IA-NET: Acceleration and Compression of Speech Enhancement using Integer-adder Deep Neural Network

Yu-Chen Lin[2], *Yi-Te Hsu*[1], *Szu-Wei Fu*[2], *Yu Tsao*[1], *Tei-Wei Kuo*[1,2,3]

[1]Research Center for Information Technology Innovation, Academia Sinica, Taiwan
[2]Department of Computer Science and Information Engineering, National Taiwan University, Taiwan
[3]Hong Kong Institute for Advanced Study, Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong

f04922077@csie.ntu.edu.tw, b01901112@ntu.edu.tw, d04922007@ntu.edu.tw,
yu.tsao@citi.sinica.edu.tw, ktw@csie.ntu.edu.tw

Abstract

Numerous compression and acceleration techniques achieved state-of-the-art results for classification tasks in speech processing. However, the same techniques produce unsatisfactory performance for regression tasks, because of the different natures of classification and regression tasks. This paper presents a novel integer-adder deep neural network (IA-Net), which compresses model size and accelerates the inference process in speech enhancement, an important task in speech-signal processing, by replacing the floating-point multiplier with an integer-adder. The experimental results show that the inference time of IA-Net can be significantly reduced by 20% and the model size can be compressed by 71.9% without any performance degradation. To the best of our knowledge, this is the first study that decreases the inference time and compresses the model size, simultaneously, while producing good performance for speech enhancement. Based on the promising results, we believe that the proposed framework can be deployed in various mobile and edge-computing devices.

Index Terms: speech enhancement, deep neural networks, inference acceleration, model compression, arithmetic circuit

1. Introduction

In the past few years, deep learning (DL)-based models have been widely used in most fields of speech processing. Owing to the deep structure and various novel mechanisms, a deep neural network (DNN) can provide state-of-the-art performance for both classification and regression tasks. Certain classification tasks such as speech recognition [1,2], speaker recognition [3,4], and speech-emotion recognition [5] aim to determine the spoken content, speaker identity, and speaker's emotion, from the speech signals. Unlike classification tasks, the outputs of regression tasks consists of continuous numbers. Speech enhancement (SE), for example, designed for mapping a noisy speech signal into an enhanced one with improved intelligibility and quality [6–16], is an important regression task in speech processing.

The recent emergence of mature wearable devices has led to an increase in the development of Internet-of-Things (IoT) applications. However, owing to their deep structures, DL-based techniques require larger storage and higher computational costs. In contrast to mainframes and computers which can be equipped with the newest hardware such as graphics processing units (GPUs) and tensor processing units (TPUs) for the development of DNNs, wearable and IoT devices are furnished with only simple processors such as central processing units (CPUs) and microcontroller units (MCUs). Therefore, researchers aim to deploy powerful DNNs in embedded devices with limited computation and storage resources, and many studies have been conducted to realize this goal through model compression.

The existing research on the compression and efficient processing of DNNs can be categorized into three classes, according to their design levels, namely, *hardware platforms*, *memory technologies*, and *software algorithms* [17]. Many researchers have studied the hardware-platform level (e.g., the Intel Knights Mill CPU [18] and Nvidia Tegra), focusing on accelerating the inference time for DNN computation. Meanwhile, much work has also been done on the memory level (e.g., eDRAM [19] and 3D memory [20]), aiming to reduce the access energy for high-density memories. Both hardware-platform and memory level need additional hardware component support. On the other hand, the software-algorithm level aims to design algorithms or model structures to compress the model sizes and improve the computation efficiency of the DNN models.

Many successful algorithms have been developed to improve the compression and acceleration of DNN models. However, during our literature survey, we observed that most of these algorithms have been designed for classification tasks [21–26]. The effects of model compression and inference acceleration on regression tasks have much fewer investigations. Among the existing algorithms, Sun et al. [27] proposed a weights-sharing and quantization technique to compress a DNN-based SE model; Their results showed that although the model size can be compressed, the performance was degraded notably. Ko et al. [26] investigated the relation between the precision scaling process and SE performance, and concluded that removing a large number of bits could cause severe performance degradation. Although these related works tried to compress the size of SE models, their results indicated that short bit-width weights were not suitable for a regression tasks, such as speech signal generation. In this paper, we propose a novel integer-adder deep neural network (IA-Net) that can accelerate the SE process and compress the model size, simultaneously.

