JOIST: A JOINT SPEECH AND TEXT STREAMING MODEL FOR ASR

Tara N. Sainath, Rohit Prabhavalkar, Ankur Bapna, Yu Zhang,
Zhouyuan Huo, Zhehuai Chen, Bo Li, Weiran Wang, Trevor Strohman

Google, Inc.

{tsainath, prabhavalkar}@google.com

ABSTRACT

We present JOIST, an algorithm to train a streaming, cascaded, encoder end-to-end (E2E) model with both speech-text paired inputs, and text-only unpaired inputs. Unlike previous works, we explore joint training with both modalities, rather than pre-training and fine-tuning. In addition, we explore JOIST using a streaming E2E model with an order of magnitude more data, which are also novelties compared to previous works. Through a series of ablation studies, we explore different types of text modeling, including how to model the length of the text sequence and the appropriate text subword unit representation. We find that best text representation for JOIST improves WER across a variety of search and rare-word test sets by 4–14% relative, compared to a model not trained with text. In addition, we quantitatively show that JOIST maintains streaming capabilities, which is important for good user-level experience.

Index Terms— end-to-end ASR, long-tail

1. INTRODUCTION

Research on E2E automatic speech recognition (ASR) systems has become increasingly prominent in recent years, with multiple research groups demonstrating the strong performance of these models [1, 2, 3, 4, 5, 6]. However, training such E2E models comes with its own set of challenges – most notably, E2E models are trained using large amounts of audio-text pairs; obtaining such hand-transcribed data is expensive and insufficient to fully cover the space of all possible words that might need to be recognized. Thus, E2E models tend to perform poorly on utterances containing words that appear infrequently in the training data (e.g., named entities) [7].

A common solution to this issue is to leverage an external neural language model (LM), trained on a much larger amount of unpaired text data, which can be incorporated into an E2E model during decoding: e.g., shallow fusion [8, 9], or rescoring [7]. While such techniques can improve rare word recognition, they have limitations: for resource-constrained tasks such as on-device speech recognition, the additional memory required to store the LM might be prohibitive (e.g. 128M parameters for the LM, relative to ∼150M parameters for the base E2E model in [7]); the high cost of running an LM at each step of the beam search may necessitate second-pass rescoring instead of shallow fusion, thus limiting the scope for improvement since with rescoring the model can only correctly recognize words which are present in the first-pass decoded lattice.

An alternative approach to address the rare-word issue is to directly train the E2E model with unpaired text-only data. Some of the earliest works in this direction have examined using text-to-speech systems to convert the text into paired audio-text pairs [10], or by incorporating cycle consistency losses (i.e., combining text-to-speech

and speech-to-text losses) [11, 12]. In addition, there has been work on injecting text into attention-based encoder-decoder models [13, 14]. An alternative approach focuses instead on creating a shared embedding space for the two modalities – speech and text [15, 16, 17, 18, 19, 20, 21] – thus improving ASR performance without increasing model parameters or decoding complexity. The use of a shared space allows training the E2E model with losses derived from paired or unpaired data, as discussed in Section 2. In the present work, we build upon these modality matching techniques in order to make them suitable for large-scale streaming state-of-the-art cascaded recurrent neural transducer (RNN-T) models [22, 23]. The contributions of this work include the following: (1) Unlike many previous works which have been applied to full-context models (i.e., full-utterance processing), we focus on streaming models [2] where words must be emitted as quickly as possible after the user speaks. We explore text injection as part of the streaming cascaded encoder model [22], which only has an acoustic look-ahead of 900ms, thus allowing for efficient low-latency streaming decoding. (2) Given the large scale of data available for our task, we explore joint training with a combination of losses computed on the supervised audio-text pairs along with the unpaired text data. We are guided by the intuition that pre-training followed by fine-tuning could be prone to forgetting the pre-trained task given the large amounts of supervised data [24]. For example, past-research has shown that pre-training has a larger impact for tasks when supervised training data is limited (less then 1,000 hours) [25, 26, 27] compared to large data sets with tens of thousands of hours and medium-sized models [28]. In addition, joint-training allows for a simpler training procedure, which is important with large-scale datasets. (3) We develop a simple solution to inject text that avoids the need for a sophisticated but more complex duration model to accurately model expected token durations. (4) Finally, we additionally optimize the model for ASR specific performance using the minimum word error rate criterion (MWER) [29] on unpaired text data by modifying the standard formulation that is only applied to audio-text pairs.

We perform a series of ablation studies on a large vocabulary voice search task to understand the performance of different text-injection schemes, both with respect to how to model duration and what type of subword unit (i.e., word-pieces or phonemes), should be used to represent the unpaired text. We find that proposed streaming JOIST configuration offers between a 4–14% relative improvement in WER across a variety of voice search and rare-word sets, compared to a baseline system that does not use any unpaired text.

2. RELATED WORK

The basic paradigm for training E2E models requires transcribed audio-text pairs; model performance is thus limited by the amount

of training data [30, 31, 32]. Many recent works have focused on improving E2E models by leveraging unpaired text, speech, or both.

Previous works which have investigated the use of unpaired speech data have focused on contrastive (e.g., [33, 34]) or reconstruction (e.g., [35, 36]) losses. Self-supervised learning approaches in natural language processsing (NLP), e.g., BERT [37], use masked language modeling (MLM) losses to pre-train encoders for NLP tasks – this leverages the fact that NLP uses discrete input representations unlike speech. Researchers have adapted these techniques for speech by deriving discrete labels for speech frames: e.g., using nearest neighbors (HuBERT [38]); vector-quantization through Gumbel softmax or online K-means (vq-wav2vec [39]) or a random (but deterministic) quantizer [40]. The wav2vec 2.0 system [41] proposed to combine the two steps – quantization and contrastive losses – which were subsequently combined with MLM losses in w2v-BERT [42]. All of these works have adopted the procedure of pre-training models with unpaired speech followed by fine-tuning the models on the paired audio-text data; notable exceptions include [24, 43], which jointly train on both losses in a multi-task framework. Such techniques tend to achieve large gains when the amount of paired speech is limited [41, 42], for example on Librispeech [44]. However, in previous work it has been observed that the gains from these techniques are limited when training with large-scale datasets [14]. Therefore, in this work, we focus on techniques which incorporate unpaired text data into the E2E model.

