Raw Waveform Encoder with Multi-Scale Globally Attentive Locally Recurrent Networks for End-to-End Speech Recognition

Max W. Y. Lam¹, Jun Wang¹, Chao Weng¹, Dan Su¹, Dong Yu²

¹Tencent AI Lab, Shenzhen, China
²Tencent AI Lab, Bellevue WA, USA

{maxwylam, joinerwang, cweng, dansu, dyu}@tencent.com

Abstract

End-to-end speech recognition generally uses hand-engineered acoustic features as input and excludes the feature extraction module from its joint optimization. To extract learnable and adaptive features and mitigate information loss, we propose a new encoder that adopts globally attentive locally recurrent (GALR) networks and directly takes raw waveform as input. We observe improved ASR performance and robustness by applying GALR on different window lengths to aggregate fine-grain temporal information into multi-scale acoustic features. Experiments are conducted on a benchmark dataset AISHELL-2 and two large-scale Mandarin speech corpora of 5,000 hours and 21,000 hours. With faster speed and comparable model size, our proposed multi-scale GALR waveform encoder achieved consistent character error rate reductions (CERRs) from 7.9% to 28.1% relative over strong baselines, including Conformer and TDNN-Conformer. In particular, our approach demonstrated notable robustness than the traditional handcrafted features and outperformed the baseline MFCC-based TDNN-Conformer model by a 15.2% CERR on a music-mixed real-world speech test set.

Index Terms: end-to-end ASR, RNN-T, feature extraction, Conformer, TDNN-Conformer

1. Introduction

End-to-end automatic speech recognition (ASR) [1][5] allows a joint optimization of components for better performances. Amongst a variety of end-to-end methods, RNN-Transducer (RNN-T) [1] has recently achieved state-of-the-art (SOTA) performances [6][10]. An RNN-T comprises an encoder-decoder architecture, where the encoder is responsible for transforming a sequence of acoustic inputs into hidden vectors and the decoder utilizes previously emitted non-blank symbols to compute prediction vectors. A recent study [14] suggests that for an RNN-T optimizing encoder and learning proper representations for acoustic inputs are remarkably more vital than the decoder.

Standard ASR systems, including the SOTA RNN-T, traditionally use handcrafted acoustic features computed over segments of fixed duration, such as band-pass filtering of signals including log Mel-filterbank values (FBANK) [2] and Mel frequency cepstral coefficients (MFCC) [13]. A potential defect of such systems is being susceptible to information loss due to an intrinsic tradeoff between temporal and frequency resolution. For example, FBANK can hardly preserve phonetic information at the frame boundaries at a fixed time resolution, especially for fast speech, nor describe or separate subtle harmonics and formants at a fixed frequency resolution; Mel-filterbank operations are prone to discard local correlations within the signal [14].

An increasing number of studies [13][22] report empirical evidence that learning directly from the waveform domain can potentially outperform the analytically handcrafted features. In the majority of the previously proposed architectures, there are typically shallow or deep convolutional layers with either 1D convolutions [16][17][19][21] or 2D convolutions [14], designed to emulate band-pass filtering or spectrogram features. While convolutional networks are good at modeling local correlations as well as invariance to local translations, they can not capture long context with shallow layers. Even with very deep layers [14] at increased computational and memory cost, they are still limited in capturing dynamic long-range global context as CNN simply applies a global averaging over the entire sequence. Existing models using convolutional, recurrent neural networks (RNNs, e.g., LSTMs [23] and GRUs [24]), or self-attention networks (SAN) each has its shortcomings: SAN, on the other hand, are well suited to model longer global context but less capable to extract fine-scale local features.

