Conformer-Based Speech Recognition On Extreme Edge-Computing Devices

Mingbin Xu[*1], **Alex Jin**[*†1], **Sicheng Wang**[1], **Mu Su**[1], **Tim Ng**[1], **Henry Mason**[1],
Shiyi Han[1], **Zhihong Lei**[1], **Yaqiao Deng**[1], **Zhen Huang**[1], **Mahesh Krishnamoorthy**[1]
[1]Apple
mingbinxu@apple.com, alexgbjin@gmail.com,
{sicheng_wang,mu_su,tim_ng,hmason,shan26,zlei,yaqiao_deng,zhen_huang,maheshk}@apple.com

Abstract

With increasingly more powerful compute capabilities and resources in today's devices, traditionally compute-intensive automatic speech recognition (ASR) has been moving from the cloud to devices to better protect user privacy. However, it is still challenging to implement on-device ASR on resource-constrained devices, such as smartphones, smart wearables, and other small home automation devices. In this paper, we propose a series of model architecture adaptions, neural network graph transformations, and numerical optimizations to fit an advanced Conformer based end-to-end streaming ASR system on resource-constrained devices without accuracy degradation. We achieve over 5.26 times faster than realtime (0.19 RTF) speech recognition on small wearables while minimizing energy consumption and achieving state-of-the-art accuracy. The proposed methods are widely applicable to other transformer-based server-free AI applications. In addition, we provide a complete theory on optimal pre-normalizers that numerically stabilize layer normalization in any L_p-$norm$ using any floating point precision.

1 Introduction

Conformer-based (Gulati et al., 2020) end-to-end (E2E) automatic speech recognition (ASR) (Yao et al., 2021; Zhang et al., 2022) with streaming capabilities (He et al., 2019) have made numerous advances recently. This has paved the way for fully neural speech recognition on resource-constrained mobile devices. These systems also have numerous advantages over conventional hybrid-HMM ASR (Hinton et al., 2012).

First, the training procedure is simplified; the entire system can be defined in a single deep learning framework such as PyTorch or TensorFlow. Second, recent work (e.g. Miao et al., 2019; Sainath et al., 2020; Li et al., 2020; Lei et al., 2023a,b) shows E2E ASR systems can provide better Word-Error-Rate (WER) when compared to conventional hybrid ASR systems. Third, with the continued advancement of deep learning applications, special hardware accelerators such as NVIDIA's Graphics Processing Units (GPU), Google's Tensor Processing Units (TPU), and Apple's Neural Engine (ANE) are becoming increasingly popular. A fully neural ASR system can best utilize such hardware advancements and operate with high throughput while minimizing energy consumption.

In this paper, we present optimizations to enable fully E2E neural network based ASR system under resource-constrained environments, such as smartphones, wearables, and home automation devices. Operating fully offline saves cloud computing resources while providing stronger user privacy (Xu et al., 2023) guarantees, as the user's speech does not need to be transmitted outside of the device.

When targeting resource constrained devices, hardware limitations present many challenges. We describe several multidisciplinary solutions we explored, including memory-aware network transformation, model structural adjustment, and numerical optimizations to address inference stability. We specifically focus on our efforts to take advantage of the inference efficiency provided by specialty hardware accelerators. We derive a theory to numerically stabilize computation of layer normalization on hardware accelerators. This stabilization technique does not require model retraining and is applicable to the computation of any L_p-$norm$.

2 Prior Work

Improving the efficiency of the Transformer architecture has seen substantial interest. Tay et al. (2023) provides a comprehensive survey primarily concentrating on model architecture improvements. Kim et al. (2023) is another noteworthy resource

[*] Equal contribution.
[†] left Apple after paper submission.

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track), pages 131–139
June 16-21, 2024 ©2024 Association for Computational Linguistics

which delves deeper into considerations specific to hardware configurations. Linear Transformer (Katharopoulos et al., 2020) is a key technique, mitigating the computationally expensive softmax function (Bridle, 1989) within the attention mechanism. Softmax is also susceptible to numeric overflow problems when computing with limited numerical range. Hoffer et al. (2018); Zhang and Sennrich (2019) discuss alternative normalization methods other than Batchnorm (Ioffe and Szegedy, 2015) and Layernorm (Ba et al., 2016) to improve computational efficiency and numerical stability in low precision environments. Principles for optimizing transformers have been described in Apple (2022) which target Apple hardware, but are generally applicable for similar devices. Within the domain of speech recognition, Squeezeformer (Kim et al., 2022) stands as a seminal work focusing on efficiency optimization, particularly with respect to the Conformer architecture. The paper uses depthwise separable convolution subsampling to substantially save computation which is central to MobileNet (Howard et al., 2017). It's worth mentioning that the majority of prior work focuses on improving training efficiency by making modifications to the existing model architecture. As a result, these changes require model retraining to achieve efficiency improvements. In contrast, our research primarily concentrates on post-training, inference-only processes while avoiding model retraining whenever possible.

