Benchmarking Neural Network Architectures for Acoustic Sensor Networks

Janek Ebbers, Jens Heitkaemper, Joerg Schmalenstroeer, Reinhold Haeb-Umbach

Department for Communication Engineering, Paderborn University, Germany
Email: {ebbers,heitkaemper,schmalen,haeb}@nt.upb.de

Abstract

Due to their distributed nature wireless acoustic sensor networks offer great potential for improved signal acquisition, processing and classification for applications such as monitoring and surveillance, home automation, or hands-free telecommunication. To reduce the communication demand with a central server and to raise the privacy level it is desirable to perform processing at node level. The limited processing and memory capabilities on a sensor node, however, stand in contrast to the compute and memory intensive deep learning algorithms used in modern speech and audio processing. In this work, we perform benchmarking of commonly used convolutional and recurrent neural network architectures on a Raspberry Pi based acoustic sensor node. We show that it is possible to run medium-sized neural network topologies used for speech enhancement and speech recognition in real time. For acoustic event recognition, where predictions in a lower temporal resolution are sufficient, it is even possible to run current state-of-the-art deep convolutional models with a real-time-factor of 0.11.

1 Introduction

Audio and speech signal processing over wireless acoustic sensor networks (WASNs) [1] has become an interesting research topic for two reasons. First, if acoustic sensor nodes are distributed over an environment, it is more probable that a sensor is close to a relevant sound source than with a single central compact microphone array. Consequently, typical signal enhancement and classification tasks, such as noise reduction, dereverberation, source separation, acoustic event detection, and speech recognition can be expected to operate at higher SNR and thus deliver better enhancement and recognition results compared to a recording by a distant microphone. A second reason for the interest in WASNs is the ubiquity of wireless communications and the inexpensiveness of sensor nodes equipped with sensing, computing and communication capabilities.

However, the computing power on a sensor node is limited compared to what is available on desktop computers or even servers. This limitation stands in contrast to the recent developments in audio and speech processing, where compute intense deep neural networks (DNNs) have made inroads in almost every enhancement and classification task. Typical high-performance networks for acoustic event detection have tens of layers or even more and several millions of learnable parameters [2]. An interesting question is therefore: are high-performance DNNs a viable option for WASNs?

An often used approach to run signal processing tasks on sensor nodes is to distribute the algorithm over the network. Parts of the algorithm are run locally on a sensor node and the nodes exchange intermediate results among each other to eventually come up with almost the same enhancement/classification performance as a centralized solution, see e.g. [3]. In the case of neural networks, it needs to be investigated if this approach is sensible because the intermediate (hidden) layers of the network are often very wide and the transmission of their activations to neighboring nodes would result in a significant communication overhead. Nevertheless, this is a potentially interesting line of research, which is, however, not pursued here.

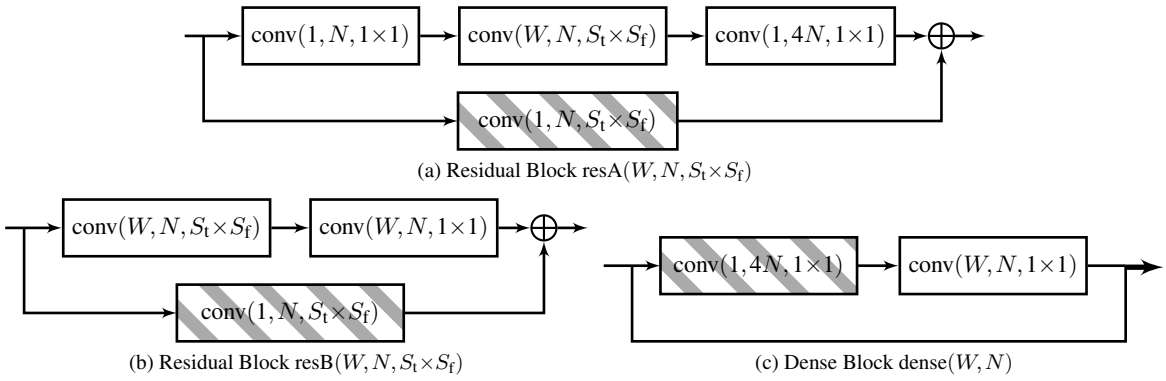
There has been a lot of work on reducing the computational and memory footprint of neural networks. One approach is to shrink weight matrices by means of singular value decomposition [4] or vector quantization [5]. Another interesting line of research applies ideas from approximate computing to reduce the computational effort [6, 7], e.g., by intentionally using errorprone computing elements such as multipliers with, however, greatly reduced energy consumption. If the network is trained with such only approximate computations, it can be made quite resilient to such imperfections. One can also introduce algorithmic changes to speed up computations or to reduce the memory footprint such as an efficient design of convolutional neural networks [8] or on-the-fly language model rescoring [9]. Finally, dedicated sensor node hardware can be employed to run DNNs.