Generally, SE methods include masking- and mapping-based approaches. The mapping-based approaches can be implemented in either the spectral or the waveform domain. Compared to the spectral-mapping-based counterparts, the waveform-mapping-based approaches have the advantage of avoiding the imperfect phase information issue. Moreover, since waveform-mapping is conducted in the time domain, signal extraction and waveform reconstruction is not required. This expedites the online process, when compared to spectral-mapping. Therefore, in the present study, we focus on a waveform-mapping-based SE approach. In addition, to realize the proposed SE system in common embedded devices, we choose the fully convolutional network (FCN) [28] model that has relatively simpler structures,

compared to the other NNs, such as convolutional neural network (CNN) [29]. Most importantly, to accelerate the inference time while compressing the model size, we compressed the parameters of IA-Net during training to fit the computing format of the integer-adder circuit during inference. Thus, our framework does not need further complex designs for hardware support; it can be easily realized in common embedded devices.

2. Electronic circuits of arithmetic

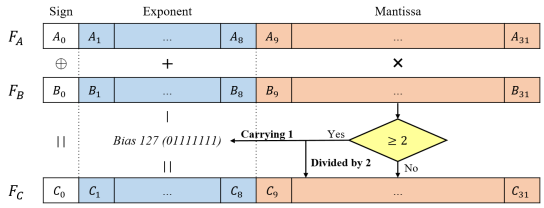

Figure 1: *Multiplication of floating point F_A and F_B.*

The single-precision floating-point representation of IEEE-754 [30] is the most common format used to represent the parameters of NNs. Its binary format is shown in Figure 1. A single-precision floating-point value, $bits$, consists of three parts: $bits[0]$ indicates the sign, $bits[1:8]$ are the exponent part, which represents a power of two ranging from -127 to $+128$, and the remaining $bits[9:31]$ contain the fractional value, also known as the mantissa (or significand or fraction), which is multiplied by 2^{exp} to obtain the decimal value[1]. Figure 1 shows an example of multiplication of two floating-point operands, $F_A \times F_B = F_C$, using a floating-point multiplier circuit. First, the output sign bit C_0 is calculated by the XOR operation of two input sign bits A_0 and B_0. Second, the effective values of the exponential terms are added together. Since the physical range of an 8-bit unsigned exponent is [0, 255], the addition result of the exponents must subtract 127, which is an offset also known as the *exponent bias* with the binary value of 01111111, shifting the result back to the range of $[-127, 128]$. For example, 16.0 multiplied by 4.0 equals 64.0; the exponent of 64.0 is 1000101, calculated by the following formula:

$$10000101 = 10000011 + 10000001 - 01111111 \quad (1)$$

where 10000011, 10000001 and 01111111 are the exponents of 16.0, 4.0, and the 127 $bias$, respectively. Finally, the two mantissa values are multiplied to obtain the result of the mantissa. Because the mantissa ranges from 1 to 2, the output of the multiplication always falls between 1 and 4. If the output mantissa is greater than or equal to 2, it will be divided by 2 and the value 1 will be carried over to the exponent part.

However, in the arithmetic logic unit (ALU), the designs of electronic circuits (e.g., adder, multiplier, divider, etc.,) can be roughly classified into integer and floating-point designs. Compared to the circuits for floating-point arithmetic, the electronic circuits for integers are less costly and more energy efficient, owing to the simpler structures. Furthermore, the designs of the other arithmetic electronic circuits are based on the integer adder. For example, the exponent part of the floating-point multiplier performs an addition, which is followed by a subtraction that can also be performed by adding the complement of the value. This inspired us to replace the floating-point multiplier with an integer adder for multiplication in the inference process of DNN-based SE. In this way, we can obtain a smaller model size and accelerate the computation, simultaneously, without requiring any additional hardware support or arithmetic circuit design.

[1]For the details of binary to decimal conversion, please refer to https://www.h-schmidt.net/FloatConverter/IEEE754.html.

