Model Compression on Acoustic Event Detection with Quantized Distillation

Bowen Shi[1], Ming Sun[2], Chieh-Chi Kao[2], Viktor Rozgic[2], Spyros Matsoukas[2], Chao Wang[2]

[1]Toyota Technological Institute at Chicago
[2]Amazon

bshi@ttic.edu, {mingsun,chiehchi,rozgicv,matsouka,wngcha}@amazon.com

Abstract

Deep convolutional neural networks (CNNs) are effective modeling architectures for acoustic event detection (AED). Those models can be computational demanding, and it is difficult to deploy CNN-based AED models for resource-constraint applications. In this paper we present a compression technique combining quantization and knowledge distillation, which we refer to quantized distillation, to train a computation-efficient long short-term memory (LSTM) model for AED with guidance of large CNN models. Our experimental results show the proposed compression technique can reduce model size to $1/8$ while further decreasing error rate of original model by more than 15%.

Index Terms: acoustic event detection, model compression, quantization, knowledge distillation

1. Introduction

Acoustic event detection (AED), the task of detecting the occurrence of certain events based on audio streams, can be widely applied in many scenarios. In surveillance systems, audio is used either independently or in conjunction with visual modality for scene analysis. For example, [1] applies AED model to detect hazardous events and perilous road situations. AED also plays an essential role in enabling widespread voice-based (Amazon Alexa, Apple Siri, and Google Assistant etc) to better understand surrounding environments, beyond only listening to human speech. It can also enhance human-computer interaction by providing context-related information [2].

Most works on AED only focus on lowering detection error rate while studies on reducing inference latency are much fewer. Models with high accuracies are often of deep architectures composed of multiple convolutional and recurrent layers [3, 4, 5, 6], thus being computation exhaustive. This raises a concern for the use of AED models in resource-constraint scenarios where computation resource and memory space are limited (e.g. mobile devices).

In this paper, we look into model compression for AED. Our objective is to reduce model size and speed up computation while preserve accuracy. Our contributions include the following two folds: (1). We study the model compression problem in the context of AED, and propose an effective compression technique combining knowledge distillation and quantization. With 4-bit quantization scheme, the original LSTM model size can be compressed to $1/8$. Meanwhile, error rates of original model can be futher reduced by more than 15% through distillation from an accurate deep convolutional model. Throughout this paper, quantization refers to quantization during training unless otherwise specified. (2). We verify the effectiveness of proposed compression technique is generic with different model sizes.

2. Related work

Neural network compression has been well explored in broad context. Knowledge distillation [7] is a commonly used technique for model compression, which consists of training a compact student network with distilled knowledge from a large teacher network. Knowledge distillation has been widely applied in automatic speech recognition (ASR) [8, 9, 10] and visual object detection [11]. Network quantization refers to compressing the original network by reducing number of bits required to represent its weights. Quantization methods have been studied from the perspective of different model architectures [12, 13] as well the specific applications (e.g. ASR [14], machine translation [15]). Low rank factorization [16, 17] is another commonly used compression approach, which estimates informative parameters by using matrix decomposition. Individual compression techniques can also be combined to achieve larger compression rate. For instance, [10] evaluates the combination of low-rank factorization, knowledge distillation and network pruning in ASR. Specifically in AED, [18] investigates compression of CNNs and their method is on simplification of architectures by introducing bottleneck layers and global pooling. [19] combines quantization and low-rank matrix factorization technique to compress multi-layer recurrent neural network.

This paper focuses on applying knowledge distillation and quantization to AED models. Similarly, [15] studies the grouping effect of above two techniques in the context of image classification and machine translation. However there also exists clear distinctions between [15] and ours in terms of methodology. The quantization scheme studied in [15] aims at reducing storage size of model, thus only model weights are quantized there. In this paper, we look into quantizing both parameters and inputs which can speed up inference by enabling efficient low-bit computations in addition to model downsizing. On aspect of knowledge distillation, we study the distillation from teacher CNN to student LSTM while teacher and student models used in [15] are of same architecture with the only difference on size (e.g. number of layers, number of units).