3 Backbone Model

Our backbone model is built upon the Conformer neural architecture (Gulati et al., 2020) as shared acoustic encoder while connectionist temporal classification (Graves et al., 2006) (CTC) and Attention-based Encoder Decoder (AED) (Chan et al., 2016) as dual decoders trained with multitask learning mechanism (Caruana, 1997).

Similar to prior work (e.g. Gulati et al., 2020), we stack transformer (Vaswani et al., 2017) layers and convolution (LeCun et al., 1998) layers alternatively to convert speech frames into high-level representation. We use a relative sinusoidal positional encoding (Dai et al., 2019) into transformer layers. Since our goal is to stream ASR on edge devices, we adopt the chunk-based attention strategy to better balance accuracy and dependency of future audio frames (Yao et al., 2021; Zhang et al., 2022).

4 Proposed Optimizations

4.1 Depthwise Separable Convolution

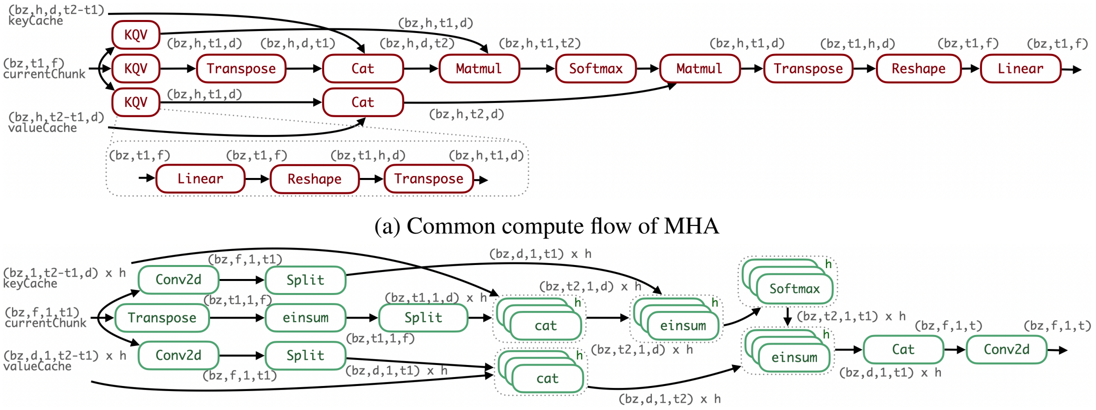
In the original Conformer encoder design (Gulati et al., 2020), the subsampling module at the beginning of the architecture is implemented using two vanilla convolution layers. Our profiling shows that vanilla convolution subsampling accounts for 32.8% of the overall computation and becomes expensive on resource-constrained devices. To alleviate this bottleneck, we used the idea of depthwise separable convolution (Howard et al., 2017; Chollet, 2017) as a drop-in replacement and reduced this computational bottleneck to 4.0% whilst maintaining the WER (Kim et al., 2022), making it particularly well-suited for inference tasks on mobile devices.

While most of the research emphasizes depthwise separable convolution's (DWS) computational efficiency and small memory footprint, its effect on reducing dynamic range of the outputs needs more study. The possible reason could be that DWS reduces the number of multiply-accumulate operations needed for the convolution filters, hence the chance of bigger values. Low numeric range is of great importance for model deployment on edge devices equipped with hardware accelerators. Those hardware often operate in low precision (e.g.fp16) to ease the burden of storage and memory and are exposed to overflow.

4.2 Memory-aware Graph Execution

In Apple's white paper (Apple, 2022) on deploying transformers on the Apple Neural Engine (ANE), *four principles* are elaborated for optimizing transformers on the ANE:

- *Principle 1: Picking the Right Data Format*
  - The (B, C, 1, S) {Batch, Channel, 1, Sequence} data format is chosen for tensor representation to align with the ANE's 4D and channels-first architecture.
- *Principle 2: Chunking Large Intermediate Tensors*
  - Utilize split and concatenation operations to divide tensor into smaller chunks and increase L2 cache residency.
- *Principle 3: Minimizing Memory Copies*
  - Minimize the number of memory operations on tensors such as reshape and transpose.
  - Represent batch matrix multiplication operations using Einstein summation layers.



