UNIVERSAL ASR: UNIFY AND IMPROVE STREAMING ASR WITH FULL-CONTEXT MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Streaming automatic speech recognition (ASR) aims to emit each hypothesized word as quickly and accurately as possible, while *full-context* ASR waits for the completion of a full speech utterance before emitting completed hypotheses. In this work, we propose a unified framework, *Universal ASR*, to train a single end-to-end ASR model with shared weights for both streaming and full-context speech recognition. We show that the latency and accuracy of streaming ASR significantly benefit from *weight sharing* and *joint training* of full-context ASR, especially with *inplace knowledge distillation*. The Universal ASR framework can be applied to recent state-of-the-art convolution-based and transformer-based ASR networks. We present extensive experiments with two state-of-the-art ASR networks, ContextNet and Conformer, on two datasets, a widely used public dataset LibriSpeech and an internal large-scale dataset MultiDomain. Experiments and ablation studies demonstrate that Universal ASR not only simplifies the workflow of training and deploying streaming and full-context ASR models, but also significantly improves both emission latency and recognition accuracy of streaming ASR. With Universal ASR, we achieve new state-of-the-art streaming ASR results on both LibriSpeech and MultiDomain in terms of accuracy and latency.

1 INTRODUCTION

"Ok Google. Hey Siri. Hi Alexa." have featured a massive boom of smart speakers in recent years, unveiling a trend towards ubiquitous and ambient Artificial Intelligence (AI) for better daily lives. As the communication bridge between human and machine, low-latency streaming ASR (*a.k.a.*, online ASR) is of central importance, whose goal is to emit each hypothesized word as quickly and accurately as possible on the fly as they are spoken. On the other hand, there are some scenarios where full-context ASR (*a.k.a.*, offline ASR) is sufficient, for example, offline video captioning on video-sharing platforms. While low-latency streaming ASR is generally preferred in most of the speech recognition scenarios, it often has worse prediction accuracy as measured in Word Error Rate (WER), due to the lack of future context compared with full-context ASR. Improving both WER and emission latency has been shown to be highly challenging (He et al., 2019; Li et al., 2020a; Sainath et al., 2020) in streaming ASR systems.

Since the acoustic, pronunciation, and language model (AM, PM, and LM) of a conventional ASR system have been evolved into a single end-to-end (E2E) all-neural network, modern streaming and full-context ASR models share most of the neural architectures and training recipes in common, such as, Mel-spectrogram inputs, data augmentations, neural network meta-architectures, training objectives, model regularization techniques and decoding methods. The most significant difference is that *streaming ASR encoders are auto-regressive models*, with the prediction of the current timestep conditioned on previous ones (no future context is permitted). It can be achieved with uni-directional LSTMs, causal convolution and left-context attention layers in streaming ASR encoders (Chiu & Raffel, 2018; Fan et al., 2018; Han et al., 2020; Gulati et al., 2020; Huang et al., 2020; Moritz et al., 2020; Miao et al., 2020; Tsunoo et al., 2020; Zhang et al., 2020; Yeh et al., 2019). Recurrent Neural Network Transducers (RNN-T) (Graves, 2012) are commonly used as the decoder in both streaming and full-context models, which predicts the token of the current input frame based on all previous tokens using uni-directional recurrent layers. Figure 1 illustrates a simplified example of the similarity and difference between streaming and full-context ASR models with E2E neural networks.