3. Methods

We start by formulating the multi-class acoustic event detection problem. Given an audio signal I (e.g. log mel-filter bank energies (LFBEs)), the task is to train a model $\mathbf{f}$ to predict a multi-hot vector $\mathbf{y} \in \{0,1\}^C$, with C being the size of event set $\mathcal{E}$, and y_c being a binary indicator whether event c is present in I. Note the prediction $\mathbf{f}(I)$ is not a distribution over event set $\mathcal{E}$ since multiple events can occur simultaneously in I. We denote $\mathcal{D}_L = \{(I, \mathbf{y})\}$ as the labeled dataset. Model $\mathbf{f}$ is trained using cross-entropy loss (see equation 4.1), where w_c is the penalty of mis-classification for positive data of class c. w_c should be tuned to balance losses computed from positive and negative instances.

$$L = - \sum_{(I,\mathbf{y}) \in \mathcal{D}_L} \sum_{c=1}^{C} \{ w_c y_c \log f_c(I) + (1-y_c) \log(1-f_c(I)) \} \tag{1}$$

Knowledge distillation Classic knowledge distillation [7] requires training a teacher model first, which is often a large network. A student model is trained with knowledge distillation loss which utilizes the logits outputs from the teacher model as soft-targets. In our multi-class setting, loss defined in equation 2 is used to train the student model m^s.

$$\begin{aligned}
L_{kd} &= \sum_{(I,\mathbf{y}) \in \mathcal{D}_L} \{ \alpha T^2 l(I, \mathbf{y}^t(I;T)) + (1-\alpha) l(I, \mathbf{y}) \} \\
l(I, \mathbf{y}') &= \sum_{c=1}^{C} \{ w_c y'_c \log m_c^s(I) + (1-y'_c) \log(1-m_c^s(I)) \} \\
\mathbf{y}^t(I;T) &= \frac{1}{1+\exp(-\frac{\mathbf{m^t(I)}}{T})}
\end{aligned} \tag{2}$$

where $\mathbf{m}^t$ and $\mathbf{m}^s$ are teacher and student model which takes in acoustic signal $\mathbf{I}$ and output logits in different event categories, $\mathbf{y}'$ denotes probability of either ground-truth ($\mathbf{y}$) or that given by teacher model ($\mathbf{y}^t(I;T)$), T and α are hyperparameters controlling the softness of teacher logits $\mathbf{z}$, and relative weight of distillation loss, respectively.

Quantization Quantization refers to representing floating-point values with n-bit integers ($n < 32$). A common quantization process consists the following steps: (1). scaling: which normalizes vectors of arbitrary range to values in $[0, 1]$, (2). quantizing: which rounds scaled values to quantized values $[0, 1]$, (3). recovering: which scales quantized values in $[0, 1]$ back to original range. The above process is formulated as equation 3.

$$\begin{aligned}
\hat{\mathbf{V}} &= \frac{\mathbf{V} - \beta}{\alpha} \\
\hat{Q}_n(\hat{\mathbf{V}}) &= \frac{[\hat{\mathbf{V}}(2^n-1)]}{2^n-1} \\
Q_n(\mathbf{V}) &= \alpha \hat{Q}_n(\hat{\mathbf{V}}) + \beta \\
\alpha &= \max_i V_i - \min_i V_i, \ \beta = \min_i V_i
\end{aligned} \tag{3}$$

As the quantization function ($\hat{Q}_n$ in equation 3) is discrete, its gradient is almost zero everywhere. To solve this problem, we apply straight-through estimator [20] to approximate the gradient computation regarding $\hat{\mathbf{V}}$. The forward and backward pass are given as equation 4.

$$\begin{aligned}
\textbf{forward}: & \hat{\mathbf{V}}^q = \hat{Q}_n(\hat{\mathbf{V}}) \\
\textbf{backward}: & \frac{\partial l}{\partial \hat{\mathbf{V}}} = \frac{\partial l}{\partial \hat{\mathbf{V}}^q}
\end{aligned} \tag{4}$$

We combine quantization and knowledge distillation for compressing AED models in a way that a quantized student model is trained with knowledge distillation loss (equation 2) using a **full-precision** teacher model. In the scope of this paper, the student model is a one-layer LSTM, and the teacher model is of convolutional architectures with many more layers. Details regarding the architecture of teacher model are discussed in the following experimental section. The structure of LSTM is shown as equation 5, where x_t, h_t, C_t are input, hidden state, cell state of LSTM at timestep t. $\{W_f, W_i, W_c, W_o\}$, $\{b_f, b_i, b_c, b_o\}$ are weight and bias parameters to be trained.