(a) Common compute flow of MHA

(b) ANE-optimized compute flow of MHA

Figure 1: bz, h and f refers to batch size, number of attention heads and feature dimension respectively, whereas $d = f/h$. Firstly, we transposed the input and output of Conformer CTC, expanding the input tensor to the desired shape of $(B, C, 1, S)$. This transformation allowed us to execute most layers on the hardware accelerator as per *Principle 1*. Additionally, we extensively employed split and concatenation operations to enhance L2 cache residency (*Principle 2*). To address the issue of undesired memory copies resulting from batched matrix multiplication layers, we replaced them with Einstein summation operations (*Principle 3*).

- *Principle 4: Handling Bandwidth-Boundness*
  - We should carefully benchmark the model performance with various batch sizes and sequence lengths and make an informed decision about the cost of memory fetches when we become bandwidth-bound on the ANE.

The key idea behind these 4 principles is being aware of high cost invoked by memory copies between CPU and our hardware accelerator. In our implementation, we adhered to the aforementioned principles. We demonstrate how to rewrite multi-head attention (MHA) in Figure 1 as an example.

More importantly, operations not supported by hardware accelerator were positioned at the beginning or end of the network graph, thus minimizing copies in the memory.

4.3 Stability of Layer Normalization

Layer normalization has become the *de facto* normalization method in transformers after *Attention is all you need* (Vaswani et al., 2017). This normalization technique is widely used in the Conformer CTC architecture. On the other hand, modern hardware accelerators for deep learning often exploit lower precision compute paths in order to reduce memory and boost computation throughput. In the Conformer model, we observed that layer normalization and hardware accelerators are often in dissonance with each other. The reason is that skip connections in the Conformer model join values of varying magnitudes to a single tensor and this often leads to numerical *underflows* or *overflows* in low precision compute paths. For example, the maximum value is 65504 in half precision floating point format (IEEE, 2008). As a contrast, the maximum value is $3.4e38$ in single precision floating point format.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}} \qquad (Layernorm). \tag{1}$$

Equation (1) is a common realization of layer normalization with respect to the L_2-$norm$, where μ and σ^2 are the mean and variance of a vector $\mathbf{x} = \{x_i | 1 \leq i \leq n, x_i \in \mathbb{R}\}$. A small ϵ is added at the bottom to avoid division by zero when σ is small. In order to compute the variance, however, we need to sum the squares of each x_i, which often leads to numerical instability in low precision compute paths. To combat this issue, we employ a technique called Mean Absolute Deviation (MAD) normalization as a pre-normalizer. We note that Layernorm is unaffected by global shifts or global re-scaling of the x_i's and will from here on assume $\mu = 0$.

Definition 1. *Given a low precision compute path with a maximum value M, an optimal L_p-norm pre-normalizer for this compute path maps any distribution of values to a bounded region, $[-D, D]$, where D is as large as possible without causing overflows during the computation of the L_p-norm.*

We note that in the above definition, we explicitly set a constraint to make D as large as possible to minimize the effect of underflow while staying below our low precision limit.

Lemma 1. *Let* $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ *be a finite vector of real numbers with* $\sum_{i=1}^{n} x_i = 0$*, and let* $S = \sum_{i=1}^{n} |x_i|$ *be its* L_1*-norm. Let* $p \geq 1$ *be a real number. We have*

$$||\mathbf{x}||_p^p = \sum_{i=1}^{n} |x_i|^p \leq 2^{1-p} S^p$$

and the maximum is attained when $\mathbf{x} = \{-\frac{S}{2}, 0, ..., 0, \frac{S}{2}\}$.

Proof: For the cases where $n = 1$ or $p = 1$, the inequality above trivially holds.

Let's now look at the case where $n \geq 2$ and $p > 1$. Let $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ be any vector of real numbers and let S be its L_1*-norm*. Consider the vector $\mathbf{v} = \{-\frac{S}{2}, 0, ..., 0, \frac{S}{2}\}$, then

$$||\mathbf{v}||_p^p = 2(\frac{S}{2})^p = 2^{1-p} S^p$$

Hence we attain the maximum value of $||\mathbf{x}||_p^p$ when $\mathbf{x} = \mathbf{v}$. We will now show that $\mathbf{v}$ is indeed the maximum.