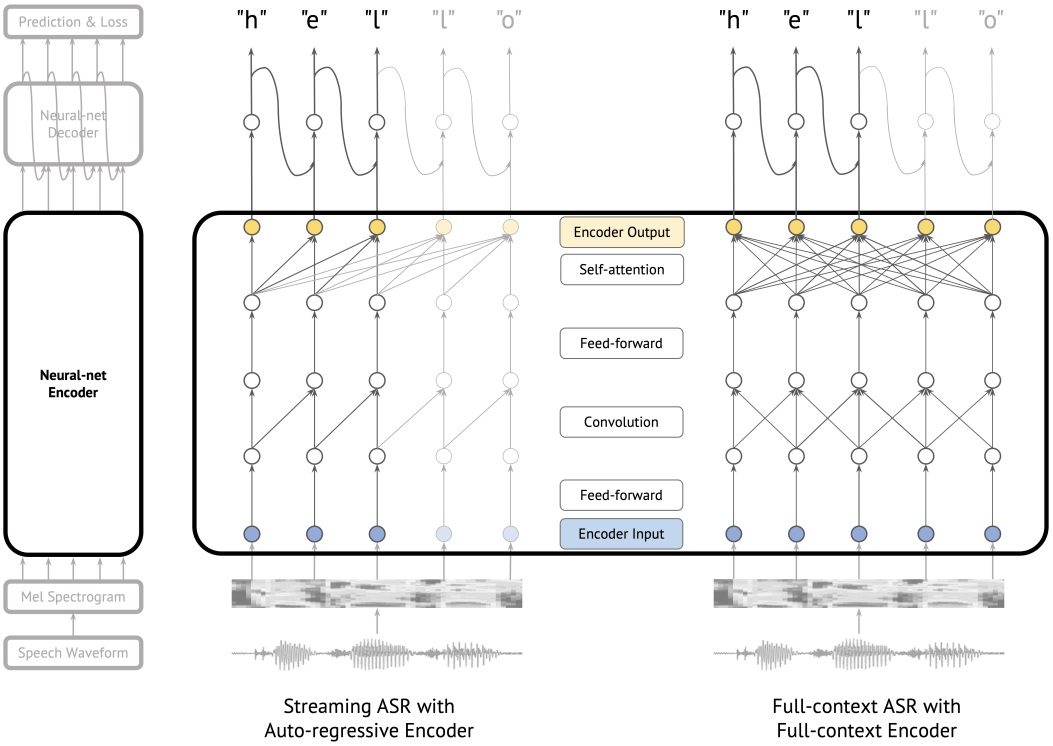

Figure 1: A simplified illustration of the similarity and difference between Streaming ASR and Full-context ASR networks. Modern end-to-end streaming and full-context ASR models share most of the neural architectures and training recipes in common, with the most significant difference in the **ASR encoder (highlighted)**. Streaming ASR encoders are auto-regressive models, with each prediction of the current timestep conditioned on previous ones (no future context). We show examples of feed-forward layer, convolution layer and self-attention layer in the encoder of streaming and full-context ASR respectively. With Universal ASR, we unify them without parameters overhead.

Albeit the similarities, streaming and full-context ASR models are usually developed, trained, and deployed separately. In this work, we propose *Universal ASR*, a framework to unify streaming and full-context speech recognition networks with shared weights. Universal ASR comes with many immediate benefits, including reduced model download and storage on devices and simplified development and deployment workflows. To accomplish this goal, we first introduce *Dual-mode Encoders*, which can run in both streaming mode and full-context mode. Dual-mode encoders are designed to reuse the same set of model weights for both modes with zero or near-zero parameters overhead. We propose the design principles of a dual-mode encoder and show examples on how to design dual-mode convolution, dual-mode pooling, and dual-mode attention layers. We also investigate into different training algorithms for Universal ASR, specifically, randomly sampled training and joint training. We show that joint training significantly outperforms randomly sampled training in terms of model quality and training stability. Moreover, motivated by *Inplace Knowledge Distillation* (Yu & Huang, 2019b) in which a large model is used to supervise a small model, we propose to *distill knowledge from the full-context mode (teacher) into the streaming mode (student) on the fly* within the same Universal ASR model, by encouraging consistency of the predicted token probabilities.

We demonstrate that the emission latency and prediction accuracy of streaming ASR significantly benefit from *weight sharing* and *joint training* of its full-context mode, especially with *inplace knowledge distillation*. We present extensive experiments with two state-of-the-art ASR networks, convolution-based ContextNet (Han et al., 2020) and conv-transformer hybrid Conformer (Gulati et al., 2020), on two datasets, a widely used public dataset LibriSpeech (Panayotov et al., 2015) (970 hours of English reading speech) and an internal large-scale dataset MultiDomain (Narayanan et al., 2018) (413,000 hours speech of a mixture across multiple domains including Voice Search, Farfield Speech, YouTube and Meetings). For each proposed technique, we also present ablation study and analysis to demonstrate and understand the effectiveness. With Universal ASR, we achieve new

state-of-the-art streaming ASR results on both LibriSpeech and MultiDomain in terms of accuracy and latency.

2 Related Work

Streaming ASR Networks. There has been a growing interest in building streaming ASR systems based on E2E Recurrent Neural Network Transducers (RNN-T) (Graves, 2012). Compared with sequence-to-sequence Listen, Attend and Spell (LAS) models (Chan et al., 2016), RNN-T models are naturally *streamable* and have shown great potentials for low-latency streaming ASR (Chang et al., 2019; He et al., 2019; Tsunoo et al., 2019; Sainath et al., 2019; Shen et al., 2019; Li et al., 2020a;b; Sainath et al., 2020; Huang et al., 2020; Moritz et al., 2020). In this work, we mainly focus on RNN-T based models. He et al. specifically studied how to optimize the RNN-T streaming ASR model for mobile devices, and proposed a bag of techniques including using layer normalization and large batch size to stabilize training; using word-piece targets (Wu et al., 2016); using a time-reduction layer to speed up training and inference; quantizing network parameters to reduce memory footprint and speed up computation; applying shallow-fusion to bias towards user-specific context. To support streaming modeling in E2E ASR models, various efforts have also been made by modifying attention-based models such as monotonic attention (Raffel et al., 2017; Chiu & Raffel, 2017; Fan et al., 2018; Arivazhagan et al., 2019), GMM attention (Graves, 2013; Chiu et al., 2019), triggered attention (TA) (Moritz et al., 2019), Scout Network (Wang et al., 2020), and approaches that segment encoder output into non-overlapping chunks (Jaitly et al., 2016; Tsunoo et al., 2020). Tsunoo et al. also applied knowledge distillation from the non-streaming model to the streaming model, but their streaming and non-streaming models do not share weights and are trained separately.

