Massive End-to-end Speech Recognition Models with Time Reduction

Weiran Wang **Rohit Prabhavalkar** **Haozhe Shan** **Zhong Meng**
Dongseong Hwang **Qiujia Li** **Khe Chai Sim** **Bo Li** **James Qin**
Xingyu Cai **Adam Stooke** **Chengjian Zheng** **Yanzhang He**
Tara Sainath **Pedro Moreno Mengibar**
{weiranwang, prabhavalkar, hzshan, zhongmeng, dongseong}@google.com

Abstract

We investigate massive end-to-end automatic speech recognition (ASR) models with efficiency improvements achieved by time reduction. The encoders of our models use the neural architecture of Google's universal speech model (USM), with additional funnel pooling layers to significantly reduce the frame rate and speed up training and inference. We also explore a few practical methods to mitigate potential accuracy loss due to time reduction, while enjoying most efficiency gain. Our methods are demonstrated to work with both Connectionist Temporal Classification (CTC) and RNN-Transducer (RNN-T), with up to 2B model parameters, and over two domains. For a large-scale voice search recognition task, we perform extensive studies on vocabulary size, time reduction strategy, and its generalization performance on long-form test sets, and show that a 900M RNN-T is very tolerant to severe time reduction, with as low encoder output frame rate as 640ms. We also provide ablation studies on the Librispeech benchmark for important training hyperparameters and architecture designs, in training 600M RNN-T models at the frame rate of 160ms.

1 Introduction

There has been a recent focus on scaling up end-to-end (E2E) ASR models, such as connectionist temporal classification (CTC, Graves et al., 2006 and neural transducers (RNN-T, Graves, 2012), to extremely large sizes (Zhang et al., 2022; Li et al., 2022; Radford et al., 2023; Zhang et al., 2023; Pratap et al., 2023), to unlock new levels of accuracy on more challenging tasks. Many of these models are motivated by the goal of training a single multilingual ASR model which can perform well across a range of languages. For example, the Universal Speech Model (USM, Zhang et al., 2023) is a 2B parameter full-context CTC model trained on YouTube data from 300+ languages; Pratap et al. (2023) train ASR systems on 1000+ languages.

The present work follows in the footsteps of these previous works, and addresses the important practical challenges of training massive E2E models: as model size increases, the cost of computing the outputs of the encoders in the model (most of the model parameters are typically in the encoder) scales poorly in attention-based encoders such as transformers (Vaswani et al., 2017) and conformers (Gulati et al., 2020). The RNN-T model, in particular, requires additional computation and memory for the prediction and joint network, which further compounds the problem.

In this work, we investigate the aforementioned issues and find that encoder output frame rate reduction can be applied repeatedly at multiple layers in the encoder to obtain a large output frame rate reduction; this in turn is critical for training models and performing inference efficiently. On a large scale voice search task with short queries, while past research incorporated time reduction to lower the final frame rate to 60ms (He et al., 2019; Ding et al., 2022), or 120ms (Cai et al., 2023) for voice search we find that we can increase time reduction to 6x with large CTC, and all the way up to 16x with large RNN-T, of the 40ms base frame rate with funnel pooling (Dai et al., 2020), without sacrificing much accuracy.

We also propose a few practical techniques to mitigate the potential accuracy loss due to time reduction while still enjoying most efficiency improvements. We provide ablation studies of these techniques on the Librispeech benchmark, for training 600M RNN-T model with a frame rate of 160ms. Our results show that time reduction is generally useful for ASR in different domains.

2 End-to-end ASR Models

In this section, we briefly describe the E2E models employed in this work. Interested readers can find

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 6206–6217

more information in recent overview articles (Li, 2022; Prabhavalkar et al., 2023).

Notations We assume that the input audio signal has been parameterized into suitable acoustic feature vectors: $\mathbf{x} = [\mathbf{x}_1, \ldots, \mathbf{x}_{T'}]$, of length T', where $\mathbf{x}_t \in \mathbb{R}^d$ (128-dimensional log-mel features, in this work). The input acoustic features are processed using an *encoder* which transforms the input into a higher-level representation: $\mathbf{h} = [\mathbf{h}_1, \ldots, \mathbf{h}_T]$, where the length of the encoded representation is typically shorter than the original input length ($T \leq T'$). We assume that we have an input transcript corresponding to each utterance: $\mathbf{y} = [y_0 = \langle \mathrm{s} \rangle, y_1, \ldots, y_U]$, where each $y_u \in \mathcal{V}$, the set of output symbols (word-pieces (Schuster and Nakajima, 2012), in this work), and $\langle \mathrm{s} \rangle$ represents a special start-of-sentence symbol.

2.1 Connectionist Temporal Classification

CTC was introduced by Graves et al. (2006), as a way to train sequence-to-sequence models which can transduce the input sequence $\mathbf{x}$, into the output sequence $\mathbf{y}$ when the alignments between the two sequences are unknown. CTC accomplishes this by modeling the conditional distribution, $P(\mathbf{y}|\mathbf{x})$, which marginalizes over all possible alignments between the two sequences:

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a} \in \mathrm{B}_{\mathrm{ctc}}(\mathbf{y})} \prod_{t=1}^{T} P(a_t|\mathbf{h}) \qquad (1)$$

where, the $\mathrm{B}_{\mathrm{ctc}}(\mathbf{y})$ corresponds to the set of all valid alignments. Specifically, an alignment $\mathbf{a} \in \mathrm{B}_{\mathrm{ctc}}(\mathbf{y})$ is a valid alignment if it contains $|\mathbf{h}(\mathbf{x})| = T$ symbols from the set of outputs augmented with a special *blank* symbol – $\mathcal{V} \cup \{\langle \mathrm{b} \rangle\}$; and if additionally, removing consecutive repeated non-blank symbols and then removing all $\langle \mathrm{b} \rangle$ symbols produces the original label sequence $\mathbf{y}$. As can be seen in (1), CTC models make a strong conditional independence assumption – that the output labels are conditionally independent given the input acoustic encoded features. However, these models work well in practice (Miao et al., 2015; Karita et al., 2019), especially with large encoders (Zhang et al., 2023). Finally, the CTC model produces one output symbol (blank, or non-blank) per encoder time step.