Grasping the fact that RNN/CNN and SAN are complementary in modeling local detail dependencies and global long-range context, we propose to adopt an architecture called Globally Attentive Locally Recurrent (GALR) network. Its advantage has been demonstrated in the Cocktail-Party problem (i.e., speech separation tasks) in our previous studies [25][27]. By a large margin, it outperforms other advanced audio separation networks using a fully convolutional network [28] and using a dual-path RNN network [29]. Built on the previous success, the unique contributions of this paper are the followings:

a) We reveal the effectiveness of the GALR networks as building modules for a waveform encoder in end-to-end ASR. One challenge in directly taking raw waveform as input is that the sequence is densely sampled in time, and thus presents the dilemma between reducing the number of timesteps and throwing away relevant information for ASR. Our work elegantly solves the critical bottleneck of self-attention networks for modeling very long sequences and makes it memorably and computationally tractable. GALR is well suited to capture not only long-range global context but also local correlations and fine-grained patterns.

b) We propose a novel architecture that jointly learns GALRs at multiple scales on raw waveforms simultaneously. The combined multi-scale features remedy the tradeoff of temporal and frequency resolution, thus allowing the model to automatically select and combine the resolutions that could most efficiently represent the needed frequencies. Meanwhile, unlike other multi-scale approaches, which generally require very deep structure at extra cost [16], our method is computationally and memorably efficient.

c) We demonstrate that our waveform encoder can outperform by a large margin over statically hand-engineered features on large vocabulary end-to-end ASR tasks. In particular, we build our baseline models using the RNN-T architecture.
2. Multi-Scale GALR Waveform Encoder

We propose a raw waveform encoder with a multi-scale globally attentive locally recurrent (GALR) architecture. As shown in Figure 1, the waveform input goes through a series of $N$ modules, each composed of $B$ GALR blocks, to encode features at different time scales. The resultant features then concatenate and can substitute the conventional FBANK or MFCC features, and instead of being static, they are adaptive and jointly learned during the end-to-end ASR model training.

2.1. Waveform Encoder

Given a raw waveform input $x \in \mathbb{R}^T$, we first split the sequence into $L_n$ half-overlapping windows, denoted by $x_1, x_2, ..., x_{L_n}$, where $x_i \in \mathbb{R}^{M_n}$, $i = 1, ..., L_n$, and $M_n$ is the window length changing from small to large while the scale index $n$ goes from 1 to $N$. These windows are 50% overlapped, and the last one is padded with a zero vector to make $L_n = \lceil 2T/M_n \rceil$. Then, analogous to traditional short-time Fourier transform (STFT), $L_n$ frames are generated by non-linearly projecting $x_i$ onto a $D$-dimension feature space, forming a matrix $F_n \in \mathbb{R}^{D \times L_n}$:

$$F_n = \text{LN}(\text{ReLU}(\text{Conv1D}([x_1, ..., x_{L_n}]))) \quad (1)$$

where $\text{Conv1D}(\cdot) = U_n \ast [x_1, ..., x_{L_n}]$, $\ast$ denotes the 1D convolution operation, $U_n \in \mathbb{R}^{D \times M_n}$ is a learnable matrix of $D$ basis vectors, $\text{ReLU}(\cdot)$ is the rectified linear unit for ensuring the non-negativity, and $\text{LN}(\cdot)$ is the layer normalization method for stabilizing and accelerating the model training.

To inherit fine-to-coarse information, we then connect the output of the $(n - 1)$-th network to $F_n$:

$$\hat{F}_n = F_n + \text{AvgPool1D}(E_{n-1}) \quad (2)$$

where $E_{n-1} \in \mathbb{R}^{D \times L_{n-1}}$ is the output (detailed in Section 2.3) of the preceding GALR block at a finer scale (i.e., $L_{n-1} > L_n$), and for initialization, we define $E_0 = 0$ for the first block, and $\text{AvgPool1D}(\cdot)$ is an averaging pooling operation that interpolates the length $L_{n-1}$ of $E_{n-1}$ to match the length $L_n$ of $F_n$.

Before conducting GALR operations, we further split $F_n$ into $S_n$ half-overlapping chunks each of length $K_n$. The first and last chunks are padded with zeros to generate $S_n = \lfloor 2L_n/K_n \rfloor$ chunks. These chunks can be represented as a 3D tensor $Q_n \in \mathbb{R}^{D \times S_n \times K_n}$.