3. Integer-adder neural network (IA-Net)

3.1. Fully convolutional neural networks

As mentioned in Section 1, most NN models are too large to be applied in an embedded device. Therefore, it is necessary to design a relatively small architecture for embedded devices. Fu et al. [31] conducted a study on raw waveform-based SE using FCNs. The only difference between FCNs and CNNs is that FCNs do not contain any fully connected layers. This provides two advantages for enhancing noisy raw-waveform speech. First, FCNs can model the raw waveforms more precisely. Training in a convolutional layer depends only on the neighboring input regions. Secondly, FCN-based SE contains a small number of model parameters, which is ideal for deployment in embedded devices. Therefore, we use FCN to perform waveform-based SE.

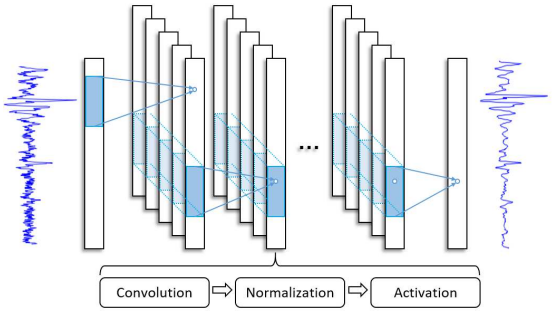

Figure 2: *The inference procedure of the FCN.*

The inference procedure of the FCN is shown in Figure 2. The noisy utterances are enhanced by the FCN model. The forward procedure of every convolutional layer can be roughly divided into three steps. First, the input speech (i.e., the output of the previous layer) is convolved by all the well-trained one-dimensional filters in the current layer. Secondly, we normalize each output channel to maintain the amplitude at the same level as that of the input raw waveform. Finally, the activation function transforms the convolved signal with a nonlinear function.

3.2. Replacing floating-point multiplier with integer-adder

To speed up the model computations, we replace the floating-point multiplier with an integer-adder, which is more time-efficient. However, the binary representation of floating-point numbers is very different from that of integers; therefore, we must design a new and appropriate mechanism to address this difficulty so that the forward procedures will not be affected. In addition, zero-value case is an special case in floating-point multiplication, wherein either of the operands involved in the multiplication is zero (i.e., all 31 bits of exponent and mantissa of the operand are 0s). The electronic circuit of the floating-point multiplier has already been designed to handle this special case. Thus, we only stored the position of zero values and utilize the original floating-point multiplier circuit to support this kind of multiplication. Finally, we adjust the format of all the floating-point values (except for the zero case) in our SE model to ensure that the values obtained by adding two floating-point numbers using an integer adder is identical to the original value obtained by multiplying them using a floating-point multiplier. The binary representation of a floating-point number has three partitions, as mentioned in Section 2. Thus, our adjustment for the binary representation is divided into three parts: the sign bit, the mantissa part, and the exponent part.

3.2.1. Sign bit

For the sign bit, we intend to substitute the XOR operator by utilizing the overflow support of the integer-adder circuit. If the addition of two sign bits is produced from these three combinations: $\{1, 0\}$, $\{0, 1\}$, or $\{0, 0\}$, then the outputs of the integer adder will be the same as the XOR operator. When the addition is produced from $\{1, 1\}$, the result will be 10, and the overflow will occur; however, the integer adder keeps only 0 in this case by its circuit design. As a result, the integer adder can obtain the same sign-bit output as the XOR of the floating-point multiplier.

3.2.2. Mantissa part

For the mantissa part, Hsu et al. [32] presented a study on SE using exponent-only floating point to quantize NN models. They claimed that full 32 bits of IEEE-754 floating-point representation were not necessary for the DNN-based SE model. Instead, exponent-only parameters were considered sufficient to obtain a high performance for SE. Therefore, we rounded the values of all the convolutional filters during training so that each value retained only the sign and exponent parts. Figure 3 shows an example of the training procedure for a floating-point filter weight.