2.2 Neural Transducers

The recurrent neural network transducer (RNN-T) was proposed by Graves (2012), as an improvement over the CTC model, to reduce conditional independence assumptions in the model. This is achieved by introducing a separate *prediction network*, which models label dependency (and thus makes model outputs conditionally dependent on the sequence of previous predictions).

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a} \in \mathrm{B}_{\mathrm{nt}}(\mathbf{y})} \prod_{\tau=1}^{T+U} P(a_\tau|\mathbf{q}_{i_\tau}, \mathbf{h}_{\tau - i_\tau})$$

where, $\mathrm{B}_{\mathrm{nt}}(\mathbf{y})$ is the set of all valid alignments – sequences of length $T + U$, which are identical to $\mathbf{y}$ after removing all blanks (i.e., each sequence has exactly T blank symbols); i_τ represents the number of non-blank labels in the partial alignment $a_1, \ldots, a_{\tau-1}$; and, $\mathbf{q}_j$ represents the output of the prediction network after processing the first $j-1$ output labels: $\mathbf{q}_j = \mathrm{PredNetwork}(y_{j-1}, \ldots, y_0)$. Notably, neural transducers can output multiple (non-blank) labels at each frame; blanks correspond to transitions to the next input encoder frame.

3 Massive Models with Time Reduction

In this work, we boost the ASR performance by tremendously scaling up the size of the E2E models. Recognizing the encoder's crucial role in E2E speech recognition, we dramatically expand encoder parameters (up to 1.8B), enabling the model to effectively leverage the vast amount of training data and improve the recognition accuracy.

The USM encoder of our E2E model is built on the convolution-augmented Transformer (Gulati et al., 2020), or Conformer architecture. Each Conformer block consists of a feed-forward module (FFN), a multi-head self-attention module (MHSA), a convolution module (CONV), and a second feed-forward module. If the input to l-th Conformer block is $\mathbf{x}^{(l)}$, then its output (and, the input to the next block), $\mathbf{x}^{(l+1)}$, is computed as:

$$\begin{aligned}
\tilde{\mathbf{x}}^{(l)} &= \mathbf{x}^{(l)} + \frac{1}{2}\mathrm{FFN}_1\big(\mathbf{x}^{(l)}\big) \\
\mathbf{x}'^{(l)} &= \tilde{\mathbf{x}}^{(l)} + \mathrm{MHSA}\big(W_q\tilde{\mathbf{x}}^{(l)}, W_k\tilde{\mathbf{x}}^{(l)}, W_v\tilde{\mathbf{x}}^{(l)}\big) \\
\mathbf{x}''^{(l)} &= \mathbf{x}'^{(l)} + \textsc{Conv}\big(\mathbf{x}'^{(l)}\big) \qquad (2) \\
\mathbf{x}^{(l+1)} &= \textsc{LayerNorm}\big(\mathbf{x}''^{(l)} + \frac{1}{2}\mathrm{FFN}_2(\mathbf{x}''^{(l)})\big)
\end{aligned}$$

For the standard conformer layer, the input length and output length are the same.

3.1 Efficiency improvements

We explore two ways to reduce the computational latency and speed up the inference.

3.1.1 Encoder Frame Rate Reduction

We improve training and inference speed in our models by reducing the sequence length of the encoder embeddings along time axis, relative to the audio length. Specifically, we employ the pooling technique in the MHSA module introduced in Funnel-Transformers (Dai et al., 2020) at selected Conformer blocks within the encoder. At a funnel self-attention layer, the entire input is used to produce the key and value sequences as usual, but strided-average pooling is applied to the time dimension of the input used in producing the query vectors. That is, we replace the MHSA module in (2) with the following:

$$\hat{\mathbf{x}}^{(l)} = \textsc{StridedPooling}(\tilde{\mathbf{x}}^{(l)}), \quad (3)$$
$$\mathbf{x}'^{(l)} = \hat{\mathbf{x}}^{(l)} + \text{MHSA}\big(W_q\hat{\mathbf{x}}^{(l)}, W_k\tilde{\mathbf{x}}^{(l)}, W_v\tilde{\mathbf{x}}^{(l)}\big).$$

The stride of pooling in (3) is the time reduction factor in this layer. This effectively downsamples the output encoder embeddings of the previous layer. In a language model (LM), this pool-query-only method was shown to provide a slight advantage over simply pooling the entire hidden embedding sequence between layers (Dai et al., 2020); likely, the higher granularity in the key and value sequences allows the network to learn a less lossy compression. Computational savings accrue in all subsequent Conformer layers according to the $O(T^2D)$ complexity for self-attention, and linearly for feed-forward networks. For inference, this directly shrinks the number of decoding steps which is $O(T)$ for CTC and $O(T+U)$ for RNN-T, thus significantly accelerating the decoding process.

3.1.2 Light-Weight Decoder

To reduce the computational latency for short voice search queries, we utilize a light-weight $|V|^2$ embedding prediction network (Botros et al., 2021) which conditions the current prediction only on the two previous non-blanks:

$$\mathbf{E}_u = \mathbf{W}_e \cdot \text{Concat}(\mathbf{E}_{u-1}, \mathbf{E}_{u-2}) \quad (4)$$

where $\mathbf{E}_n$ is the embedding vector at decoder step n and $\mathbf{W}_e$ is the projection matrix.

3.2 Compensation for Accuracy Loss

While time reduction benefits efficiency, it can be detrimental for longer, acoustically and linguistically rich utterances. This is because critical information such as long-term dependencies and prosodic cues gets inevitably compressed during the reduction process, resulting in degraded accuracy. We propose methods to mitigate the potential accuracy loss caused by time reduction, which are effective for different tasks in our experiments.

