Breaking Down Power Barriers in On-Device Streaming ASR: Insights and Solutions

Yang Li[*†2] **Yuan Shangguan**[*‡3] **Yuhao Wang**[1] **Liangzhen Lai**[1]
Ernie Chang[1] **Changsheng Zhao**[1] **Yangyang Shi**[1] **Vikas Chandra**[1]

[1]Meta [2]Iowa State University [3]Google

Abstract

Power consumption plays a crucial role in on-device streaming speech recognition, significantly influencing the user experience. This study explores how the configuration of weight parameters in speech recognition models affects their overall energy efficiency. We found that the influence of these parameters on power consumption varies depending on factors such as invocation frequency and memory allocation. Leveraging these insights, we propose design principles that enhance on-device speech recognition models by reducing power consumption with minimal impact on accuracy. Our approach, which adjusts model components based on their specific energy sensitivities, achieves up to 47% lower energy usage while preserving comparable model accuracy and improving real-time performance compared to leading methods.

1 Introduction

Streaming automatic speech recognition (streaming ASR) enables real-time transcription of speech to text with latency typically under 500 milliseconds, supporting applications such as interface navigation, voice commands, real-time communication, and accessibility on mobile and wearable devices. However, high power consumption poses a significant challenge, limiting usability by requiring frequent recharges. Improving the energy efficiency of on-device streaming ASR is therefore essential for enhancing user experience.

We focus on on-device streaming ASR models, particularly the Neural Transducer (Graves, 2012), which combines an Encoder for acoustic modeling, a Predictor for language modeling, and a Joiner to integrate their outputs (see Figure 1). Widely regarded as the standard for on-device streaming ASR (Graves et al., 2013; He et al., 2019; Li et al., 2021), the Neural Transducer excels in balancing computational efficiency and accuracy. We train and evaluate over 180 Neural Transducer models[1], exploring architectures including Emformer (Shi et al., 2021) and Conformer (Gulati et al., 2020) while varying component sizes. This extensive study reveals how the components impact accuracy, real-time factor (RTF),[2] and power consumption.

Our analysis reveals several key findings: (1) Energy usage in streaming ASR models is driven by memory traffic for loading weights, which depends on the invocation frequency of components and their memory hierarchy placement. (2) Invocation frequencies vary widely, with the Joiner being called far more often than the Predictor, and the Predictor more than the Encoder. Despite comprising only 5–9% of the model's size, the Joiner accounts for 48–73% of its power consumption. (3) We identify an exponential relationship between model accuracy and encoder size, suggesting new directions for streaming ASR research.

Building on these insights, we propose a targeted compression strategy to optimize energy efficiency with minimal accuracy loss. This approach evaluates power and accuracy sensitivity for each component, prioritizing compression of components with higher power sensitivity and lower accuracy sensitivity. Specifically, we focus on compressing the Joiner first, followed by the Predictor and Encoder, and aim to store the Joiner's weights in energy-efficient local memory. Experiments on LibriSpeech (Panayotov et al., 2015) and Public Video datasets show our method reduces energy usage by up to 47% and lowers RTF by up to 29%, while maintaining comparable accuracy to state-of-the-art compression strategies. Unlike previous

[*]Co-first authors.

[†]Corresponding author (jerryyangli@gmail.com). Work partially done while employed at Meta and partially while at Iowa State University.

[‡]Work done while employed at Meta.

[1]Traning each model requires 640-960 V100 GPU hours.

[2]RTF is the ratio of inference time to the speech segment duration, with lower values indicating faster processing.

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Industry Track), pages 616–626

approaches, our method effectively leverages the diverse runtime characteristics of ASR components, showcasing its superior efficiency.

This paper makes the following contributions:

- Power consumption analysis: We reveal that ASR component energy usage depends not only on model size but also on invocation frequency and memory placement. This challenges the prevailing belief that larger components inherently consume more energy, emphasizing the role of operational dynamics and memory management.
- Energy-efficient design: We propose design guidelines that reduce energy consumption by up to 47% and RTF by up to 29% while maintaining comparable model accuracy to state-of-the-art methods.
- Accuracy-size relationship: We uncover an exponential relationship between model accuracy and encoder size, showing diminishing gains with larger encoders and advocating for more efficient use of computational and memory resources in on-device streaming ASR.

An earlier version of this paper was released as a preprint on arXiv (Li et al., 2024b).

2 Background

2.1 On-Device Streaming ASR

The Neural Transducer, introduced in (Graves, 2012), is the state-of-the-art solution for on-device streaming speech recognition (Graves et al., 2013; He et al., 2019; Li et al., 2021). It aligns audio and text (Prabhavalkar et al., 2024) by integrating a compact language model and acoustic model within a single framework, making it ideal for resource-constrained devices due to its reduced memory footprint (Shangguan et al., 2019; Venkatesh et al., 2021). With sub-500 millisecond latency, it meets the demands of streaming applications, and it is widely adopted by leading companies for on-device ASR (Li et al., 2024a; Le et al., 2023; Wang et al., 2023; Radfar et al., 2022).