To improve the latency of RNN-T streaming models, Li et al. investigated additional early and late penalties on Endpointer prediction (Chang et al., 2019) to reduce the emission latency, and employed the minimum word error rate (MWER) training (Prabhavalkar et al., 2018) to remedy accuracy degradation. Sainath et al. further proposed to improve quality by using two-pass models (Sainath et al., 2019), *i.e.*, a second-pass LAS-based rescore model on top of the hypotheses from first-pass RNN-T streaming output. More recently, Li et al. proposed parallel rescoring by replacing LSTMs with Transformers (Vaswani et al., 2017) in rescoring models. Chang et al. further proposed Prefetching to reduce system latency by submitting partial recognition results for subsequent processing such as obtaining assistant server responses or second-pass rescoring before the recognition result is finalized. Unlike these approaches, our work explores the unification of streaming and full-context ASR networks, thus can be generally applied as an add-on technique without requiring extra runtime support during inference.

Weight Sharing for Multi-tasking. Sharing model weights of a deep neural network for multiple tasks has been widely explored in the literature to reduce overall model sizes. In the broadest sense, tasks can refer to different objectives or same objective but different settings, ranging from natural language processing and speech recognition to computer vision and reinforcement learning. In speech recognition, Kannan et al. employed a single ASR network for multilingual ASR, and showed accuracy improvements over monolingual ASR systems. Wu et al. proposed dynamic sparsity neural networks (DSNN) for speech recognition on mobile devices with resource constraints. A single trained DSNN (Wu et al., 2020) can transform into multiple networks of different sparsities for adaptive inference in real-time. Chang et al. trained a single RNN-T model with LSTMs (Hochreiter & Schmidhuber, 1997) for Joint Endpointing (*i.e.*, predicting both recognition tokens and the end of an utterance transcription) in streaming ASR systems.

Another related research work in Computer Vision is Slimmable Neural Networks (Yu et al., 2018; Yu & Huang, 2019a;b; Yu et al., 2020). Yu et al. proposed an approach to train a single neural network running at different widths, permitting instant and adaptive accuracy efficiency trade-offs at runtime. We also adapt the training rules introduced in slimmable networks, that is, using independent normalization layers for different sub-networks (tasks) as conditional parameters and using the prediction of teacher network to supervise student network as inplace distillation. Unlike slimmable networks in which a large model is used to supervise a small model, we propose to distill the knowledge from full-context mode (teacher) into streaming mode (student) on the fly within the same Universal ASR model.

3 UNIVERSAL ASR

Most neural sequence transduction networks for ASR have an encoder-decoder structure (Graves, 2012; Sainath et al., 2020; He et al., 2019; Li et al., 2020a), as shown in Figure 1. Without loss of generality, here we discuss how to design Universal ASR networks under the most commonly used RNN-T model (Graves, 2012). In RNN-T models, we first extract mel-spectrogram feature from input speech waveform. The Mel-spectrogram feature is then fed into a neural-net encoder, which usually consists of feed-forward layers, RNN/LSTM layers, convolution layers, attention layers, pooling (time-reduction) layers, and residual or dense connections. In neural-net encoders, streaming ASR model requires all components to be auto-regressive, whereas full-context ASR model has no such requirement. The ASR decoder then predicts the token of current frame based on the output from the encoder and previous predicted tokens (inference) or target tokens (training with teacher forcing (Williams & Zipser, 1989)). The decoder is commonly an auto-regressive model in both streaming and full-context ASR models, thus is *fully shared* in Universal ASR. The prediction from decoder is finally used either in decoding algorithm during inference (*e.g.*, beam search) or learning algorithm during training (*e.g.*, RNN-T loss).

As discussed above and shown in Figure 1, it becomes clear that the major difference between streaming and full-context ASR models is in the *neural-net encoder*. In the following, we will first discuss the design principles of *dual-mode encoder* to support both streaming and full-context ASR. We provide examples including dual-mode convolution, dual-mode average pooling, and dual-mode attention layers, which are widely used in the state-of-the-art ASR networks ContextNet (Han et al., 2020) and Conformer (Gulati et al., 2020). We will then discuss the training algorithm of Universal ASR networks including joint training and inplace knowledge distillation.

3.1 DUAL-MODE ENCODER

Unifying streaming and full-context ASR models requires two design principles of *Dual-mode Encoder*:

1. Each layer in a dual-mode encoder should be either dual-mode or streaming (*a.k.a.*, causal). Since streaming encoder has to be auto-regressive which prohibits any future context, any full-context (*a.k.a.*, non-causal) layer violates this constraint.
2. The design of a dual-mode layer should not introduce significant amount of additional parameters, compared with its streaming model. We aim at supporting full-context ASR on top of the streaming model with near-zero parameters overhead.