3.2.1 Rotary Positional Embeddings

Maintaining positional information, in the presence of pooling, is useful especially for long audio. We apply rotary positional embeddings (RoPE, Su et al., 2023) to the MHSA module at each conformer layer. Compared to the similar relative positional embeddings (RPE, Dai et al., 2019), RoPE has the advantage of not requiring additional trainable parameters nor longer training latency. In layers where the sequence length is shortened by a factor of 2, keys and values have their usual positional indices (0, 1, 2, 3, 4, 5, ...) while the shortened query sequence uses (1, 3, 5,...) to maintain consistent positional information despite shortening. Similarly, in upsampling layers the lengthened sequence uses (0, 0, 1, 1, 2, 2...) as indices.

3.2.2 Larger Batch Size

We find that increasing the number of utterances in a training mini-batch, namely batch size, can be effective at offsetting the WER degradation resulted from time reduction. In funnel encoders where the input sequence is downsampled by a factor of 4, the number of tokens per batch is similarly reduced. To maintain the usual tokens per batch, a larger batch size may be required.

3.2.3 Encoder Frame Upsampling

To alleviate the information loss caused by funnel pooling in early encoder layers, we propose upsampling the encoder embeddings at later encoder layers. This injects additional information back into the sequence, and allow learning to recover lost details and improve model performance (Ronneberger et al., 2015; Dai et al., 2020).

The total upsampling rate at higher layers should be lower than the total reduction rate at lower layers to ensure an effective decimation of encoder output frames and decoding steps. Note that, our method still enjoys the decrease of encoder computation even if the encoder embeddings are upsampled back to the original frame rate at the input.

3.2.4 Language Model Fusion

LM fusion (Gulcehre et al., 2015; Chorowski and Jaitly, 2017) is a simple yet effective way of injecting text information, potentially learned on large amounts of unpaired text-only data, into E2E ASR

systems. With the much smaller number of output frames and on-TPU/GPU beam search, shallow fusion with a neural LM becomes a viable candidate for enhancing model accuracy.

To achieve best accuracy with LM fusion, we make use of the notion of an internal language model (ILM), $P_{\text{ILM}}(\mathbf{y})$, which represents the LM learned by the E2E model based on the training data; the ILM can thus be subtracted out from the posterior distribution, before fusion with an external LM, $P_{\text{EXT}}(\mathbf{y})$, during decoding, by adding tunable hyperparameters α, and β which can be set based on a development set to produce the most likely hypothesis (Morgan and Bourlard, 1990; Variani et al., 2015; Kanda et al., 2017; McDermott et al., 2019; Variani et al., 2020; Meng et al., 2021):

$$\mathbf{y}^* = \arg\max_{\mathbf{y}} \log P_{\text{E2E}}(\mathbf{y}|\mathbf{x}) - \alpha \log P_{\text{ILM}}(\mathbf{y}) + \beta \log P_{\text{EXT}}(\mathbf{y}).$$

ILMs can be estimated for CTC (Sak et al., 2015; Li et al., 2019), RNN-T (Meng et al., 2021) and the hybrid autoregressive transducer (HAT, Variani et al., 2020) variant of RNN-T.

4 Related Work in Time Reduction

Time reduction has been a useful technique for achieving a balance between input and output lengths for ASR, and has evolved over time with the choice of neural architecture and modeling units. In Vanhoucke et al. (2013); Miao et al. (2016), the lower frame rates were achieved by concatenating input frames and striding, for DNN or LSTM models predicting context-dependent HMM states. Hihi and Bengio (1996); Koutnik et al. (2014); Chan et al. (2016) proposed to use hierarchical/pyramidal RNNs where outputs of consecutive steps are combined before feeding to the next layer. After the community switched to end-to-end systems and word-piece type modeling units, a popular frame rate is 40ms as achieved by convolutional subsampling, and is adopted by widely-used open-source libraries (Watanabe et al., 2018). In this work, we also use such a configuration for the input to our encoder, and perform funnel pooling atop.

With the similar intuition of reducing sequence lengths at encoder layers, Efficient Conformer (Burchi and Vielzeuf, 2021) used convolution and strided attention to produce shortened sequences in a progressive manner. Squeezeformer (Kim et al., 2022) applied progressive downsampling in the decoder and included upsampling at the end to recover the original sequence length with a U-net-like design; a similar idea was studied in Andrusenko et al. (2023). Compared to these prior work, we investigate the use of time reduction with much larger model sizes and more extreme reduction rates, on multiple datasets from different domains, and explore additional techniques to make up potential accuracy loss. Prabhavalkar et al. (2024) explore extreme encoder output reduction rates for RNN-T on a large-scale voice search task with as low frame rate as 2.56 secs, by using funnel pooling at later encoder layers.

Another line of work (Tian et al., 2021; Wang et al., 2023; Yang et al., 2023) proposed to use CTC to (irregularly) select encoder output frames (or to skip confident blank frames) for RNN-T modeling, effectively reducing the frame rate for the decoder. However, such a CTC-based encoder output selection does not save computations in the encoder, but only reduces the computation in downstream modules. Our goal in this work is to aggressively reduce the sequence length early in the architecture for computational savings.

5 Experiments on Voice Search Queries

Our first set of experiments focus on a short voice search task with in-house datasets.

5.1 Datasets

For a majority of the experiments, we use 520M utterances of voice search queries for training; the total amount of audio is 490K hours and the average duration per utterance is 3.4 seconds. A small percentage of the training data is human transcribed while the rest is pseudo-labeled by a 600M bidirectional RNN-T teacher (Hwang et al., 2022). We tokenize training transcripts with word-piece models (Schuster and Nakajima, 2012).