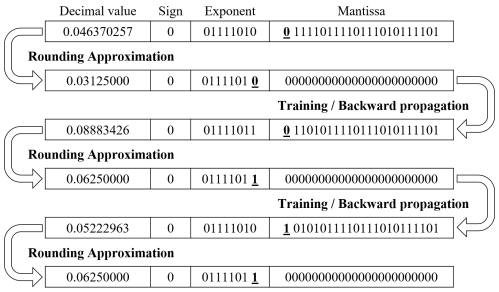

Figure 3: *Training procedure for a floating-point filter weight.*

3.2.3. Exponent part

In the case of the exponent part, there are two problems when replacing the floating-point multiplier with an integer adder. First, the sign bit may be affected by the result of exponent overflow when adding the exponent bits of two floating-point values. Therefore, we restrict the values of all signals and filters within the decimal range of $[-1, 1]$ by normalization. The binary value of the unsigned exponent part is restricted within the range of $[01111111, 00000000]$ (where 01111111 is the exponent part of ± 1 and 00000000 is the exponent of ± 0). In this way, the sign bit will not be affected by the result of exponent overflow.

Another problem is that the unsigned exponent of the result must subtract 127, the $bias$, after the addition of the exponents of two floating points as mentioned in Section 2. The floating-point multiplier has a complex circuit design, which can meet this requirement. However, in our software solution, we need to avoid increasing the number of instructions during inference, as it may slow down the inference time. Therefore, we solve this problem by adjusting the trained model. For the subtraction of *bias*, we fairly separate the subtraction of 127 into two subtractions of 64, 01000000, and 63, 00111111, since 2^{-64} and 2^{-63}, are far less than the absolute floating-point values (except for 0s) of the speech channels and filters, and they will not be affected.

We distribute the two subtractions of 64 and 63 among the speech channels and filters, respectively. Figure 4 presents the adjustment of a trained convolutional layer, where S is the speech signal, F denotes all the filters of convolution, and σ is the standard deviation of normalization. The differences between the adjusted and the original trained convolutional layers are the σ of normalization and each filter value of F. In convolution, we divide the values of all filters F by 2^{63} so that the exponent part of each value is additionally subtracted by 00111111 during the convolution of input speech S and filters F'. In normalization, the adjusted σ' is equal to $\sigma * 2^{64}$ so that all the values of the speech channels are additionally divided by 2^{64} during the normalization; that is, the exponent part of each value is subtracted by 01000000. In this way, the subtraction problem of $bias$ (127) is solved. Figure 5 shows an example of the conversion of the exponent part, in detail. Except for the first convolutional layer, which cannot be sped up since the input noisy utterances are not divided by 2^{64}, all the other convolutional layers behind can be accelerated by using our method.

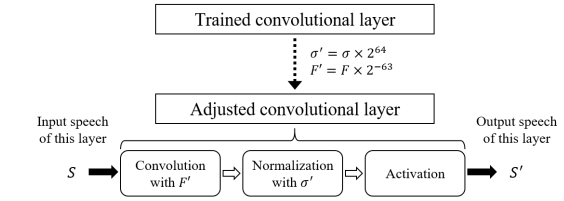

Figure 4: *Adjustment of a trained convolutional layer.*

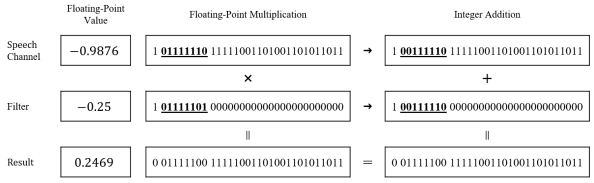

Figure 5: *An example of replacing the floating-point multiplier with an integer adder while performing floating-point multiplication.*

4. Experiment and analysis

4.1. Experimental Setup

In our SE experiments, we used the TIMIT corpus [33] as our training and test sets. We intentionally designed both the noise types and the SNR levels of the training and test sets to be mismatched in order to make the experimental conditions more realistic. For the training set, all of the 4620 training utterances from the TIMIT corpus were used, and they were further corrupted with 100 different types of noise (including stationary and non-stationary noise types) at eight signal-to-noise (SNR) levels (from -10 dB to 25 dB with a step of 5 dB) in a random manner. For the test set, we randomly selected another 100 distinct utterances from the TIMIT test set and corrupted these utterances using three other noise signals (engine, street, and two speakers) at four SNR levels (-6, 0, 6, and 12 dB).