The architecture comprises three components: an Encoder, a Predictor, and a Joiner (Figure 1). The Encoder processes chunks of audio (C_1, ..., C_t), each consisting of frames ($\mathbf{x}_{t,1}$, ..., $\mathbf{x}_{t,n}$) with 80-dimensional log Mel-filterbank features derived from a 25 ms sliding window with a 10 ms step. The Encoder maps frames to embeddings ($\mathbf{enc}_{t,j}$).

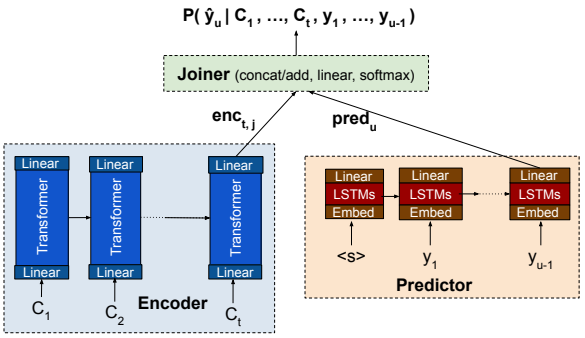


Figure 1: A schematic representation for the Transformer-based Neural Transducer.

The Predictor uses previously predicted tokens (y_1, ..., y_{u-1}) to forecast the embedding of the next token ($\mathbf{pred}_u$). The Joiner combines the embeddings from the Encoder and Predictor, processes them through a feedforward network, and applies a softmax to generate the probability distribution over sentence-piece targets and a "blank" token indicating the end of a frame's transcription.

Recent studies (Shi et al., 2021; Moritz et al., 2020; Dong et al., 2018; Zhang et al., 2020; Yeh et al., 2019; Gulati et al., 2020; Wang et al., 2020; Karita et al., 2019) show a preference for Transformer-based Encoders in Neural Transducers. We implement the Encoder using Emformer (Shi et al., 2021) and Conformer (Gulati et al., 2020), two Transformer variants optimized for streaming. These designs enable chunk-based frame processing, reducing Encoder invocation frequency compared to the Predictor and Joiner, which process frames individually. The Predictor is invoked per meaningful output token, while the Joiner operates for both meaningful tokens and frequent "blank" tokens. This results in a hierarchy of invocation frequency: the Joiner is used most, followed by the Predictor, and then the Encoder.

2.2 Mobile and Wearable Devices

As shown in Figure 2, mobile and wearable devices feature processors such as mobile CPUs, GPUs, and hardware accelerators, all optimized for energy efficiency. For example, a neural network accel-

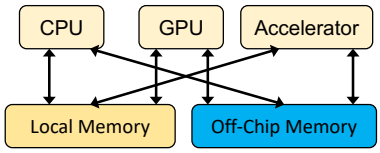
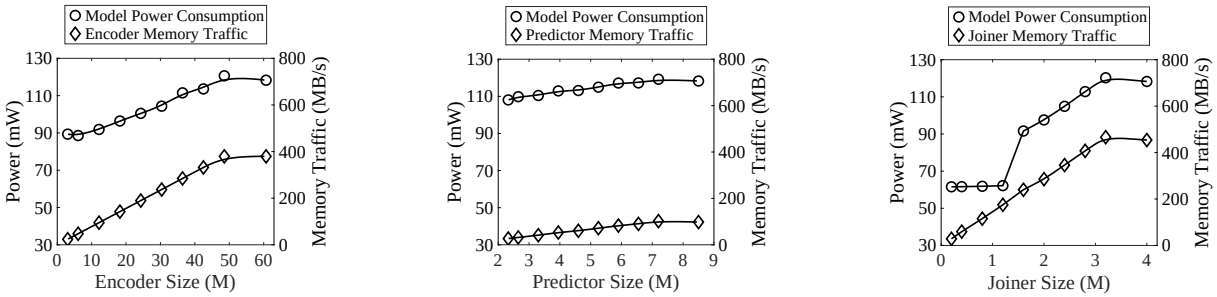


Figure 2: Architecture of mobile and wearable devices.



(a) Compressing Encoder (b) Compressing Predictor (c) Compressing Joiner

Figure 3: Models trained on LibriSpeech: Model power consumption with compressing an individual component (Encoder, Predictor, or Joiner) while keeping the sizes of the other two components constant.

	Encoder	Predictor	Joiner
Size (M)	60.70	8.50	4.00
Compute Power (mW)	0.80	0.03	0.19
Memory Power (mW)	47.78	12.33	57.13
Invocation Frequency (Hz)	6.25	11.53	113.50

Table 1: A typical model trained on LibriSpeech.

erator highlighted by (Lee et al., 2018) achieves 5 GOPS/mW (INT8), consuming just 1 mW for 5 billion INT8 operations per second. These processors interact with two memory types: *local memory* (e.g., SRAM, eDRAM, on-chip DRAM) and *off-chip memory* (e.g., DRAM). Local memory offers faster, energy-efficient access, with 64-byte read/write operations taking 0.5–20 ns and consuming 1.1–1.5 pJ/byte (Li et al., 2019). In contrast, off-chip memory is slower and less efficient, with 64-byte operations taking 50–70 ns and using about 120 pJ/byte (Li et al., 2019). This stark energy efficiency gap makes memory operations a dominant energy drain in on-device streaming ASR.