We use both real audio and TTS-generated data for evaluation. The real audio utterances are representative of typical voice search traffic, with an average duration of 3.9 seconds. Our development set consists of 9K real audio utterances (denoted as VS-dev); we use a separate held-out test set consisting of 5K utterances for testing (denoted as VS-test). The TTS sets contain rare proper nouns (RPN) appearing fewer than 5 times in the training set. Each TTS set contains 10K utterances and covers one of five domains: Maps (denoted as RPNM), News (RPNN), Play (RPNP), Search query logs (RPNS), and Youtube (RPNY); they have average

durations of 5.9, 10.1, 5.3, 5.4, and 5.8 seconds respectively. We use VS-dev, RPNM and RPNN for tuning the model architecture and other hyperparameters and report final WERs on the rest sets.

To improve model accuracy on rare words, we fuse CTC and RNN-T with neural LMs. We train transformer LMs of 128M and 1B parameters with word-pieces compatible with those of E2E models. The LM training data consists of transcripts of ASR training data, and text-only data containing textual search queries from the five domains of RPN sets (Huang et al., 2022). All acoustic and text training data is anonymized and adheres to Google AI Principles (Google).

5.2 Model Architectures

We use the 128-dimensional log Mel-filterbank energies (extracted from 32ms window and 10ms shift) as the frontend features. After two 2D-convolution layers, both with strides (2,2), the resulting feature sequence has a frame rate of 40ms and becomes the input to our Conformer architecture. This architecture mimics that of Google's universal speech model (USM, Zhang et al., 2023). The number of attention heads used in Conformer blocks is 8, and the intermediate dimension of the FFNs is 4 times the model dimension. The Conformer blocks use local self-attention with a large attention span, and the encoder output has a large enough receptive field to cover the entire utterance.

We explore two different encoder configurations for CTC, with different sizes: the smaller encoder configuration consists of 24 Conformer layers of dimension 768, leading to a total of 340M parameters; the larger encoder has 32 Conformer layers of dimension 1536, leading to a total of 1.8B parameters. For RNN-T, the encoder consists of 16 Conformer layers of dimension 1536, resulting in a total of 870M parameters. We use a $|V|^2$ embedding decoder (Botros et al., 2021), i.e., the prediction network computes LM features based on two previous non-blank tokens, which works well on voice search data. The RNN-T model output uses the HAT factorization (Variani et al., 2020) to benefit external LM integration.

Following Ding et al. (2022), we start funnel pooling at layer 4 (layer index is zero-based), and apply pooling in subsequent layers to achieve the desired factor of time reduction. As an example, if we perform pooling at layers 4 and 5, each with the reduction factor 2, we achieve a total reduction factor 4 for layers 6 and onwards, i.e., the sequence length after layer 6 is 1/4 of the original input length and the frame rate at the encoder output is 160ms, which is also the frame rate at which the decoder operates. Note that (Ding et al., 2022) used funnel pooling for on-device modeling with stringent latency requirements, whereas here we use funnel pooling for the full-context model to speed up training and inference. We observe that the ASR accuracy turns out to be more tolerant of time reduction.

Each model is trained with the Adafactor optimizer (Shazeer and Stern, 2018) with a batch size of 4096 utterances. We train CTC models to 500K steps and RNN-T models to 300K steps, by which point WERs on development sets have stabilized.

For inference, we perform beam search (with path merging) with a beam size of 8. For LM fusion, two types of prior probabilities were used for CTC: the uniform prior over non-blanks as implemented by the blank probability downscaling technique (Sak et al., 2015), versus the uni-gram prior based on the model posteriors on training set (Li et al., 2019). We perform internal LM score subtraction for fusion with HAT (Variani et al., 2020).

5.3 CTC Results

Intuitively, with a larger vocabulary, the label sequences are shorter and we can afford more time reduction. The hard constraint for CTC is that, since it emits only one token (blank or non-blank) at each encoder output frame, the encoder output sequence length must remain longer than the label sequence length (RNN-T however is not subject to this constraint). During CTC training, we discard utterances that violate this constraint, although such cases are uncommon in the VS sets.

We report WERs of a selection of CTC models in Table 1. As baselines, we report the WERs with 4K vocabulary at a 40ms frame rate, a setup closely following that of the USM architecture (Zhang et al., 2023). We match these baselines on VS with the 16K vocabulary size and much lower frame rates of 160ms and 240ms: for 340M CTC, we obtain an improvement from 4.7% to 4.5% on VS-dev, while for 1.8B CTC, we match the 4.2% obtained with 40ms frame rate. For supervised training with 1.8B CTC, we observe a 4x speedup in training time on TPU with a 160ms frame rate, and a 4.5x speedup with a 240ms frame rate compared to the 40ms model (despite the use of a more costly softmax operation due to larger vocabulary). We observe that with a 320ms frame rate, the WER on VS-dev

Model size (FR)	VS-dev	RPNM	RPNN
baselines: no pooling, vocab size=4096			
340M (40ms)	4.7	15.1	12.4
1.8B (40ms)	4.2	14.2	**10.4**
with funnel pooling, vocab size=16384			
340M (160ms)	4.5	15.3	16.6
340M (240ms)	4.5	14.9	21.6
1.8B (40ms)	**4.2**	14.8	10.5
1.8B (160ms)	4.3	**13.7**	12.7
1.8B (240ms)	**4.2**	13.8	17.3
1.8B (320ms)	5.0	14.0	26.4
1.8B (40ms) + 128M LM fusion			
blank downscaling	**3.8**	11.1	8.8
model-based prior	**3.8**	10.8	**8.6**
1.8B (160ms) + 128M LM fusion			
blank downscaling	**3.8**	10.7	11.0
model-based prior	**3.8**	10.5	10.8
1.8B (160ms) + 1B LM fusion			
blank downscaling	**3.8**	10.1	10.4
model-based prior	**3.8**	**9.8**	10.1

Table 1: WERs (%) on dev sets by CTC, with different architectures, vocabulary sizes, and encoder output frame rates (FR). Funnel pooling is employed at layers 4 and 5, with factors 2,2 for 160ms frame rate and 3,2 for a 240 frame rate.

