End-to-End ASR: from Supervised to Semi-Supervised Learning with Modern Architectures

A Preprint

Gabriel Synnaeve[*]
Facebook, NYC
gab@fb.com

Qiantong Xu[*]
Facebook, Menlo Park
qiantong@fb.com

Jacob Kahn[*]
Facebook, Menlo Park
jacobkahn@fb.com

Tatiana Likhomanenko[*]
Facebook, Menlo Park
antares@fb.com

Edouard Grave[*]
Facebook, Paris
egrave@fb.com

Vineel Pratap
Facebook, Menlo Park
vineelkpratap@fb.com

Anuroop Sriram
Facebook, Menlo Park
anuroops@fb.com

Vitaliy Liptchinsky
Facebook, Menlo Park
vitaliy888@fb.com

Ronan Collobert[*]
Facebook, Menlo Park
locronan@fb.com

March 23, 2020

Abstract

We study pseudo-labeling for the semi-supervised training of ResNet, Time-Depth Separable ConvNets, and Transformers for speech recognition, with either CTC or Seq2Seq loss functions. We perform experiments on the standard LibriSpeech dataset, and leverage additional unlabeled data from LibriVox through pseudo-labeling. We show that while Transformer-based acoustic models have superior performance with the supervised dataset alone, semi-supervision improves all models across architectures and loss functions and bridges much of the performance gaps between them. In doing so, we reach a new state-of-the-art for end-to-end acoustic models decoded with an external language model in the standard supervised learning setting, and a new absolute state-of-the-art with semi-supervised training. Finally, we study the effect of leveraging different amounts of unlabeled audio, propose several ways of evaluating the characteristics of unlabeled audio which improve acoustic modeling, and show that acoustic models trained with more audio rely less on external language models.

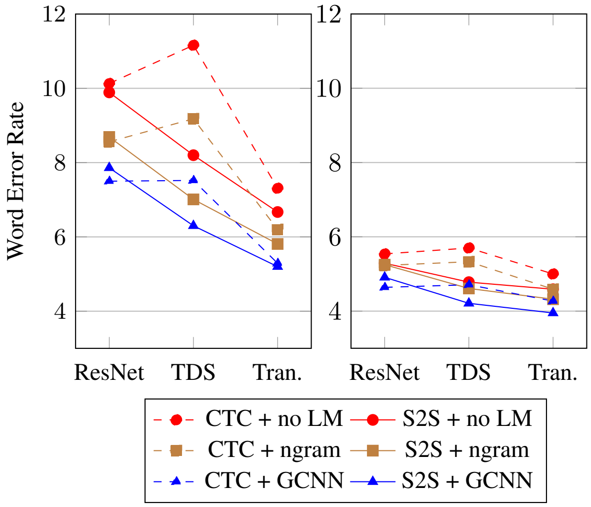
1 Introduction

End-to-end speech recognition models are simpler to implement and train than bootstrapped systems. Even given recent promising results from these systems, best-results for common benchmarks are still dominated by classical ASR models; systems requiring force alignment may leave some performance aside for each training step. We set out to study end-to-end systems on LibriSpeech [37] and, without any algorithmic contribution, see if they can be made to perform as well as more complex training pipelines. The difficulties involved in properly optimizing acoustic models with Connectionist Temporal Classification (CTC) [16] or sequence-to-sequence (Seq2Seq) [47] (v.s. cross-entropy, for instance) combined with more readily-available regularization techniques for classical pipelines make this comparison challenging. Our best acoustic models nonetheless reach 5.17% WER on `test-other`, showing that end-to-end models can compete with traditional pipelines.

As in other domains, self and semi-supervised learning in ASR, where a pretrained network generates and trains on its own labels, yields improvements [49]. In end-to-end ASR, pseudo-labeling and self-training can be quite

[*] Equal contribution.

Figure 1: WERs on `dev-other` across AM architectures and loss functions. *Left*: WERs of different models trained on LIBRISPEECH with and without beam-search decoding. Transformer AM architectures outperform others by a large margin. *Right*: WERs of models trained on LIBRIVOX. All models trained on LIBRIVOX significantly outperform their LIBRISPEECH counterparts. The gap between Transformer AMs and other models is much smaller with LIBRIVOX data.



effective, and its effectiveness is further improved when more data is available [25]. In this setting, we train a model on LIBRISPEECH, then use that model in conjunction with a language model to generate pseudo-labels from unlabeled audio. We show that with this training scheme, our results *without an external language model* (LM) reach state-of-the-art results that *use an external language model*, with 2.28% and 4.88% Word Error Rate (WER) on `test-clean` and `test-other` respectively. With LM beam-search decoding and rescoring, we reach 2.09% and 4.11% WER on the test set.

While many advances in end-to-end ASR come as the result of neural architecture search [41, 55, 6], we additionally show that simple semi-supervision via pseudo-labeling significantly bridges the performance gap between a variety of different model architectures and loss functions, as shown in Figure 1. In particular, with enough unlabeled audio, Transformer, ResNet, and depthwise-separable convolution-based acoustic models give similar performance with both CTC and Seq2Seq loss functions, suggesting that new techniques in semi-supervision may facilitate equally-significant gains in ASR performance while being applicable to a multitude of end-to-end setups.

