Adaptive Compression of Supervised and Self-Supervised Models for Green Speech Recognition

Mouaad Oujabour, Leila Ben Letaifa, Jean-françois Dollinger, Jean-Luc Rouas

Adaptive Compression of Supervised and Self-Supervised Models for Green Speech Recognition

Mouaad Oujabour*, Leila Ben Letaifa*, Jean-François Dollinger*, Jean-Luc Rouas†,
*CESI LINEACT UR 7527, Nancy, France
†LaBRI, CNRS UMR 5800 Univ. de Bordeaux, Bordeaux INP Talence, France

***Abstract*—Computational power is crucial for the development and deployment of artificial intelligence capabilities, as the large size of deep learning models often requires significant resources. Compression methods aim to reduce model size making artificial intelligence more sustainable and accessible. Compression techniques are often applied uniformly across model layers, without considering their individual characteristics. In this paper, we introduce a customized approach that optimizes compression for each layer individually. Some layers undergo both pruning and/or quantization, while others are only quantized, with fuzzy logic guiding these decisions. The quantization precision is further adjusted based on the importance of each layer. Our experiments on both supervised and self-supervised models using the librispeech dataset show only a slight decrease in performance, with about 85% memory footprint reduction.**

***Index Terms*—Adaptive compression, Self Supervised models, speech recognition, pruning, quantization, fuzzy logic**

I. Introduction

Artificial Intelligence (AI) achieves remarkable success in a wide array of applications. This success stems from the development of deeper and wider Deep Neural Network (DNN) architectures, which enhance the model's ability to learn intricate patterns for specific tasks. This is especially evident in computer vision, in natural language processing and audio processing, including speech recognition. However, the deployment of such large models comes with significant computing and financial costs and contributes to a substantial carbon footprint [1]. This not only challenges the inclusivity of AI but also raises environmental concerns [2]. To address these issues, reducing model size is crucial. Several methods exist for compressing DNNs, including quantization, pruning, knowledge distillation, parameter sharing, and matrix factorization [3]–[9].

Today, most DNN applications rely on supervised learning, which requires labeled data — a process that is often time-consuming and costly. In contrast, human learning begins unsupervised, as infants learn language through observation, and later through supervised tasks like reading and writing. To mimic this process, self-supervised learning (SSL) frameworks have been developed. In speech processing, models like Wav2vec 2.0 [10], HUBERT [11], and WavLM [12] excel with minimal annotated data by pretraining on large unlabeled datasets, followed by fine-tuning on smaller labeled datasets.

Several works in the literature have focused on large model compression, but only a limited number address SSL models [13], [14]. Among these, in [15], the authors applies knowledge distillation (KD) to the Wav2vec acoustic model, achieving a 4.8x compression ratio, though with a WER increase of 3.62. In [16], genetic algorithms are proposed for structured pruning of the Wav2vec2 XLSR53 model, resulting in a slight WER increase of 0.21% with 40% pruning. In [17], the authors employ symmetric linear quantization to reduce the precision of weights and activations of a BERT model to INT8. To our knowledge, quantization has not yet been applied to speech SSL models.

Previous research often applies a uniform compression method across all layers. However, recent studies reveal that weight distributions vary by type and position within the network [5], [18]. For example, layers with many critical weights need higher quantization precision, while layers with mostly low-magnitude weights are more suitable for pruning. We propose a customized approach that selects the optimal compression method for each layer individually. Some layers will undergo both quantization and/or pruning, while others will be only quantized, with fuzzy logic guiding the decision process.

II. Model compression

Focusing on Green AI [1] to minimize computational costs while preserving performance, we prioritize techniques with minimal parameter tuning. We explore two model compression strategies, quantization and pruning, which are not only easily applied to pre-trained models but also ideal for rapid deployment on mobile devices.