First we note that since $\sum_{i=1}^{n} x_i = 0$, the sum of all the negative x_i's must be exactly the opposite of the sum of all the positive x_i's. Furthermore, we can partition the x_i's into two sets, P and N, where

$$N := \{x_i | x_i < 0, x_i \in \mathbf{x}\}, \text{and} \sum_{x_i < 0} x_i = -\frac{S}{2}$$

$$P := \{x_i | x_i \geq 0, x_i \in \mathbf{x}\}, \text{and} \sum_{x_i \geq 0} x_i = \frac{S}{2}$$

If we have exactly one non-zero value in both P and N, then our vector must be $\mathbf{v}$. W.L.O.G., assume we have two non-zero values, $x_j \geq x_k > 0$ and $x_j, x_k \in P$.

Claim: $(x_j + x_k)^p > x_j^p + x_k^p$.

Let's consider the L^p*-space* on $\mathbb{R}^2$ with *p-norm* $||\mathbf{u}||_p := (|u_1|^p + |u_2|^p)^{1/p}$. Let $\mathbf{y} = (x_j, 0)$ and $\mathbf{z} = (0, x_k)$. Applying *Minkowski Inequality* gives us $x_j + x_k > (x_j^p + x_k^p)^{1/p}$ and the claim holds.

Following what we have shown above, $||\mathbf{x}||_p^p$ is strictly increasing if we replace x_j and x_k with $x_j* = 0$ and $x_k* = x_j + x_k$. We note that this replacement does not change the mean or the value of S. By symmetry, the same holds for N. We may continue this replacement process until there's only one non-zero value left in both N and P, and since this process monotonically increases $||\mathbf{x}||_p^p$, we conclude that $||\mathbf{x}||_p^p \leq 2^{1-p} S^p$ and we attain the maximum when $\mathbf{x} = \mathbf{v}$. We will now use the above lemma to prove a useful theorem.

Theorem 1. (Optimal Low Precision Pre-normalizer Theorem). *Let* $\mathbf{x} = \{x_1, x_2, ..., x_n\}$ *be a finite vector of real numbers with* $\sum_{i=1}^{n} x_i = 0$*. Let* M *be the maximum value of our low precision path. Then,*

$$f(\mathbf{x}) = \frac{\mathbf{x}}{\frac{1}{2}(\frac{2}{M})^{1/p} \sum_{i=1}^{n} |x_i|}$$

is an optimal L_p*-norm pre-normalizer for this compute path.*

Proof: From Lemma 1, we know that $||\mathbf{x}||_p^p$ attains the maximum value when $\mathbf{x} = \mathbf{v} = \{-\frac{S}{2}, 0, ..., 0, \frac{S}{2}\}$, where S is the L_1*-norm* of $\mathbf{x}$. Thus it suffices to prove that $f(\mathbf{v})$ satisfies *Definition 1.*

$$||f(\mathbf{v})||_p^p = \sum_{j=1}^{n} |\frac{v_j}{\frac{1}{2}(\frac{2}{M})^{1/p} \sum_{i=1}^{n} |v_i|}|^p \tag{2}$$

$$= \Big(\frac{|-\frac{S}{2}|}{\frac{1}{2}(\frac{2}{M})^{1/p} \sum_{i=1}^{n} |v_i|}\Big)^p + \tag{3}$$

$$\Big(\frac{|\frac{S}{2}|}{\frac{1}{2}(\frac{2}{M})^{1/p} \sum_{i=1}^{n} |v_i|}\Big)^p \tag{4}$$

$$= \Big(\frac{\frac{S}{2}}{\frac{1}{2}(\frac{2}{M})^{1/p} S}\Big)^p + \Big(\frac{\frac{S}{2}}{\frac{1}{2}(\frac{2}{M})^{1/p} S}\Big)^p \tag{5}$$

$$= \frac{M}{2} + \frac{M}{2} = M \tag{6}$$

As shown above, the largest possible value attainable after applying our pre-normalizer is precisely M, the maximum value of our low precision path. □

Corollary 1. $f(\mathbf{x}) = \frac{\mathbf{x}}{\frac{\sqrt{2}}{512} \sum_{i=1}^{n} |x_i|}$ *is an optimal low precision pre-normalizer for* L_2*-norm on the FP16 compute path.*

On a practical note, the pre-normalizer we used for our experiment was the one from Lemmas A1 and A2 (B) with $n = 512$, which gave a slightly lower normalization constant than what Corollary 1 suggests. This worked well in our setup because attaining or even getting close to the maximum value as stated in Lemma 1 requires atypical distribution of values with very few extreme values and everything else being 0. This does not happen in practice, however, with the most common distribution of values observed being Gaussian.