2 Models

2.1 Acoustic Models

In this section, we present the three families of acoustic models (AMs) studied. All AMs output probability distributions over tokens. In particular, we use a set of 10k word pieces [44, 29] generated from the *SentencePiece* toolkit[1]. The choice to use a fixed set of 10k word pieces is made for the simplicity of the comparative study, not the result of a limitation. Similarly, all AMs take 80-channel log-melfiltrbanks as input, with STFTs computed on Hamming windows strided by 10ms. This window size is 25ms for Transformer models and 30ms for TDS and ResNet models. All models are trained end-to-end with either CTC or Seq2Seq loss. Given the huge difference between the amounts of data, we prepare two sets of architectures: one for training only on labeled LIBRISPEECH and one for unlabeled LIBRIVOX.

ResNet Acoustic Models. ResNets were first introduced in the domain of computer vision [21] and have since been successfully applied to speech recognition [53, 43, 31, 50]. ResNets are composed of several blocks of convolutions (in our case only 1-D convolutions), with skip connections to reduce the effect of vanishing gradients in deep neural networks. In particular, our ResNet encoder includes 42 convolutional layers each with a kernel size of 3. The encoder first maps the input to an embedding space of size 1024 using a single convolutional layer with stride 2; 12 blocks of three 1-D convolutions each follow. Each of the convolutional layers is followed by ReLU, dropout and

[1] `https://github.com/google/sentencepiece`

LayerNorm [2]. Both the dropout and the number of hidden units increases with the depth of the network. Specific convolution layers are inserted between ResNet blocks in order to upsample when the hidden representation size increases. Our architecture performs significant pooling with respect to the input (16 frames in total, equating to 160 milliseconds) – in addition to the first strided convolutional layer, 3 max pooling layers (each with stride 2) are distributed across the depth of the network (after blocks 3, 7 and 10). Nearly identical encoder architectures are used in front of CTC and Seq2Seq loss functions; the Seq2Seq encoder has its last bottleneck layer removed and lower dropout in deeper layers. The Seq2Seq self-attention decoder for the ResNet architecture is the same as that used with the TDS convolutional AM described below. To better fit the unlabeled data, we increase the model size by increasing the number of channels in each convolution layer.

Time-Depth Separable (TDS) Convolution Acoustic Models. We extend the TDS block [20] (which is composed of one 2-D convolution layer and two fully-connected layers with ReLU, LayerNorm and residual connections in between), by increasing the number of channels in the feature maps spanning the two internal fully-connected layers by a factor $F > 1$, so as to increase model capacity. Following [20], 3 sub-sampling layers, i.e. 1-D convolution layers with stride 2, are adopted to ensure an optimal context size for the encoder. For training with only labeled data, we have three groups of TDS blocks with $F = 3$ after each sub-sampling layers. There are 5, 6, and 10 blocks in each group, containing 10, 14, and 18 channels, respectively. To increase model capacity for unlabeled data, the three groups of TDS blocks, having fewer 4, 5, and 6 blocks and $F = 2$ in each, are equipped with much larger 16, 32, and 48 channels. All convolutions in both TDS and sub-sampling layers have kernel shapes of 21×1. Identical encoder architectures are shared between CTC and Seq2Seq.

Our Seq2Seq self-attention decoder performs R rounds of attention through the same N-layers of RNN-GRU each with a hidden unit size of 512 in conjunction with the same efficient key-value attention as in [20, 48]:

$$\mathbf{S}^r = \textsc{SoftMax}(\frac{1}{\sqrt{d}}\mathbf{K}^\top \mathbf{Q}_t^{r-1})\mathbf{V}, \tag{1}$$

where $[\mathbf{K}, \mathbf{V}]$ is 512-dimensional encoder activation and $\mathbf{Q}_t^r = g(\mathbf{Q}_{t-1}^r, \mathbf{Q}_t^{r-1}) + \mathbf{S}^r$ is the query vector at time t in round r, generated by the GRU $g(\cdot)$. The initial $\mathbf{Q}_t^0$ is a 512-dimensional token embedding, and the final $\mathbf{Q}_t^R$ is linearly projected to output classes for token classification. In our experiments, N and R are both set to either 2 or 3 based on validation performance. We use dropout in all TDS blocks and GRUs to prevent overfitting.

Transformer-Based Acoustic Models. Our transformer-based acoustic models have a small front-end: 3 (LIBRISPEECH AMs) or 6 (LIBRIVOX AM) layers of 1-D convolutions each of kernel width 3 and respective input and output sizes $(80, D_c)$, $(D_c/2, D_c)$, $[(D_c/2, D_c)$, $(D_c/2, D_c)$, $(D_c/2, D_c),]$ $(D_c/2, D_{tr} \times 2)$, with $D_c = 1024$ or 2048. Each convolution is followed by a GLU activation function [12] and are striding by 2 each (for 3 consecutive layers), or every other layer (for 6 layers). The output of the front-end for all models is thus strided by 8 frames (80 ms). After the front-end, each Transformer block has 4 attention heads followed by a feedforward network (FFN) with one hidden layer and a ReLU non-linearity. There are two configurations of Transformer blocks: one 24 layer configuration (only for the LIBRISPEECH CTC AM) with dimension $D_{tr} = 1024$ for the self-attention and 4096 for the FFN, and one 36 layer configuration with dimension $D_{tr} = 768$ for the self-attention and 3072 for the FFN. Specifically, given a sequence of T vectors of dimension d, the input is represented by the matrix $\mathbf{H^0} \in \mathbb{R}^{d \times T}$, following exactly [48]:

$$\begin{aligned}\mathbf{Z}^i &= \textsc{Norm}(\textsc{SelfAttention}(\mathbf{H}^{i-1}) + \mathbf{H}^{i-1}),\\ \mathbf{H}^i &= \textsc{Norm}(\text{FFN}(\mathbf{Z}^i) + \mathbf{Z}^i),\end{aligned}$$