We show examples below by applying the above two design principles to ContextNet (Han et al., 2020) and Conformer (Gulati et al., 2020), in which the encoders are composed of pointwise operators (feed-forward net, residual connections, activation layers, striding, dropout, *etc.*), convolution, average pooling, self-attention and normalization layers.

Pointwise operators are naturally dual-mode layers. Neural network layers that connect input and output neurons within each timestep (no across-connections among different timesteps) are often referred as pointwise operators (Chollet, 2017), including feed-forward layers (*a.k.a.*, fully-connected layers or 1×1 convolution layers), activation layers (*e.g.*, ReLU, Swish (Ramachandran et al., 2017)), residual and dense connections (He et al., 2016; Huang et al., 2017), striding layers, dropout layers (Srivastava et al., 2014) and element-wise multiplications. As there is no information propagation through time, pointwise operators are naturally dual-mode layers and can be directly used in Universal ASR encoders.

Dual-mode Convolution. Convolution layers, however, convolve feature across its neighbor timesteps within a fixed window (*e.g.*, kernel size is 3, 5, or larger), and has been widely used in sequence modeling (Gehring et al., 2017; Han et al., 2020; Gulati et al., 2020). In conv-based streaming ASR models, causal convolution layers (Oord et al., 2016) are used where the convolution window is biased to the left (self-included). As shown in Figure 2 on the left, to support both streaming and full-context modes with shared weights, we first construct a normal symmetric convolution of kernel size k which will be applied in full-context mode. Then we mimic the causal convolution of kernel size $(k+1)/2$ by constructing a Boolean mask and multiplying with the full-

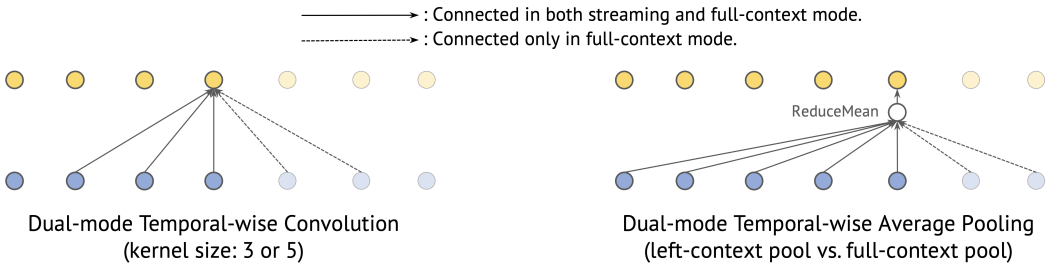

Figure 2: Dual-mode convolution and average pooling layer for Universal ASR.

context convolution kernel before applying the actual convolution of streaming mode in Universal ASR encoders.

The design of dual-mode convolution introduces $(k-1)/2$ additional parameters to support full-context convolution (k) compared with streaming convolution ($(k+1)/2$). However, we note that in convolution-based models, these temporal-wise convolution layers only take a tiny amount of total model size and most of the weights are on 1×1 convolution layers which are fully shared pointwise operators. For example, in ContextNet (Han et al., 2020), temporal-wise convolution has less than 1% of total model size, thus parameters overhead is negligible.

Dual-mode Average Pooling. Squeeze-and-excitation (Hu et al., 2018) (SE) modules are used in ContextNet to enhance the global context encoding. Each SE module is a sequential stack of average pooling (through time) layer, feed-forward layer, activation layer, another feed-forward layer and elementwise multiplication. To support both modes, dual-mode average pooling layer is used as shown in Figure 2 on the right. Dual-mode average pooling layer is parameter-free thus does not introduce additional model parameters. It also trains in parallel in streaming mode, easily implemented with "cumsum" function in both TensorFlow and PyTorch.

Dual-mode Self-attention. Self-attention (*a.k.a.* intra-attention) is an attention mechanism weighting different positions of a single sequence in order to compute a representation of the same sequence. It is heavily used in Conformer (Gulati et al., 2020) ASR networks. The attention layer itself is parameter-free (projection layers before attention are fully shared), and is composed of matrix multiplication of the key and the query, followed by softmax over keys, before another matrix multiplication with the value. As shown in Figure 3, in dual-mode attention layer, the softmax is performed on the left context only in streaming mode (rectangle with solid line), compared with the full-context mode (rectangle with dash line). We find this simple form of dual-mode self-attention works well in practice.

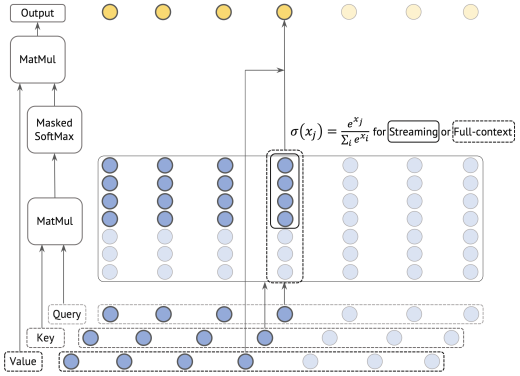

Figure 3: Dual-mode self-attention layer.