4.4 Scaling of Softmax

Another common constraint on hardware accelerators is their limited support in complex operations. For example, hardware accelerators may choose to omit support for exponential operations (Hu et al., 2018; Li et al., 2018). In such cases, we seek to implement such operations in memory instead, namely using lookup tables (LUT). However, since LUTs are slow and expensive in terms of memory consumption, we would like the tables to be as small as possible. To this end, we introduce a technique called conditional re-scaling for softmax layers:

$$\mathbf{x} = \begin{cases} \frac{4096\mathbf{x}}{\max(\mathbf{x})} & \text{if } max(\mathbf{x}) > 4096 \\ \mathbf{x} & \text{otherwise.} \end{cases}$$

To interpret the above transformation, we first assume that our LUT gives reasonably accurate approximation for x_i's below 4096. Next we take FP16 as an example of our low precision compute paths. We note that for values greater than 4096, gaps between values jump in increments of 4 according to IEEE 754-2008 (IEEE, 2008). Under such scenario, the softmax function behaves similarly to an argmax operation. Since gaps of values between 2048 and 4096 jump in increments of 2, the "argmax behavior" is largely preserved after the re-scaling and exponentiation.

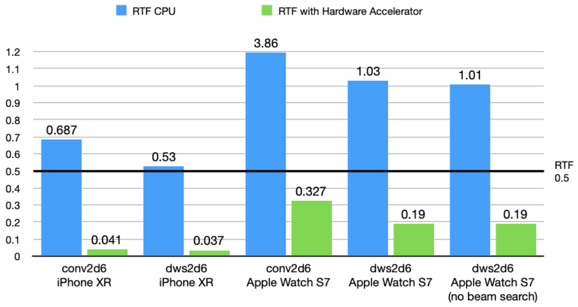


Figure 2: Realtime Factor (RTF) of the original Conformer CTC vs Depthwise Separable Convolution (DWS) architectures. Blue and green bars represent the RTF on CPU and hardware accelerators, respectively. We also added a horizontal line at 0.5 to illustrate required RTF for ASR to process in realtime.

5 Experiments and Results

5.1 Setup

The training corpus contains 17k-hour audio-transcript pairs where the audio is randomly sampled from anonymized virtual assistant queries and human-annotated. We curate 20k queries in the same manner to form an accuracy test set. We use it to examine the accuracy of the optimizations. 200 queries are sampled from the accuracy test set and serve as the performance test set. The audio is decoded lightweightedly with CTC prefix beam search so as to rule out as many computationally intensive components as possible (Graves et al., 2006). The data choice and the training recipe do not play important role in the experiments because the proposed methods focus on hardware acceleration. The experiments are conducted on iPhone XR and Apple Watch Series 7.

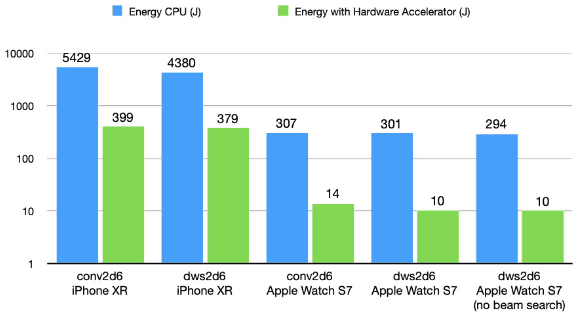


Figure 3: Energy consumption (in joules) for 200 queries of the original Conformer CTC vs Depthwise Separable Convolution (DWS) architectures. Blue and green bars represent the values on CPU and hardware accelerators, respectively. The y-axis is in log scale.

Two models (*conv2d6* and *dws2d6*) are trained with the same hyper-parameters but minor difference in subsampling strategy, summarized in Appendix A. Another two models (*conv2d6x22* and *dws2d6x22*) are trained with the same configuration except that the input to the first Conformer block is scaled by a factor of square root of the IO dimension described in (Vaswani et al., 2017). Additionally we decode greedily on watch to show that **encoder's workload dominates**.

5.2 Performance

High performance is critical in an ASR system in order to process a user's request in real time. To benchmark the performance, we define a notion of Realtime Factor (RTF) as $RTF = processingTime/audioDuration$. It is clear from the definition that lower RTF values are desirable. On real devices, users may often multitask or the operating system may occasionally use computing resources in the background. Therefore an RTF value of at least 0.5 is a reasonable target. As we can see from Figure 2, models running on CPUs do not meet our RTF target of 0.5 and the perfor-

model w/ multiplier	overflow	model w/o multiplier	overflow
conv2d6x22	6.85%	conv2d6	3.26%
dws2d6x22	6.85%	dws2d6	0.25%

Table 1: Layernorm overflow statistics when the proposed transform in Section 4.3 is not applied