In our study, we ran streaming ASR models on a Google Pixel-5 smartphone, measuring RTF and workload statistics including the number of operations and component invocations. These workload metrics remain consistent across device platforms. Therefore, the power analysis derived from these metrics applies broadly to other mobile and wearable devices. We modeled ASR power consumption using established methodologies (Li et al., 2024a; Micron, 2006; Li et al., 2017; Lee et al., 2009), leveraging computing and memory power parameters from authoritative literature in the circuits community (Lee et al., 2018; Li et al., 2019). Our setup includes a hardware accelerator, 2 MB of local memory (1.5 MB for weights and 0.5 MB for activations), and 8 GB of off-chip memory, with local memory treated as a scratchpad for flexible allocation. This setup does not represent a specific commercial hardware platform or product; rather, it serves as a general model that is broadly representative of most mobile and wearable devices.

3 Power and Accuracy Analysis of On-Device Streaming ASR

In this section, we use Adam-pruning (Yang et al., 2022), a state-of-the-art weight pruning technique for speech recognition,[3] to adjust the sizes of the Encoder, Predictor, and Joiner in ASR models. This generates ASR models of varying sizes, enabling analysis of their power consumption and accuracy, yielding key insights.

3.1 Power Analysis

Table 1 summarizes the characteristics of a typical on-device streaming ASR model trained on LibriSpeech (Panayotov et al., 2015), including size, component invocation frequency, computing power, and memory power. The data reveals that computing power accounts for less than 1% of total power, with memory power dominating due to frequent weight loading. Although the Encoder holds over 83% of the weights, the Joiner, invoked 18 times more often, generates 1.2 times more memory traffic and consumes more power. This challenges the prevailing belief that larger components consume more energy, highlighting the importance of operational dynamics in energy optimization.

Figure 3 examines power consumption by compressing individual components (Encoder, Predictor, or Joiner) while keeping the others unchanged. The results show that power closely tracks memory traffic, which depends on component size and invocation frequency. Notably, compressing the Joiner

[3]Adam-pruning is detailed in Appendix A.

below 1.2M parameters does not reduce power further, as its weights then fit into energy-efficient local memory, minimizing data-loading energy costs. This underscores the strategic advantage of placing the most energy-intensive components in local memory to optimize energy efficiency.

We also investigate the effects of input stride and chunk size—two key hyperparameters of streaming ASR—on the model's power consumption, revealing some interesting observations. Detailed results are provided in Appendix D.

3.2 Accuracy Analysis

Figures 4 and 5 show the word error rates for compressed models on LibriSpeech's test-clean and test-other sets. Reducing component sizes generally increases word error rates.[4] Among the components, the Predictor is least sensitive to compression, indicating that using a smaller Predictor or omitting it entirely has minimal impact on accuracy. In contrast, the Encoder and Joiner are more sensitive to compression, with encoder size showing an exponential relationship to word error rate:

$$\text{Word Error Rate} = \exp\left(a \cdot \text{encoder\_size} + b\right) + c \quad (1)$$

Fitting this function yielded parameters a, b, and c with adjusted R-squared values of 0.9832 (test-clean) and 0.9854 (test-other), confirming the model's strong fit. Similar trends were observed in other datasets (Appendix C). This exponential relationship suggests diminishing returns with larger encoder sizes, encouraging the community to rethink encoder design in ASR systems.

4 ASR Energy Efficiency Optimization

We aim to minimize the power consumption of streaming ASR models with minimal performance impact by evaluating the **power** and **accuracy sensitivities** of the Encoder, Predictor, and Joiner components. These sensitivities quantify the change in power consumption and performance, respectively, for a unit reduction in component size:

$$\begin{aligned}\text{Power Sensitivity}_{\text{component}} &:= \frac{\Delta\text{Power}}{\Delta\text{Size}_{\text{component}}} \\ \text{Accuracy Sensitivity}_{\text{component}} &:= \frac{\Delta\text{Accuracy}}{\Delta\text{Size}_{\text{component}}}\end{aligned} \quad (2)$$

[4]Variability in Predictor and Joiner compression curves stems from randomness in training and pruning.

Here, component refers to the Encoder, Predictor, or Joiner, and accuracy is inversely related to the word error rate.

The power consumption of on-device streaming ASR is primarily due to loading model weights from memory. Power sensitivity is therefore expressed as:

$$\begin{aligned}&\text{Power Sensitivity}_{\text{component}} \\ &= \frac{\Delta(\text{size} \times \text{invocation frequency} \times \text{memory energy unit})}{\Delta\text{size}} \\ &= \text{invocation frequency} \times \text{memory energy unit}\end{aligned} \quad (3)$$

with the memory energy unit representing the energy required to load a byte from memory, we adopt 1.5pJ/byte for local memory and 120pJ/byte (Li et al., 2019) for off-chip memory. Component size determines whether weights fit in energy-efficient local memory or power-hungry off-chip memory, influencing power sensitivity.