A. Quantization

Model quantization reduces the size of the neural networks by using lower precision values of the weight or activation [6]. The two main approaches are Post-Training Quantization (PTQ) and Quantization-Aware Training (QAT). A standard quantization function $Q(x)$ converts a floating-point value x to an integer :

$$Q(x) = \text{Int}\left(\frac{x}{S}\right) - Z \tag{1}$$

where S is a floating-point scaling factor, and Z is an integer zero point, representing the zero value in the quantization

scheme. The Int$(\cdot)$ function rounds the scaled x to the nearest integer. This approach is known as uniform quantization because all x values are scaled by the same factor S, leading to evenly spaced quantized values. Non-uniform quantization, with its variable spacing between quantized values, can more effectively capture signal information, but it is challenging to implement efficiently on standard hardware. As a result, uniform quantization is the preferred method.

Clipping range selection, or calibration, can be done using the signal's minimum and maximum values $\alpha = x_{min}$ and $\beta = x_{max}$ resulting in asymmetric quantization as the range may not be centered. Alternatively, symmetric quantization sets $\alpha = -\beta$ often using the maximum absolute values $-\alpha = \beta = \max(|x_{max}|, |x_{min}|)$. Asymmetric quantization typically narrows the clipping range, which is important for imbalanced weights or activations like those following ReLU. Setting the zero point to $Z = 0$ simplifies symmetric quantization.

B. Pruning

Pruning removes unimportant weights/components, by zeroing values close to zero. Formally, a neural network model can be defined as a function family $f(x, W)$ where x denotes the network architecture and W its parameters. Pruning a neural network involves taking an existing model $f(x, W)$ and generating a new model $f(x, W')$ such that $W' = M \odot W$ where $M \in \{0, 1\}^{|W|}$ is a binary mask to set some parameters to zero and $\odot$ is the elementwise product operator [19].

The pruning techniques include : – Unstructured pruning [20] removes individual weights, creating sparse matrices – Structured pruning removes entire blocks, such as rows, columns, neurons, or attention heads [21]. In this paper, our focus is on unstructured pruning, as it targets the smallest model elements without significant performance loss. Unstructured pruning introduces sparsity, creating irregular memory access patterns but sparse matrix representations [22] or specialized hardware [23] can address this issue. Pruning can be done iteratively between training epochs, applied once after training [18], or integrated during fine-tuning [24]. Pruning can be applied globally, removing a fraction of parameters across the entire model, or locally, targeting a specific percentage of parameters within each layer [19].

III. Adaptive Compression

We propose an adaptive compression method using fuzzy logic [25] to evaluate weight importance in each layer and dynamically select the optimal compression strategy. By analyzing the statistical distribution of weight magnitudes (e.g., minimum, maximum, median, standard deviation), the method defines fuzzy membership functions to classify weights as *low*, *medium*, or *high* importance.

1) Fuzzy Membership Functions: Fuzzy logic allows us to assign degrees of membership to different importance classes low, medium, or high for each weight based on its magnitude. In the adaptive method, we used basically trapezoidal and triangular membership functions to describe the fuzzy sets for *low, medium,* and *high* importance as shown on Fig. 1.

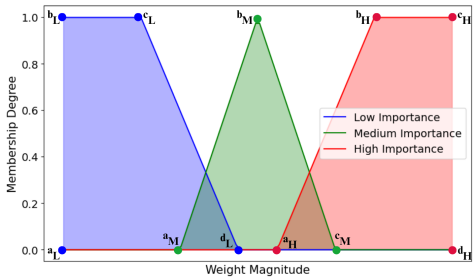

Fig. 1. Fuzzy membership functions for weight importance in the adaptive compression method.