In these techniques, either the software or the hardware is device specific to accommodate for resource constraints. However, a question to be answered in the first place is, what if we use off-the-shelf hardware and ubiquitous neural network software implementations, how complex a network one can afford to be? This is the question we are addressing in this contribution. We consider a typical widely used low-cost hardware platform, a Raspberry Pi Model 3b+ device, which, admittedly, is not a very low-cost and parsimonious sensor node, and benchmark typical high-performance network topologies on this device. We are considering two tasks typically performed on WASNs, acoustic event detection and automatic speech recognition. Interestingly, the platform can afford quite complex models and still offer real-time processing capabilities, as we will demonstrate in our experiments. This renders device-specific modifications for the most part unnecessary.

This paper is organized as follows. With a focus on acoustic event recognition first the corresponding DNN approach is explained and the employed DNN topologies will be presented. Secondly, the considered speech recognition system will be reviewed. Finally, the event and speech recognition systems will be benchmarked on the Raspberry Pi while recognition performance is evaluated on Google's AudioSet and the CHiME-4 robust speech recognition challenge, respectively.

2 Acoustic Event Recognition

Acoustic event recognition asks for predictions whether an event is active or inactive in a certain time segment. In particular, it is possible that multiple events are active at the same time. Therefore, we perform K binary classifications rather than a single one, with K being the number of events of interest. Due to the availability of large scale databases, we aim at making predictions in time segments of 10 s here, which is a resolution high enough to perform monitoring and diarization tasks. To do so, neural network models are used to output logits



(a) Residual Block resA$(W, N, S_\mathrm{t}\times S_\mathrm{f})$

(b) Residual Block resB$(W, N, S_\mathrm{t}\times S_\mathrm{f})$

(c) Dense Block dense(W, N)

Figure 1: Convolutional building blocks

$\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_T] = \mathrm{DNN}(\mathbf{X})$ in a higher temporal resolution first, where $\mathbf{X}$ denotes the input feature sequence and $\mathbf{y}_t$ denote the t-th K-dimensional logit vector.

Here, log-mel-band-energies (LMBEs) are used as input extracted from the short time Fourier transform (STFT) of the audio signal. Do note that the number of time steps T in the output does not necessarily need to match the number of time steps in the input here. The logits are finally averaged $\overline{\mathbf{y}} = \frac{1}{T}\sum_{t=1}^{T}\mathbf{y}_t$ and segment-level posteriors are obtained using the sigmoid activation $\tilde{\mathbf{z}} = \sigma(\overline{\mathbf{y}})$. The models are trained using the binary cross entropy loss

$$L(\tilde{\mathbf{z}}, \mathbf{z}) = -\sum_{k=1}^{K}\left[z_k \log \tilde{z}_k + (1 - z_k)\log(1 - \tilde{z}_k)\right]$$

with z_k denoting the binary target for the k-th event.