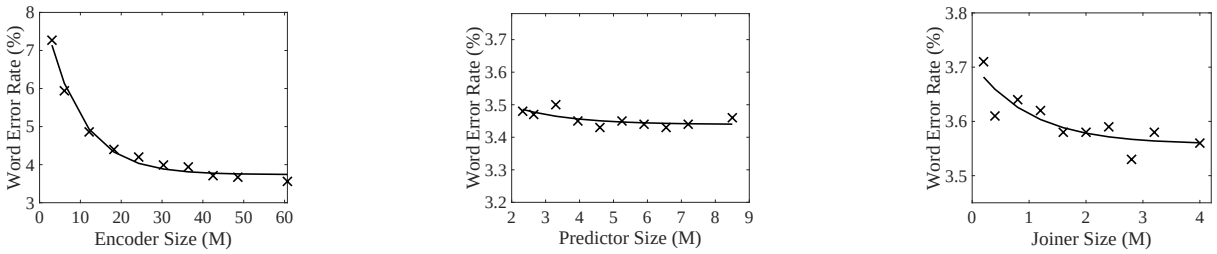
Accuracy sensitivity is calculated by progressively reducing a component's size, observing the effect on model accuracy, and fitting an exponential function to describe the relationship. The derivative of this function quantifies accuracy sensitivity.

Finally, we use the power-to-accuracy sensitivity ratio to prioritize compression decisions:

$$\text{power-to-accuracy sensitivity ratio} = \frac{\text{power sensitivity}}{\text{accuracy sensitivity}} \quad (4)$$

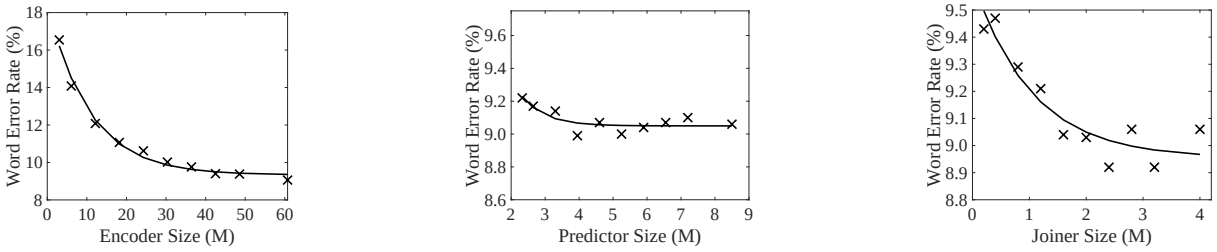
A higher ratio identifies components where compression provides the greatest power savings for minimal accuracy loss, helping determine the optimal compression order for on-device ASR models.

Our compression algorithm starts with a fully uncompressed model and iteratively reduces its size to achieve a user-defined power reduction target (e.g., "reduce power by 60 mW"). At each step, we calculate the power-to-accuracy sensitivity ratio for each component and compress the one with the highest ratio. In Neural Transducer models, the Joiner typically starts with the highest ratio due to its high power sensitivity from frequent invocation. Once its size is reduced enough to fit into energy-efficient local memory, its ratio decreases, and the Predictor becomes the next priority. The Predictor is compressed until it reaches its user-defined minimum size, beyond which further compression would cause significant accuracy loss due to the exponential relationship between accuracy and size. The Encoder is then compressed similarly, followed by additional compression of the Joiner if more power reduction is required.



(a) Compressing Encoder (b) Compressing Predictor (c) Compressing Joiner

Figure 4: Models trained on LibriSpeech: Word error rate on Test-Clean with compressing an individual component (Encoder, Predictor, or Joiner) while keeping the sizes of the other two components constant.



(a) Compressing Encoder (b) Compressing Predictor (c) Compressing Joiner

Figure 5: Models trained on LibriSpeech: Word error rate on Test-Other with compressing an individual component (Encoder, Predictor, or Joiner) while keeping the sizes of the other two components constant.

The compression order is thus: Joiner → Predictor → Encoder → Joiner. Our algorithm determines only the compression order between components, delegating the pruning of weight parameters within a selected component to existing compression methods. This makes our approach compatible with any existing compression algorithm.

5 Experiments

5.1 Datasets and Models

We conduct experiments on two datasets: LibriSpeech and Public Video (details in Appendix B).

LibriSpeech, from audiobooks, contains 960 hours of training data and two evaluation sets: Test-Clean, with easily transcribed recordings, and Test-Other, featuring recordings with accents or poor audio quality. Public Video, an in-house dataset of de-identified audio from publicly available English videos (with consent), includes 148.9K hours of training data and two evaluation sets: Dictation (5.8K hours of open-domain conversations) and Messaging (13.4K hours of audio messages).

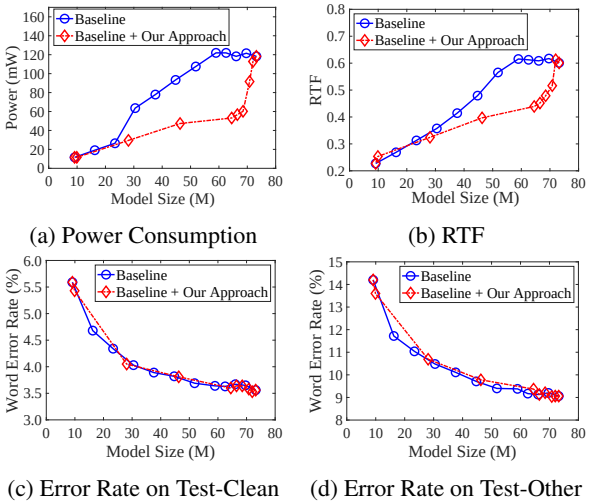
For LibriSpeech, we use Emformer models (Shi et al., 2021) with a 40ms input stride and 160ms chunk size. For Public Video, we use Conformer models (Gulati et al., 2020) with a 60ms input stride and 300ms chunk size.