a) Trapezoidal Membership Function: This function is used to classify weights into two categories: *low* or *high* importance. The corresponding membership function, denoted by $\mu_k(x)$, where $k \in \{\text{low}, \text{high}\}$, is given by:

$$\mu_k(x; a_k, b_k, c_k, d_k) = \begin{cases} 0 & \text{if } x \le a_k \\ \frac{x - a_k}{b_k - a_k} & \text{if } a_k < x \le b_k \\ 1 & \text{if } b_k < x \le c_k \\ \frac{d_k - x}{d_k - c_k} & \text{if } c_k < x \le d_k \\ 0 & \text{if } x > d_k \end{cases} \quad (2)$$

b) Triangular Membership Function: For weights classified as *medium importance*, we use a triangular membership function. The triangular membership function $\mu_{\text{medium}}(x; a_{\text{M}}, b_{\text{M}}, c_{\text{M}})$ is defined as:

$$\mu_{\text{medium}}(x; a_{\text{M}}, b_{\text{M}}, c_{\text{M}}) = \begin{cases} 0 & \text{if } x \le a_{\text{M}} \\ \frac{x - a_{\text{M}}}{b_{\text{M}} - a_{\text{M}}} & \text{if } a_{\text{M}} < x \le b_{\text{M}} \\ \frac{c_{\text{M}} - x}{c_{\text{M}} - b_{\text{M}}} & \text{if } b_{\text{M}} < x \le c_{\text{M}} \\ 0 & \text{if } x > c_{\text{M}} \end{cases} \quad (3)$$

2) Membership Degree Calculation: For each weight w in a layer, we calculate its degree of membership in each importance class as *low, medium,* or *high*. Let $x = |w|$, the magnitude of the weight. The membership degrees are $\mu_{\text{low}}(x)$, $\mu_{\text{medium}}(x)$ and $\mu_{\text{high}}(x)$. These degrees describe the extent to which a weight belongs to each importance class.

3) Defuzzification and Decision-Making: Defuzzification converts fuzzy weight classifications into concrete actions. Based on the percentage of weights in each importance category (*low, medium, high*), the method applies the appropriate compression strategy. Layers with a majority of low-importance weights are pruned or quantized with low precision, while those with medium or high-importance weights are quantized with higher precision.

IV. Experiments and Results

We conducted experiments comparing quantization, pruning, and the proposed adaptive method under identical conditions. Each technique was applied post-training in a one-shot manner and evaluated by memory footprint and Word Error Rate (WER). Compression was measured by storage

efficiency for quantization and sparsity for pruning. To ensure fairness, we introduced a unified compression rate for doubly compressed models.

A. Baseline systems

We utilize automatic speech recognition (ASR) models trained with the ESPnet toolkit [26] on the LibriSpeech dataset [27], which comprises approximately 1000 hours of 16kHz English speech recordings. Of these, around 960 hours are dedicated to training, with the remaining hours evenly split between development (dev) and testing (test) sets. The dataset distinguishes between two categories: clean data (test-clean and dev-clean) and other data (test-other and dev-other). Other data refer to recordings that are more challenging due to factors such as background noise, unclear pronunciation, or varied accents. Instead, clean data consist of recordings with clear and high-quality audio. We chose to evaluate the Transformer architecture [28] and some of its variants that are the Conformer [29], the Branchformer [30] and the E-Branchformer [31] because of their high performance in End to End ASR. We also used the Wav2Vec, the Hubert and the WavLM SSL models. These models are referred to as $Transf$, $Conf$, $Branch$, $Ebranch$, $W2V$, Hub and Wlm. Results are reported in TABLE I.

TABLE I
BASELINE SYSTEMS' MODELS: NUMBER OF PARAMETERS (MILLIONS), MEMORY FOOTPRINT (MEGABYTES) AND WORD ERROR RATE (%).

Characteristic	Trans	Conf	Branch	Ebranch	W2V	Hub	Wlm
Parameters	99	93	116	148	432	433	431
Mem.	397	373	596	553	1734	1731	1727
Mem. zip	369	345	433	467	1176	1179	1174
WER							
Test-clean	3.3	2.9	2.4	2.2	2.5	2.0	2.0
Test-other	8.0	7.3	5.3	4.6	6.3	4.2	4.2
Dev-clean	3.0	2.9	2.1	2.0	2.3	1.9	1.9
Dev-other	7.9	7.1	5.2	4.6	6.6	4.1	4.2

Among classical DNNs, E-Branchformer is the most performant but also the largest. For SSL models, all have comparable and significant memory sizes, with Hubert and WavLM showing superior performance than Wav2vec.