Dual-mode Normalization. Moreover, following Yu et al. (2018), we also find the normalization statistics like means and variances are different in streaming and full-context modes. Thus, for normalization layers including BatchNorm (Ioffe & Szegedy, 2015) and LayerNorm (Ba et al., 2016) in Universal ContextNet and Universal Conformer, we instantiate two separate norm layers dedicated to streaming and full-context mode respectively.

3.2 Training Universal ASR Networks

The training algorithm of Universal ASR networks is outlined in Algorithm 1. In this section, we discuss two important training techniques: joint training and inplace knowledge distillation.

Joint Training. To train Universal ASR networks, given a batch of data in each training iteration, we can either randomly sample one from two modes to train, or train both modes and aggregate

Algorithm 1 Pseudocode of training Universal ASR networks.

```
# Requires: data_loader; context manager with support of mode switching by network.mode();
    universal_network with support of running both modes under context manager;

for x, y in data_loader: # Load a minibatch of speech input x and text label y.
   with universal_network.mode('fullcontext'): # Switch context to 'fullcontext' mode.
      # Compute full-context prediction given speech input x and text label y.
      fullcontext_pred = universal_network.forward_encoder_decoder(x, y)
      # Compute RNN-T loss of full-context mode.
      fullcontext_loss = rnnt_loss(fullcontext_pred, y)

   with universal_network.mode('streaming'): # Switch context to 'streaming' mode.
      # Compute streaming prediction given speech input x and text label y.
      streaming_pred = universal_network.forward_encoder_decoder(x, y)
      # Compute RNN-T loss of streaming mode.
      streaming_loss = rnnt_loss(streaming_pred, y)

   # Add inplace knowledge distillation loss (full-context prediction as teacher).
   distill_loss = inplace_distill_loss(streaming_pred, stop_gradient(fullcontext_pred))

   # Compute total loss as a sum of full-context, streaming and distillation losses.
   loss = fullcontext_loss + streaming_loss + distill_loss
   loss.backward() # Update weights.
```

their losses. In the former approach, referred as *randomly sampled training*, we can control the importance of streaming and full-context modes by setting different sampling probabilities during training. In the latter approach, referred as *joint training*, importance can also be controlled by assigning different loss weights to balance streaming and full-context modes. Empirically we find joint training leads to better model qualities overall thus is adopted in all of our experiments. We will show an ablation study comparing randomly sampled training and joint training. In all of our experiments, we treat streaming and full-context mode to be equally important by assigning equal importance during training.

Inplace Knowledge Distillation. Additionally we propose to distill knowledge from the full-context mode (teacher) into the streaming mode (student) on the fly within the same Universal ASR model, by encouraging consistency of the predicted token probabilities. Since in each iteration we always compute predictions of both modes, the teacher prediction comes for free (no additional computation or memory cost), as shown in Algorithm 1. We use the efficient knowledge distillation introduced by Panchapagesan et al., which is based on the KL-divergence between full-context and streaming over the probability of three parts: P_{label}, P_{blank} and $1 - P_{label} - P_{blank}$. We note that the prediction of full-context mode (teacher) usually has lower latency (since it has no incentive to delay its output), thus we can control the target emission latency of streaming mode (student) by shifting the prediction of full-context mode, before applying distillation loss. We do a small-scale hyper-parameter sweep from -2 to 2 frames to shift for ContextNet and Conformer in our experiments.

4 EXPERIMENTS

4.1 MAIN RESULT

Measuring Latency. Latency measurement is itself challenging for streaming ASR systems. Motivated by Prefetching (Chang et al., 2020) technique, we measure latency as the difference of two timestamps: 1) when the last token is emitted in the finalized recognition result; 2) the end of the speech when a user finishes speaking. We find this is especially descriptive of user experience in real-world ASR applications like Voice Search. ASR models that capture stronger contexts can emit the full hypothesis even before they are spoken, leading to a **negative latency**. Moreover, instead of naively averaging latency over all utterances, we report both median and 90th percentile of all utterances in test set, denoted as **Latency@50** and **Latency@90**, to better characterize latency by excluding outlier utterances. To evaluate the model quality, we report WER only for full-context models and both WER and latency for streaming models (full-context latency is meaningless).

Datasets. We conduct our experiments on two datasets: a public widely used dataset LibriSpeech (Panayotov et al., 2015) (1,000 hours of English reading speech) and an internal large-scale dataset MultiDomain (163,000 hours speech, 77 million utterances of a mixture across multiple domains including Voice Search, YouTube, and Meetings). Table 1 summarizes the information

and statistics of two datasets. For LibriSpeech, we report our evaluation results on TestClean and TestOther (noisy) sets and compare with other published baselines. For MultiDomain, we report our evaluation results on Voice Search test set and compare with our reproduced baselines. For fair comparisons, on each dataset we train and report our models and baselines with the same settings (number of training iterations, hyper-parameters, optimizer, regularization, *etc*.).