Frame rate (factors)	VS-dev	RPNM	RPNN
vocab size=4096, decoder size=10M			
40ms	3.8	**12.1**	**11.0**
240ms (3x2)	3.8	12.6	12.1
320ms (2x2x2)	3.8	12.9	14.2
vocab size=16384, decoder size=33M			
160ms (2x2)	**3.7**	12.3	12.8
240ms (3x2)	3.8	12.5	12.2
320ms (2x2x2)	3.8	12.5	13.6
400ms (5x2)	3.8	12.7	15.6
480ms (3x2x2)	3.9	12.6	19.5
640ms (2x2x2x2)	3.9	12.9	28.7
vocab size=32768, decoder size=65M			
320ms (2x2x2)	3.8	12.9	14.8
640ms (2x2x2x2)	3.9	12.7	25.9

Table 2: Dev set WERs (%) by RNN-T with various vocabulary sizes and encoder output frame rates (FR). The encoder contains 870M parameters. Funnel pooling starts at layer 4 and applies to adjacent layers. In parenthesis, we use the notation "3x2" to indicate that layer 4 has a reduction factor of 3, and layer 5 has a factor of 2, leading to the final frame rate of 240ms.

significantly degrades to 5.0%, and this could not be alleviated by increasing the vocabulary size to 32K. This suggests that a too coarse time resolution does not work well with the CTC loss and its underlying independence assumptions.

The trend of WER on RPNM is similar to that of VS-dev. However, we do observe worse degradation on RPNN as we apply heavier time reduction. As mentioned in Sec 5.1, the transcriptions of RPNN come from the News domain which has different linguistic characteristics from voice search, and the audio length is quite longer (10 secs on average) than the VS set (4 secs on average). We hypothesize that models with lower frame rates may not generalize well to unseen audio length, and further investigate this issue in Sec 5.5.

We then perform LM fusion with 1.8B CTC models with frame rates 40ms and 160ms. The results we present in Table 1 uses LM weights that achieve a good balance between VS-dev and RPN sets. We observe a significant WER reduction from 4.2% to 3.8% with a smaller 128M LM already on the in-domain VS-dev set, and a reduction from 13.7% to ∼11% on RPNM. Among the two prior estimation methods, model-based prior (Li et al., 2019) consistently outperforms blank probability downscaling (Sak et al., 2015) over different WER operating points. When increasing the external LM size to 1B, we could not further improve on VS-dev but achieve sizeable gains on in-domain rare word test sets, e.g., with model-based prior, WER is reduced from 10.5% to 9.8% for RPNM.

5.4 RNN-T Results

We conduct a similar set of time reduction experiments with RNN-T, and the results are shown in Table 2. Overall RNN-T is quite robust to different vocabulary sizes. With the 16K vocabulary, we have studied the most time reduction settings, and observed only small WER degradations from 40ms frame rate all the way to 640ms frame rate on VS-dev and RPNM.

Comparing the models at the same frame rate of 160ms and vocabulary size 16K, RNN-T has a WER of 3.7% on VS-dev, outperforming the 4.2% by CTC by a large margin. This demonstrates the benefit of having a learnable decoder feature for modeling label dependency in end-to-end ASR.

When fusing the best RNN-T model with a 128M LM, we observe interestingly that it tends to significantly degrade RPNN (with heavy deletion errors) and the WERs are quite sensitive to

Frame rate	VS-dev	RPNM	RPNN
Voice search training			
40ms	3.8	12.1	11.0
240ms	3.8	12.5	12.2
640ms	3.9	12.9	28.7
Multi-domain training			
40ms	**3.6**	13.3	6.8
240ms	**3.6**	13.6	6.9
640ms	3.8	12.6	9.5
+ 128M LM fusion			
40ms	**3.6**	**11.4**	**6.7**
240ms	3.7	11.5	7.1

Table 3: Dev set WERs (%) by 900M RNN-T with different frame rates trained on multi-domain data. The 40ms model uses a vocabulary of 4096 while the others use a vocabulary of 16384. First three rows are taken from Table 2.

internal and external LM weights. We hypothesize this is partly due to the bias of the internal LM of HAT learned purely on short utterances. In the next section, we provide further evidence for this, by demonstrating that training on length-diverse data improves the robustness to shallow fusion.

5.5 Adding Long-Form Training Data

To verify that the poor performance of end-to-end models on RPNN was due to the lack of longer training audio, we repeat several RNN-T experiments with additional multi-domain training data (Narayanan et al., 2019). Most notably, we include segmented YouTube audio data containing 220M utterances with an average duration of 9.8 seconds, giving us a total of 600K hours of data.

The comparisons between RNN-T models trained on voice search data and multi-domain data are presented in Table 3. For vastly different frame rates 40ms, 240ms and 640ms, the additional long-form training data improves the WERs on both VS-dev and RPNN significantly. With only voice search training data, the WER gap on RPNN between the two frame rates used to be very large (12.2% vs 28.7%), whereas with multi-domain training, the gap is significantly reduced (6.9% vs 9.5%) with much improved absolute WERs.

When performing shallow fusion for the multi-domain 240ms frame rate model with the 128M LM, we achieve a better balance between in-domain and out-of-domain test sets, improving RPNM from 13.6% to 11.5% and without affecting RPNN much.