5.2 Baselines and Evaluation Methodologies

Our method identifies the most critical model component for compression to maximize energy savings. The specific compression technique applied to the identified component is beyond our scope.

We compare two scenarios: a uniform application of a baseline compression technique across the entire model ("baseline") and an enhanced version where the same technique is guided by our approach to strategically prioritize components ("baseline + our approach"). This comparison demonstrates the power savings achieved by our method and highlights the benefits of strategic component prioritization.

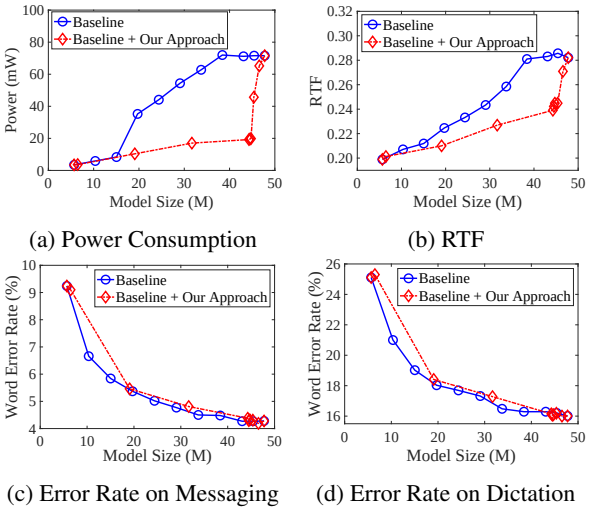
Our experiments use Adam-prune (Yang et al., 2022), the state-of-the-art compression technique for speech recognition models. While we employ the strongest available baseline, the choice or number of baselines is not critical, because our primary focus is on demonstrating consistent power savings achieved by integrating our approach with the baseline, irrespective of the baseline's inherent performance. Stronger baselines yield higher accuracy, and weaker baselines result in lower accuracy; however, the relative power savings for a given model size remain consistent. Therefore, the baseline selection does not affect our objective of highlighting power efficiency enhancement.



(a) Power Consumption (b) RTF

(c) Error Rate on Test-Clean (d) Error Rate on Test-Other

Figure 6: Models trained on LibriSpeech under different sizes and compression schemes.



(a) Power Consumption (b) RTF

(c) Error Rate on Messaging (d) Error Rate on Dictation

Figure 7: Models trained on Public Video under different sizes and compression schemes.

5.3 Results on LibriSpeech

Figure 6 (a) shows the power consumption across different model sizes. Our method achieves significant power savings compared to the baseline for models between 30–76 MB. For models under 30 MB, further compression results in minimal-size components, reducing differences between methods and leading to similar power consumption.

Figure 6 (b) illustrates the Real-Time Factor (RTF). Interestingly, while focusing on energy efficiency, our method improves RTF, indicating faster inference. This is due to prioritizing compression of heavily used components, which more significantly reduces overall inference time.

Figures 6 (c) and (d) show that word error rates remain consistent across model sizes, demonstrating that our method preserves baseline accuracy. Overall, Figures 6 (a)–(d) highlight that our approach reduces energy consumption by up to 47% and RTF by 29% while maintaining accuracy comparable to the baseline.

5.4 Results on Public Video

Figures 7 (a)–(d) show the power consumption, RTF, and accuracy for models of various sizes trained on the Public Video dataset. Our method reduces energy consumption by up to 38% and RTF by 15% while preserving accuracy.

5.5 Discussion

As hardware technology advances, on-chip local memory in mobile and wearable devices continues to expand, allowing an increasing portion of Neural Transducer model weights to be stored locally. This shift enhances energy efficiency by leveraging the high energy efficiency of the on-chip memory. Simultaneously, these advancements may enable the deployment of more complex speech model architectures—previously infeasible for on-device or streaming scenarios due to model size and hardware constraints—as viable on-device streaming solutions. Consequently, we believe that power consumption will remain an important bottleneck in on-device streaming speech recognition. When new architectures incorporate multiple components with varying invocation frequencies, each component exhibits distinct power sensitivities. Our proposed energy efficiency optimization guidelines, which account for differences in power-to-accuracy sensitivity across model components, remain highly relevant in such cases. By adopting these guidelines, power consumption can be significantly reduced, fostering broader development, applicability, and deployment of on-device streaming speech recognition technology.

6 Related Work

This study is the first to analyze the operational dynamics and memory placement of model components to enhance energy efficiency in on-device streaming ASR. The most relevant prior works focus on ASR compression and power optimization.

