. ACM, 2006, pp. 535–541.

[22] J. Shen, N. Vesdapunt, V. N. Boddeti, and K. M. Kitani, "In teacher we trust: Learning compressed models for pedestrian detection," *arXiv preprint arXiv:1612.00478*, 2016.

[23] A. Romero, N. Ballas, S. E. Kahou, A. Chassang, C. Gatta, and Y. Bengio, "Fitnets: Hints for thin deep nets," *arXiv preprint arXiv:1412.6550*, 2014.

[24] A. Mesaros, T. Heittola, A. Diment, B. Elizalde, A. Shah, E. Vincent, B. Raj, and T. Virtanen, "DCASE 2017 challenge setup: Tasks, datasets and baseline system," in *DCASE*, 2017.

[25] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *MLSP*. IEEE, 2015, pp. 1–6.

[26] Z. Zhang, S. Xu, S. Cao, and S. Zhang, "Deep convolutional neural network with mixup for environmental sound classification," in *PRCV*. Springer, 2018, pp. 356–367.

[27] X. Zhang, Y. Zou, and W. Shi, "Dilated convolution neural network with leakyrelu for environmental sound classification," in *DSP*. IEEE, 2017, pp. 1–5.

[28] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *ICASSP*. IEEE, 2017, pp. 776–780.

[29] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," *arXiv preprint arXiv:1801.06601*, 2018.

[30] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *ICML*, 2016, pp. 2849–2858.