where $\mathbf{Z}$ is the output of the self-attention layer, with a skip connection, and $\mathbf{H}$ is the output of the FFN layer, with a skip connection. As is standard: our NORM is LayerNorm, and self-attention is defined as in Eq. 1, but with $\mathbf{K} = \mathbf{W}_K\mathbf{H}$, $\mathbf{Q} = \mathbf{W}_Q\mathbf{H}$, and $\mathbf{V} = \mathbf{W}_V\mathbf{H}$. For CTC-trained models, the output of the encoder $\mathbf{H}^{L_e}$ is followed by a linear layer to the output classes. For Seq2Seq models, we have an additional decoder, which is a stack of 6 Transformers with encoding dimension 256 and 4 attention heads. The probability distribution of the transcription is factorized as:

$$p(y_1, ..., y_n) = \prod_{i=1}^{n} p(y_i \mid y_0, ..., y_{i-1}, \mathbf{H}^{L_e}), \tag{2}$$

where y_0 is a special symbol indicating the beginning of the transcription. For all layers (encoder and decoder – when present), we use dropout on the self-attention. We also use layer drop [14], dropping entire layers at the FFN level.

2.2 Language Models

In this section, we present external language models (LMs) used in beam-search decoding. We consider n-gram LMs as well as convolutional [12] (GCNN) and Transformer-based LMs. For n-gram and GCNN LMs, we train both word-based and word-piece models, and only a word-level Transformer LM. All word-piece LMs are trained on the set of 10k word pieces as outlined in Section 2.1. This ensures that the set of word pieces is consistent across both of the output distributions of the acoustic models and the candidates the language model scores during beam-search decoding.

For the word-piece and word-level GCNN models, we use the GCNN-14B architecture from [12] with embedding size 1024 and dropout 0.1. The word-level Transformer LM has the same architecture as [3]'s *Google Billion Words* model; we use 16 attention heads and 20 decoder layers with embedding, input and output dimensions of 1280 and 6144 for the FFN with dropout of 0.1.

3 Dataset and Language Models for Pseudo-Labeling

3.1 Unlabeled Audio Dataset Preparation

LIBRIVOX[2] is a large collection of freely-available audiobooks. Using tools provided with the LIBRI-LIGHT dataset [26], we select 72K hours of read speech from English book listings and run several preprocessing steps. After filtering samples to remove readings of duplicate text and corrupted audio, we remove all audio for which the speaker has overlap with a sample in LIBRISPEECH. We run voice action detection (VAD) using the wav2letter++ framework [42] on the resulting collection of audio with a CTC model trained on LIBRISPEECH, and segment the result into chunks no greater than 36 seconds; the resulting audio corpus contains 53.8K hours of read speech.

We then generate pseudo-labels for this audio using the recipe described in [25]. To generate the pseudo-labels, we use a Transformer AM trained on LIBRISPEECH with CTC loss that achieves a 6.20% WER on dev-other when decoded with a 4-gram word LM – the same model as is listed in Table 5 in the Appendix. We pseudo-label all audio using this AM and run beam-search decoding with a 4-gram word LM described below in Section 3.2.

3.2 Text Corpus Preparation and n-gram LM Training

The LIBRISPEECH language model corpus[3] contains text from 14500 public domain books taken from the Gutenberg project[4]. Given that pseudo-labels are generated with a beam-search decoding procedure that integrates a language model, it is important that the corpus used to train the language model does not have overlap with the unlabeled audio, else information about the ground truth labels for that unlabeled audio may be explicitly embedded in the LM. We remove all text from the LIBRISPEECH language model training corpus that is ground truth for any of the unlabeled audio from the subset of LIBRIVOX.

To do so, we follow several steps. Firstly, we filter out all books from the LIBRISPEECH LM corpus with IDs present in LIBRIVOX. Secondly, after normalizing all titles (removing punctuation, casing, and non-alphanumeric tokens), we remove all titles with zero Levenshtein distance between titles from the LIBRIVOX and the LIBRISPEECH LM corpuses. We use a Levenshtein metric over words rather than tokens for improved performance. We then find titles with nonzero but low similarity scores isolated via the following conditions. Given two book title strings s_1 and s_2, and constants α and β:

$$\max\{|s_1|, |s_2|\} - \min\{|s_1|, |s_2|\} < \alpha \cdot \min\{|s_1|, |s_2|\} \ \ \&$$

$$\text{Levenshtein}(s_1, s_2) \leq \beta \cdot \max\{|s_1|, |s_2|\}$$

where 0.75 and 0.3 were used as values for α and β, respectively. Titles that are manually matched are removed to create the final corpus; 13% of the original LIBRISPEECH-LM corpus was filtered with the aforementioned steps.

Before training LMs, we normalize the filtered corpus so as to mimic the original normalization procedure found in LIBRISPEECH. 88% of our normalized/filtered corpus has identical normalized text compared to the original LIBRISPEECH LM corpus. As a result of our using a different tokenizer, sentence boundaries may differ across corpuses, as may abbreviations (e.g. we map '&c' to 'et cetera').

A 4-gram language model is trained with the resulting corpus using the KenLM toolkit [23] and the top 200k words as vocabulary. The model is trained without pruning (183k of the top 200k words are the same as the original LIBRISPEECH LM corpus). This model is then used at beam-search decoding time in conjunction with an acoustic model trained on LIBRISPEECH to generate pseudo-labels on the subset of LIBRIVOX detailed in Section 3. During beam-search decoding we use a lexicon which is constructed from the LIBRISPEECH train sets only.