3 Neural Network Topologies

DNNs have been shown to be able to model high-level abstractions in data by using many layers of non-linear transformations each increasing the level of abstraction. Besides standard fully connected layers $\mathrm{fc}(N)$, with N denoting the number of units in the layer in this context, more sophisticated convolutional and recurrent layers allow to process structured input data, such as images and audio, more efficiently and facilitate training of deep structures. In the following, we will briefly review the basic building blocks used in the neural network topologies considered afterwards. For a more detailed explanation, however, it is referred to the literature. [10]

3.1 Building Blocks

Convolutional layers $\mathrm{conv}(W, N, S_\mathrm{t}\times S_\mathrm{f})$ perform discrete convolutions of its input feature maps with learnable kernels. It is defined by the kernel size $W\times W$, the number of kernels N and the stride $S_\mathrm{t}\times S_\mathrm{f}$. Convolutional layers are followed by batch normalization [11] and ReLU activation functions if not stated otherwise. Additional pooling stages $\mathrm{pool}(P_\mathrm{t}\times P_\mathrm{f})$ with the pool size $P_\mathrm{t}\times P_\mathrm{f}$ reduce the size of the feature maps and make extraction invariant to small translations.

In recent years several techniques and networks have been proposed to enable increasingly deeper models. In this work we consider residual and dense networks, which use skip connections to facilitate training as explained in the following.

Residual blocks [12] implement multiple convolutional layers in a feed-forward architecture. Additionally, residual connections are added, which bypass the convolutions by summing the convolutional output with the block's input as shown in Fig. 1a and Fig. 1b. Note that the convolutional layers might change the number or size of the feature maps due to a different number of kernels or striding, respectively. In that case, the gray hashed convolution is applied on the residual connection to match the input to the blocks output shape.

Dense blocks [13] perform one or more convolutions and concatenate the resulting feature maps and the input feature maps to build its output. To not change the size of the feature maps within a block, no striding is used. A dense block is shown in Fig. 1c. Because the total number of feature maps might become very large, optional bottle neck layers can be added, which apply a 1×1 convolution before the block's actual $W\times W$ convolution to reduce the number of input feature maps.

Recurrent layers $\mathrm{rnn}(N)$ are designed to process sequential data such as audio. Unlike fully connected and convolutional layers, which only have feedforward connections, recurrent layers also contain feedback connections. This means the output at a certain time step does not only depend on the current input but also on previous inputs from the sequence. To access information from the future as well, bidirectional recurrent layers process the input sequence in positive and negative time direction using two recurrent sub-layers with N units each and forward the concatenation of size $2N$ to the next layer. To mitigate the vanishing gradient problem through time [14], sophisticated recurrent architectures have been developed, which will be reviewed in the following.

Long-short-term-memories (LSTMs) employ, in addition to the feedback of the layer's output, an inner state, which allows memorizing information over long periods. Input gates, forget gates and output gates allow to store, delete and access information.

Gated recurrent units (GRUs) can be understood as a (simpler) variation of the LSTM. They drop the inner state and only use the layers output as a feedback. However, to not suffer from the vanishing gradient problem, the feedback is not transformed directly but reset and update gates are used to add information from the feedback to the input and to update the feedback to the new output, respectively. It has been shown that for some tasks GRUs perform better than LSTMs [15] as will also be observed in our experiments.

3.2 Topologies

First, we consider purely convolutional neural networks (CNNs), which are slightly modified versions of the weak acoustic labels (WAL) Net [16, 17], ResNet-50 [12] and DenseNet-121 [13]. While the first has been specifically designed for the considered event recognition task and have shown to achieve state-of-the-art, the two latter originate from the image recognition community but have also

WAL Net	ResNet	DenseNet
2×conv(3, 16, 1×1) pool(2×2)	conv(7, 16, 2×2)	
2×conv(3, 32, 1×1) pool(2×2)	pool(2×2)	
2×conv(3, 64, 1×1) pool(2×2)	3×resA(3, 64, 1×1) resA(3, 128, 2×2)	6×dense(3, 32) conv(1,128,1×1) pool(2×2)
2×conv(3, 128, 1×1) pool(2×2)	3×resA(3, 128, 1×1) resA(3, 256, 2×2)	12×dense(3, 32) conv(1,256,1×1) pool(2×2)
2×conv(3, 256, 1×1) pool(2×2)	5×resA(3, 256, 1×1) resA(3, 512, 2×2)	24×dense(3, 32) conv(1,512,1×1) pool(2×2)
conv(3, 512, 1×1) pool(2×2)	2×resA(3, 512, 1×1) pool(2×2)	16×dense(3, 32) pool(2×2)
conv(2, 1024, 1×1) (no padding)		
fc(K)		