There have also been recent works investigating the use of unpaired text data into the E2E model. [10] converts the unpaired text data into audio utterances using a text-to-speech (TTS) system, thus making the data amenable to supervised training. A related approach consists of using cycle consistency losses – using TTS in combination with ASR to train with unpaired data [11, 12]. One of the main disadvantages of these techniques is the high computational cost involved in converting text into audio through TTS. Another approach is to distill knowledge from an LM into the E2E model [45].

An alternative approach, most closely related to our work, focuses on mapping the two modalities, audio and text, into a shared space. For example, Bapna et al. propose SLAM which uses MLM losses for the text and w2v-BERT losses for the unpaired speech to learn audio/text representations, with additional losses to align the two modalities [15]; the work is further generalized in the mSLAM approach to use multiple languages. A similar approach – dubbed STPT by Tang et al. [17] – uses BART [46] and wav2vec 2.0 [41] to train on unpaired text and speech, respectively, along with phoneme prediction and standard ASR losses on the paired data to align representations; SPLAT, proposed by Chung et al. [20], uses masked reconstruction losses for unpaired speech, a pre-trained BERT model for the unpaired text and a set of alignment losses (token-level or sequence-level) to align representations. The SpeechT5 system of Ao et al. [21] utilizes an encoder-decoder model which operates in a shared latent space; the system is combined with a set of pre-nets (to map speech/text into the shared encoder/decoder input representation) and post-nets which map the decoder output into speech/text. All of the above mentioned works focus on encoder-decoder architectures which are non-streaming; in this work, we focus on recognition using streaming RNN-T models [47, 48].

Our work is most closely related to two recent works that have investigated techniques to incorporate text-only data into RNN-T based models [18, 19]. Thomas et al. [18] propose a *textogram* – an input representation created by repeating one-hot embeddings of each input text tokens a fixed number of times (graphemes, in [18]). The textograms are stacked together with standard log-mel features along the time dimension. When training on text-only data, the input

log-mel features are set to zero; when training on audio-text paired data, the textogram features are set to zero; in either case, the model is trained with the standard RNN-T loss on the output text tokens. In order to ensure that the task of mapping from the textogram to output text tokens is not trivial, a subset of the input textogram features are masked. The model, once pre-trained, is fine-tuned for a downstream spoken language understanding task. In MAESTRO, Chen et al. [19] propose to up-sample the input text tokens using a duration prediction model similar to that used in TTS, which is jointly trained with the rest of the model. In order to ensure that the representations learned from speech and text are aligned, the MAESTRO approach adds consistency losses to align the output representations from the two modalities using the paired data. Although both of the aforementioned works [18, 19] are applied to RNN-T models, these works use full-context encoders (i.e., full utterance processing) and have not been explored in the context of streaming ASR.

3. JOIST: IMPROVING E2E ASR WITH UNPAIRED TEXT

In the present work, we simplify and expand on the techniques presented in [18, 19] to build a solution for streaming RNN-T models [22, 23]. We assume that we have examples of transcribed audio-text pairs: $\mathcal{S} = \{(x_s, y_s)\}$ (in this work, x_s corresponds to stacked log-mel feature frames; y_s corresponds to word-pieces [49]). In addition, we assume that we have (a much larger) set of unpaired text data, $\mathcal{T}$. Since the text data can be tokenized in multiple ways (e.g., as a sequence of phonemes, or word-pieces), for notational convenience we also represent the unpaired text data as a pair: $\mathcal{T} = \{(x_t, y_t)\}$ (in this work, x_t corresponds to either phonemes or word-pieces; y_t always corresponds to word-pieces), similar to [17, 19]. Note that x_t and y_t are both derived from the same unpaired text.
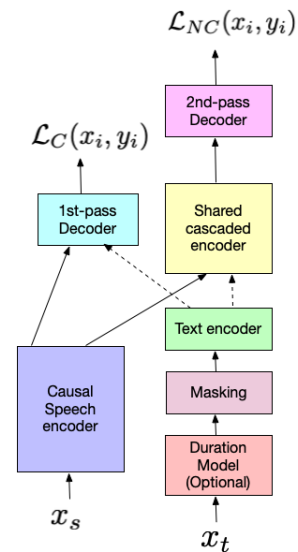

Fig. 1. JOIST architecture for training with speech and text inputs in the cascaded encoder framework [22].

Our proposed model, which we call JOIST, is depicted in Figure 1. The model is based on the cascaded encoder framework [22], which contains two Hybrid Autoregressive Transducer (HAT) [48] decoders: the first (blue) operates on the output of the *causal speech encoder* (i.e., zero right context frames); the second (pink) operates on the *shared non-causal cascaded encoder* which has access to 900ms of right context frames. We refer to these two decoders as the

first-pass and the second-pass decoders, respectively. The cascaded encoder framework is motivated by the goal of having a low output latency decoder (the causal decoder) which can be used to quickly display first-pass results to the screen; the outputs of the non-causal decoder (delayed by 900ms, because of the right context) can be computed in parallel and used to update the results displayed to the screen later following [7]. We denote the output probabilities from the *first-pass* decoder on the paired audio-text as $P_{\text{C}}(y_s|x_s)$, and the probabilities from the *second-pass decoder* on the paired audio-text as $P_{\text{NC}}(y_s|x_s)$.

To be able to train with unpaired text-data, we up-sample the input text representation, x_t, following [18, 19]. However, unlike [19], we use a simpler, parameter-free duration model, as described in Section 3.2. The up-sampled text representation is masked (to ensure that the task is sufficiently challenging for the model) and fed to a text encoder. The output of the text encoder can be fed to the first-pass decoder to generate $P_{\text{C}}(y_t|x_t)$, or to the second-pass decoder after passing through the shared encoder to generate $P_{\text{NC}}(y_t|x_t)$.