B. Quantization

We reduced the model precision from 32 bits floats to 8, 4, and 2 bits integersusing the symetric PTQ dynamic quantization method with Quanto software[1]. TABLE II shows that all models are robust to 8 bits quantization but less so to 4 bits one. The best performance is achieved by the E-Branchformer, Hubert, and Wavlm models. Overall, the error rate remains stable or increases slightly, with a maximum absolute rise of 0.2% for clean data (test and dev) and up to 0.6% for noisy data. Quantizing to 2-bit integers significantly degrades performance on clean data. So, we chose not to evaluate it on the remaining dataset. Regarding memory size, according to TABLE III, it is reduced by more than 3.6 times for 8 bits quantization and by 6.3 times for 4 bits quantization.

[1]https://github.com/huggingface/optimum-quanto/tree/main

TABLE II
WORD ERROR RATE (WER) AFTER QUANTIZATION TO 2, 4, AND 8 BITS.

Models	Data	Initial	Qint8	Qint4	Qint2
Transf	Test_clean	3.3	3.3	3.5	94.0
	Test_other	8.0	8.0	8.6	
	Dev_clean	3.0	3.0	3.2	
	Dev_other	7.9	7.9	8.4	
Conf	Test_clean	2.9	3.0	3.0	33.2
	Test_other	7.3	7.4	7.6	
	Dev_clean	2.9	2.9	3.0	
	Dev_other	7.1	7.3	7.4	
Branch	Test_clean	2.4	2.4	2.4	22.7
	Test_other	5.3	5.3	5.6	
	Dev_clean	2.1	2.2	2.2	
	Dev_other	5.2	5.2	5.4	
E-branch	Test_clean	2.2	2.2	2.2	16.5
	Test_other	4.6	4.6	4.7	
	Dev_clean	2.0	2.0	2.0	
	Dev_other	4.6	4.6	4.7	
Hubert	Test_clean	2.0	2.0	2.0	66.3
	Test_other	4.2	4.2	4.2	
	Dev_clean	1.9	1.9	1.9	
	Dev_other	4.1	4.1	4.2	
Wav2Vec	Test_clean	2.5	2.5	2.6	100.0
	Test_other	6.3	6.3	6.6	
	Dev_clean	2.3	2.3	2.4	
	Dev_other	6.6	6.6	6.9	
WavLm	Test_clean	2.0	2.0	2.1	11.4
	Test_other	4.2	4.2	4.3	
	Dev_clean	1.9	1.9	2.0	
	Dev_other	4.2	4.2	4.2	

TABLE III
MEMORY FOOTPRINT (MEGABYTES) AFTER QUANTIZATION.

Models	Qint8		Qint4		Qint2	
	Mem.	Mem. zip	Mem.	Mem. zip	Mem.	Mem. zip
Transf	108	97	65	59	41	34
Conf	131	119	95	87	75	66
Branch	129	111	79	69	50	41
E-branch	163	138	99	87	63	51
Hubert	512	443	332	279	231	171
Wav2Vec	512	444	332	279	230	171
WavLm	510	442	330	277	229	170

C. Pruning

Local unstructured pruning is applied to each linear layer to all the models using the pruning rates 40% and 60%. According to TABLE IV, not all models are equally robust to pruning. For traditional models, the Branchformer and E-Branchformer architectures appear to be over-parameterized, as they can be pruned by up to 50% without any loss in performance. These models are larger compared to transformers and conformers and include more MLP layers. In the context of SSL, the Hubert and WavLM architectures show considerably greater robustness to pruning than Wav2vec. This enhanced robustness is likely due to their training processes, which rely on masked prediction.