VS-test	RPNP	RPNS	RPNY
1.8B CTC, 160ms frame rate			
4.9	39.8	23.1	26.0
+ Multi-domain training data			
4.8	38.2	22.3	24.8
+ 128M LM fusion with model prior			
4.5	**32.6**	**15.8**	**19.4**
900M RNN-T, 40ms frame rate			
4.5	37.0	20.0	22.7
+ Multi-domain training data			
4.4	36.5	20.1	22.7
+ 128M LM fusion with ILM			
4.4	34.0	16.9	20.4
900M RNN-T, 240ms frame rate			
4.5	37.8	20.6	23.3
+ Multi-domain training data			
4.4	36.4	19.9	22.3
+ 128M LM fusion with ILM			
4.4	33.9	16.9	20.2
Multi-domain 120M RNN-T (Ding et al., 2022) 60ms frame rate, 0.9s look-ahead			
5.0	35.9	19.2	23.2

Table 4: Test set WERs (%) by CTC and RNN-T.

5.6 Final Evaluation

Finally, we compare the best configurations from both CTC and RNN-T on the test sets, namely VS-test, RPNP, RPNS, and RPNY. Taking into account both in-domain and out-of-domain performance, we choose the 1.8B CTC model with 160ms frame rate, the 900M RNN-T model with 40ms frame rate, and the 900M RNN-T model with 240ms frame rate. The results are shown in Table 4. Relative merits between methods are consistent with the performance on development sets. That is, with the same voice search training data, the 900M RNN-T outperforms the 1.8B CTC on all test sets, and by 8.0% relative margin for the in-domain VS-test set (4.5% vs 4.9%). After fusing the multi-domain models with neural LMs, we observe improvements on all test sets, and that the performance gap between CTC and RNN-T is largely removed. As a reference, we provide results of another small RNN-T model trained on the same multi-domain data, with a 60ms frame rate and a limited right context of 0.9s, from previous work (Ding et al., 2022).

Encoder (870M)	
Output Frame Rate (ms)	Runtime (ms)
40	153
80	107
160	70
240	67
320	63
640	63
Per-step Decoder Runtime	
V2 Decoder (10M params)	1.2
V2 + 128M LM Fusion	6.3

Table 5: Runtime of 900M RNN-T. Vocabulary size is 4096. Inference beam size is 8.

5.7 Inference Benchmark

To verify that our method leads to much faster inference, we benchmark the runtime of the components of the 900M RNN-T model on TPU v3 (tpu). We measure the running times for processing a batch of 8 utterances with the duration of 15 seconds, and the results are shown in Table 5.

We observe a 60% reduction of encoder runtime (from 153ms to 53ms), with 8x funnel reduction. Note the number of decoder steps we need to run for RNN-T is the encoder output length + label sequence length. And 8x funnel pooling would reduce the encoder output length by a factor of 8. Given the average audio length for voice search traffic is around 4 seconds, with an average label length of 12 tokens, our method significantly reduces the number of decoding steps and therefore decoder runtime. External LM fusion does increase the decoder model size and therefore per-step runtime (1.2ms -> 6.3ms), although the cost is not prohibitively expensive (considering also that modern chips can be quite faster).

6 Experiments on Librispeech

We now turn to the Librispeech benchmark (Panayotov et al., 2015) and provide ablation studies for other techniques that mitigate accuracy loss.

We train offline RNN-T models and evaluate them on the testsets of the Librispeech corpus. The Librispeech corpus contains utterances of audio books, with an average duration of 12.3 seconds, and an maximum duration of around 30 seconds (Moritz et al., 2021). We train the models with both the Librispeech training set which contains 960 hours of speech, and the Speechstew dataset (Chan et al., 2021) which contains 5140 hours of speech.

Funnel	ROPE	dev clean	dev other	test clean	test other	test avg.
None	N	1.95	4.51	**2.10**	4.59	3.35
None	Y	**1.88**	**4.30**	2.11	**4.54**	**3.33**
4x	N	2.03	4.68	2.19	4.98	3.59
4x	Y	2.14	4.40	2.23	4.72	3.48

Table 6: WER (%) on Librispeech with and without RoPE. Training is performed on the Librispeech training set with a batch size of 4096 utterances.

All the encoders we considered have the same number of trainable parameters (626.6M), with 24 layers, encoder dimension 1024, and convolution kernel size of 5. The attention module has a left context of 129 tokens and a right context of 128 tokens. For funnel encoders, we used a total of 4x reduction, done through a 2x reduction in layer 4 and another in layer 5. For encoders with upsampling, a 2x or 4x upsampling is done in the last layer. We focus on RNN-T models in this section which are equipped with a single-layer LSTM decoder, with an output vocabulary size of 1024.

6.1 Positional embeddings

In Table 6, we provide ablation studies for using RoPE in models with and without funnel pooling. RoPE provides a 2% relative improvement to averaged test WER in the case of 4x pooling, and we use it in experiments of following sections.

6.2 Training batch size

We study the effects of varying the global batch size during training, with results shown in Table 7. We found that larger batch sizes consistently led to significant WER gains for the funnel encoders while having a much smaller effect for non-funnel encoders. For example, with the Speechstew training set, doubling the training batch size from 2048 to 4096 improves the averaged test WER from 3.30% to 2.97% for the model with 4x time reduction.

With properly tuned batch sizes, the models with 160ms frame rate achieves only slightly worse accuracies than those with 40ms frame rate: averaged test WERs are 3.48% vs 3.31% for Librispeech training, and 2.97% vs 2.90% for Speechstew training. To our knowledge, these are the best WERs on Librispeech at the low frame rate of 160ms, under the same training setups. We would like to also mention that we could not push the frame rate to 320ms without clear WER degradation on Lib-

Funnel (FR)	batch size	dev clean	dev other	test clean	test other	test avg.
Librispeech training						
None (40ms)	512	1.91	4.32	**2.01**	4.56	**3.29**
	1024	1.97	**4.26**	2.07	4.59	3.33
	2048	**1.88**	4.30	2.11	**4.54**	3.33
	4096	1.93	4.32	2.04	4.58	3.31
4x (160ms)	512	2.01	4.84	2.21	5.14	3.68
	1024	1.97	4.61	2.23	4.78	3.51
	2048	2.14	4.40	2.23	4.72	3.48
	4096	2.08	4.64	2.22	4.73	3.48
Speechstew training						
None (40ms)	2048	**1.75**	4.02	1.93	3.97	2.95
	4096	1.76	**3.70**	**1.88**	**3.91**	**2.90**
4x (160ms)	2048	1.93	4.49	2.09	4.51	3.30
	4096	1.79	4.01	1.93	4.01	2.97

Table 7: Librispeech WERs (%) of funnel RNN-T with varying batch sizes.

rispeech (as we could for voice search queries), and it is future work to understand the limiting factor.