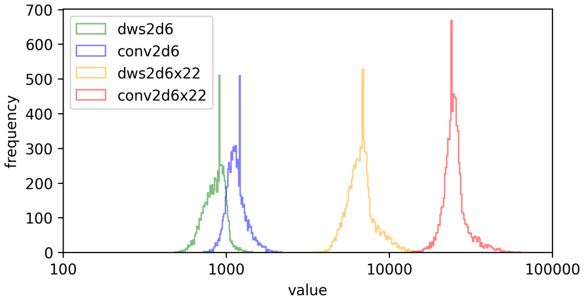


Figure 4: Distribution of the max value between vanilla convolution and DWS in log scale.

mance is substandard on the watch. By leveraging deep learning hardware accelerators, we are able to bring the RTF down by an order of a magnitude for both model variants and achieve the performance goal. On Apple Watch, it is 5.26 times faster.

5.3 Energy

Another important aspect to consider when executing an ASR system on device is the energy consumption. Energy consumption is particularly vital on mobile devices and wearables. We report the energy reduction from using hardware accelerators in Figure 3, where we again see reduction by an order of a magnitude.

5.4 Numeric Stability

In Figure 4 we compare the distribution of maximum value of each chunk's subsampling output during a chunk-based decoding procedure between vanilla convolution and DWS over the performance test set. Empirically the dynamic range of DWS subsampling is a few times smaller than that of the vanilla 2D convolution. When we compare *dws2d6* against *dws2d6x22* or *conv2d6* against *conv2d6x22*, we observe one or two orders of magnitude dynamic range increase introduced by the square root multiplier. Therefore, switching to DWS and removing the multiplier are crucial to keep the subsampling in low-precision-friendly area. Similarly, we plot the distribution of maximum value of each chunk for the Layernorms in Figure 5. Due to residual connections, the enlarged effect of the subsampling output is cascading, 4i.e. large subsampling output increases the chance of overflow in upper layers. In Table 1, we collected overflow statistics of the un-modified Layernorm.

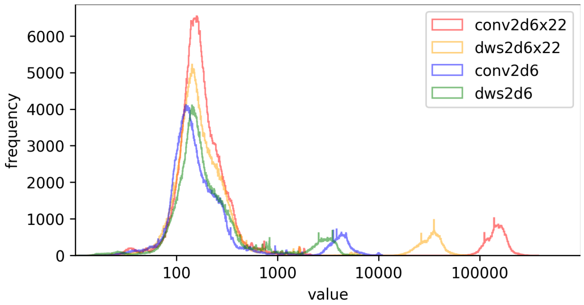


Figure 5: Distribution of Layernorm's input's max value in log scale.

model	WER (FP16)	WER (FP32)
conv2d6	4.45%	4.41%
dws2d6	4.55%	4.56%
conv2d6x22	4.57%	4.47%
dws2d6x22	4.57%	4.49%
conv2d6x22 + modified Softmax	4.76%	4.72%

Table 2: WER comparison of FP16 and FP32

5.5 Quality

We compare the WER of the models on various settings and observed that (1) The difference between FP16 and FP32 is negligible, (2) DWS and vanilla convolution yield almost same accuracy and (3) feature scale-up from the transformer work is not necessary. *conv2dx22* has an almost overflow dynamic range. We apply the softmax modification in Section 4.4 on top of *conv2dx22*. There is a slight WER regression. However, such WER regression does not affect user experience when WER is already low.

6 Conclusions

Through architectural and numerical optimizations, we demonstrate that Conformer CTC ASR models are capable of running on resource-constrained devices such as mobile phones, and wearables. The optimizations preserve recognition accuracy while performing faster than real time and consuming lesser energy. Our theoretical findings of techniques in numerical stabilization is applicable to a wide range of deep learning models and computing tasks.

References

Apple. 2022. Deploying transformers on the apple neural engine. https://machinelearning.apple.com/research/neural-engine-transformers. Accessed: 2023-06-18.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

John Bridle. 1989. Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28:41–75.

William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals. 2016. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016, Shanghai, China, March 20-25, 2016*, pages 4960–4964. IEEE.

François Chollet. 2017. Xception: Deep learning with depthwise separable convolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 1800–1807. IEEE Computer Society.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2978–2988. Association for Computational Linguistics.

Alex Graves, Santiago Fernández, Faustino J. Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, volume 148 of *ACM International Conference Proceeding Series*, pages 369–376. ACM.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, and Ruoming Pang. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 5036–5040. ISCA.