[2] https://librivox.org

[3] http://www.openslr.org/11/

[4] https://www.gutenberg.org/

The perplexity difference between the 4-gram LM trained on the filtered corpus and the 4-gram LM trained on original LIBRISPEECH LM corpus is small. The word perplexity of each model is shown in Table 1. Beam-search decoding of a Transformer AM trained on LIBRISPEECH with an LM trained on the filtered corpus results in only a 0.05% absolute WER regression on `dev-other` compared to decoding with an n-gram trained on the full corpus.

4 Decoding

Decoding is designed to select the best transcription by leveraging both the posteriors of an acoustic model (AM) and the perplexity of a language model (LM). We perform one-pass beam-search decoding with a single external LM. Optionally, to further improve performance, we use stronger NN-based LMs to rescore the beam.

4.1 Beam-search Decoder

In our experiments, we use lexicon-based and lexicon-free beam-search decoders following [8, 32] with either n-gram or GCNN LMs. The lexicon-based decoder, whose search space is limited to the words in the lexicon, is used for CTC models with a word-level LM. The lexicon-free decoder is capable of generating words with arbitrary spelling and is used for S2S models with a word-piece LM. The decoder takes as input posteriors from an acoustic model, a prefix trie built on a lexicon, and an external LM. We tune the language model weight α and the word insertion penalty β on validation sets (`dev-clean` and `dev-other`). The decoder outputs a transcription $\hat{\mathbf{y}}$ that maximizes:

$$\log P_{AM}(\hat{\mathbf{y}}|\mathbf{x}) + \alpha \log P_{LM}(\hat{\mathbf{y}}) + \beta|\hat{\mathbf{y}}|.$$

To stabilize the Seq2Seq beam search, we introduce an EOS-penalty γ to hypothesis that have finished in an end-of-sentence token. γ is tuned together with other hyper-parameters and our experiments show that this strategy effectively prevents the decoder from early-stopping. To improve decoding efficiency, we also incorporate the thresholding technique in [20] and strategies mentioned in [54] including hypothesis merging, score caching, and batched LM forwarding. For CTC decoding, following [39], only the blank token is considered if its posterior probability is greater than 0.95.

4.2 Rescoring

After acquiring the transcriptions of the N-best hypotheses from the one-pass beam-search decoder, we use an external word-level GCNN LM and a Transformer LM to evaluate their log-probabilities, denoted as $\log P_1(\hat{\mathbf{y}})$ and $\log P_2(\hat{\mathbf{y}})$ respectively. We then perform rescoring to reorder the hypotheses according to the following score:

$$\log P_{AM}(\hat{\mathbf{y}}|\mathbf{x}) + \alpha_1 \log P_1(\hat{\mathbf{y}}) + \alpha_2 \log P_2(\hat{\mathbf{y}}) + \beta|\hat{\mathbf{y}}|,$$

where α_1, α_2, β are hyper-parameters of the rescoring algorithm optimized on the validation set and $|\hat{\mathbf{y}}|$ is the transcription length in characters (including the spaces between words). In order to diversify the hypotheses in the beam, to increase the probability that the correct transcription is included, we usually relax the threshold in the decoder and increase beam size when dumping beam candidates.

5 Experiments

5.1 Technical Details

We use the standard splits for LIBRISPEECH and the standard LIBRISPEECH LM corpus for LM training. Models are trained using the wav2letter++[5] toolkit [42]; we will release reproduction steps and pre-trained models.

5.1.1 Acoustic Model Training

All hyper-parameters including model architecture are cross-validated on `dev-clean` and `dev-other`. Given that we have a large family of models, for simplicity and clarity, we only report ranges of configurations and hyper-parameters.

We use plain SGD with momentum to train ResNet and TDS models, and Adagrad [13] to train Transformers. Models are trained on 64 GPUs each with an overall batch size of 256 for ResNet and TDS and 320 for Transformer. With only LIBRISPEECH, all models converged in under a week; with pseudo-labels from LIBRIVOX, training required 2-3 weeks. The initial learning rate for ResNet models is [0.05, 0.5], while for TDS and Transformer models, the range

[5] `https://github.com/facebookresearch/wav2letter`

Table 1: Word-level perplexities of LMs on LibriSpeech. Perplexity is computed without unknown words.

Language Model	dev-clean	dev-other
word 4-gram	148.0	136.6
w/o LibriVox	152.8	140.0
wp 6-gram	145.4	133.7
wp GCNN (188M)	61.7	61.9
word GCNN (319M)	57.0	57.9
word Transf. (429M)	48.2	50.2

decreases to [0.01, 0.03]. Specifically, for Transformers, we apply a linear learning rate warm-up schedule for either 32k or 64k updates. For fully-supervised training with LibriSpeech, the learning rate is halved every 90 epochs for Transformer models, and 150 epochs for ResNet and TDS models. With LibriVox, however, we only halve the learning rate once in the middle of the training. For TDS and ResNet models, we use momentum in the range [0.1, 0.6]. With respect to regularization, we use 0.2 dropout everywhere (front-end, encoder, decoder), and layer drop for all Transformer blocks. Dropout in TDS blocks and ResNet convolutions is in the range [0.05, 0.2] and increases with depth. For Seq2Seq training, we run 3 epochs of attention-window pretraining, and use 99% of teacher forcing (1% of uniform output sampling). We also use 10% dropout in the decoder for TDS (and 10% dropout and 10% layer drop in the decoder for Transformers), together with 5% label smoothing, 1% random sampling and 1% word piece sampling. We also use SpecAugment [38] with an LD policy for all models.