2.2. GALR Blocks

As shown in the middle of Figure 1, the building blocks in each waveform encoder, we adopt the same GALR architecture originally proposed for our speech separation tasks. In this work, we find that it can generalize surprisingly well for ASR. For integrity, we briefly describe the GALR architecture here. The left diagram in Figure 1 illustrates the connections within a GALR block. For each block, we denote the input as $Q_{n,b} \in \mathbb{R}^{D \times S_n \times K_n}$, where $n = 1, ..., N$ is the scale index, $b = 1, ..., B$ is the block index, and $Q_{n,1} = Q_n$. Each GALR block contains two primary layers—a local RNN layer and a global SAN layer, corresponding to the intra- and the inter-chunk processing, respectively.

2.2.1. Local RNN Layer

The local RNN layer with $D$ hidden nodes captures short-term dependencies within each chunk:

$$\tilde{L}_{n,b} = \text{Linear}(\text{RNN}(Q_{n,b}[s, :]), s = 1, ..., S_n) \quad (3)$$

where Linear($\cdot$) is a trainable linear mapping with a bias term enabled, RNN($\cdot$) is an RNN model, e.g., GRU or LSTM, and $Q_{n,b}[s, :] \in \mathbb{R}^{D \times K_n}$ refers to the local sequence within the $s$-th chunk, each of length $K_n$. The output of the local RNN then goes through LN($\cdot$) followed by a residual connection:

$$\hat{L}_{n,b} = \text{LN}(\tilde{L}_{n,b}) + Q_{n,b} \quad (4)$$

2.2.2. Global SAN Layer

Then, we transpose the 3D tensor $\hat{L}_{n,b} \in \mathbb{R}^{D \times S_n \times K_n}$ to process inter-chunk sequences, each of length $S_n$. A multi-head
SAN layer\textsuperscript{[32]} is used to capture the global dependencies:
\begin{equation}
G_{n,b} = \text{UpSmpl} \left( \text{SAN} \left( \text{DownSmpl} \left( \mathcal{L}_{n,b} \right) \right) \right),
\end{equation}
where SAN(\cdot), UpSmpl(\cdot), and DownSmpl(\cdot) are the self-attentive, upsampling, and downsampling operations, respectively. Readers are referred to\textsuperscript{[26]} for more details of these operations, which we omit here for simplicity.

Afterward, the output of the \textit{b}-th GALR block is passed forward with a residual connection as the input for the \textit{(b + 1)-th} GALR block as follows:
\begin{equation}
\mathcal{Q}_{n,b+1} = \text{LN}(G_{n,b}) + \mathcal{L}_{n,b},
\end{equation}

\subsection{2.3. Merging Chunks}
As shown in the middle diagram of Figure\textsuperscript{1} after processing through the \textit{B} GALR blocks, we merge the chunks to transform the 3D tensor back to a sequence of feature vectors for the downstream ASR task. Inspired by the mask estimation in\textsuperscript{[26]}, merging operations entail a non-linear mapping, without which we observed a loss degradation in our ablation study. In the merging operations, we use a Swish-gated 2D convolutional layer for the non-linear mapping:
\begin{equation}
F_n = \text{Conv2D} \left( \text{Swish} \left( \mathcal{Q}_{n,b} \right) \right),
\end{equation}
where Conv2D(\cdot) is a point-wise 2D convolutional layer with a \textit{1} \times \textit{1} kernel, and Swish(\cdot) is the element-wise Swish activation function\textsuperscript{[33]}. Afterward, we merge the \textit{S}_n chunks using OverlapAdd\textsuperscript{[20]} to transform the 3D tensor \( F_n \in \mathbb{R}^{D \times S_n \times N_n} \) back to a feature matrix:
\begin{equation}
E_n = \text{OverlapAdd} \left( F_n \right),
\end{equation}
which is also passed to the next GALR encoder to establish the fine-to-coarse information flow, as indicated in Eq.\textsuperscript{2}.