3.1. Loss Computation

The model is trained by jointly optimizing both decoders using audio-text pairs in addition to the unpaired text. If we denote $\mathcal{L}_{\text{C}}(y, x) = -\log P_{\text{C}}(y|x)$ and $\mathcal{L}_{\text{NC}}(y, x) = -\log P_{\text{NC}}(y|x)$ as the negative log likelihood of the causal and the non-causal decoders, respectively, we define the overall loss as:

$$\begin{aligned} \mathcal{L}_{\text{CE}} = {} & \lambda_1 [\mathcal{L}_{\text{C}}(y_s, x_s) + \mathcal{L}_{\text{NC}}(y_s, x_s)] \\ & + \lambda_2 [\mathcal{L}_{\text{C}}(y_t, x_t) + \mathcal{L}_{\text{NC}}(y_t, x_t)] \end{aligned} \tag{1}$$

where, λ_1 is the weight corresponding to the paired audio-text data and λ_2 is the weight on the unpaired text-only data. As can be seen in (1), we weight the casual and non-causal decoders equally in the loss function. In practice, the losses are computed over a mini-batch of examples; in training, we use 50% paired audio-text and unpaired 50% text examples in each mini-batch. Unlike previous work, we do not add additional MLM or consistency losses from the text encoder [15, 17, 19] which simplifies the overall training procedure. Evaluations of the impact of these and other losses in the JOIST framework are left as future work.

3.2. Duration Modeling to Up-Sample Text Representations

Previous works [18, 19] have demonstrated the importance of up-sampling the text representations in order to create representations that can be easily aligned with the speech modality. In this work, we consider a number of schemes for this purpose.

Fixed Repetition: In this scheme, each text sub-word unit (word-piece or phoneme) is replicated a fixed number of times, exactly following the approach proposed in [18]. The drawback of this approach is that this does not match the actual expected durations of various sub-word units in practice (e.g., vowels tend to be longer than consonants; word-pieces with more characters tend to have longer durations). Thus, repetition by a fixed amount is a simple but crude approximation. We consider a fixed repetition length of 3, which corresponds to 180ms per unit of x_t, in this work.

Random Repetition: To address the shortcomings of fixed repetition, we also consider random repetition. In this approach, the text representation is varied by randomly repeating each unit by sampling from a uniform distribution between 1 and 3 (i.e., 60ms, 120ms, or 180ms per unit of x_t). The potential benefit of random repetition is that it simulates some of the variation that we might expect to see in

the distribution of sub-word units. It must be noted however, that as with the fixed repetition scheme, this is still a crude approximation.

Sub-Word Distribution : In this approach, we model the distribution of each sub-word unit using a Gaussian distribution. Given the paired audio-text data, we generate forced-alignments using a baseline system [50] to estimate phoneme and word alignments for each word in the transcript; these are used to compute statistics of the number of frames corresponding to each phoneme or word in the supervised training data. We decompose each word into its constituent word-pieces and evenly distribute the words total frames amongst its constituent word-pieces. Thus, by accumulating statistics over the entire training set we can compute the sufficient statistics of the Gaussian distribution – the mean and standard deviation – for each unit. We can repeat each unit by sampling from it's corresponding Gaussian distribution. This is more exact then fixed or random repetition, but is still an approximation since it ignores contextual effects as each unit is sampled independently.

Align+Sub-Word Distribution: We can always use all of the text in the paired audio-text set, $\mathcal{S}$, to augment the unpaired text data, $\mathcal{T}$ – in effect treating the text in the paired data as unpaired text. In this specific case we up-sample text examples in $\mathcal{T}$ based on the true number of frames for each unit, obtained using a forced-alignment [51]; as before, we divide up the total number of frames in the word amongst its constituent word-pieces. For text data in $\mathcal{T}$, for which audio (and thus, forced-alignments) are not available, we use subword distribution to up-sample the text.

3.3. MWER

Instead of optimizing model log-likelihoods as in (1), the Minimum Word Error Rate (MWER) [29] strategy minimizes the expected number of word errors. Specifically, in the standard MWER criterion, given a speech utterance, x, corresponding ground-truth text, y^*, and a set of N-best hypotheses, y_i, $(1 \leq i \leq N)$, we minimize the MWER loss proposed in [29]:

$$\mathcal{L}^{\text{MWER}}(y^*, x) = \sum_{y_i} \left[\frac{P(y_i|x)}{\sum_i P(y_i|x)} \right] \left[\mathcal{W}(y_i, y^*) - \frac{\sum_i \mathcal{W}(y_i, y^*)}{N} \right] \tag{2}$$

where, $\mathcal{W}(y, y^*)$ corresponds to the number of word errors between the hypothesis, y, and the ground-truth, y^*. To stabilize training, the MWER loss is interpolated with the standard cross-entropy loss, after initializing from a model that has converged under the maximum likelihood criterion in (1). MWER training has been shown to improve WER by 5–20% in previous works [29, 52, 53].

In the present work, we adapt MWER training by noting that we can compute the MWER loss using the paired audio-text data (i.e., the standard MWER loss paths using (x_s, y_s)), but *also using the unpaired text representations* (i.e., (x_t, y_t)). This leads to a novel MWER loss formulation which allows us to train the model using both paired audio-text as well as the unpaired text:

$$\begin{aligned} \mathcal{L} = {} & \lambda_1 \left[\mathcal{L}_{\text{C}}^{\text{MWER}}(y_s, x_s) + \mathcal{L}_{\text{NC}}^{\text{MWER}}(y_s, x_s) \right] \\ & + \lambda_2 \left[\mathcal{L}_{\text{C}}^{\text{MWER}}(y_t, x_t) + \mathcal{L}_{\text{NC}}^{\text{MWER}}(y_t, x_t) \right] + \alpha \mathcal{L}_{\text{CE}} \end{aligned} \tag{3}$$

where, $\mathcal{L}_{\text{C}}^{\text{MWER}}$ and $\mathcal{L}_{\text{NC}}^{\text{MWER}}$ represent the MWER losses in (2) computed using the first-pass and second-pass decoders, respectively, and α represents the interpolation weight for the CE loss. As with CE training in (1), we weight causal and non-causal losses equally.