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{C_t} &= \tanh(W_c \cdot [h_{t-1}, x_t] + b_i) \\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t * \tanh(C_t)
\end{aligned} \tag{5}$$

Operations in LSTM include matrix multiplication, elementwise multiplication and activation computation. Quantization of those operators follow the equation in 6. The value of bias involved in a linear transform is not quantized (see equation 6) due to its negligible number of parameters compared to that of weight matrices

$$\begin{aligned}
&\mathbf{y} = \mathbf{W}\mathbf{x} + \mathbf{b} \rightarrow \mathbf{y} = Q_n(\mathbf{W}) Q_n(\mathbf{x}) + \mathbf{b} \\
&\mathbf{y} = \mathbf{x}_1 * \mathbf{x}_2 \rightarrow \mathbf{y} = Q_n(\mathbf{x}_1) * Q_n(\mathbf{x}_2) \\
&\mathbf{y} = F(\mathbf{x}) \rightarrow \mathbf{y} = Q_n(F(\mathbf{x})) \\
&F: \sigma, \tanh
\end{aligned} \tag{6}$$

We quantize every operations in LSTM with its corresponding quantization operators in equation 6. Note we quantize both input and model parameters of one operator. Though quantization of input does not have any impact on the storage size of the model, computation is accelerated as more values fit into one operation in parallel setting. In practice the overhead of value quantization (equation 3) is negligible. The specific acceleration ratio depends on the hardware and will not be covered in this paper. We empirically find quantizing C_t to low bits will lead to the divergence of model training. This may be related to its design of potential unbounded value. Therefore we quantize it with higher precision than other parameters. Similar practice is adopted in previous work on quantization of recurrent neural networks [13].

4. Experiments

4.1. Experimental Setting

Data The dataset we use is a subset from Audioset [21], which contains a large amount of 10-second audio clips. In particular, we select dog sound, baby crying and gunshots as the target events. These three events included in Audioset amount to 13,460, 2,313 and 4,083 respectively, and we use all of them. In addition to these three events, we randomly selected 36,036 examples from all other audio clips in Audioset as negative samples. We randomly split the whole subset for training (70%), validation (10%) and test (20%). Additional efforts has been made to ensure the distribution of events roughly the same across different sets.

Implementation details We compute log mel-filter bank energy (LFBE) features for each audio clip. It is calculated with window size of 25 ms and hop size of 10 ms. The number of mel coefficients is 64, which gives us log-mel spectrogram feature of size 998×64 for each audio clip. Features are further normalized via global cepstral mean and variance normalization (CMVN).

We use DenseNet [22] with 63 layers as the teacher model. The DenseNet we use contains 4 dense blocks with respectively 3, 6, 12 and 8 dense layers, where each layer is composed of batch normalization, ReLU, 1×1 convolution, batch normalization, ReLU and 3×3 convolution. We apply average pooling over all timesteps before feeding into a fully connected layer for classification. We experiment with different CNN architectures including ResNet [23] and DenseNet with different number of layers. The DenseNet-63 gives us the best performance on validation set and we use it as the backbone teacher model.

As mentioned in previous section, one-layer LSTM with 256 hidden units is used as the student model. Only the hidden state of last timestep is used for prediction. In quantization experiments, we will experiment 8-bit and 4-bit quantization with training. We did not quantize the model with less than 4-bit, as it results in significant performance degradation. Both parameters and input are quantized to the target number of bit (8 or 4). The cell state (C_t in equation 5) is quantized to 16-bit across all our experiments as quantizing C_t to low-bits will lead to divergence in training. As a baseline, we compare our quantization scheme with post-mortem (PM) quantization, where the model is quantized after training only.

For all experiments we use Adam optimizer with learning rate of 0.001 and batch size of 64. We tuned penalty on positive loss (w_c in equation) on validation set and found setting it to be the ratio between positive and negative examples of each class in the training set gives overall best results. This practice also prevents us from tuning w_c for every class.

Evaluation Metric We evaluate the performance of models based on area under curve (AUC) and equal error rate (EER) on detection error tradeoff (DET) curve (vertical axis: false negative rate (FNR), horizontal axis: false positive rate (FPR)). Note as both AUC and EER are computed with DET curve thus lower value indicates better performance. We report their values on individual events as well as average over all three events. To evaluate the compression effect, we measure number of parameters, size of parameters (in MB) and number of floating point operations (FLOPs). Specifically in comparison between full-precision and quantized models we use size and number of parameters as metric. This is because floating point operations are turned into low-bit integer operations in our quantization scheme.