\subsection{2.4. Multi-scale Features}
Current SOTA ASR systems (e.g., RNN-T\textsuperscript{[10]} typically use analytical acoustic features computed over a fixed window size of 25ms and a stride length of 10ms. Consequently, the time and frequency resolution have been fixed a priori rather than being adaptive or learnable using the training data and the end-to-end model learning. These irreversible operations inevitably cause information loss at a landing stage of scaffold for ASR.

In the light of the successful fine-grained modeling of GALR networks, we are able to flexibly and effectively handle frames over fine-to-coarse windows (i.e. using 0.2ms to 2ms windows for the Cocktail-Party problem\textsuperscript{[25]}). The fine-to-coarse range could be larger in ASR. For example, striking \textit{80-dimension MFCC} by about 0.1% absolute reductions on average character error rates (CERs)\textsuperscript{[30]}, Conformer and TDNN-Conformer, were taken as our reference systems. They were built on two types of input representations: 1) Traditional static features, to select the strongest static features, we empirically examined the performances on the largest training set (21khrs) with regard to MFCC and FBANK features both following the recipe in Kaldi\textsuperscript{[50]} (PyKaldi\textsuperscript{[37]} was used to read the features extracted in Kaldi). On the \textit{Spon} test set, 40-dimension high-resolution MFCC outperformed 80-dimension FBANK by about 0.1% absolute reduction on average character error rates (CERs). A similar observation was also made in\textsuperscript{[35]}. Consequently, we selected MFCC for our baseline systems using handcrafted features. 2) End-to-end learnable features, for which we also trained baselines using the single-scale and multi-scale convolution-based approach\textsuperscript{[10]} to directly learn filterbanks from the raw waveform. All systems were trained following the RNN-T encoder model, e.g., Conformer\textsuperscript{[10]}.

3. Evaluation and Analysis
3.1. Experimental Setup
3.1.1. Data Preparation
Our models were trained on three datasets at 16kHz sampling rate: the 1,000-hour AISHELL-2 benchmark\textsuperscript{[34]}, a \textit{4} \times 5000-hour (5khrs) and a \textit{2} \times 21,000-hour (21khrs) Mandarin speech datasets\textsuperscript{[35]}. The models were evaluated on a variety of test sets, covering 1) \textit{Mic}: the benchmark AISHELL-2 test set, 2) \textit{Read}: 1-hour read speech (Read), 3) \textit{Spon}: 2-hour spontaneous speech, and 4) \textit{Music}: 2-hour speech with background music collected from a real-world video-sharing platform.

3.1.2. Model Setup and Training Details
Two cutting-edge encoder architectures that gave SOTA performances\textsuperscript{[30]}, Conformer and TDNN-Conformer, were taken as our reference systems. They were built on two types of input representations: 1) Traditional static features. To select the strongest static features, we empirically examined the performances on the largest training set (21khrs) with regard to MFCC and FBANK features both following the recipe in Kaldi\textsuperscript{[50]} (PyKaldi\textsuperscript{[37]} was used to read the features extracted in Kaldi). On the \textit{Spon} test set, 40-dimension high-resolution MFCC outperformed 80-dimension FBANK by about 0.1% absolute reduction on average character error rates (CERs). A similar observation was also made in\textsuperscript{[35]}. Consequently, we selected MFCC for our baseline systems using handcrafted features. 2) End-to-end learnable features, for which we also trained baselines using the single-scale and multi-scale convolution-based approach\textsuperscript{[10]} to directly learn filterbanks from the raw waveform. All systems were trained following the RNN-T, which was implemented in the PIKA\textsuperscript{[38]} library and conducted on 16 NVIDIA Tesla V100 GPU devices. For simplicity, we refer our readers to\textsuperscript{[38]} for the detailed RNN-T setups.