Table 1: Purely convolutional neural networks used for acoustic event recognition. Input layer is at the top of the table.

Plain(k) + RNN	WRN(k) + RNN	Dense + RNN
conv(3, 16, 1×1)		
2×conv(3, 16k, 1×1) pool(2×2)	resB(3, 16k, 1×1) resB(3, 32k, 2×2)	7×dense(3, 16) conv(1,64,1×1) pool(2×2)
2×conv(3, 32k, 1×1) pool(1×2)	resB(3, 32k, 1×1) resB(3, 64k, 1×2)	12×dense(3, 16) conv(1,128,1×1) pool(1×2)
2×conv(3, 64k, 1×1) pool(5×2)	resB(3, 64k, 1×1) pool(5×2)	8×dense(3, 16) pool(5×2)
fc(1024)		
2×rnn(512)		
fc(1024)		
fc(K)		

Table 2: Hybrid convolutional-recurrent models.

been intensively used for and achieve state-of-the-art performance in event recognition [2, 18]. For these models 128 LMBEs are extracted using a frame length of 20 ms with an overlap of 10 ms. The architectures are outlined in Tab. 1. They consist of six stages each reducing feature size and time resolution by a factor of two. Convolutions in these stages use padding such that a reduction of size is only achieved by pooling and striding. A final convolutional layer without padding extracts 1024 features/0.64 s and a final fully connected layer outputs logits for K events. The Residual blocks and the dense blocks with bottleneck layers used here consist of three and two convolutional layers, respectively, resulting in a total of 13, 51 and 122 layers for the WAL Net, ResNet and DenseNet, respectively.

Secondly, we propose variants of CNN + recurrent neural network (RNN) hybrids shown in Tab. 2. The CNNs are inspired by the four block structure of wide residual networks (WRNs) [19]. Residual and the dense blocks consist of two and one conv layers here, respectively. Because the proposed models use fewer pooling/striding stages, only 64 LMBEs are extracted here in frames of 40 ms length with an overlap of 20 ms. The CNNs in total reduce the feature map size by factors of eight and ten in feature and time dimensions, respectively. Hence, the number of time steps that need to be processed by the following layers is reduced from 50/s to 5/s making it more efficiently for execution on devices with limited computational power. A linear layer maps the CNN output to 1024 features per time step, which build the input to the classifier network consisting of two recurrent layers followed by two linear layers.

4 Speech Recognition

For speech recognition a DNN / hidden Markov Model acoustic model (AM) is employed. Thus, the DNN is trained to provide the acoustic emission scores. An optional DNN based mask estimator aims at masking the speech signal and the noise signal, allowing to compute speech and noise statistics and perform speech enhancement beforehand. For a more detailed overview of this approach please refer to [20].

First, an STFT spectrogram with frame lengths of 25 ms, a hop size of 10 ms and FFT length of 512 is computed. In contrast to acoustic event recognition, masks and acoustic posteriors need to be provided in a higher temporal resolution which is why no temporal pooling and striding can be performed resulting in higher computational expenditure. Thus, only medium sized networks can be deployed on the Raspberry Pi. The topologies used are smaller versions of the networks proposed in [20, 21], which achieved very competitive performance on the benchmarking tasks CHiME-3, CHiME-4, and REVERB. The mask estimator network consists of one recurrent layer RNN with 256 units followed by three fully connected hidden and one output layers with 1024 and 512 units, respectively. For the subsequent AM 64 LMBEs are extracted. Here, WRN + RNN hybrid models are used similar to those from the acoustic event recognition. However, no temporal pooling and striding is performed and the last pooling operation is replaced by a stride in feature dimension in the initial conv layer. In contrast to batch normalization, statistics for normalization are only accumulated over time here, which has shown to yield better performance. Further, only a single recurrent layer with 256 units is used and the second recurrent layer is replaced by a fully connected layer with 1024 units.