6.1 On-Device ASR Compression

Ghodsi et al. (2020) demonstrated that removing recurrent layers from the Predictor in Neural Transducer models does not degrade word-error rates,

enabling stateless operation and potential compression. Botros et al. (2021) proposed parameter sharing between the Predictor and Joiner embedding matrices, introducing a weighted-average embedding to capture Predictor token history and reduce footprint. Shangguan et al. (2019) reduced Predictor size by replacing LSTM units with sparsified Simple Recurrent Units (SRU) and adapted Encoders with sparsified CIFG LSTMs. Yang et al. (2022) applied Supernet-based neural architecture search to optimize layer sparsity, balancing accuracy and size. While these works focused on reducing model size or RTF, they did not address power consumption, which is the central goal of our study.

6.2 On-Device ASR Power Optimization

Efforts to optimize Neural Transducer power consumption often involve modifying cell architectures. Li et al. (2024a) introduced folding attention, reducing model size and power consumption by 24% and 23%, respectively, without sacrificing accuracy. Venkatesh et al. (2021) streamlined LSTM cells and designed a deeper, narrower model, reducing off-chip memory access by 4.5x and energy costs by 2x, with minimal accuracy loss. Our work differs by examining the runtime behaviors of Neural Transducer components to guide compression strategies specifically toward energy optimization.

7 Conclusion

Power consumption is a critical challenge for on-device streaming ASR, impacting device recharge frequency and user experience. This study analyzed power usage in ASR models, revealing its dependence on model size, invocation frequency, and memory placement. Notably, the Joiner consumes more power than the larger Encoder and Predictor due to its higher invocation frequency and off-chip memory usage. We also identified an exponential relationship between word error rate and encoder size.

Based on these insights, we developed guidelines for model compression to enhance energy efficiency. Applying these guidelines to the LibriSpeech and Public Video datasets achieved up to 47% energy savings and a 29% reduction in RTF, maintaining accuracy comparable to state-of-the-art methods. These findings highlight the potential of targeted optimizations to advance sustainable and energy-efficient on-device streaming speech recognition.

References

Rami Botros, Tara N Sainath, Robert David, Emmanuel Guzman, Wei Li, and Yanzhang He. 2021. Tied & Reduced RNN-T Decoder. In *INTERSPEECH*.

Linhao Dong, Shuang Xu, and Bo Xu. 2018. Speech-Transformer: A No-Recurrence Sequence-to-Sequence Model for Speech Recognition. In *ICASSP*.

Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein. 2020. RNN-Transducer with Stateless Prediction Network. In *ICASSP*.

Alex Graves. 2012. Sequence Transduction with Recurrent Neural Networks. *ICML Workshop on Representation Learning*.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech Recognition with Deep Recurrent Neural Networks. In *ICASSP*.

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*.

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 *ICASSP*.

Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, Shinji Watanabe, Takenori Yoshimura, and Wangyou Zhang. 2019. A Comparative Study on Transformer vs RNN in Speech Applications. In *ASRU*.

Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. 2015. Audio Augmentation for Speech Recognition. In *INTERSPEECH*.

Duc Le, Frank Seide, Yuhao Wang, Yang Li, Kjell Schubert, Ozlem Kalinli, and Michael L. Seltzer. 2023. Factorized Blank Thresholding for Improved Runtime Efficiency of Neural Transducers. In *ICASSP*.

Benjamin C. Lee, Engin Ipek, Onur Mutlu, and Doug Burger. 2009. Architecting Phase Change Memory as a Scalable DRAM Alternative. In *ISCA*.

Jinmook Lee, Changhyeon Kim, Sanghoon Kang, Dongjoo Shin, Sangyeob Kim, and Hoi-Jun Yoo. 2018. UNPU: A 50.6TOPS/W unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision. In *ISSCC*.

Bo Li, Anmol Gulati, Jiahui Yu, Tara N Sainath, Chung-Cheng Chiu, Arun Narayanan, Shuo-Yiin Chang, Ruoming Pang, Yanzhang He, James Qin, et al. 2021. A Better and Faster End-to-End Model for Streaming ASR. In *ICASSP*.

Haitong Li, Mudit Bhargav, Paul N. Whatmough, and H.-S. Philip Wong. 2019. On-Chip Memory Technology Design Space Explorations for Mobile Deep Neural Network Accelerators. In *DAC*.

Yang Li, Saugata Ghose, Jongmoo Choi, Jin Sun, Hui Wang, and Onur Mutlu. 2017. Utility-Based Hybrid Memory Management. In *CLUSTER*.

Yang Li, Liangzhen Lai, Yuan Shangguan, Forrest N Iandola, Ernie Chang, Yangyang Shi, and Vikas Chandra. 2024a. Folding Attention: Memory and Power Optimization for On-Device Transformer-based Streaming Speech Recognition. In *ICASSP*.

Yang Li, Yuan Shangguan, Yuhao Wang, Liangzhen Lai, Ernie Chang, Changsheng Zhao, Yangyang Shi, and Vikas Chandra. 2024b. Not All Weights Are Created Equal: Enhancing Energy Efficiency in On-Device Streaming Speech Recognition. *arXiv preprint arXiv: 2402.13076*.

Micron. 2006. Technical Note TN-47-04: Calculating Memory System Power for DDR2. Technical report.

Niko Moritz, Takaaki Hori, and Jonathan Le Roux. 2020. Streaming Automatic Speech Recognition with the Transformer Model. *arXiv preprint arXiv:2001.02674*.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR Corpus based on Public Domain Audio Books. In *ICASSP*.