\textsuperscript{1}http://github.com/tencent-ailab/pika
For the Conformer models, we reproduced the small, medium, and large Conformer encoders, with setups as described in [10], and denoted them as Conf-S, Conf-M, and Conf-L. Moreover, the TDNN-Conformer model (denoted as TDNN-Conf) has shown better performances in [30] than TDNN-Transformer [8] and Conformer [10]. Our evaluation also showed a consistent conclusion.

For the multi-scale GALR networks, we set \( N = 3, B = 1, \{ M_1, M_2, M_3 \} = \{ 100, 200, 400 \}, \{ K_1, K_2, K_3 \} = \{ 48, 24, 12 \}, \{ C_1, C_2, C_3 \} = \{ 8, 4, 2 \} \). The GALR feature dimension (\( D \)) was set to 128 for AISHELL-2 and to 256 for 5khrs and 21khrs datasets, respectively. For a fair comparison, all models had a comparable number of parameters to its counterpart. We removed the MFCC feature extraction step and the first encoder layers, which were the first 8/4/3 conformer layers for the S/M/L Conformer models and 3 TDNN layers for the TDNN-Conformer models, and replaced them with the multi-scale GALR waveform encoder, while the overall sizes matched the respective baseline models.

3.2. Results and Discussion

3.2.1. Performances on Benchmark AISHELL-2

| Encoder | #Params. (M) | CERs (%) |
|---------|-------------|----------|
| Conf-S [10] | 8.7 | 8.3 | 17.1 | 12.3 |
| Conf-M [10] | 27.2 | 26.9 | 15.2 | 11.9 |
| Conf-L [10] | 114 | 114 | 13.3 | 11.7 |
| TDNN-Conf [30] | 102 | 99.0 | 12.7 | 11.7 |

First and foremost, we trained various types of RNN-T encoders on the benchmark AISHELL-2 [34]. As shown in Table 1 across different configurations of the SOTA Conformer encoders, applying the multi-scale GALR waveform encoder in place of the handcrafted features consistently obtained CER reductions, from 7.9% to 28.1% relatively. The results also indicated that the traditional TDNN-Conformer outperformed the traditional Conformers, echoing the report by [30], therefore, we selected the TDNN-Conformer encoder as the reference models for the following evaluations in larger scales.

It was also worth mentioning that, to focus on the comparisons concerning different encoder architectures, all the ASR performances through out this paper were obtained from a pure RNN-T system, without using any additional resource or technique (e.g., NNLM, MBR [8] or LAS rescoring [9]), which intrinsically remedy the tradeoff of temporal and frequency resolution, thus mitigating the information loss for end-to-end ASR. Our evaluations demonstrate the effectiveness of directly learning a waveform encoder based on GALR, which was a larger and more realistic industrial dataset. We also used a variety of test sets with a broader diversity such as different speaker speaking styles (i.e., read and spontaneous) and scenarios (with music interference in a video-sharing platform). As shown in Table 2, the model using the proposed multi-scale GALR encoder obtained consistently lower CERs than the traditional TDNN-Conformer. The result showed that the advantage of our approach could sustain in large-scale data and generalize well to complex scenarios. Especially, the CER for Music was reduced by a large margin of 15.2% relatively. Please note that our baseline model was fairly strong using TDNN, which is successful in modeling context information. This demonstrated the powerful capability of the proposed GALR to capture global dynamic context, such as in the unseen complex scenarios with non-stationary music interference as illustrated here. This evaluation empirically proves the robustness of withstanding interference by directly learning a waveform encoder based on GALR. Meanwhile, our approach sped up the processing notably from 239 to 382 characters per second.

4. Conclusions

We propose a novel architecture that jointly learns GALR at multiple different scales on raw waveforms simultaneously, which intrinsically remedy the tradeoff of temporal and frequency resolution, thus mitigating the information loss for end-to-end ASR. Our evaluations demonstrate the effectiveness of directly learning waveform encoder based on GALR, as opposed to building end-to-end ASR on top of traditional features. Moreover, our method is computationally and memory efficient. Future work includes exploring this novel architecture in a cocktail party scenario to solve the speech separation and ASR tasks simultaneously in an end-to-end solution.