3.4. Streaming Metrics

An important focus of our work is to ensure that the model can be used to produce streaming first-pass recognition results with low latency (i.e., the time between when the user speaks, and the system outputs a sub-word unit). Since we are now injecting an additional source of text-data, where naturally RNN-T would prefer to delay and see more text, it is important to ensure that latency metrics are not degraded with JOIST. We therefore quantify the *streaming quality* of the system in terms of the following latency metrics.

In streaming ASR systems, it is important to detect when the user has finished speaking, so that next fulfillment step can be triggered as quickly as possible. *Endpointer latency* measures the time difference between when the user finishes speaking and when the system predicts an end of sentence (EOS) token [54, 55]; a lower endpointer latency allows for faster fulfillment and system response and is thus desirable. We report both the median (i.e. 50th percentile, **EP50**) and the 90th percentile (**EP90**) endpointer latency.

An additional desirable feature is to ensure that the system also has low latency while outputting all intervening words – i.e., low latency for the partial hypotheses generated by the first-pass decoder, which will be displayed to the screen. We measure this by computing *partial latency* – the time difference between when the first correct partial hypothesis is generated by the model and when the user finishes speaking [54]. In this work, we report 50th (**PR50**) and 90th percentile (**PR90**) partial latency.

Finally, in order to create the best user experience, we would like to ensure that the hypotheses generated by the first- and second-pass decoders are as similar as possible. If not, the outputs presented on the screen will change constantly, which causes *too much screen flickering*. The *Prefetch Hit Rate* (**PFHR**) calculates the percentage of utterances where the hypotheses flip between the first- and second-pass decoders, at the utterance level.

3.5. Novelty of Proposed Method

Now that we have described JOIST, in this section we further highlight its novelty. First, most previous works which have investigated techniques to directly incorporate unpaired text data into the model have focused on non-streaming encoder-decoder architectures [13, 14, 15, 16, 17, 20, 21]. Although two recent works [18, 19] apply these approaches to an RNN-T model, they only consider full-context (i.e., bi-directional) encoders, and are thus unsuitable for streaming speech recognition. To the best of our knowledge, our work, JOIST, is the first to investigate whether it is possible to improve *streaming* end-to-end transducer models using unpaired text without synthesis [56]. An additional benefit of the proposed approach is its simplicity: unlike previous works, we focus on joint multi-task training of supervised and unsupervised objectives using a parameter-free duration model, thus greatly simplifying the overall process. The proposed techniques also demonstrate that it is possible to optimize the model for ASR criteria such as minimum word error rate (MWER) [29] using unpaired text data, which opens up new research directions. To the best of our knowledge, our work is the first to demonstrates that it is possible to obtain gains using text even when using large-scale supervised training sets.

4. EXPERIMENTS

4.1. Training Sets

The proposed techniques are evaluated on a large-scale voice search task. Our first set of experiments are conducted using a super-

vised training set, referred to as *Train Set A*, that consists of ∼300 million United States English multidomain audio-text pairs, which include domains such as Search, Dictation, YouTube and Telephony [32]. All domains are anonymized and hand-transcribed, except for YouTube where the transcription is done in a semi-supervised fashion [57]. Since the effectiveness of various techniques often reduces as the size of the training data increases, in order to test robustness we also consider an even larger dataset, *Train Set B*, which consists of ∼650 million United States English multidomain audio-text pairs, spanning similar domains as above; the 'supervised' text corresponding to these utterances is obtained using a 600M-parameter teacher system trained on *Train Set A* [58].

In addition to the diverse multi-condition training sets, we increase robustness by using multi-condition training data to simulate noisy conditions [59]; generating data at both 8KHz and 16KHz, with equal probability, to reduce acoustic mismatch due to sampling rates [60, 61]; and using SpecAug [62]. Noisy data is generated at signal-noise-ratio (SNR) from 0 to 30 dB, with an average SNR of 12 dB, and with T60 times ranging from 0 to 900ms, averaging 500ms. Noise segments are sampled from YouTube and daily life noisy environmental recordings.

Our unpaired text data consists of more than 100B utterances and spans the domains of Maps, News, Google Play, Web and YouTube, and is thus more than two-orders of magnitude larger than our supervised sets. In addition, we incorporate all text data from the supervised sets, Train Sets A and B, which we add to the unpaired text data. In order to ensure that the text data does not degrade quality on the base voice search task, we sample text data from the unsupervised and supervised sets with the same probability so that each unpaired-text minibatch contains 50% data from Train Set A/B and 50% unpaired text data, following the standard practice for training N-gram [63] and maximum-entropy [64] LMs.

4.2. Evaluation Sets

Results are reported on multiple test sets which measure the systems ability to recognize the *head* (i.e., relatively common words) as well as the *long tail* of rare words. The *Search* test set includes around 12K Voice Search utterances with an average length of 5.5 seconds. They are anonymized and hand-transcribed, and are representative of Google's Voice Search traffic. In addition, to measure accuracy while recognizing the long-tail of rare words, we create synthetic *rare word test sets*, as described in [65]. Specifically, we look for words in the LM training data that occur rarely (i.e., less than 5 times) in the supervised training sets A and B. We construct test sets for each of the 5 domains (i.e., Maps, News, Play, Query, YouTube) by selecting utterances containing rare words and synthesizing them using a TTS system [66].

4.3. Modeling Architecture

Our proposed JOIST architecture is modeled as follows. All speech-text pairs use a 128-dimensional log-mel feature frontend computed on 32 msec windows with a 10ms hop. Features from four consecutive frames are stacked together, and sub-sampled by a factor of 3 to generate 512-dimensional features at a 30ms frame rate. These are appended with 16-dimensional one-hot domain-id vectors [32], to obtain x_s. The input speech features, x_s, are fed to the *causal speech encoder*, which consists of 5 conformer layers [67] with causal convolution and left-context attention to ensure that the causal speech encoder does not have access to any right context frames. The self attention layers in the conformer use multi-headed attention with 8

heads, and a convolution kernel size of 15. We use a stacking layer after the second conformer block, which down-samples the input by a factor of 2, so that the effective frame rate at the output of the causal speech encoder is 60ms.