5.1.2 Language Model Training

All LMs in this section are trained on the standard LibriSpeech LM corpus. All word-level LMs use the same vocabulary for training. n-gram LMs are trained with the KenLM toolkit [23], while the GCNN and Transformer LMs are trained with fairseq toolkit[6] [36]. The word-level 4-gram and GCNN are trained in the same way as [32]. We also train a 6-gram word-piece LM, which has a similar context size to a word-level 4-gram LM, and prunes 5-grams appearing once and 6-gram appearing twice or fewer. The word-piece and word-level GCNN models are trained with Nesterov accelerated gradient descent [35] on 8 GPUs for 22 epochs with a step-wise learning rate schedule starting from 1 and decreasing by a factor of 5 when loss stabilized. Gradient clipping and weight normalization are used following [12]. The word-level Transformer LM is trained with Nesterov accelerated gradient descent on 128 GPUs for 100 epochs with an inverse square root learning rate schedule. During the first 16000 iterations, a warm-up schedule that linearly increases the learning rate from 0 to 1 is used. Word-level perplexities of all LM variants can be found in Table 1.

5.2 Results

LibriSpeech Results. All our results for LibriSpeech are listed in Table 5 in the Appendix. We present results under three scenarios: without any decoding nor external LM, with one-pass decoding only, and with decoding followed by beam rescoring. The decoding beam size is usually 50 and 500 for Seq2Seq and CTC respectively. We use a beam size of 250 for CTC decoding with a GCNN LM. We train strong baselines on simple ResNet architectures and improve the TDS models significantly compared to past results [20]. These convolutional models outperform end-to-end biLSTM models from [33]. Our best acoustic models are Transformers-based and reach 6.98% without any decoding on `test-other`and 5.17% with decoding and rescoring, demonstrating that end-to-end training can perform as well as traditional bootstrapped systems.

LibriVox Results. Assuming all pseudo-labels are ground-truth, we train acoustic models on a combination of the 960 hours of labeled audio from LibriSpeech in conjunction the pseudo-labeled audio from LibriVox, where batches are uniformly sampled (without weighting) from both LibriSpeech and LibriVox datasets. Transformer AMs with both CTC and Seq2Seq loss were trained for 5 days on this combined dataset, achieving WERs on `test-other` of 4.88% and 2.28% on `test-clean` without decoding or use of an LM, which is state-of-the-art even amongst pipelines that use an LM. Results with decoding/rescoring are shown in Table 2, where we reach 2.09% and 4.11% on `test-clean` and `test-other` , respectively, and are further improvements on the state-of-the-art.

Table 2: WERs on LibriSpeech development and test sets. Our best results are shown in the bottom section (with the number of parameters), and are both trained with Seq2Seq loss. Full results can be found in Table 5 in the Appendix.

AM		LM		Dev		Test	
type	lexicon	type	lexicon	clean	other	clean	other
Fully Conv. [54]	letter	GCNN	word	3.1	9.9	3.3	10.5
Conv. Transf. [34]	5k WP	-	-	4.8	12.7	4.7	12.9
TDS Conv. [20]	10k WP	GCNN	-	5.0	14.5	5.4	15.6
Decoding	10k WP	GCNN	10k WP	3.0	8.9	3.3	9.8
LAS [38]	16k WP	-	-			2.8	6.8
Decoding	16k WP	RNN	16k WP			2.5	5.8
biLSTM + attn. [33]	10k BPE	-	-	4.3	12.9	4.4	13.5
+ Transf. decoding	10k BPE	Transf.	10k BPE	2.6	8.4	2.8	9.3
HMM/biLSTM [33]	12k CDp	4gram+LSTM	word	2.2	5.1	2.6	5.5
+ Transf. rescoring	12k CDp	+ Transf.	word	1.9	4.5	2.3	5.0
Transformers [28]	BPE	RNN	word	2.2	5.6	2.6	5.7
Conv. Transf. [19]	6k triphones	3gram, rescored + TDNN + LSTM	word	1.8	5.8	2.2	5.7
Conv. Transf. [51]	chenones	4gram	word			2.60	5.59
+ Transf. rescoring	chenones	Transf.	word			2.26	4.85
Transf. (270M) – LibriSpeech	10k WP	-	-	2.54	6.67	2.89	6.98
+ Decoding/Rescoring	10k WP	GCNN + Transf.	word	2.10	4.79	2.33	5.17
Transf. (296M) – LibriVox	10k WP	-	-	2.12	4.59	2.28	4.88
+ Decoding/Rescoring	10k WP	GCNN + Transf.	word	2.00	3.65	**2.09**	**4.11**

Table 3: WERs of a Transformer AM architecture outlined in section 2.1 trained with Seq2Seq loss on LibriSpeech with different amounts of pseudo-labeled audio from LibriVox.

Training Dataset (Hours)	dev-clean	dev-other
LS only	2.54	6.67
LS + 1k LV	2.35	5.56
LS + 3k LV	2.21	5.16
LS + 10k LV	2.11	4.95
LS + 53.8k LV	2.11	4.59

5.3 Ablations

Varying the amount of unlabeled audio. In this study, we train on several different randomly-selected subsets of pseudo-labels from the original collection generated as described in Section 3. Results are given in Table 3. Increasing the amount of pseudo-labels strictly improves performance. The listed 53.8k hour result is using the fully-prepared dataset as outlined in Section 3.1. WERs given are without decoding after 800k iterations of training.