Rohit Prabhavalkar, Takaaki Hori, Tara N. Sainath, Ralf Schlüter, and Shinji Watanabe. 2024. End-to-End Speech Recognition: A Survey. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 32:325–351.

Martin Radfar, Rohit Barnwal, Rupak Vignesh Swaminathan, Feng-Ju Chang, Grant P Strimel, Nathan Susanj, and Athanasios Mouchtaris. 2022. ConvRNN-T: Convolutional Augmented Recurrent Neural Network Transducers for Streaming Speech Recognition. In *INTERSPEECH*.

Yuan Shangguan, Jian Li, Qiao Liang, Raziel Alvarez, and Ian McGraw. 2019. Optimizing Speech Recognition for the Edge. *MLSys On-device Intelligence Workshop*.

Yangyang Shi, Yongqiang Wang, Chunyang Wu, Ching-Feng Yeh, Julian Chan, Frank Zhang, Duc Le, and Mike Seltzer. 2021. Emformer: Efficient Memory Transformer Based Acoustic Model for Low Latency Streaming Speech Recognition. In *ICASSP*.

Ganesh Venkatesh, Alagappan Valliappan, Jay Mahadeokar, Yuan Shangguan, Christian Fuegen, Michael L. Seltzer, and Vikas Chandra. 2021. Memory-Efficient Speech Recognition on Smart Devices. In *ICASSP*.

Weiran Wang, Ding Zhao, Shaojin Ding, Hao Zhang, Shuo-Yiin Chang, David Rybach, Tara N. Sainath, Yanzhang He, Ian McGraw, and Shankar Kumar. 2023. Multi-output RNN-T Joint Networks for Multi-Task Learning of ASR and Auxiliary Tasks. In *ICASSP*.

Yongqiang Wang, Abdelrahman Mohamed, Due Le, Chunxi Liu, Alex Xiao, Jay Mahadeokar, Hongzhao Huang, Andros Tjandra, Xiaohui Zhang, Frank Zhang, Christian Fuegen, Geoffrey Zweig, and Michael L. Seltzer. 2020. Transformer-based Acoustic Modeling for Hybrid Speech Recognition. In *ICASSP*.

Haichuan Yang, Yuan Shangguan, Dilin Wang, Meng Li, Pierce Chuang, Xiaohui Zhang, Ganesh Venkatesh, Ozlem Kalinli, and Vikas Chandra. 2022. Omni-Sparsity DNN: Fast Sparsity Optimization for On-Device Streaming E2E ASR via Supernet. In *ICASSP*.

Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L. Seltzer. 2019. Transformer-Transducer: End-to-End Speech Recognition with Self-Attention. *arXiv preprint arXiv:1910.12977*.

Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. 2020. Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss. In *ICASSP*.

A Details of Adam-Pruning Algorithm

Adam-pruning is an iterative method designed to prune a model or its components. Each pruning step is executed over N training epochs. During each step, Adam-pruning evaluates the square of the gradient ($E\left[\left(\frac{\partial l}{\partial w}\right)^2\right]$) for every non-sparse parameter w in the model. A larger square of the gradient suggests that pruning the parameter would result in a substantial change in the model's performance. Based on this, Adam-pruning prunes only the parameters with the top K smallest gradient squares at the end of each pruning step. After M such steps, Adam-pruning reduces the model to a desired level of sparsity.

B Details of the Datasets

B.1 LibriSpeech

LibriSpeech (Panayotov et al., 2015), is a prominent corpus extensively utilized in speech recogni-

tion research. This corpus features 960 hours of English speech, sourced from audiobooks available through the LibriVox project, which are in the public domain. It includes two main evaluation sets tailored for different testing scenarios:

- Test-Clean: This subset consists of high-quality, clean audio recordings. It provides an ideal condition for benchmarking the baseline performance of speech recognition systems due to its clarity and ease of transcription.
- Test-Others: This subset encompasses recordings that present a variety of challenges, such as accents, background noises, and lower recording qualities. It serves as a stringent testing environment to evaluate the robustness and adaptability of speech recognition technologies under less-than-ideal conditions.

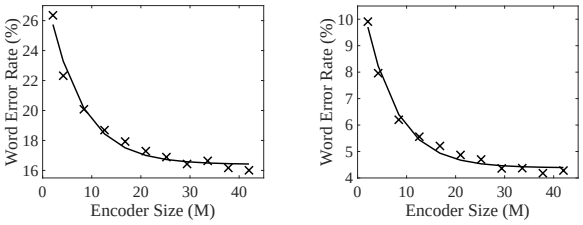
B.2 Public Video

The Public Video dataset, an in-house collection, is derived from 29.8K hours of audio extracted from English public videos. This dataset has been ethically curated with the consent of video owners and further processed to ensure privacy and enhance quality. We de-identify the audio, aggregate it, remove personally identifiable information (PII), and add simulated reverberation. We further augment the audio with sampled additive background noise extracted from publicly available videos. Speed perturbations (Ko et al., 2015) are applied to create two additional copies of the training dataset at 0.9 and 1.1 times the original speed. We apply distortion and additive noise to the speed-perturbed data. These processing steps eventually result in a total of 148.9K hours of training data. For evaluating the performance of models trained on this dataset, we use the following two test sets:

- Dictation: This subset consists of 5.8K hours of human-transcribed, anonymized utterances, sourced from a vendor. Participants were asked to engage in unscripted open-domain dictation conversations, recorded across various signal-to-noise ratios (SNR), providing a diverse assessment environment.
- Messaging: This subset comprises 13.4K hours of utterances, sourced from a vendor. It features audio messages recorded by individuals following scripted scenarios intended for an unspecified recipient. These utterances are generally shorter and incorporate more noise than those in the dictation subset, offering a different dimension to evaluate ASR systems.