The text input, x_t, is constructed by generating one-hot embeddings of either 4,096 word-pieces [49] or 46 phonemes depending on the experiment. These are then up-sampled using the duration model, and masked before feeding them to the text encoder. Following [15, 68], we mask 15% of the up-sampled text IDs with spans of length 5. The text encoder is a simple embedding table which takes sub-word IDs as inputs and generates corresponding embeddings. We set $\lambda_1 = 0.1$, and $\lambda_2 = 0.2$, in (1) and (3), respectively.

The bulk of the processing of JOIST is performed by the *shared cascaded encoder*, and we devote most of the model's capacity to this block. The shared cascaded encoder consists of 12 conformer layers; the first 5 have access to three frames of right context each, for a total of $3 \times 5 \times 60\text{ms} = 900\text{ms}$ of acoustic right context. This specific model structure follows [23], where it was shown to provide a good trade-off between accuracy and latency.

Both RNN-T decoders (first- and second-pass) consist of a joint network (a single feed-forward layer with 640 units) and an embedding prediction network [69] which uses an embedding dimension of 640, and conditions on only the last two labels. In total, each decoder contains 15.2M parameters. All models use the Hybrid Autoregressive Transducer (HAT) factorization [48] to predict 4,096 word pieces [49]. Furthermore, all models are trained with FastEmit [70] to encourage the model to not delay predictions. Overall, the total model size is $\sim$169M parameters.

For all models, we discard the text-encoder during inference, and evaluate the RNN-T decoders using input speech utterances. Unless otherwise indicated, all WERs are computed using the second-pass decoder. Models are decoded with a beam size of 8.

We also compare JOIST to rescoring a lattice generated from the second-pass decoder using a LM. We train a conformer LM, following [7], which has a look-back attention context of 31. The LM contains 12 conformer layers [67] each of which has a model dimension of 768 and a feed-forward layer dimension of 2048.

5. RESULTS

5.1. Full-Context Models

Since all text injection methods have been explored in the context of full-context encoder layers, in our first set of experiments, we compare previous approaches explored in the literature to our proposed method. For this set of experiments, we represent the text input representation, x_t, using word-pieces, and we use *Train Set A* as our paired audio-text data, $\mathcal{S}$. Full-context means that the left and right context for self-attention in all conformer layers (both the causal and shared cascaded encoders), is set to allow the model to access all frames in the utterance. Results of our full context experiments are presented in Table 1, where "S" corresponds to the *Search* test set; the rare word test sets are denoted as "M" (*Maps*), "N" (*News*), "P" (*Google Play*), "Q" (*Search Queries*) and "Y" (*YouTube*).

The baseline model, $B0$, is a model that is trained without any text data. $E0$ corresponds to a JOIST model that uses unpaired text, but does not up-sample the text tokens (i.e., no duration model), similar to [17], except that we do not use a MLM loss and inject wordpieces. As can be seen in Table 1, without replication, $E0$ is no better then baseline $B0$, and is much worse on the *News* set. Fixed repetition of the word-piece tokens, which is very similar to [18], but in a joint training setup ($E1$), however, can achieve a WER improvement

of 2–5% relative over $B0$ on the rare-word test sets. The alternative duration modeling schemes random distribution ($E2$), and sub-word distribution ($E3$) improve performance over the baseline, $B0$, but do not outperform fixed repetition ($E1$).

Finally, as a comparison to other methods $B1$ shows joint-training of speech-text and text with SLAM [15], where we concatenate the outputs from speech encoder and text encoder before passing to the shared encoder. SLAM works in tasks where the encoder is pre-trained using SLAM, and then fine-tuned using a supervised loss. However, in joint-training the conformer layers learn to compute attention over speech and text jointly in training, which is missing in inference, leading to the high WER. We compare against MAESTRO [19] in the next section.

Our goal in these initial set of experiments confirm the importance of duration modeling and also that joint training is an effective yet simple method to optimize mixed input systems. We use these initial findings to help guide our experiments with streaming models in the next section, where we also investigate the impact of using phonemes vs. word-pieces as our text representation.

Exp	Model	S	M	N	P	Q	Y
B0	no text	4.8	11.9	8.2	36.1	19.3	22.6
B1	SLAM	75.9	87.0	99.4	87.1	84.7	91.2
E0	no rep	4.8	11.9	8.5	35.8	19.5	22.6
E1	fixed rep	**4.6**	**11.4**	**7.9**	**35.7**	**18.9**	**22.1**
E2	random rep	4.7	11.8	8.2	35.9	19.0	22.2
E3	sub-wrd dist	4.9	11.8	8.0	36.2	19.5	22.2

Table 1. WER for WPM Text Injection; Full-Context Model

5.2. Streaming Models

First, we repeat the experiment of representing unpaired data x_t in terms of word-pieces, but using a streaming architecture described in Section 4.3. Once again, we use *Train Set A* as our supervised training set, $\mathcal{S}$. Our results are presented in Table 2. Our baseline model ($B2$) corresponds to an E2E model that is trained solely on the paired audio-text data. We leave out SLAM from the comparison, as Table 1 showed a degradation due to the speech-text concatenation. We observe similar trends to the non-streaming model: as long as we up-sample the text representations ($E5$–$E7$), we can obtain a 2–4% relative improvement in WER over the baseline ($B2$) on the rare word sets; however, JOIST with no repetition $E4$ does not improve over the baseline. Finally, we also compare to $B3$, a jointly-trained word-piece based MAESTRO model, which uses a TTS-based duration model and consistency losses, with the same cascaded encoder architecture as all other models in the table. This model is on-par with the parameter-free duration-modeling results from $E5$–$E7$. Improvements with alternative architectures have been seen with MAESTRO, though we leave that for future work.