Table 1: Summary of datasets we used in our experiments.

Dataset Name	**# Hours**	**# Utterances**	**Speech Domain**
LibriSpeech (Panayotov et al., 2015)	~ 970	$\sim 281,000$	Single domain of English reading speech.
MultiDomain (Narayanan et al., 2018)	$\sim 413,000$	$\sim 287,000,000$	Multiple domains including: Voice Search, Farfield Speech, YouTube and Meetings.

ASR Networks. We use two recent state-of-the-art ASR networks to demonstrate the effectiveness of our proposed methods, ContextNet (Han et al., 2020) and Conformer (Gulati et al., 2020). The encoder of ContextNet is based on depthwise-separable convolution (Chollet, 2017) and squeeze-and-excitation modules (Hu et al., 2018). In depthwise-separable convolution of Universal ContextNet, the weights of 1×1 convolutions are fully shared between streaming and full-context mode, whereas for temporal-wise convolution we follow the design of Dual-mode Convolution proposed in Section 3.1. Note that in ContextNet, temporal-wise convolutions only take less than 1% of the model size thus the parameters overhead of full-context mode is negligible compared with steaming mode. In squeeze-and-excitation modules, we use dual-mode average pooling layers (Section 3.1) to support both streaming and full-context mode without additional parameters.

Conformer (Gulati et al., 2020) combines convolution and transformer to model both local and global dependencies of speech sequences in a parameter-efficient way. In Universal Conformer, we replace all convolution and transformer layers with their dual-mode correspondents (Section 3.1). Moreover, for normalization layers including BatchNorm (Ioffe & Szegedy, 2015) and LayerNorm (Ba et al., 2016) in Universal ContextNet and Universal Conformer, we instantiate two separate norm layers for streaming and full-context mode respectively.

Table 2: Summary of our results on MultiDomain dataset (Narayanan et al., 2018). We report WER on Voice Search test set. Compared with standalone ContextNet and Conformer models, Universal ASR models have slightly higher accuracy and much better streaming latency. ASR models that capture stronger contexts can emit the full hypothesis even slightly before they are spoken, leading to a *negative latency*.

Method	**Mode**	**# Params (M)**	**VS Test**	**Latency@50**	**Latency@90**
ContextNet	Full-context	133	5.1	——	——
Conformer	Full-context	142	5.2	——	——
Conventional (Sainath et al., 2020)	Streaming	7.2G	6.6	460	870
LSTM (Sainath et al., 2020)	Streaming	179	6.4	440	770
ContextNet (Han et al., 2020)	Streaming	133	6.1	160	310
Conformer (Gulati et al., 2020)	Streaming	142	6.1	160	300
Universal ContextNet	Full-context	133	4.9	——	——
	Streaming		6.0 (-0.1)	10 (-150)	220 (-90)
Universal Conformer	Full-context	142	5.0	——	——
	Streaming		6.0 (-0.1)	-50 (-210)	130 (-170)

Training Details and Results. We train our models exactly following our baselines ContextNet (Han et al., 2020) and Conformer (Gulati et al., 2020), using Adam optimizer (Kingma & Ba, 2014), SpecAugment (Park et al., 2019) and a transformer learning rate schedule (Vaswani et al., 2017) with warm-up (Goyal et al., 2017). Our main results are summarized in Table 2 and Table 3. As shown in the tables, the streaming mode in Universal ASR models has significantly better latency and similar or higher WER results, surpassing other baselines including conventional models, LSTM-based transducers (Sainath et al., 2020), transformer-transducers (Zhang et al., 2020) and some others.

Table 3: Summary of our results on Librispeech dataset (Panayotov et al., 2015). We report WER on TestClean and TestOther (noisy) set. Compared with standalone ContextNet and Conformer models, Universal ASR models have both higher accuracy in average and better streaming latency.

Method	**Mode**	**# Params (M)**	**Test Clean/Other**	**Latency@50**	**Latency@90**
LSTM-LAS	Full-context	360	2.6 / 6.0	——	——
QuartzNet-CTC	Full-context	19	3.9 / 11.3	——	——
Transformer	Full-context	139	2.4 / 5.6	——	——
ContextNet	Full-context	31.4	2.4 / 5.4	——	——
Conformer	Full-context	30.7	2.3 / 5.0	——	——
ContextNet	Streaming	31.4	4.5 / 10.0	70	270
Conformer	Streaming	30.7	4.6 / 9.9	140	280
Universal ContextNet	Full-context	31.8	2.3 / 5.3	——	——
	Streaming		3.9 (-0.6) / 8.5 (-1.5)	40 (-30)	160 (-110)
Universal Conformer	Full-context	30.7	2.5 / 5.9	——	——
	Streaming		3.7 (-0.9) / 9.2 (-0.7)	10 (-130)	90 (-190)

4.2 Ablation Study

In this section, we perform various ablation studies to support and understand the effectiveness of each technique in Universal ASR. We train Universal ContextNet on LibriSpeech training set with exactly same settings and report WER, Latency@50 and Latency@90 on TestOther set of streaming mode. We specifically study three techniques and their combinations including *weight sharing*, *joint training* and *inplace knowledge distillation*.