6.3 Upsampling

Given that funnel pooling early in the architecture does cause WER degradation, we investigate whether we can compensate the loss by upsampling the sequence at a later layer. We perform funnel upsampling at the last encoder layer (layer 23) to keep as much efficiency improvements as possible.

Since the downsampling factor in our encoder is 4, upsampling by a factor of 2 returns sequences half the length of encoder inputs; upsampling by a factor of 4 recovers the original length. We provide the results in Table 8. We observe a noticable improvement for Librispeech-trained models: the averaged test WER improves from 3.48% to 3.34% with 4x upsampling, which is close to the 3.31% achieved by the model without funnel pooling. On the other hand, upsampling appears to be less useful for Speechstew-trained models.

7 Conclusions

In this paper, we have investigated the use of time reduction by funnel pooling with massive E2E ASR models. Our experiments on a large-scale voice search task demonstrate its effectiveness with two major E2E ASR models, CTC and RNN-T, and show that for large models, funnel pooling significantly reduces inference and training costs without sacrificing much accuracy, achieving encoder output frame rate as low as 640ms. We have similar success on the Librispeech benchmark with a reduced frame rate of 160ms, given proper training hyperparameters and carefully designed architectures. We believe our techniques are generally useful for speech recognition and will further extend their usage to much longer audio.

Funnel size	dev clean	dev other	test clean	test other	test avg.
Librispeech training					
4x	2.14	4.40	2.23	4.72	3.48
4x-2x	**1.91**	4.46	**2.14**	4.69	3.42
4x-4x	1.99	**4.38**	2.17	**4.50**	**3.34**
Speechstew training					
4x	**1.79**	4.01	**1.93**	4.01	**2.97**
4x-2x	1.92	**3.86**	1.99	4.05	3.02
4x-4x	1.91	4.01	2.01	**3.94**	2.98

Table 8: Librispeech WER (%) of funnel RNN-T with upsampling. The notation "4x-2x" indicates 4 times downsampling combined with 2 times upsampling. Batch size is 4096.

8 Limitations

Our results cover both an important industrial use case and a widely used public benchmark, and demonstrate that time reduction shall be a generally useful technique and our simple algorithm sets the baseline for future development in this direction. Our preliminary results on YouTube transcription (not presented in the current paper) show similar successes, for RNN-T models with the frame rate of 320ms. However, we could not possibly cover all use cases of end-to-end ASR. Another limitation is that our study is performed exclusively on English datasets, and it is interesting to understand how the method behaves on other languages.

References

System architecture. https://cloud.google.com/tpu/docs/system-architecture-tpu-vm#tpu_v3. Accessed: 2024-03-01.

Andrei Andrusenko, Rauf Nasretdinov, and Aleksei Romanenko. 2023. Uconv-conformer: High reduction of input sequence length for end-to-end speech recognition. In *ICASSP*.

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

Maxime Burchi and Valentin Vielzeuf. 2021. Efficient conformer: Progressive downsampling and grouped attention for automatic speech recognition. In *ASRU*.

Xingyu Cai, David Qiu, Shaojin Ding, Dongseong Hwang, Weiran Wang Antoine Bruguier, Rohit Prabhavalkar, Tara Sainath, and Yanzhang He. 2023. Efficient cascaded streaming asr system via frame rate reduction. In *ASRU*.

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 *ICASSP*.

William Chan, Daniel Park, Chris Lee, Yu Zhang, Quoc Le, and Mohammad Norouzi. 2021. Speechstew: Simply mix all available speech recognition data to train one large neural network. *arXiv preprint arXiv:2104.02133*.

Jan Chorowski and Navdeep Jaitly. 2017. Towards better decoding and language model integration in sequence to sequence models. In *Interspeech*.

Zihang Dai, Guokun Lai, Yiming Yang, and Quoc Le. 2020. Funnel-Transformer: Filtering out sequential redundancy for efficient language processing. In *NeurIPS*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Shaojin Ding, Weiran Wang, Ding Zhao, Tara N. Sainath, Yanzhang He, Robert David, Rami Botros, Xin Wang, Rina Panigrahy, Qiao Liang, Dongseong Hwang, Ian McGraw, Rohit Prabhavalkar, and Trevor Strohman. 2022. A unified cascaded encoder ASR model for dynamic model sizes. In *Interspeech*.

Google. Artificial Intelligence at Google: Our Principles.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. In *ICML Workshop on Representation Learning*.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. In *ICML*.

Anmol Gulati, James Qin, Chung-Cheng Chiu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. In *Interspeech*.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv:1503.03535*.

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*.

Salah Hihi and Yoshua Bengio. 1996. Hierarchical recurrent neural networks for long-term dependencies. In *NeurIPS*.

W. Ronny Huang, Cal Peyser, Tara N. Sainath, Ruoming Pang, Trevor Strohman, and Shankar Kumar. 2022. Sentence-select: Large-scale language model data selection for rare-word speech recognition. In *Interspeech*.

Dongseong Hwang, Khe Chai Sim, Zhouyuan Huo, and Trevor Strohman. 2022. Pseudo label is better than human label. In *Interspeech*.

Naoyuki Kanda, Xugang Lu, and Hisashi Kawai. 2017. Minimum Bayes risk training of CTC acoustic models in maximum a posteriori based decoding framework. In *ICASSP*.