To evaluate the performance and better understand the properties of our IA-Net, we conducted a series of experiments on a personal computer with only one core in the Intel(R) Core(TM) i7-6700 3.40 GHz CPU, with Ubuntu 16.0.4 as the operating system and 8192 MB memory. To evaluate the improvement in speed, we simulated the procedure of each convolutional layer using C language. Each input speech was simulated with 81920 samples of random floating-point values. The $clock$ function of $time.h$, a C library, was used to evaluate the real inference time while enhancing the noisy speech utterances. To evaluate the performance of the enhancement, we used the standardized

objective evaluation metrics, including the perceptual evaluation of speech quality (PESQ) [34], the short-time objective intelligibility measure (STOI) [35], and the speech distortion index (SDI) [36]. PESQ was designed to evaluate the quality of processed speech, and the score ranged from -0.5 to 4.5. A high PESQ score indicates that the enhanced speech is close to the clean speech. On the other hand, STOI was designed to compute the speech intelligibility, and the scores ranges from 0 to 1. A higher STOI score indicates better speech intelligibility. We used time-domain SDI, which measured the distortion by computing the normalized mean square error. A lower SDI value indicates a less speech distortion.

4.2. Experimental Result

In this experiment, we prepared two FCN models with different depths. The 10-layer FCN (FCN-10) had ten convolutional layers. The first nine layers consisted of 30 filters, with a filter size of 55. There was only one filter with a size of 55 in the last layer to generate the enhanced single-channel speech. For the 12-layer FCN (FCN-12), two additional convolution layers were added. These two FCN models were compared with the proposed IA-Net in terms of performance enhancement, inference time, and model size. For a fair comparison, the structure of the models and the number of epochs for training were kept the same for the original full- precision model and the compressed model.

Table 1: *Detailed scores of PESQ, testing time and model size for the original and IA-Net under specific SNR conditions.*

	FCN-10		FCN-12	
	Original	IA-Net	Original	IA-Net
SNR	PESQ	PESQ	PESQ	PESQ
-6	1.381	1.444	1.379	**1.514**
0	1.843	1.877	1.843	**1.920**
6	**2.304**	2.281	2.297	2.303
12	**2.729**	2.700	2.715	2.691
Ave.	2.064	2.076	2.058	**2.107**
Time (s)	208.233	**174.986**	252.148	208.387
Size (KB)	1759	**495**	2147	604

Table 1 shows the experimental results. It can be observed that the inference time and model sizes are significantly reduced by the IA-Net method. The acceleration rate of the inference time is 1.19x (208.233 to 174.986) and 1.21x (252.148 to 208.387) for the FCN-10 and FCN-12 models, respectively. Meanwhile, for both FCN-10 and FCN-12, the model sizes of the IA-Nets are only 28.1% of the original models. Another observation from Table 1 is that the IA-Net outperforms the original model in terms of the PESQ scores under low-SNR conditions (0 and -6 dB SNRs). For example, the PESQ scores are improved by approximately 10% (from 1.379 to 1.514), compared to the original model, under -6 dB SNR for FCN-12. The results suggest that IA-Net not only effectively accelerates the computation effectively but also provides better SE results. We believe that the improvements come from enhanced generalization capability with removing redundant parameters in the original SE models.

Moreover, although FCN-12 IA-Net has two more layers to compute, the inference time is almost the same as that of the original FCN-10. However, FCN-12 IA-Net outperforms the original FCN-10 in terms of the PESQ. This result suggests that, instead of training a model with less number of layers, training a deeper model with compression can yield better performance in terms of computation efficiency and enhanced speech quality.