Exp	Model	S	M	N	P	Q	Y
B2	no text	5.2	12.5	9.0	37.4	20.0	23.2
B3	MAESTRO	5.6	12.2	9.1	36.2	20.3	23.0
E4	no rep	5.3	12.3	15.1	36.9	20.0	23.4
E5	fixed rep	5.3	12.1	**8.8**	37.0	**19.6**	**23.1**
E6	random rep	**5.2**	**12.0**	9.0	**36.7**	19.8	23.3
E7	sub-wrd dst	5.2	12.1	9.0	36.9	19.7	23.3

Table 2. WER for WPM Text Injection; Streaming Model

Next, we contrast the benefits of different duration modeling schemes using phonemes versus word-pieces in representing un-

paired text, x_t. These experiments are conducted using the much larger *Train Set B*, which uses a teacher model to generate the paired audio-text data, $\mathcal{S}$. Our results are presented in Table 3. We have omitted results on the Search test set for clarity since it typically does not change between techniques, but will present them in the final section. Since the various word-piece based duration modeling techniques perform similarly, we only list the random repetition baseline $E8$ for brevity. As can be seen in the table, systems which use phoneme representations ($E9$–$E12$) outperform word-pieces; There is not a huge difference in performance between the different duration modeling strategies, similar to what we found with word-pieces. Since $E11$ provides a very slight edge, we will choose that for subsequent exepriments. Overall, $E11$ provides between a 4–14% relative improvement in WER over $B4$. The number of phonemes is roughly 3-times that of wordpieces, and it is possible that this finer-granularity is inherently a better model for duration and thus helps with quality. In the future, we will also compare to a grapheme text representation, to understand if the finer granularity helps, or if gains come because phonemes are typically better for long-tail words compared to graphemes/wordpieces [71].

Exp	Model	M	N	P	Q	Y
B4	Baseline, no Text	13.9	9.4	37.9	21.6	24.4
E8	random rep, WPM	13.7	9.3	37.6	20.9	24.2
E9	fixed rep, phn	**13.0**	9.1	32.7	**18.7**	21.5
E10	random rep, phn	13.1	9.2	**31.2**	18.9	21.3
E11	sub-wrd dst, phn	13.1	9.6	32.6	**18.7**	**21.2**
E12	align+dist, phn	13.1	**8.5**	33.2	19.5	22.2

Table 3. WER for phonemes vs. word-piece based text injection

5.3. Comparison to Neural LM

As stated in the introduction, neural LM is a very common approach used to improve the quality of rare word recognition [9]. In this section, we look at a standard cascaded encoder trained only on audio-text pairs ($B4$), that is then rescored by a neural LM ($B5$). The oracle WER of the lattice is also shown in Table 4. We compare this to the best phn-JOIST system ($E11$), and also rescore this with a LM ($E13$). Table 4 shows that a LM with the base system ($B5$), still does better then JOIST alone ($E11$) on half of the long-tail sets, though requires an additional 128M parameter LM. However, if we apply the LM to JOIST ($E13$), this outperforms $B5$ on all sets. Moreover, the oracle WER as well as the relative WER improvement of $E13$ is much larger then $B5$, which confirms our hypothesis that JOIST helps to bring rare word hypotheses into the beam, which leads to even better quality with the LM.

Exp	Model	M	N	P	Q	Y
B4	no text	13.9	9.4	37.9	21.6	24.4
B5	B4 + neural LM	10.3	9.8	33.8	14.9	20.2
	oracle	7.0	8.2	21.7	9.5	14.0
E11	JOIST	13.1	9.6	32.6	18.7	21.2
E13	JOIST + neural LM	**9.7**	**9.4**	**28.5**	**12.9**	**17.1**
	oracle	6.4	7.9	15.7	7.5	10.8

Table 4. WER for Neural LM

5.4. Final Best System: Phoneme JOIST

We take the best system – the phoneme-based JOIST from $E11$ – and investigate quality and latency compared to a cascaded encoder model trained only on paired data.

5.4.1. Quality: MWER Training

Table 5 shows $B4$, the baseline cascaded encoder, and $B6$, the baseline after MWER training using paired-only data. In contrast, $E11$ is the JOIST model and $E14$ the model after MWER training, using the method from Section 3 to train on paired and unpaired data. Further gains are seen with $E14$, particularly on rare-word sets. In addition, we also evaluate the 1st-pass WER for both systems after MWER training, which appears to be around 9.3%. In the next section, we will discuss metrics around flickering.

Exp	Model	S	M	N	P	Q	Y
B4	CascEnc	6.2	13.9	9.4	37.9	21.6	24.4
B6	B4 + MWER	5.8	13.5	9.1	37.5	20.8	24.0
E11	JOIST	6.1	13.1	9.6	32.6	18.7	21.2
E14	E11 + MWER	**5.8**	**12.7**	9.4	**32.0**	**18.3**	**20.7**

Table 5. WER for MWER Experiments

5.4.2. Streaming Metrics

An important focus of our work is to ensure that JOIST has good streaming recognition performance compared to a cascaded encoder, which we quantify in Table 6. First, we see that both the endpointer (EP50, EP90) and partial (PR50, PR90) latencies between JOIST ($E14$) and Cascaded Encoder ($B6$) are on par. Second, the flickering between the 1st and 2nd pass, as measured by PFHR, is also on-par.

Exp	EP50	EP90	PR50	PR90	PFHR
B6	410	710	20	430	0.79
E14	410	700	40	430	0.79

Table 6. WER and Latency on Search Test Set

5.4.3. Comparison on Logs Data

Finally, we compare the no-text cascaded encoder with phoneme JOIST, by running a "side-by-side" (SxS) on unseen, real-audio search data. In the SxS, we collect 114 utterances which generate different hypotheses when decoded with the two systems, and send these utterances to two human raters. Based on these ratings, we report five statistics based on the SxS: *Changed* – % of utterances in which the two models produced different hypotheses; *Wins* – the # of utts the JOIST is correct and Cascaded Encoder is incorrect; *Losses* – the # of utts JOIST is incorrect and Cascaded Encoder is correct; *Neutral* – the # of utts both models are both correct or incorrect;

The table shows that more than 10% of the traffic is changed with JOIST, and it has more wins then the Cascaded Encoder. A closer look at the errors shows wins in many rare words, due to the text injection.

Changed (%)	Win	Loss	Neutral
10.6%	23	16	75

Table 7. SxS: Cascaded Encoder vs. JOIST

6. ACKNOWLEDGEMENTS

7. REFERENCES

[1] J. Li, Y. Wu, Y. Gaur, et al., "On the Comparison of Popular End-to-End Models for Large Scale Speech Recognition," in *Proc. Interspeech*, 2020.