5 Experimental Evaluation

We conduct benchmarking on a Raspberry Pi Model 3b+ device [22], which hosts a recent Raspbian Stretch installation including a Kernel version 4.14. The Pi is equipped with 1 GB LPDDR2 RAM memory and an ARM A53 quad-core processor running at 1.4 GHz. For additional information on our hardware please refer to [23]. We also provide an installation guide on how to setup a WASN consisting of Raspberry Pis [24]. On the Raspberry Pi we first build *Bazel* and subsequently *Tensorflow* [25] following the guide from [26]. For *Tensorflow* 1.5.0 we documented all required installation steps on our project website [24]. *Tensorflow* 1.5.0 in combination with *Bazel* 0.8.0 compiled on the Pis in a few hours, whereas newer versions of *Tensorflow* either broke down during the compilation process or after installation when importing *Tensorflow*. The investigated *Tensorflow* models are then executed on the Pi using floating-point arithmetic with 32 Bit precision. Execution time on the Raspberry Pi is measured by the real time factor (RTF) stating the ratio between execution time and length of the processed time segment. RTFs have been determined using *Tensorflow*'s benchmarking tool by processing 100 segments with a length of 10 s each.

For the event recognition task Google's AudioSet [27] is used for training and evaluation. The whole database consists of 2 M *Youtube* audio clips with a maximum length of 10 s and sampling frequency of 44.1 kHz. Each of the clips was manually labeled with event tags from a hierarchical ontology of K=527 different event labels in total. Due to limited resources, however, we use the bal-

Model	#Params/10^6	RTF	mAP	mAUC
WAL Net	**5.0**	**0.11**	19.1 %	92.5 %
ResNet	32.5	0.36	18.4 %	92.4 %
DenseNet	11.6	0.31	19.7 %	93.0 %
Plain + BGRU	$3.3+11.0$	$0.19+0.14$	**22.0 %**	**94.0 %**
WRN + BGRU	$4.9+11.0$	$0.43+0.14$	21.7 %	**94.0 %**
Dense + BGRU	$2.7+11.0$	$0.37+0.14$	21.8 %	93.8 %
Plain + BLSTM	$3.3+14.2$	$0.19+0.16$	19.4 %	93.5 %
Plain + GRU	$3.3+5.0$	$0.19+0.07$	20.3 %	93.3 %
Plain + LSTM	$3.3+6.3$	$0.19+0.08$	17.8 %	93.0 %

Table 3: Acoustic event recognition performance in terms of number of model parameters, real time factor on Raspberry Pi, mean average precision and mean area under curve. For all hybrid models k=4 is used.

anced training set here consisting of 22K audio clips with at least 59 examples per event. Validation is performed on a small portion of the unbalanced training set with at least 7 examples per event. For the final evaluation the official evaluation dataset is used comprising 20 K audio clips with at least 59 examples per event. However, do note that at the time of downloading around 2 % of the *Youtube* videos were no longer available.

Recognition performance is measured in terms of mean average precision (mAP) and mean area under curve (mAUC) [28]. The average precision (AP) for a single event class is computed as the average of precision values obtained for all thresholds accepting a new positive example. The area under curve (AUC) for a single event is given as the area under the receiver operating characteristic curve. The mAP and mAUC are then obtained as the mean of APs and AUCs over all event classes. For both metrics high values are desirable. Training is performed for 50 epochs using Adam optimization with a learning rate of 10^{-4} and an early stopping criterion of 6 epochs without improvement of the mAP on the validation set. The stated mAP and mAUC values represent one-shot results.