During the training we distill knowledge from full-context mode (teacher) into streaming mode (student) on the fly within the same universal model. Inplace distillation comes for free as shown in training Algorithm 1. But what if we simply share weights and jointly train them without distillation? As shown in the second row of Table 4, the model without inplace distillation has worse results compared to the baseline.

Given a batch of data for each training iteration, we train both modes and aggregate their losses. We also show results of *randomly sampled training* in the third row of Table 4, which leads to even worse performance. Note that with randomly sampled training, we cannot apply inplace distillation easily either because in each training iteration there is only one prediction from either streaming mode or full-context mode.

Weight sharing reduces the model size which is one of the major motivation of Universal ASR. However, what if we simply train two individual models and use knowledge distillation with full-context model as the teacher? As shown in the last row of Table 4, the results are better than other ablation but still worse than the Universal ASR baseline. It might indicate that weight sharing itself encourages learning better deep representation for streaming ASR.

Table 4: Ablation studies of weight sharing, joint training and inplace distillation. We report WER on TestOther (noisy) set (Panayotov et al., 2015) using ContextNet with same training settings.

Weight Sharing	**Joint Training**	**Inplace Distillation**	**TestOther**	**Latency@50**	**Latency@90**
✔	✔	✔	8.5	40	160
✔	✔	✘	10.2 (+1.7)	120 (+80)	310 (+150)
✔	✘	✘	10.6 (+2.1)	90 (+50)	290 (+130)
✘	✔	✔	9.9 (+1.4)	50 (+10)	210 (+50)

5 Conclusion

In this work, we have proposed a unified framework, *Universal ASR*, to unify and improve streaming ASR by joint full-context modeling. We hope our exploration will inspire streaming models in other fields such as simultaneous machine translation and video processing.

REFERENCES

Naveen Arivazhagan, Colin Cherry, Wolfgang Macherey, Chung-Cheng Chiu, Semih Yavuz, Ruoming Pang, Wei Li, and Colin Raffel. Monotonic infinite lookback attention for simultaneous machine translation. In *ACL*, 2019.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4960–4964. IEEE, 2016.

Shuo-Yiin Chang, Rohit Prabhavalkar, Yanzhang He, Tara N Sainath, and Gabor Simko. Joint endpointing and decoding with end-to-end models. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5626–5630. IEEE, 2019.

Shuo-Yiin Chang, Bo Li, David Rybach, Yanzhang He, Wei Li, Tara Sainath, and Trevor Strohman. Low latency speech recognition using end-to-end prefetching. In *Interspeech*. ISCA, 2020.

Chung-Cheng Chiu and Colin Raffel. Monotonic chunkwise attention. *arXiv preprint arXiv:1712.05382*, 2017.

Chung-Cheng Chiu and Colin Raffel. Monotonic chunkwise attention. In *International Conference on Learning Representations*, 2018.

Chung-Cheng Chiu, Wei Han, Yu Zhang, Ruoming Pang, Sergey Kishchenko, Patrick Nguyen, Arun Narayanan, Hank Liao, Shuyuan Zhang, Anjuli Kannan, Rohit Prabhavalkar, Zhifeng Chen, Tara Sainath, and Yonghui Wu. A comparison of end-to-end models for long-form speech recognition. In *ASRU*, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Ruchao Fan, Pan Zhou, Wei Chen, Jia Jia, and Gang Liu. An online attention-based model for speech recognition. *arXiv preprint arXiv:1811.05247*, 2018.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Alex Graves. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

Alex Graves. Generating sequences with recurrent neural networks, 2013.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

Wei Han, Zhengdong Zhang, Yu Zhang, Jiahui Yu, Chung-Cheng Chiu, James Qin, Anmol Gulati, Ruoming Pang, and Yonghui Wu. Contextnet: Improving convolutional neural networks for automatic speech recognition with global context. *arXiv preprint arXiv:2005.03191*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al. Streaming end-to-end speech recognition for mobile devices. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6381–6385. IEEE, 2019.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Wenyong Huang, Wenchao Hu, Yu Ting Yeung, and Xiao Chen. Conv-transformer transducer: Low latency, low frame rate, streamable end-to-end speech recognition. *arXiv preprint arXiv:2008.05750*, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio. An online sequence-to-sequence model using partial conditioning. In *Advances in Neural Information Processing Systems 29*, pp. 5067–5075, 2016.

Anjuli Kannan, Arindrima Datta, Tara N Sainath, Eugene Weinstein, Bhuvana Ramabhadran, Yonghui Wu, Ankur Bapna, Zhifeng Chen, and Seungji Lee. Large-scale multilingual speech recognition with a streaming end-to-end model. *arXiv preprint arXiv:1909.05330*, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Bo Li, Shuo-yiin Chang, Tara N Sainath, Ruoming Pang, Yanzhang He, Trevor Strohman, and Yonghui Wu. Towards fast and accurate streaming end-to-end asr. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6069–6073. IEEE, 2020a.