D. Adaptive compression

Each model is compressed layer by layer as follows: For each layer, a membership function categorizes the weights into three groups: L (low), M (medium), and H(high). To define the parameters of these membership functions, let us denote

TABLE IV
WORD ERROR RATE (WER) FOR THE PRUNING RATES: 40% AND 60%

Models	**Pr = 40%**	**Pr = 60%**
Transf	3.6	10.7
Conf	3.1	5.2
Branch	2.4	2.7
E-branch	2.1	2.3
Hubert	2.3	3.1
Wav2Vec	4.1	18.5
WavLm	2.1	2.9

$|w|$ the weight magnitude. The following relationships then hold under tuple notation:

$$(a_M, b_M, c_M) = (median(|w|) - \beta, \quad median(|w|), \quad median(|w|) + \beta) \tag{4}$$

$$(a_L, b_L, c_L, d_L) = (min(|w|), min(|w|), \quad min(|w|) + std(|w|), \quad median(|w|) - \alpha) \tag{5}$$

$$(a_H, b_H, c_H, d_H) = (max(|w|), max(|w|), \quad max(|w|) - std(|w|), \quad median(|w|) + \alpha) \tag{6}$$

with α and β variable parameters.

We apply two experiments: three-class compression and two-class compression.

1) Three-class compression: : If class L has the highest cardinality, the layer is pruned and quantized to 8 bits. If class M has the highest cardinality, the layer is quantized to 4 bits. Otherwise, it is quantized to 8 bits. TABLE V show the results for the following values :
($\alpha 1$, $\beta 1$) = ($\frac{\texttt{std\_magnitude}}{2}$, $\frac{\texttt{std\_magnitude}}{4}$) and
($\alpha 2$, $\beta 2$) = $\frac{\texttt{std\_magnitude}}{1}$, $\frac{\texttt{std\_magnitude}}{2}$)

TABLE V
RESULTS OF THE THREE-CLASS COMPRESSION: $(\alpha 1, \beta 1)$ AND $(\alpha 2, \beta 2)$, %Ł, %H AND %M THE PERCENTAGE OF RESPECTIVE COMPRESSED LAYERS AND Pr THE PRUNING RATE OF THE PRUNED LAYERS.

Model	WER	Mem.z	%L	%M	%H	%Pr
	(α1,β1)					
Transf	3.3	88.06	1.17	85.96	12.87	48
Conf	3.1	118.97	0.68	97.28	2.04	48
Branch	2.4	88.7	26.57	58.94	14.49	59
E branch	2.2	107.82	24.72	63.29	11.99	53
Hubert	2.0	308.46	0.00	91.82	8.18	0
Wav2Vec	2.6	299.82	0.62	93.12	6.25	48
WavLm	2.1	313.66	1.47	89.44	9.09	54
	(α2,β2)					
Transf	3.4	97.25	44.45	32.16	23.39	36
Conf	3.1	120.54	54.42	27.21	18.37	36
Branch	2.4	94.76	54.11	27.54	18.35	49
E branch	2.2	117.8	56.93	25.09	17.98	45
Hubert	2.1	357.45	38.05	34.91	27.04	36
Wav2Vec	4.4	358.52	37.81	32.50	29.69	36
WavLm	2.1	370.2	42.82	31.09	26.09	36

Using $(\alpha 1, \beta 1)$, the WER has either remained the same or increased slightly. The majority of layers in all models were not pruned, except for the Branchformer and E-Branchformer, where a quarter of the layers were pruned without performance loss. With $(\alpha 2, \beta 2)$, we focused on increasing the proportion of weights in the *low* class since 60% pruning rate led to a WER increase of 0.3, 0.1, and 0.9 respectively for the Branchformer, E-Branchformer, and WavLm (see TABLE IV). We find that the model size is typically larger than with 4-bit quantization but smaller than with 8-bit quantization. Branchformer offers the best trade-off, reducing size by 75% with slight increasing the WER.