Results are presented in Tab. 3. Note that originally reported mAP and mAUC for the WAL Net are 21.3 % and 92.7 % [16], respectively, whereas we obtain 19.1 % and 92.5 % here. This may be explained by slightly different evaluation sets (due to the missing *Youtube* clips) as well as further implementation details. The WAL Net is reported as the current state of the art for balanced training, which can be executed on the Raspberry Pi with a RTF of 0.11. We can see that it is also possible to run the dramatically deeper ResNet and DenseNet architectures around three times faster than real time. However, no or only little improvement over the WAL Net can be achieved. This indicates that the training data used here might be too small to take advantage of the more complex ResNet and DenseNet architectures. The hybrid models allow to improve recognition performance over the purely convolutional networks and are still executable in real-time with a big margin. Using a plain seven-layer CNN combined with bidirectional GRU (BGRU) layers improves mAP and mAUC by 2.3 % and 1.0 %, respectively, compared to purely convolutional networks, and are better than what was stated in [16] as state of the art. Surprisingly GRUs perform noticeably better than LSTM layers here. If we aim at continuous online processing on a Raspberry Pi, only unidirectional models can be used, which perform worse than the bidirectional models here but still better than purely convolutional networks.

To evaluate speech recognition performance of the proposed models the real evaluation dataset of the fourth CHiME challenge [29] is used. The CHiME data was recorded on tablets with six microphones and a sampling rate of 16 kHz in four different noisy environments with negligible reverberation. The training procedure of [20] is adopted. Recognition performance is measured in terms of word error rate (WER) given as the ratio of Deletions, Insertions and Substitutions performed by the system and the number of actual words in the reference. Note that a trigram language model without rescoring from the KALDI toolkit [30] was used for decoding, which is not part of the benchmarking as we focus on neural network architectures here. The decoding, however, could be performed on a second node in an acoustic sensor network.

Model	#Params/10^6	RTF	WER
WRN(4) + BLSTM	$4.9+6.2$	$0.69+0.78$	20.0 %
+ BLSTM Mask	$+4.2$	$+0.37$	**11.4 %**
WRN(2) + LSTM	**1.7 + 4.6**	**0.26 + 0.71**	46.2 %
+ LSTM Mask	$+3.4$	$+0.34$	18.1 %

Table 4: Speech recognition performance in terms of number of parameters, real time factor on Raspberry Pi and word error rates.

Results are shown in Tab. 4. We can see that for continuous real time online processing on a single node only the smaller WRN(2) + LSTM acoustic model without beamforming can be used, which has a rather poor recognition performance of 46.2 % though. However, speech enhancement greatly improves results and we argue that the mask estimation and beamforming part can be performed on one node in a WASN, e.g. the node acquiring the signal, whereas the recognition is performed on another node. Note that mask estimation is only performed on a single channel here, which has shown to not degrade performance and saves computation time. If we are further able to perform offline speech recognition and distribute our AM on two different nodes, a recognition performance of 11.4 % can be achieved, which is, in this setting without language model rescoring, close to state of the art [31].

6 Conclusions

The experimental study reported in this paper has shown that even large neural networks for acoustic event recognition, which achieve state of the art classification performance, can be run on a Raspberry Pi much faster than real-time. Even a sophisticated automatic speech recognition system, which achieved competitive performance on recent challenge tasks, can be run below realtime when distributed on multiple sensor nodes. These are good news, because it means that for many real-world tasks a cloud-based solution is not necessary, which greatly enhances the user's confidence that his/her data are kept private. Certainly, the hardware platform used in this study was not very low-cost. For less powerful platforms, adjustments of the algorithms may be necessary. But for many scenarios it occurs, that devoting research efforts to improving recognition performance is more reasonable than spending efforts on downsizing algorithms to reduce computational and memory requirements.

Acknowledgements

This work has been supported by *Deutsche Forschungsgemeinschaft* under contract no. HA 3455/15-1 and SCHM 3301/1-1 within the Research Unit FOR 2457 (acoustic sensor networks), and under contract no. HA 3455/14-1.