4.2. Results

Effect of knowledge distillation Table 1 shows the AUC and EER of LSTM, Dense-63 and LSTM trained with knowledge distillation (LSTM+KD), all in full-precision. Comparison of three models on size and FLOPs are shown in table 2. DenseNet-63 achieves much lower AUC and EER compared to LSTM, which has smaller number of parameters. The better performance of DenseNet-63 can be explained by its deeper architecture. Despite its high accuracy, Dense-63 is much more computation intensive compared to LSTM as can be seen from table 2. The relative low performance of LSTM is due to its simpler architecture as well as the mechanism of recurrence, where gradient vanishing can occur when sequence is very long. Despite of huge difference between these two architectures, the knowledge learned by convolutional model can be partly distilled to the LSTM. As shown in table 1, AUC and EER for three events are much reduced for LSTM trained with knowledge distillation.

Table 1: *AUC and EER of* ***full-precision*** *models (w and w/o knowledge distillation). For both AUC and EER, lower is better.*

Full-precision models		Dense-63	LSTM	LSTM+KD
AUC (%)	Dog	4.32	8.65	6.35
	Baby	2.20	7.91	4.19
	Gunshot	2.07	7.14	3.22
	Avg	2.86	7.90	4.59
EER (%)	Dog	11.11	16.60	14.07
	Baby	6.56	15.58	10.21
	Gunshot	6.41	13.07	9.19
	Avg	8.03	15.08	11.06

Table 2: *Comparison of full-precision teacher and student models on number of parameters, parameter size and FLOPs*

	Dense-63	LSTM	LSTM+KD
# params (M)	2.28	0.33	0.33
Param size (MB)	8.70	1.26	1.26
FLOPs (G)	1.38	0.32	0.32

Effect of quantized distillation We further show results of quantized distillation for LSTM in table 3 . The structure of LSTM (i.e number of hidden units, number of layers) is same as the one used in table 1. The result of 16-bit quantization is not shown as quantization on this high precision does not have much impact on the performance compared to full-precision. Overall, the proposed quantized distillation scheme outperforms PM across different events and bit widths. We notice the performance of 8-bit quantized models is close to its full-precision counterpart even with simple PM quantization scheme. However, the degradation of this naive method with 4-bit is bigger and quantization-aware training leads to minor accuracy loss.

Table 3: *Performance of* ***quantized*** *LSTM trained with knowledge distillation. All models have the same 0.33M parameters with different memory footprints. Full: 1.262MB, 8-bit: 0.321MB, 4-bit: 0.165MB. For both AUC and EER, lower is better. Full: full-precision model, QT: quantization training, PM: post-mortem.*

LSTM+KD (H=256, default)		Full precision	QT 8-bit	PM 8-bit	QT 4-bit	PM 4-bit
AUC (%)	Dog	6.35	7.01	7.12	7.16	9.37
	Baby	4.19	5.00	5.14	5.65	6.95
	Gunshot	3.22	3.97	4.09	4.35	5.76
	Avg	4.59	5.33	5.45	5.72	7.36
EER (%)	Dog	14.07	14.61	14.70	14.72	17.15
	Baby	10.21	10.19	11.02	12.56	13.61
	Gunshot	9.19	10.23	9.68	10.31	11.76
	Avg	11.16	11.68	11.80	12.53	14.17

How quantized distillation works in varying model sizes To verify how the proposed technique works with different sizes of student models, we vary the number of hidden units of LSTM, and show results in table 4. In addition to previous observations, we find that the gap between our quantized training scheme and PM grows as model scales down across all bit width. Specifically we calculate the relative degradation which is defined as the increased AUC/EER divided by the original AUC/EER on full-precision. The difference of relative degradation from full-precision to 8-bit between two quantization scheme increases from 1.9% to 11.0% in AUC and from 1.1% to 8.5% in EER when number of hidden units is reduced from 256 to 64 . Similar finding holds for 4-bit setting as well. To a highly compact model, quantization during training is important to prevent the accuracy drop.

Table 4: *AUC (table a) and EER (table b) of small student models trained with knowledge distillation. H: Number of hidden units of LSTM. H=128: 0.067M parameters, H=64: 0.033M parameters. For both AUC and EER, lower is better.*

(a). AUC(%)		Full precision	QT 8-bit	PM 8-bit	QT 4-bit	PM 4-bit
H=128	Dog	6.90	7.06	7.49	7.91	9.32
	Baby	4.55	5.10	5.07	5.91	7.63
	Gunshot	3.80	4.02	4.10	4.25	8.84
	Avg	5.08	5.39	5.53	6.02	8.60
H=64	Dog	7.68	7.82	8.81	8.71	11.97
	Baby	5.05	5.12	5.87	6.19	7.98
	Gunshot	4.36	4.94	5.08	5.80	11.49
	Avg	5.70	5.96	6.59	6.90	10.48