C Accuracy of ASR Models Trained on Public Video

We applied compression to the Encoder of the ASR model trained using the Public Video dataset. The impact of this compression on word error rates across two evaluation sets, Dictation and Messaging, is depicted in Figures 8 (a) and (b). To analyze the data, we employed the function outlined in Equation 1, which proved to be an excellent fit; the predictions derived from this function align closely with the observed data. Quantitatively speaking, the adjusted R-squared values—0.9760 for Dictation and 0.9851 for Messaging—underscore the exponential relationship between word error rate and encoder size, reaffirming this pattern's consistency across different datasets.



(a) Dictation (b) Messaging

Figure 8: Models trained on the Public Video dataset: Word error rate with compressing Encoder while keeping the size of Predictor and Joiner.

D Impact of Input Stride and Chunk Size on Model Accuracy and Power Usage

Input stride and chunk size are two essential hyperparameters for on-device streaming ASR. Input stride defines the time window over which input frames are combined into an aggregated frame that is then fed into the model. Chunk size refers to the time duration over which these aggregated frames are processed together as a batch by the model. In this section, we examine how varying these parameters affects the performance and power consumption of the Neural Transducer.

We first vary the input stride from 20 milliseconds to 40 milliseconds and evaluate the accuracy and power consumption of four models trained on LibriSpeech: a dense model, a model with 80% sparsity in its encoder, a model with 80% sparsity in its predictor, and a model with 80% sparsity in its joiner. The results are provided in Tables 2 and 3.

Word Error Rate (%)	Input Stride	Dense Model	80% Sparse Encoder	80% Sparse Predictor	80% Sparse Joiner
Test-Clean	20ms	3.61	4.72	3.61	4.17
	40ms	3.56	4.86	3.60	3.64
Test-Other	20ms	9.13	11.90	9.13	9.58
	40ms	9.06	12.08	9.14	9.29

Table 2: Impact of input stride on the model accuracy trained on LibriSpeech.

Model Power Consumption (mW)	Input Stride	Dense Model	80% Sparse Encoder	80% Sparse Predictor	80% Sparse Joiner
	20ms	131	104	123	62
	40ms	118	92	110	62

Table 3: Impact of input stride on the power consumption of models trained on LibriSpeech.

Word Error Rate (%)	Chunk Size	Dense Model	80% Sparse Encoder	80% Sparse Predictor	80% Sparse Joiner
Test-Clean	160ms	3.56	4.86	3.60	3.64
	320ms	3.50	4.60	3.50	3.52
Test-Other	160ms	9.06	12.08	9.14	9.29
	320ms	8.82	11.75	8.83	8.90

Table 4: Impact of chunk size on the model accuracy trained on LibriSpeech.

Model Power Consumption (mW)	Chunk Size	Dense Model	80% Sparse Encoder	80% Sparse Predictor	80% Sparse Joiner
	160ms	118	92	110	62
	320ms	94	86	87	38

Table 5: Impact of chunk size on the power consumption of models trained on LibriSpeech.

Our findings are as follows:

- Observation 1: A smaller stride can have both positive and negative effects on model performance.
- Observation 2: A smaller stride generally increases power consumption.

Regarding the first observation, input stride is used to enhance training and inference efficiency by reducing sequence length. While a smaller stride better preserves local acoustic features and improves performance, it also introduces risks such as greater sensitivity to noise and loss of broader contextual information. A stride of 4–6 is commonly chosen to balance accuracy and efficiency.

As for the second observation, in streaming ASR, a smaller stride increases the number of segments, resulting in more frequent decoding of blank tokens and thus more frequent invocation of the joiner, which raises power consumption. However, if the joiner is compressed to fit within the SRAM, this increased invocation has minimal impact on power usage, due to the high energy efficiency of SRAM.

We also vary the chunk size from 160ms to 320ms and measure the accuracy and power consumption of four models: a dense model, a model with 80% sparsity in its encoder, a model with 80% sparsity in its predictor, and a model with 80% sparsity in its joiner. The results are provided in Tables 4 and 5. Our observations are as follows:

- Observation 3: Increasing the chunk size generally improves model accuracy.
- Observation 4: Larger chunk sizes reduce model power consumption.

For the third observation, larger chunk sizes enable the encoder to capture relationships between segments more effectively, improving performance. However, smaller chunk sizes have the advantage of lowering decoding latency.

As for the fourth observation, in streaming ASR, a larger chunk size decreases the frequency at which the encoder is invoked, thereby reducing memory power usage and overall power usage.