Wei Li, James Qin, Chung-Cheng Chiu, Ruoming Pang, and Yanzhang He. Parallel rescoring with transformer for streaming on-device speech recognition. *arXiv preprint arXiv:2008.13093*, 2020b.

Haoran Miao, Gaofeng Cheng, Changfeng Gao, Pengyuan Zhang, and Yonghong Yan. Transformer-based online ctc/attention end-to-end speech recognition architecture. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6084–6088. IEEE, 2020.

Niko Moritz, Takaaki Hori, and Jonathan Le Roux. Triggered attention for end-to-end speech recognition. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5666–5670. IEEE, 2019.

Niko Moritz, Takaaki Hori, and Jonathan Le. Streaming automatic speech recognition with the transformer model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6074–6078. IEEE, 2020.

Arun Narayanan, Ananya Misra, Khe Chai Sim, Golan Pundak, Anshuman Tripathi, Mohamed Elfeky, Parisa Haghani, Trevor Strohman, and Michiel Bacchiani. Toward domain-invariant speech recognition via large scale training. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pp. 441–447. IEEE, 2018.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. Librispeech: an asr corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5206–5210. IEEE, 2015.

Sankaran Panchapagesan, Daniel S Park, Chung-Cheng Chiu, Alex Gruenstein, and Quoc V Le. Knowledge distillation for rnn-transducer models. *arXiv preprint*, 2020.

Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le. Specaugment: A simple data augmentation method for automatic speech recognition. *arXiv preprint arXiv:1904.08779*, 2019.

Rohit Prabhavalkar, Tara N Sainath, Yonghui Wu, Patrick Nguyen, Zhifeng Chen, Chung-Cheng Chiu, and Anjuli Kannan. Minimum word error rate training for attention-based sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4839–4843. IEEE, 2018.

C. Raffel, M. Luong, P. J. Liu, R.J. Weiss, and D. Eck. Online and Linear-Time Attention by Enforcing Monotonic Alignments. In *Proc. ICML*, 2017.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Tara N Sainath, Ruoming Pang, David Rybach, Yanzhang He, Rohit Prabhavalkar, Wei Li, Mirkó Visontai, Qiao Liang, Trevor Strohman, Yonghui Wu, et al. Two-pass end-to-end speech recognition. *arXiv preprint arXiv:1908.10992*, 2019.

Tara N Sainath, Yanzhang He, Bo Li, Arun Narayanan, Ruoming Pang, Antoine Bruguier, Shuo-yiin Chang, Wei Li, Raziel Alvarez, Zhifeng Chen, et al. A streaming on-device end-to-end model surpassing server-side conventional model quality and latency. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6059–6063. IEEE, 2020.

Jonathan Shen, Patrick Nguyen, Yonghui Wu, Zhifeng Chen, Mia X Chen, Ye Jia, Anjuli Kannan, Tara Sainath, Yuan Cao, Chung-Cheng Chiu, et al. Lingvo: a modular and scalable framework for sequence-to-sequence modeling. *arXiv preprint arXiv:1902.08295*, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Emiru Tsunoo, Yosuke Kashiwagi, Toshiyuki Kumakura, and Shinji Watanabe. Towards online end-to-end transformer automatic speech recognition. *arXiv preprint arXiv:1910.11871*, 2019.

Emiru Tsunoo, Yosuke Kashiwagi, and Shinji Watanabe. Streaming transformer asr with blockwise synchronous inference. *arXiv preprint arXiv:2006.14941*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Chengyi Wang, Yu Wu, Shujie Liu, Jinyu Li, Liang Lu, Guoli Ye, and Ming Zhou. Low latency end-to-end streaming speech recognition with a scout network. *arXiv preprint arXiv:2003.10369*, 2020.

Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Zhaofeng Wu, Ding Zhao, Qiao Liang, Jiahui Yu, Anmol Gulati, and Ruoming Pang. Dynamic sparsity neural networks for automatic speech recognition. *arXiv preprint arXiv:2005.10627*, 2020.

Ching-Feng Yeh, Jay Mahadeokar, Kaustubh Kalgaonkar, Yongqiang Wang, Duc Le, Mahaveer Jain, Kjell Schubert, Christian Fuegen, and Michael L Seltzer. Transformer-transducer: End-to-end speech recognition with self-attention. *arXiv preprint arXiv:1910.12977*, 2019.

Jiahui Yu and Thomas Huang. Autoslim: Towards one-shot architecture search for channel numbers. *arXiv preprint arXiv:1903.11728*, 2019a.

Jiahui Yu and Thomas S Huang. Universally slimmable networks and improved training techniques. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1803–1811, 2019b.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations*, 2018.

Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. Bignas: Scaling up neural architecture search with big single-stage models. *arXiv preprint arXiv:2003.11142*, 2020.

Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar. Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7829–7833. IEEE, 2020.