[2] Y. He, T. N. Sainath, R. Prabhavalkar, et al., "Streaming End-to-end Speech Recognition For Mobile Devices," in *Proc. ICASSP*, 2019.

[3] C.-C. Chiu, T. N. Sainath, Y. Wu, et al., "State-of-the-art Speech Recognition With Sequence-to-Sequence Models," in *Proc. ICASSP*, 2018.

[4] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *Proc. ICASSP*, 2017.

[5] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proc. ASRU*, 2019.

[6] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, "A new training pipeline for an improved neural transducer," in *Proc. Interspeech*, 2020.

[7] T. N. Sainath, Y. He, Narayanan, et al., "An Efficient Streaming Non-Recurrent On-Device End-to-End Model with Improvements to Rare-Word Modeling," in *Proc. Interspeech*, 2021.

[8] J. Chorowski and N. Jaitly, "Towards Better Decoding and Language Model Integration in Sequence to Sequence Models," in *Proc. Interspeech*, 2017.

[9] A. Kannan, Y. Wu, P. Nguyen, et al., "An analysis of incorporating an external language model into a sequence-to-sequence model," in *Proc. ICASSP*, 2018.

[10] Z. Chen, Y. Zhang, A. Rosenberg, et al., "Injecting Text in Self-Supervised Speech Pretraining," in *Proc. ASRU*, 2021.

[11] T. Hori, R. Astudillo, T. Hayashi, et al., "Cycle-Consistency Training for End-to-End Speech Recognition," in *Proc. ICASSP*, 2019.

[12] A. Tjandra, S. Sakti, and S. Nakamura, "Listening While Speaking: Speech Chain by Deep Learning," in *Proc. ASRU*, 2017.

[13] B. Yusuf, A. Gandhe, and A. Sokolov, "USTED: Improving ASR with a Unified Speech and Text Encoder-Decoder," in *Proc. ICASSP*, 2022.

[14] T. N. Sainath, R. Pang, R. J. Weiss, et al., "An Attention-Based Joint Acoustic and Text on-Device End-To-End Model," in *Proc. ICASSP*, 2020.

[15] A. Bapna, Y.-A. Chung, N. Wu, et al., "SLAM: A Unified Encoder for Speech and Language Modeling via Speech-Text Joint Pre-Training," *arXiv preprint arXiv:2110.10329*, 2021.

[16] A. Bapna, C. Cherry, Y. Zhang, et al., "mSLAM: Massively Multilingual Joint Pre-Training for Speech and Text," *CoRR*, vol. abs/2202.01374, 2022.

[17] Y. Tang, H. Gong, N. Dong, et al., "Unified Speech-Text Pretraining for Speech Translation and Recognition," in *Proc. ACL*, 2022.

[18] S. Thomas, H. J. Kuo, B. Kingsbury, and G. Saon, "Towards Reducing the Need for Speech Training Data to Build Spoken Language Understanding Systems," in *Proc. ICASSP*, 2022.

[19] Z. Chen, Y. Zhang, A. Rosenberg, et al., "MAESTRO: Matched Speech Text Representations through Modality Matching," in *Proc. ICASSP*, 2022.

[20] Y.-A. Chung, C. Zhu, and M. Zeng, "SPLAT: Speech-Language Joint Pre-Training for Spoken Language Understanding," in *Proc. of NAACL-HLT*, 2021.

[21] J. Ao, R. Wang, L. Zhou, et al., "SpeechT5: Unified-Modal Encoder-Decoder Pre-Training for Spoken Language Processing," in *Proc. ACL*, 2022.

[22] A. Narayanan, T. N. Sainath, R. Pang, et al., "Cascaded encoders for unifying streaming and non-streaming ASR," in *Proc. ICASSP*, 2021.

[23] T. N. Sainath, Y. He, A. Narayanan, et al., "Improving the Latency and Quality of Cascaded Encoder," in *Proc. ICASSP*, 2022.

[24] J. Bai, B. Li, Y. Zhang, et al., "Joint Unsupervised and Supervised Training for Multilingual ASR," in *Proc. ICASSP*, 2022.

[25] A. Mohamed and G. Hinton, "Phone Recognition using Restricted Boltzmann Machines," in *Proc. ICASSP*, 2010.

[26] T. N. Sainath, B. Kingsbury, B. Ramabhadran, P. Fousek, P. Novak, and A. Mohamed, "Making Deep Belief Networks effective for large vocabulary continuous speech recognition," in *Proc. ASRU*, 2011.

[27] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011.

[28] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proc. Interspeech*, 2014.

[29] R. Prabhavalkar, T. N. Sainath, Y. Wu, et al., "Minimum Word Error Rate Training for Attention-based Sequence-to-sequence Models," in *Proc. ICASSP*, 2018.

[30] R. Prabhavalkar, K. Rao, T. N. Sainath, et al., "A Comparison of Sequence-to-sequence Models for Speech Recognition," in *Proc. Interspeech*, 2017.

[31] K. Irie, R. Prabhavalkar, A. Kannan, et al., "On the Choice of Modeling Unit for Sequence-to-Sequence Speech Recognition," *Proc. Interspeech*, 2019.

[32] A. Narayanan, R. Prabhavalkar, C.-C. Chiu, et al., "Recognizing Long-Form Speech Using Streaming End-to-End Models," in *Proc. ASRU*, 2019.

[33] A. Van Den Oord, Y. Li, and O. Vinyals, "Representation Learning with Contrastive Predictive Coding," *arXiv preprint arXiv:1807.03748*, 2018.

[34] S. Schneider, A. Baevski, R. Collobert, and M. Auli, "wav2vec: Unsupervised Pre-training for Speech Recognition," in *Proc. Interspeech*, 2019.

[35] Y.-A. Chung and J. Glass, "Generative Pre-Training for Speech with Autoregressive Predictive Coding," in *Proc. ICASSP*, 2020.