Figure 6 reports the average PESQ, STOI, and SDI scores

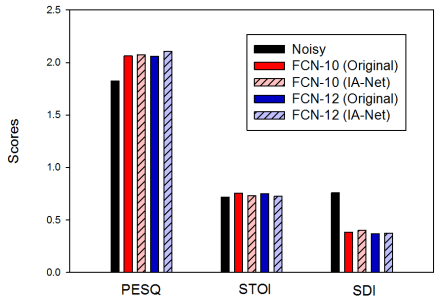

Figure 6: *Average scores of PESQ, STOI and SDI of the SE*

over the three noise types and four SNR levels. From the figure, it can be seen that all of the four models (original/IA-Net FCN-10, and FCN-12) considerably improve the speech quality and intelligibility and reduce the speech distortions in unprocessed noisy speeches. It is also seen that for both FCN-10 and FCN-12, IA-Net achieves performance similar to that of the original model. Finally, Figure 7 (a), (b), (c), and (d) show the speech waveforms of clean, noisy, FCN-12 (Original), and FCN-12 (IA-Net), respectively. From the figure, we can observe that the enhanced waveforms of FCN-12 (Original) and FCN-12 (IA-Net) are very similar, and they both effectively remove the noise components from the noisy waveforms. The results suggest that, in addition to significant acceleration, IA-Net maintains denoising capabilities similar to that of the original model.

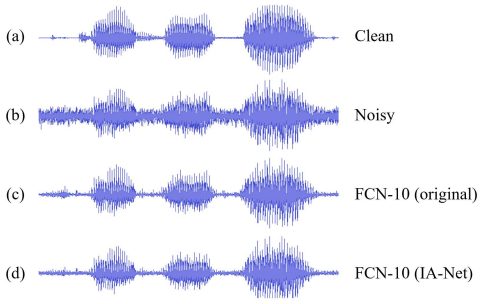

Figure 7: *Waveforms of a TIMIT utterance: (a) clean speech; (b) noisy speech (street noise at 0 dB); (c) speech enhanced by FCN-10 (Original) ; (d) speech enhanced by FCN-10 (IA-Net).*

5. Conclusions

In this work, we proposed a novel IA-Net technique for speech enhancement, as the existing compression and acceleration techniques designed for classification provided unsatisfactory results for estimation or regression tasks. The IA-Net compressed the DNN model by rounding the approximation in training so that the floating-point values could be multiplied using integer-adder circuit during inference. Compared to the original FCN models, IA-Net improved the inference time by 1.19x to 1.21x and compressed the model size by 71.9%. However, with such significant acceleration and compression, the quality and intelligibility scores were even better in terms of the PESQ. For example, FCN-12 (IA-Net) could produce 2.38% (from 2.058 to 2.107) PESQ improvement compared to FCN-12 (Original). The results suggested that, by using the proposed IA-Net technique, we could install the compressed SE system in embedded devices with acceleration, for application in various of computing units.

6. References

[1] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, 2013, pp. 6645–6649.

[2] L. Deng, J. Li, J.-T. Huang, K. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams *et al.*, "Recent advances in deep learning for speech research at microsoft," in *Proc. ICASSP*, 2013, pp. 8604–8608.

[3] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5329–5333, 2018.

[4] Y. Liu, Y. Qian, N. Chen, T. Fu, Y. Zhang, and K. Yu, "Deep feature for text-dependent speaker verification," *Speech Communication*, vol. 73, pp. 1–13, 2015.

[5] H. M. Fayek, M. Lech, and L. Cavedon, "Evaluating deep learning architectures for speech emotion recognition," *Neural networks : the official journal of the International Neural Network Society*, vol. 92, pp. 60–68, 2017.

[6] D. Wang and J. Chen, "Supervised speech separation based on deep learning: An overview," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 10, pp. 1702–1726, Oct 2018.

[7] X. Lu, Y. Tsao, S. Matsuda, and C. Hori, "Speech enhancement based on deep denoising autoencoder," in *Proc. Interspeech*, 2013, pp. 436–440.

[8] Y. Xu, J. Du, L. R. Dai, and C. H. Lee, "A regression approach to speech enhancement based on deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 1, pp. 7–19, Jan 2015.

[9] S.-W. Fu, Y. Tsao, and X. Lu, "SNR-aware convolutional neural network modeling for speech enhancement," in *Proc. Interspeech*, 2016, pp. 3768–3772.

[10] H. Erdogan, S. W. J. R. Hershey, and J. L. Roux, "Phase-sensitive and recognition-boosted speech separation using deep recurrent neural networks," in *Proc. ICASSP*, 2015, pp. 708–712.

[11] S. Pascual, A. Bonafonte, and J. Serr, "Segan: Speech enhancement generative adversarial network," in *Proc. Interspeech*, 2017, pp. 3642–3646.