Generating pseudo-labels with an LM containing overlapping text. As discussed in Section 3.1, using an LM to generate pseudo-labels that was trained with a corpus that includes ground truth text from unlabeled audio introduces an overlap that may unrealistically improve the quality of pseudo-labels. We show that the effect of using an LM trained with a small amount of overlapping text to generate pseudo-labels has only a small effect on the performance of models trained on those pseudo-labels.

Table 4 contains results for Transformer AMs with both CTC and Seq2Seq loss as described in 2.1 trained on pseudo-labels generated with a decoding step that uses an LM trained on an overlapping versus non-overlapping corpus. The models used are of the same architecture as described in Section 2.1. There is a small improvement in `dev-other` performance for pseudo-labels generated from an overlapping LM, but both models generalize very similarly.

Training on pseudo-labels only. Models trained on LibriVox pseudo-labels alone outperform models trained on LibriSpeech. As outlined in Section 5, all acoustic models are trained on a combination of LibriSpeech and pseudo-labeled LibriVox audio. In this setup, it is difficult to disambiguate the importance of the pseudo-labeled audio compared to supervised data from LibriSpeech. To test the quality of pseudo-labels in isolation, we trained a

[6] https://github.com/pytorch/fairseq

Table 4: WERs of a Transformer AM when trained with pseudo-labels generated with a decoder integrating an LM that contains overlapping text with unlabeled audio versus an LM with no overlap. Results are shown after decoding with the word 4-gram language model described in Section 2.2.

MODEL	OVERLAP	DEV-OTHER	TEST-OTHER
TRANS. S2S	NO	4.58	4.90
	YES	4.51	4.87
TRANS. CTC	NO	4.92	5.47
	YES	4.80	5.33

CTC-based Transformer model similar to that described in Section 2.1 to compare directly with the CTC-based transformer AM used to generate the pseudo-labels described in Section 3. We compare the resulting AM-only performance on the LIBRISPEECH development sets. Without decoding, the resulting LIBRIVOX pseudo-label-only model achieves WERs of 2.38% and 5.43% on `dev-clean` and `dev-other` respectively, which improves over the LIBRISPEECH-only baseline's 2.99% and 7.31%, respectively. The volume, quality, and diversity of the generated pseudo-labels alone are sufficient to generate superior results as compared to a model trained only on LIBRISPEECH. The model trained on LIBRISPEECH and LIBRIVOX pseudo-labels achieves an improved 2.28% and 4.99% on `dev-clean` and `dev-other`, respectively.

5.4 End-to-End Acoustic Models Learn a Language Model: Removing the LM from ASR

In the sections that follow, we show two results. We first give a simple experimental framework to demonstrate that acoustic models trained on speech learn nontrivial language models, and that training on additional audio facilitates learning better acoustic representations. We then show that with a large collection of pseudo-labeled audio, well-trained acoustic models no longer benefit much from decoding with an external language model in most cases.

AMs learning LM: transcribing shuffled audio. The language modeling properties of end-to-end acoustic models are briefly discussed in [4], where an AM trained with CTC is shown to learn an implicit language model based on its predicted posteriors for words with multiple spelling variants. Still other results show that fusing an LM with an AM during training can improve performance [46, 7, 52]. These previous works use RNN-based acoustic models, which possess infinite receptive fields and processes most or all of an input utterance during a single forward pass. We show that modern convolutional architectures have large receptive fields and also condition on internal word representations learned directly from audio.

Figure 2: `dev-other` WERs without decoding across acoustic models and loss functions for original and shuffled versions of `dev-other` across three settings. Each plot uses the following original and shuffled audio: *Left*: original and shuffled `dev-other` audio segmented using ASG. *Middle*: audio generated by TTS vocoder for the original and shuffled transcriptions from `dev-other`. *Right*: original and shuffled audio for a subset of `dev-other` recorded by the paper's authors.

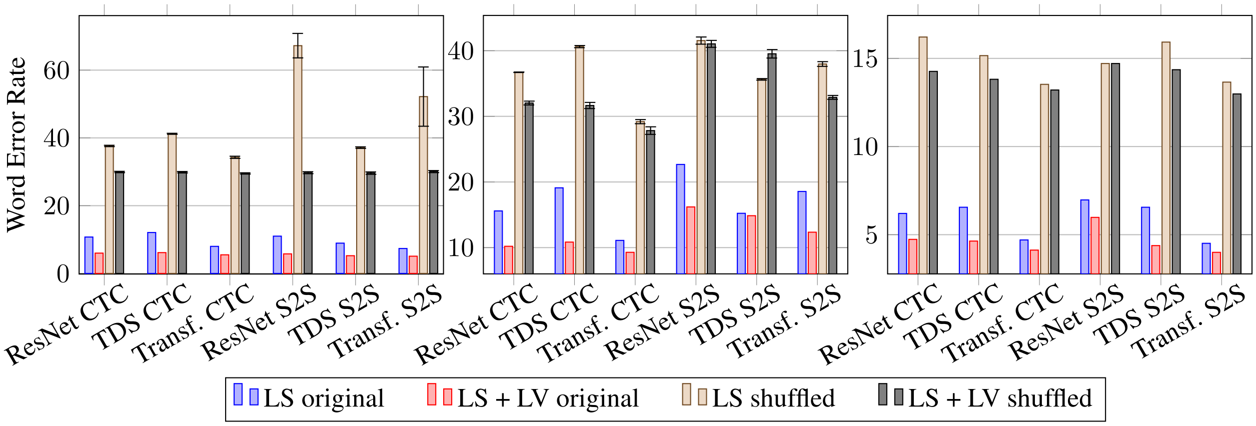
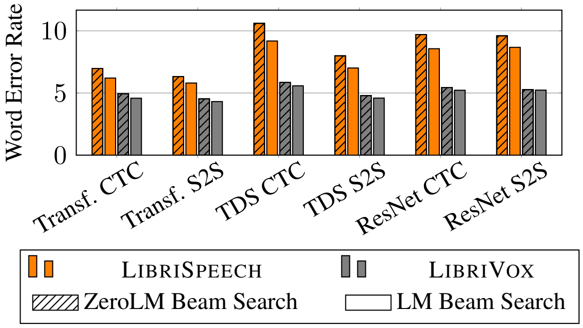