[36] A. T. Liu, S.-W. Yang, P.-H. Chi, et al., "Mockingjay: Unsupervised Speech Representation Learning with Deep Bidirectional Transformer Encoders," in *Proc. ICASSP*, 2020.

[37] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proc. NAACL-HLT*, 2018.

[38] W.-N. Hsu, B. Bolte, Y.-H. H. Tsai, et al., "HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3451–3460, 2021.

[39] A. Baevski, S. Schneider, and M. Auli, "vq-wav2vec: Self-Supervised Learning of Discrete Speech Representations," in *Proc. ICLR*, 2020.

[40] C.-C. Chiu, J. Qin, Y. Zhang, et al., "Self-Supervised Learning with Random-Projection Quantizer for Speech Recognition," in *Proc. ICML*, 2022.

[41] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A Framework for Self-Supervised Learning of Speech Representations," *Proc. Neurips*, 2020.

[42] Y.-A. Chung, Y. Zhang, W. Han, et al., "w2v-bert: Combining Contrastive Learning and Masked Language Modeling for Self-Supervised Speech Pre-Training," in *Proc. ASRU*, 2021.

[43] C. Talnikar, T. Likhomanenko, R. Collobert, and G. Synnaeve, "Joint Masked CPC And CTC Training For ASR," in *Proc. ICASSP*, 2021.

[44] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR Corpus based on Public Domain Audio Books," in *Proc. ICASSP*, 2015.

[45] Y. Kubo, S. Karita, and M. Bacchiani, "Knowledge Transfer from Large-scale Pretrained Language Models to End-to-end Speech Recognizers," in *Proc. ICASSP*, 2022.

[46] M. Lewis, Y. Liu, N. Goyal, et al., "BART: Denoising Sequence-to-Sequence Pre-Training for Natural Language Generation, Translation, and Comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[47] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech Recognition with Deep Neural Networks," in *Proc. ICASSP*, 2012.

[48] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid Autoregressive Transducer (HAT)," in *Proc. ICASSP*, 2020.

[49] M. Schuster and K. Nakajima, "Japanese and Korean voice search," in *Proc. ICASSP*, 2012.

[50] G. Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models," in *Proc. Interspeech*, 2016.

[51] P. J. Moreno, C. F. Joerg, J.-M. Van Thong, and O. Glickman, "A Recursive Algorithm for The Forced Alignment of Very Long Audio Segments," in *Proc. ICSLP*, 1998.

[52] J. Guo, G. Tiwari, J. Droppo, et al., "Efficient Minimum Word Error Rate Training of RNN-Transducer for End-to-End Speech Recognition," in *Proc. Interspeech*, 2020.

[53] C. Weng, C. Yu, J. Cui, et al., "Minimum Bayes Risk Training of RNN-Transducer for End-to-End Speech Recognition," in *Proc. Interspeech*, 2020.

[54] B. Li, A. Gulati, J. Yu, et al., "A Better and Faster End-to-End Model for Streaming ASR," in *Proc. ICASSP*, 2021.

[55] S. Chang, R. Prabhavalkar, Y. He, et al., "Joint Endpointing and Decoding with End-to-End Models," in *Proc. ICASSP*, 2019.

[56] Z. Chen, Y. Zhang, A. Rosenberg, B. Ramabhadran, P. Moreno, and G. Wang, "Tts4pretrain 2.0: Advancing the use of Text and Speech in ASR Pretraining with Consistency and Contrastive Losses," in *Proc. ICASSP*, 2022.

[57] H. Liao, E. McDermott, and A. Senior, "Large Scale Deep Neural Network Acoustic Modeling with Semi-supervised Training Data for YouTube Video Transcription," in *Proc. ASRU*, 2013.

[58] D. Hwang, K. Sim, Z. Huo, and T. Strohman, "Pseudo Label Is Better Than Human Label," in *Proc. ICASSP*, 2022.

[59] C. Kim, A. Misra, K. Chin, et al., "Generation of Large-Scale Simulated Utterances in Virtual Rooms to Train Deep-Neural Networks for Far-Field Speech Recognition in Google Home," in *Proc. Interspeech*, 2017.

[60] D. Yu, M. L. Seltzer, J. Li, et al., "Feature learning in deep neural networks-studies on speech recognition tasks," in *Proc. ICLR*, 2013.

[61] J. Li, D. Yu, J. Huang, and Y. Gong, "Improving Wideband Speech Rcognition using Mixed-bandwidth Training Data in CD-DNN-HMM," in *Proc. SLT*, 2012.

[62] D. S. Park, W. Chan, Y. Zhang, et al., "SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, 2019.

[63] C. Allauzen and M. Riley, "Bayesian Language Model Interpolation for Mobile Speech Input," in *Proc. Interspeech*, 2011.

[64] F. Biadsy, K. Hall, P. J. Moreno, and B. Roark, "Backoff Inspired Features for Maximum Entropy Language Models," in *Proc. Interspeech*, 2014.

[65] C. Peyser, S. Mavandadi, T. N. Sainath, et al., "Improving Tail Performance of a Deliberation E2E ASR Model Using a Large Text Corpus," in *Proc. Interspeech*, 2020.

[66] X. Gonzalvo, S. Tazari, C. Chan, et al., "Recent Advances in Google Real-time HMM-driven Unit Selection Synthesizer," in *Proc. Interspeech*, 2016.

[67] A. Gulati, J. Qin, C.-C. Chiu, et al., "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech*, 2020.

[68] M. Joshi, D. Chen, Y. Liu, et al., "SpanBERT: Improving Pretraining by Representing and Predicting Spans," *Transactions of the Association for Computational Linguistics*, vol. 8, 2020.

[69] R. Botros and T. N. Sainath, "Tied & Reduced RNN-T Decoder," in *Proc. Interspeech*, 2021.

[70] J. Yu, C.-C. Chiu, B. Li, et al., "FastEmit: Low-latency Streaming ASR with Sequence-level Emission Regularization," in *Proc. ICASSP*, 2021.

[71] A. Bruguier, R. Prabhavalkar, G. Pundak, and T. N. Sainath, "Phoebe: Pronunciation-aware Contextualization for End-to-end Speech Recognition," in *Proc. ICASSP*, 2019.