2) Two-class compression: : According to paragraph IV-B, WER changes little with the 4-bit quantization of all layers but increases significantly with 2-bit quantization. The approach is to use 2 bits for insignificant layers and 4 bits for the remaining layers. The adaptive compression is applied as follows: if class L has the highest cardinality, the layer is quantized to 4 bits; otherwise, it is quantized to 2 bits. TABLE VI show the compression results.

TABLE VI
RESULTS OF THE TWO-CLASS COMPRESSION

Model	WER	Mem .zip	%H	%L
Transf	5.8	54	92.49	7.51
Conf	4.7	84	95.97	4.03
Branch	2.7	54	47.37	52.63
E branch	2.4	74	66.17	33.83
Hubert	3.2	239	76.56	23.44
Wav2Vec	15.3	233	73.60	26.40
WavLm	2.8	237	74.93	25.07

The memory size falls between 4-bit and 2-bit quantization. While the WER is higher than that of the quantization into 4 bits, it remains much better than 2-bit quantization, offering a balance between the two. For Branchformer, 2-bit quantization of all layers yields a WER of 16.5%, 4-bit gives 2.2%, and a mix of 34% of the layers at 2 bits and 66% at 4 bits results in 2.4% WER with a memory reduction to 15%. For WavLM, 2-bit quantization gives a WER of 11.4%, 4-bit achieves 2.0%, and a 25%-75% mix results in 2.8% WER.

V. CONCLUSIONS

Large DNN models are resource-intensive and compression techniques can reduce model size without compromising performance, making Artificial intelligence more sustainable and accessible. Typically, compression methods are applied uniformly across all network layers, neglecting their individual characteristics. This work introduces a refined approach that adapts compression methods to each layer's specific needs, optimizing performance. Our experiments with supervised and self-supervised models show only a slight performance reduction. BranchFormer achieves the best balance, reducing memory size by up to 85% with just a 0.2% increase in WER.

ACKNOWLEDGMENT

This work was supported by the FVLLMONTI project, funded by the European Union's Horizon 2020 program (grant agreement No. 101016776).

REFERENCES

[1] R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," *Commun. ACM*, vol. 63, no. 12, p. 54–63, nov 2020. [Online]. Available: https://doi.org/10.1145/3381831

[2] G. Sastry, L. Heim, H. Belfield, M. Anderljung, M. Brundage, J. Hazell, C. O'Keefe, G. K. Hadfield, R. Ngo, K. Pilz *et al.*, "Computing power and the governance of artificial intelligence," *arXiv preprint arXiv:2402.08797*, 2024.

[3] S. S. Jash Rathod, Nauman Dawalatabad and D. Gowda, "Multi-stage progressive compression of conformer transducer for on-device speech recognition." in *INTERSPEECH*, 2022.

[4] L. B. Letaifa and J.-L. Rouas, "Transformer model compression for end-to-end speech recognition on mobile devices," in *European Signal Processing Conference, EUSIPCO*, 2022.

[5] L. Ben Letaifa and J.-L. Rouas, "Variable scale pruning for transformer model compression in end-to-end speech recognition," *Algorithms. Special Issue "Recent Advances in Machine Learning Algorithms"*, 2023.

[6] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision*. Chapman and Hall/CRC, 2022, pp. 291–326.

[7] M. B. Noach and Y. Goldberg, "Compressing pre-trained language models by matrix decomposition," in *International Joint Conference on Natural Language Processing*, 2020.

[8] D. Bekal, K. Gopalakrishnan, K. Mundnich, S. Ronanki, S. Bodapati, and K. Kirchhoff, "A metric-driven approach to conformer layer pruning for efficient asr inference." INTERSPEECH, 2023.

[9] Y. Wang and J. Li, "Residualtransformer: Residual low-rank learning with weight-sharing for transformer layers," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 11 161–11 165.

[10] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 12 449–12 460.