Figure 3: WER on `dev-other` for models trained on LibriSpeech and LibriSpeech + LibriVox after decoding with and without the 4-gram LM described in Section 2.2. The gain from LM beam-search decoding for models trained on LibriVox is much smaller compared to that for models trained on LibriSpeech.



If an AM learns a robust LM, the acoustic model will less effectively predict utterances of high underlying word-perplexity; the model will rely on its acoustic representations to predict words without context, providing a good proxy for the quality of its learned acoustic representations. In the experiments that follow, we introduce a simple "shuffled transcription" task in which models transcribe LibriSpeech `dev-other` with utterances corresponding to both unshuffled and shuffled transcriptions. Experiments are performed in three audio settings to eliminate bias when scrambling words. First, with a TTS model, unshuffled and shuffled sentences are forwarded through a WaveRNN vocoder [27] trained on the LJSpeech dataset[7] using the Mozilla TTS toolkit[8]. In the second setting, audio is segmented at the word level using a convolutional stride 2 letter-based AM trained with ASG loss [8], then re-spliced together in the given shuffled order. Finally, the paper's authors recorded unshuffled and shuffled utterances from a subset of `dev-other`.

Figure 2 contains the WERs across audio settings on `dev-other` without decoding. Both CTC and Seq2Seq models perform poorly across the board on shuffled audio, but AMs trained with LibriVox pseudo-labels are able to learn better acoustic representations which improve performance on shuffled inputs for which their internal LMs is not predictive. Full results can be found in Table 9 in the Appendix.

With enough unlabeled audio, decoding with an LM doesn't improve performance. The importance of the language model to the success of the pseudo-labeling is known; [25] show that in the end-to-end setting, as the quality of the language model used to generate the pseudo-label decreases even marginally, the quality of the model trained on the resulting pseudo-labels decreases. In what follows, we show that through the self-training procedure, decoding an acoustic model trained on LibriVox pseudo-labels generated with the help of a language model gives very small improvements compared to models trained only on LibriSpeech.

Results are shown in Figure 3. We use a beam-search decoding procedure without an LM ("Zero-LM") to disambiguate the effect of beam search on WER, and evaluate on `dev-other` to provide a better lower bound for how much decoding with the LM can improve performance (decoder parameters are also optimized on `dev-other`). The models for which results are shown are trained on pseudo-labels from LibriVox generated with an n-gram language model without an overlapping text corpus (see the ablation in Sections 5.3 and 2.2). Decoding with the LM gives little to no gain for models trained on LibriSpeech + LibriVox and a much more significant gain for those models trained only on LibriSpeech, suggesting information from the 4-gram LM used to generate pseudo-labels on LibriVox has thoroughly diffused into AMs trained with those labels. Full results can be found in Table 5 in the Appendix.

6 Related Work

Deep neural networks were reintroduced in ASR with HMMs [24], and many of state-of-the-art models still rely on force alignment [18, 33, 28]. Nonetheless, there have been increasingly competitive end-to-end results trained with CTC [15, 1], ASG [8, 54], LF-MMI [17], sequence-to-sequence [4, 5], transduction [41, 22], and differentiable decoding [9]. *Listen Attend and Spell* [4] is a family of end-to-end models based on biLSTMs which achieved state-of-the-art results with improved regularization through data augmentation [38]; we consequently use SpecAugment in all of our experiments. Seq2Seq models are not limited to RNNs; time-depth separable convolutions also give strong

[7] https://keithito.com/LJ-Speech-Dataset/

[8] https://github.com/mozilla/TTS

results [20]. Our best models are transformer-based, as [33, 28], which give good results in Seq2Seq settings even without external LMs [34]. In ASR, semi-supervised pseudo-label-style self-training has been explored generally in end-to-end settings in [45, 30, 25] for both low-resource [49, 11] and large-scale [40] setups.

7 Discussion

We presented state-of-the-art results on LIBRISPEECH with end-to-end methods. While allowing for lexicon-free decoding, the 10k word-piece tokens used during training limit the amount of striding we can use in our model architectures and can be replaced by AMs outputting words with an arbitrary lexicon [10]. As relative WER gains due to language models shrink (from ≈20% relative-WER without LIBRIVOX to ≈10% with, for GCNN decoding), and as we showed that AMs learn LM-level information, differentiable decoding [9] is a possible avenue for single-stage AM + LM joint training.

We show the effectiveness of a simple pipeline that does not require many training steps. In light of our semi-supervised results without decoding or an LM, we think Seq2Seq/CTC losses, transducers, and differentiable decoding are viable methods to achieve end-to-end state-of-the-art results, without external LMs, through semi-supervised learning.

References

[1] Amodei, D., Ananthanarayanan, S., Anubhai, R., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pp. 173–182, 2016.

[2] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Baevski, A. and Auli, M. Adaptive input representations for neural language modeling. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=ByxZX20qFQ`.