Shigeki Karita, Nelson Enrique Yalta Soplin, Shinji Watanabe, Marc Delcroix, Atsunori Ogawa, and Tomohiro Nakatani. 2019. Improving Transformer-based end-to-end speech recognition with connectionist temporal classification and language model integration. In *Interspeech*.

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

Jan Koutnik, Klaus Greff, Faustino Gomez, and Juergen Schmidhuber. 2014. A clockwork RNN. In *ICML*.

Bo Li, Ruoming Pang, Yu Zhang, Tara N. Sainath, Trevor Strohman, Parisa Haghani, Yun Zhu, Brian Farris, Neeraj Gaur, and Manasa Prasad. 2022. Massively multilingual ASR: A lifelong learning solution. In *ICASSP*.

Jinyu Li. 2022. Recent advances in end-to-end automatic speech recognition. *APSIPA Transactions on Signal and Information Processing*, 11(1).

Qiujia Li, Chao Zhang, and Philip C. Woodland. 2019. Integrating source-channel and attention-based sequence-to-sequence models for speech recognition. In *ASRU*.

Erik McDermott, Hasim Sak, and Ehsan Variani. 2019. A density ratio approach to language model fusion in end-to-end automatic speech recognition. In *ASRU*.

Zhong Meng, Sarangarajan Parthasarathy, Eric Sun, Yashesh Gaur, Naoyuki Kanda, Liang Lu, Xie Chen, Rui Zhao, Jinyu Li, and Yifan Gong. 2021. Internal language model estimation for domain-adaptive end-to-end speech recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 243–250. IEEE.

Yajie Miao, Mohammad Gowayyed, and Florian Metze. 2015. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In *ASRU*.

Yajie Miao, Jinyu Li, Yongqiang Wang, Shi-Xiong Zhang, and Yifan Gong. 2016. Simplifying long short-term memory acoustic models for fast training and decoding. In *ICASSP*.

Nelson Morgan and Hervé Bourlard. 1990. Continuous speech recognition using multilayer perceptrons with hidden Markov models. In *ICASSP*.

Niko Moritz, Takaaki Hori, and Jonathan Le Roux. 2021. Capturing multi-resolution context by dilated self-attention. In *ICASSP*.

Arun Narayanan, Rohit Prabhavalkar, Chung-Cheng Chiu, David Rybach, Tara Sainath, and Trevor Strohman. 2019. Recognizing long-form speech using streaming end-to-end models. In *ASRU*.

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. 2023. End-to-end speech recognition: A survey. *arXiv preprint arXiv:2303.03329*.

Rohit Prabhavalkar, Zhong Meng, Weiran Wang, Adam Stooke, Xingyu Cai, Yanzhang He, Arun Narayanan, Dongseong Hwang, Tara N Sainath, and Pedro J Moreno. 2024. Extreme encoder output frame rate reduction: Improving computational latencies of large end-to-end models. In *ICASSP*.

Vineel Pratap, Andros Tjandra, Bowen Shi, Paden Tomasello, Arun Babu, Sayani Kundu, Ali Elkahky, Zhaoheng Ni, Apoorv Vyas, Maryam Fazel-Zarandi, et al. 2023. Scaling speech technology to 1,000+ languages. *arXiv preprint arXiv:2305.13516*.

Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. 2023. Robust speech recognition via large-scale weak supervision. In *ICML*.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241.

Hasim Sak, Andrew W. Senior, Kanishka Rao, Ozan Irsoy, Alex Graves, Françoise Beaufays, and Johan Schalkwyk. 2015. Learning acoustic frame labeling for speech recognition with recurrent neural networks. In *ICASSP*.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *ICASSP*.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. *arXiv preprint arXiv:1804.04235*.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2023. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, page 127063.

Zhengkun Tian, Jiangyan Yi, Ye Bai, Jianhua Tao, Shuai Zhang, and Zhengqi Wen. 2021. Fsr: Accelerating the inference process of transducer-based models by applying fast-skip regularization. *arXiv preprint arXiv:2104.02882*.

Vincent Vanhoucke, Matthieu Devin, and Georg Heigold. 2013. Multiframe deep neural networks for acoustic modeling. In *ICASSP*.

Ehsan Variani, Erik McDermott, and Georg Heigold. 2015. A Gaussian mixture model layer jointly optimized with discriminative features within a deep neural network architecture. In *ICASSP*.

Ehsan Variani, David Rybach, Cyril Allauzen, and Michael Riley. 2020. Hybrid autoregressive transducer (HAT). In *ICASSP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Yongqiang Wang, Zhehuai Chen, Chengjian Zheng, Yu Zhang, Wei Han, and Parisa Haghani. 2023. Accelerating RNN-T training and inference using CTC guidance. In *ICASSP*.

Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, Adithya Renduchintala, and Tsubasa Ochiai. 2018. ESPnet: End-to-end speech processing toolkit. In *Interspeech*.

Yifan Yang, Xiaoyu Yang, Liyong Guo, Zengwei Yao, Wei Kang, Fangjun Kuang, Long Lin, Xie Chen, and Daniel Povey. 2023. Blank-regularized ctc for frame skipping in neural transducer. *arXiv preprint arXiv:2305.11558*.

Yu Zhang, Wei Han, James Qin, Yongqiang Wang, Ankur Bapna, Zhehuai Chen, Nanxin Chen, Bo Li, Vera Axelrod, Gary Wang, et al. 2023. Google USM: Scaling automatic speech recognition beyond 100 languages. *arXiv preprint arXiv:2303.01037*.

Yu Zhang, Daniel S Park, Wei Han, James Qin, Anmol Gulati, Joel Shor, Aren Jansen, Yuanzhong Xu, Yanping Huang, Shibo Wang, et al. 2022. Bigssl: Exploring the frontier of large-scale semi-supervised learning for automatic speech recognition. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1519–1532.