[11] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, K. Lakhotia, R. Salakhutdinov, and A. Mohamed, "Hubert: Self-supervised speech representation learning by masked prediction of hidden units," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29. IEEE, 2021, pp. 3451–3460.

[12] S. Chen, C. Wang, Z. Chen, Y. Wu, Y. Liang, Y. Q. Fan, M. Z. Chang, S. Liu *et al.*, "Wavlm: Large-scale self-supervised pre-training for full stack speech processing," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30. IEEE, 2022, pp. 346–360.

[13] C.-I. J. Lai, Y. Zhang, A. H. Liu, S. Chang, Y.-L. Liao, Y.-S. Chuang, K. Qian, S. Khurana, D. Cox, and J. Glass, "Parp: Prune, adjust and re-prune for self-supervised speech recognition," *Advances in Neural Information Processing Systems*, vol. 34, pp. 21 256–21 272, 2021.

[14] Y. Peng, K. Kim, F. Wu, P. Sridhar, and S. Watanabe, "Structured pruning of self-supervised pre-trained models for speech recognition and understanding," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[15] Z. Peng, A. Budhkar, I. Tuil, J. Levy, P. Sobhani, R. Cohen, and J. Nassour, "Shrinking bigfoot: Reducing wav2vec 2.0 footprint," *arXiv preprint arXiv:2103.15760*, 2021.

[16] O. Ludwig and T. Claes, "Compressing wav2vec2 for embedded applications," in *2023 IEEE 33rd International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2023, pp. 1–6.

[17] O. Zafrir, G. Boudoukh, P. Izsak, and M. Wasserblat, "Q8bert: Quantized 8bit bert," in *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing-NeurIPS Edition (EMC2-NIPS)*. IEEE, 2019, pp. 36–39.

[18] L. B. Letaifa and J.-L. Rouas, "Fine-grained analysis of the transformer model for efficient pruning," in *International Conference on Machine Learning and Applications ICMLA*, 2022.

[19] D. Blalock, J. J. Gonzalez Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" in *Proceedings of Machine Learning and Systems*, I. Dhillon, D. Papailiopoulos, and V. Sze, Eds., vol. 2, 2020, pp. 129–146.

[20] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *proceedings ICLR*, 2016.

[21] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 1, p. 1–18, 2017.

[22] G. Nazli, J. Ankit, and S. Qian, "Comparative analysis of sparse matrix algorithms for information retrieval," *computer*, vol. 2, 2003.

[23] G. A. Alireza Amirshahi, Joshua Alexander Harrison Klein and D. Atienza, "Tic-sat: Tightly-coupled systolic accelerator for transformers." 28th Asia and South Pacific Design Automation Conference, 2023, pp. 657–663.

[24] M. Gupta and P. Agrawal, "Compression of deep learning models for text: A survey," *Computer Science. ACM Trans. Knowl. Discov. Data*, 2020.

[25] G. Klir and B. Yuan, *Fuzzy sets and fuzzy logic*. Prentice hall New Jersey, 1995, vol. 4.

[26] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "Espnet: End-to-end speech processing toolkit." INTERSPEECH, 2018, pp. 2207–2211.

[27] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: an asr corpus based in public domain audio books," in *ICCASP*, 2015.

[28] L. Dong, S. Xu, and B. Xu, "Speech-transformer: A no-recurrence sequence-to-sequence model for speech recognition." IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP, 2018.

[29] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *INTERSPEECH*, Oct. 2020.

[30] Y. Peng, S. Dalmia, I. Lane, and S. Watanabe, "Branchformer: Parallel mlp-attention architectures to capture local and global context for speech recognition and understanding," in *International Conference on Machine Learning*. PMLR, 2022, pp. 17 627–17 643.

[31] K. Kim, F. Wu, Y. Peng, J. Pan, P. Sridhar, K. J. Han, and S. Watanabe, "E-branchformer: Branchformer with enhanced merging for speech recognition," in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 84–91.