[12] L. M. Sun, J. Du, L.-R. Dai, and C.-H. Lee, "Multiple-target deep learning for lstm-rnn based speech enhancement," *2017 Hands-free Speech Communications and Microphone Arrays (HSCMA)*, pp. 136–140, 2017.

[13] S. Wang, K. Li, Z. Huang, S. M. Siniscalchi, and C.-H. Lee, "A transfer learning and progressive stacking approach to reducing deep model sizes with an application to speech enhancement," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5575–5579, 2017.

[14] J. Li, L. Deng, Y. Gong, and R. Häb-Umbach, "An overview of noise-robust automatic speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, pp. 745–777, 2014.

[15] M. Kolbk, Z.-H. Tan, and J. Jensen, "Speech intelligibility potential of general and specialized deep neural network based speech enhancement systems," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 1, pp. 153–167, Nov 2017.

[16] Z. Chen, S. Watanabe, H. Erdogan, and J. R. Hershey, "Speech enhancement and recognition using multi-task learning of long short-term memory recurrent neural networks," in *Proc. Interspeech*, 2015, pp. 1–5.

[17] V. Sze, Y. Chen, T. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec 2017.

[18] *Intel Architecture Instruction Set Extensions Programming Reference*, Intel, Santa Clara, CA, USA, Apr. 2017.

[19] D. Keitel-Schulz and N. Wehn, "Embedded dram development: Technology, physical design, and application issues," *IEEE Design Test of Computers*, vol. 18, no. 3, pp. 7–15, May 2001.

[20] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, pp. 380–392.

[21] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[22] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, "Incremental network quantization: Towards lossless cnns with low-precision weights," in *Proc. ICLR*, 2017.

[23] P. H. Hung, C. H. Lee, S. W. Yang, V. S. Somayazulu, Y. K. Chen, and S. Y. Chien, "Bridge deep learning to the physical world: An efficient method to quantize network," in *Proc. SiPS*, 2015, pp. 1–6.

[24] R. Prabhavalkar, O. Alsharif, A. Bruguier, and L. McGraw, "On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition," in *Proc. ICASSP*, 2016, pp. 5970–5974.

[25] Y. Wang, J. Li, and Y. Gong, "Small-footprint high-performance deep neural network-based speech recognition using split-vq," in *Proc. ICASSP*, 2015, pp. 4984–4988.

[26] J. H. Ko, J. Fromm, M. Philipose, I. Tashev, and S. Zarar, "Precision scaling of neural networks for efficient audio processing," *arXiv preprint arXiv:1712.01340*, 2017.

[27] H. Sun and S. Li, "An optimization method for speech enhancement based on deep neural network," in *IOP Conference Series: Earth and Environmental Science*, vol. 69, no. 1. IOP Publishing, 2017, p. 012139.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[30] I. of Electrical and E. Engineers, "Ieee standard for binary floating-point arithmetic," *ANSI/IEEE Std 754-1985*, 1985.

[31] S. Fu, T. Wang, Y. Tsao, X. Lu, and H. Kawai, "End-to-end waveform utterance enhancement for direct evaluation metrics optimization by fully convolutional neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 9, pp. 1570–1584, 2018.

[32] Y. Z. Hsu, Y.-C. Lin, S.-W. Fu, Y. Tsao, and T.-W. Kuo, "A study on speech enhancement using exponent-only floating point quantized neural network (eofp-qnn)," *CoRR*, vol. abs/1808.06474, 2018.

[33] J. S. Garofolo, L. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, "Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1," *NASA STI/Recon technical report n*, vol. 93, 1993.

[34] I.-T. Recommendation, "Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs," *Rec. ITU-T P. 862*, Jan 2001.

[35] C. H. Taal, R. C. Hendriks, R. Heusdens, and J. Jensen, "An algorithm for intelligibility prediction of time–frequency weighted noisy speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 7, pp. 2125–2136, Sept 2011.

[36] J. Chen, J. Benesty, Y. Huang, and S. Doclo, "New insights into the noise reduction wiener filter," *IEEE Transactions on audio, speech, and language processing*, vol. 14, no. 4, pp. 1218–1234, 2006.