(b). EER(%)		Full precision	QT 8-bit	PM 8-bit	QT 4-bit	PM 4-bit
H=128	Dog	14.42	14.59	15.19	15.03	17.25
	Baby	10.22	10.89	11.41	12.27	15.58
	Gunshot	9.19	9.12	9.63	10.58	16.87
	Avg	11.28	11.53	12.08	12.63	16.57
H=64	Dog	15.01	15.28	16.78	15.76	20.70
	Baby	11.31	11.30	12.35	11.11	14.29
	Gunshot	9.83	10.13	10.67	12.05	18.85
	Avg	12.05	12.24	13.27	12.97	17.95

5. Conclusion

We study the model compression problem in the context of acoustic event detection. Our compression scheme is a combination of quantization-aware training and knowledge distillation. Experimental results show that the performance of a simple one-layer LSTM can be greatly improved via knowledge distillation with a large scale convolutional neural network as teacher model, which does not increase number of parameters. This model can be further compressed to an order of magnitude smaller with preserved accuracy by quantization training. Analysis on multiple student models further verify the robustness of our proposed method, as well as its advantage over naive quantization scheme.

6. References

[1] N. Almaadeed, M. Asim, S. Al-maadeed, A. Bouridane, and A. Beghdadi, "Automatic detection and classification of audio events for road surveillance applications," vol. 18, p. 1858, 06 2018.

[2] A. Shah, A. Kumar, A. G. Hauptmann, and B. Raj, "A closer look at weak label learning for audio events," *CoRR*, vol. abs/1804.09288, 2018.

[3] S. Hershey, S. Chaudhuri, D. P. W. Ellis, J. F. Gemmeke, A. Jansen, C. Moore, M. Plakal, D. Platt, R. A. Saurous, B. Seybold, M. Slaney, R. Weiss, and K. Wilson, "Cnn architectures for large-scale audio classification," in *ICASSP*, 2017.

[4] Y. Xu, Q. Kong, Q. Huang, W. Wang, and M. D. Plumbley, "Convolutional gated recurrent neural network incorporating spatial features for audio tagging," in *IJCNN*, 2017.

[5] N. Takahashi, M. Gygli, B. Pfister, and L. Gool, "Deep convolutional neural networks and data augmentation for acoustic event detection," *CoRR*, 2016.

[6] E. Cakir and T. Virtanen, "Convolutional recurrent neural networks for rare sound event detection," in *DCASE2017*, pp. 27–31.

[7] G. E. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," in *CoRR*, 2015.

[8] A. Waters and Y. Chebotar, "Distilling knowledge from ensembles of neural networks for speech recognition," in *Interspeech*, 2016.

[9] L. Lu, M. Guo, and S. Renals, "Knowledge distillation for small-footprint highway networks," in *ICASSP*, 2017.

[10] R. Pang, T.N.Sainath, R. Prabhavalkar, S. Gupta, Y. Wu, S. Zhang, and C. Chiu, "Compression of end-to-end models," in *Interspeech*, 2018.

[11] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models with knowledge distillation," in *NIPS*, 2017.

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *Journal of Machine Learning Research*, vol. 18, 2018.

[13] Q. He, H. Wen, S. Zhou, Y. Wu, C. Yao, X. Zhou, and Y. Zou, "Effective quantization methods for recurrent neural networks," arXiv:1611.10176, 2016.

[14] R. Alvarez, R. Prabhavalkar, and A. Bakhtin, "On the efficient representation and execution of deep acoustic models," in *Interspeech*, 2016.

[15] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *ICLR*, 2018.

[16] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *ICASSP*, 2013.

[17] R. Prabhavalkar, O. Alsharif, A. Bruguier, and I. McGraw, "On the compression of recurrent neural networks with an application to lvcsr acoustic modeling for embedded speech recognition," in *ICASSP*, 2016.

[18] Y. Wu and T. Lee, "Reducing model complexity for dnn based large-scale audio classification," in *ICASSP*, 2018.

[19] B. Shi, M. Sun, C.-C. Kao, V. Rozgic, S. Matsoukas, and C. Wang, "Compression of acoustic event detection models with low-rank matrix factorization and quantization training," *NeurIPS workshop on Compact Deep Neural Networks with industrial applications*, 2018.

[20] Y. Bengio, N. Leonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," CoRR, abs/1308.3432, 2013.

[21] J. F. Gemmeke, D. 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.

[22] G. Huang, Z. Liu, L. Maaten, and K. Weinberger, "Densely connected convolutional networks," in *CVPR*, 2017.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.