Yanzhang He, Tara N. Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, Qiao Liang, Deepti Bhatia, Yuan Shangguan, Bo Li, Golan Pundak, Khe Chai Sim, Tom Bagby, Shuo-Yiin Chang, Kanishka Rao, and Alexander Gruenstein. 2019. Streaming end-to-end speech recognition for mobile devices. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 6381–6385. IEEE.

Geoffrey Hinton, Li Deng, Dong Yu, George E. Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N. Sainath, and Brian Kingsbury. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. 2018. Norm matters: efficient and accurate normalization schemes in deep networks. *Advances in Neural Information Processing Systems*, 31.

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861.

Ruofei Hu, Binren Tian, Shouyi Yin, and Shaojun Wei. 2018. Efficient hardware architecture of softmax layer in deep neural network. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pages 1–5. IEEE.

IEEE. 2008. Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, pages 1–70.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 5156–5165. PMLR.

Sehoon Kim, Amir Gholami, Albert E. Shaw, Nicholas Lee, Karttikeya Mangalam, Jitendra Malik, Michael W. Mahoney, and Kurt Keutzer. 2022. Squeezeformer: An efficient transformer for automatic speech recognition. In *NeurIPS*.

Sehoon Kim, Coleman Hooper, Thanakul Wattanawong, Minwoo Kang, Ruohan Yan, Hasan Genc, Grace Dinh, Qijing Huang, Kurt Keutzer, Michael W. Mahoney, Yakun Sophia Shao, and Amir Gholami. 2023. Full stack optimization of transformer inference: a survey. *CoRR*, abs/2302.14017.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11):2278–2324.

Zhihong Lei, Ernest Pusateri, Shiyi Han, Leo Liu, Mingbin Xu, Tim Ng, Ruchir Travadi, Youyuan Zhang, Mirko Hannemann, Man-Hung Siu, and Zhen Huang. 2023a. Personalization of ctc-based end-to-end speech recognition using pronunciation-driven subword tokenization. *CoRR*, abs/2310.09988.

Zhihong Lei, Mingbin Xu, Shiyi Han, Leo Liu, Zhen Huang, Tim Ng, Yuanyuan Zhang, Ernest Pusateri, Mirko Hannemann, Yaqiao Deng, and Man-Hung Siu. 2023b. Acoustic model fusion for end-to-end speech recognition. In *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU 2023, Taipei, Taiwan, December 16-20, 2023*, pages 1–7. IEEE.

Jinyu Li, Rui Zhao, Zhong Meng, Yanqing Liu, Wenning Wei, Sarangarajan Parthasarathy, Vadim Mazalov, Zhenghao Wang, Lei He, Sheng Zhao, and Yifan Gong. 2020. Developing RNN-T models surpassing high-performance hybrid models with customization capability. In *Interspeech 2020, 21st Annual Conference of the International Speech Communication Association, Virtual Event, Shanghai, China, 25-29 October 2020*, pages 3590–3594. ISCA.

Zhenmin Li, Henian Li, Xiange Jiang, Bangyi Chen, Yue Zhang, and Gaoming Du. 2018. Efficient fpga implementation of softmax function for dnn applications. In *2018 12th IEEE International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, pages 212–216. IEEE.

Haoran Miao, Gaofeng Cheng, Pengyuan Zhang, Ta Li, and Yonghong Yan. 2019. Online hybrid ctc/attention architecture for end-to-end speech recognition. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 2623–2627. ISCA.

Tara N. Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-Yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, Chung-Cheng Chiu, David Garcia, Alexander Gruenstein, Ke Hu, Anjuli Kannan, Qiao Liang, Ian McGraw, Cal Peyser, Rohit Prabhavalkar, Golan Pundak, David Rybach, Yuan Shangguan, Yash Sheth, Trevor Strohman, Mirkó Visontai, Yonghui Wu, Yu Zhang, and Ding Zhao. 2020. A streaming on-device end-to-end model surpassing server-side conventional model quality and latency. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2020, Barcelona, Spain, May 4-8, 2020*, pages 6059–6063. IEEE.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2023. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6):109:1–109:28.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Mingbin Xu, Congzheng Song, Ye Tian, Neha Agrawal, Filip Granqvist, Rogier C. van Dalen, Xiao Zhang, Arturo Argueta, Shiyi Han, Yaqiao Deng, Leo Liu, Anmol Walia, and Alex Jin. 2023. Training large-vocabulary neural language models by private federated learning for resource-constrained devices. In *IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP 2023, Rhodes Island, Greece, June 4-10, 2023*, pages 1–5. IEEE.