References

[1] A. Bertrand, "Applications and trends in wireless acoustic sensor networks: A signal processing perspective," in *2011 18th IEEE Symposium on Communications and Vehicular Technology in the Benelux (SCVT)*, pp. 1–6, Nov 2011.

[2] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 131–135, IEEE, 2017.

[3] A. Bertrand and M. Moonen, "Distributed Adaptive Node-Specific Signal Estimation in Fully Connected Sensor Networks - Part II: Simultaneous and Asynchronous Node Updating," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5277–5291, 2010.

[4] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. INTERSPEECH*, 2013.

[5] Y. Wang, J. Li, and Y. Gong, "Small-footprint high-performance deep neural network-based speech recognition using split-vq," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4984–4988, April 2015.

[6] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "Impact: imprecise adders for low-power approximate computing," in *Proceedings of the 17th IEEE/ACM international symposium on Low-power electronics and design*, pp. 409–414, IEEE Press, 2011.

[7] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu, "ApproxANN: An approximate computing framework for artificial neural network," in *DATE workshop*, 2015.

[8] M. Meyer, L. Cavigelli, and L. Thiele, "Efficient convolutional neural network for audio event detection," *arXiv preprint arXiv:1709.09888*, 2017.

[9] X. Lei, A. Senior, A. Gruenstein, and J. Sorensen, "Accurate and compact large vocabulary speech recognition on mobile devices," in *Proc. INTERSPEECH*, 2013.

[10] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[11] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[13] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 1, p. 3, 2017.

[14] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[16] A. Kumar, M. Khadkevich, and C. Fugen, "Knowledge transfer from weakly labeled audio using convolutional neural network for sound events and scenes," *arXiv preprint arXiv:1711.01369*, 2017.

[17] A. Shah, A. Kumar, A. G. Hauptmann, and B. Raj, "A closer look at weak label learning for audio events," *arXiv preprint arXiv:1804.09288*, 2018.

[18] B. McMahan and D. Rao, "Listening to the world improves speech command recognition," *arXiv preprint arXiv:1710.08377*, 2017.

[19] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[20] J. Heymann, L. Drude, and R. Haeb-Umbach, "Wide residual BLSTM network with discriminative speaker adaptation for robust speech recognition," in *CHiME4 Workshop*, 2016.

[21] J. Heymann, L. Drude, and R. Haeb-Umbach, "A generic neural acoustic beamforming architecture for robust multi-channel speech processing," *Computer Speech and Language*, 2017.

[22] Raspberry Pi, "https://www.raspberrypi.org/," ARM documentation, 2018.

[23] DFG FOR 2457, "https://upb.de/asn/hardware," Hardware documentation, Acoustic Sensor Networks Project, 2018.

[24] DFG FOR 2457, "https://upb.de/asn/software," Installation guide, 2018.

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, *et al.*, "Tensorflow: a system for large-scale machine learning.," in *OSDI*, vol. 16, pp. 265–283, 2016.

[26] S. Abrahams, "github.com/samjabrahams/tensorflow-on-raspberry-pi/blob/master/guide.md," Installation guide for Tensorflow, 2018.

[27] 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 *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*, pp. 776–780, IEEE, 2017.

[28] T. Virtanen, M. D. Plumbley, and D. Ellis, *Computational Analysis of Sound Scenes and Events*. Springer, 2018.

[29] E. Vincent, S. Watanabe, A. A. Nugraha, J. Barker, and R. Marxer, "An analysis of environment, microphone and data simulation mismatches in robust speech recognition," *Computer Speech & Language*, vol. 46, pp. 535–557, 2017.

[30] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, *et al.*, "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*, no. EPFL-CONF-192584, IEEE Signal Processing Society, 2011.

[31] M. Kitza, A. Zeyer, R. Schlüter, J. Heymann, and R. Haeb-Umbach, "Robust online multi-channel speech recognition," in *Speech Communication; 12. ITG Symposium; Proceedings of*, pp. 1–5, VDE, 2016.