Zhuoyuan Yao, Di Wu, Xiong Wang, Binbin Zhang, Fan Yu, Chao Yang, Zhendong Peng, Xiaoyu Chen, Lei Xie, and Xin Lei. 2021. Wenet: Production oriented streaming and non-streaming end-to-end speech recognition toolkit. In *Interspeech 2021, 22nd Annual Conference of the International Speech Communication Association, Brno, Czechia, 30 August - 3 September 2021*, pages 4054–4058. ISCA.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12360–12371.

Binbin Zhang, Di Wu, Zhendong Peng, Xingchen Song, Zhuoyuan Yao, Hang Lv, Lei Xie, Chao Yang, Fuping Pan, and Jianwei Niu. 2022. Wenet 2.0: More productive end-to-end speech recognition toolkit. In *Interspeech 2022, 23rd Annual Conference of the International Speech Communication Association, Incheon, Korea, 18-22 September 2022*, pages 1661–1665. ISCA.

A Hyper Parameters

conv2d6x22 follows the recipe of (Yao et al., 2021; Zhang et al., 2022), where the subsampling output is multiplied by $\sqrt{512}$ before being fed into the first conformer layer. The multiplier is originated from the transformer work (Vaswani et al., 2017). Its hyper-parameters are summarized in Table 3.

dws2d6x22 is produced by replacing vanilla convolutional subsampling with depthwise separable convolution (DWS). Their difference is compared in Table 4.

conv2d6 is indentical to conv2dx22 except that multiplier is not applied.

dws2d6 is same as dws2dx22 but without applying the multiplier.

hyper-parameters	**values**
#layers (encoder)	12
#layers (decoder)	3
#heads	8
layer IO dimension	512
feedforward dimension	2048

Table 3: Common hyper-parameters in the experiments

model	**channel**	**kernel**	**stride**	**group**
conv2d6	1 → 512	(3,3)	(2,2)	1
	512 → 512	(5,5)	(3,3)	1
dws2d6	1 → 512	(3,3)	(2,2)	1
	512 → 512	(5,5)	(3,3)	512
	512 → 512	(1,1)	(1,1)	1

Table 4: Different subsampling hyper-parameters. Convolution in the same group are applied sequentially.

B Mean Absolute Deviation Normalization on Example Distributions

Definition A1. *A desirable low precision pre-normalizer maps a distribution of values to a bounded region, $[-C, C]$, for some small C.*

Lemma A1. $f(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\frac{1}{n}\sum_{i=1}^{n}|x_i|}$ *is a desirable low precision pre-normalizer for uniform distributions.*

Proof: suppose $X \sim unif[-L, L]$ and $\mathbf{x}$ is a vector of x_i's sampled from X. Consider the limit of the denominator of our normalizer as $n \to \infty$,

$$\lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n}|x_i| = \mathbb{E}[|\mathbf{x}|] = \int_{-L}^{L}\frac{|x|}{2L}dx = \frac{L}{2}.$$

Thus, $f(\mathbf{x}) = \frac{2\mathbf{x}}{L} \sim unif[-2, 2]$.

Lemma A2. $f(\boldsymbol{x}) = \frac{\boldsymbol{x}}{\frac{1}{n}\sum_{i=1}^{n}|x_i|}$ *is a desirable low precision pre-normalizer for normal distributions.*

Proof: suppose $X \sim N(0, \sigma)$ and $\mathbf{x}$ is a vector of x_i's sampled from X. Consider the limit of the denominator of our normalizer and $n \to \infty$,

$$\begin{aligned}\lim_{n\to\infty}\frac{1}{n}\sum_{i=0}^{n}|x_i| &= \mathbb{E}[|\mathbf{x}|]\\ &= \frac{1}{\sigma\sqrt{2\pi}}\int_{-\infty}^{\infty}|x|e^{-\frac{1}{2}(\frac{x}{\sigma})^2}dx\\ &= \frac{2}{\sigma\sqrt{2\pi}}\int_{0}^{\infty}xe^{-\frac{1}{2}(\frac{x}{\sigma})^2}dx\\ &\quad (by\ symmetry)\\ &= \sqrt{\frac{2}{\pi}}\sigma.\end{aligned}$$

Let $x = k\sigma$ for some real k, $f(x) = k\sqrt{\frac{\pi}{2}}$. When $k = \pm 4$, $f(x) = \pm 5.01$. In other words, $f(x) \in [-5.01, 5.01]$ with 99.99% probability.

The two lemmas above illustrate the effect of our MAD normalizer on a couple of common distributions. Empirically, we observed no overflow during our subsequent Layernorm computation after we prepended our pre-normalizer. Let us now look at the theory behind a bit more rigorously.