[4] Chan, W., Jaitly, N., Le, Q. V., and Vinyals, O. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *ICASSP*, 2016.

[5] Chiu, C.-C., Sainath, T., Wu, Y., et al. State-of-the-art speech recognition with sequence-to-sequence models. *ICASSP*, 2018.

[6] Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, E., et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778. IEEE, 2018.

[7] Chorowski, J. and Jaitly, N. Towards better decoding and language model integration in sequence to sequence models. *arXiv preprint arXiv:1612.02695*, 2016.

[8] Collobert, R., Puhrsch, C., and Synnaeve, G. Wav2letter: an end-to-end convnet-based speech recognition system. *arXiv preprint arXiv:1609.03193*, 2016.

[9] Collobert, R., Hannun, A., and Synnaeve, G. A fully differentiable beam search decoder. In *ICML*, pp. 1341–1350, 2019. URL `http://proceedings.mlr.press/v97/collobert19a.html`.

[10] Collobert, R., Hannun, A., and Synnaeve, G. Word-level speech recognition with a dynamic lexicon. *arXiv preprint arXiv:1906.04323*, 2019.

[11] Cui, J., Kingsbury, B., Ramabhadran, B., Saon, G., Sercu, T., Audhkhasi, K., Sethy, A., Nussbaum-Thom, M., and Rosenberg, A. Knowledge distillation across ensembles of multilingual models for low-resource languages. 2017.

[12] Dauphin, Y. N., Fan, A., Auli, M., and Grangier, D. Language modeling with gated convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 933–941. JMLR.org, 2017.

[13] Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(Jul):2121–2159, 2011.

[14] Fan, A., Grave, E., and Joulin, A. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019.

[15] Graves, A. and Jaitly, N. Towards end-to-end speech recognition with recurrent neural networks. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pp. II–1764–II–1772. JMLR.org, 2014. URL `http://dl.acm.org/citation.cfm?id=3044805.3045089`.

[16] Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376, 2006.

[17] Hadian, H., Sameti, H., Povey, D., and Khudanpur, S. End-to-end speech recognition using lattice-free mmi. In *Interspeech*, pp. 12–16, 2018.

[18] Han, K. J., Chandrashekaran, A., et al. The capio 2017 conversational speech recognition system, 2017.

[19] Han, K. J., Prieto, R., Wu, K., and Ma, T. State-of-the-art speech recognition using multi-stream self-attention with dilated 1d convolutions, 2019.

[20] Hannun, A., Lee, A., Xu, Q., and Collobert, R. Sequence-to-sequence speech recognition with time-depth separable convolutions. *Interspeech 2019*, Sep 2019. doi: 10.21437/interspeech.2019-2460.

[21] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

[22] He, Y., Sainath, T. N., Prabhavalkar, R., et al. Streaming end-to-end speech recognition for mobile devices. *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019. doi: 10.1109/icassp.2019.8682336.

[23] Heafield, K. Kenlm: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*, pp. 187–197. Association for Computational Linguistics, 2011.

[24] Hinton, G., Deng, L., Yu, D., Dahl, G., rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*, 2012.

[25] Kahn, J., Lee, A., and Hannun, A. Self-training for end-to-end speech recognition. *arXiv preprint arXiv:1909.09116*, 2019.

[26] Kahn, J., Rivire, M., Zheng, W., Kharitonov, E., Xu, Q., Mazar, P., Karadayi, J., Liptchinsky, V., Collobert, R., Fuegen, C., Likhomanenko, T., Synnaeve, G., Joulin, A., Mohamed, A., and Dupoux, E. Libri-light: A benchmark for asr with limited or no supervision, 2019.

[27] Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A. v. d., Dieleman, S., and Kavukcuoglu, K. Efficient neural audio synthesis. *arXiv preprint arXiv:1802.08435*, 2018.

[28] Karita, S., Chen, N., Hayashi, T., et al. A comparative study on transformer vs rnn in speech applications, 2019.

[29] Kudo, T. and Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

[30] Li, B., Sainath, T. N., Pang, R., and Wu, Z. Semi-supervised training for end-to-end models via weak distillation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2837–2841. IEEE, 2019.

[31] Li, J., Lavrukhin, V., Ginsburg, B., Leary, R., Kuchaiev, O., Cohen, J. M., Nguyen, H., and Gadde, R. T. Jasper: An end-to-end convolutional neural acoustic model. In *Interspeech*, 2019.

[32] Likhomanenko, T., Synnaeve, G., and Collobert, R. Who needs words? lexicon-free speech recognition. *arXiv preprint arXiv:1904.04479*, 2019.

[33] Lscher, C., Beck, E., Irie, K., et al. Rwth asr systems for librispeech: Hybrid vs attention. *Interspeech 2019*, Sep 2019. doi: 10.21437/interspeech.2019-1780.

[34] Mohamed, A., Okhonko, D., and Zettlemoyer, L. Transformers with convolutional context for asr, 2019.

[35] Nesterov, Y. A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2). In *Doklady AN USSR*, volume 269, pp. 543–547, 1983.

[36] Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

[37] Panayotov, V., Chen, G., Povey, D., and Khudanpur, S. Librispeech: an ASR corpus based on public domain audio books. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pp. 5206–5210. IEEE, 2015.

[38] Park, D. S., Chan, W., Zhang, Y., et al. Specaugment: A simple data augmentation method for automatic speech recognition. *Interspeech 2019*, Sep 2019. doi: 10.21437/interspeech.2019-2680.

[39] Park, J., Boo, Y., Choi, I., Shin, S., and Sung, W. Fully neural network based speech recognition on mobile and embedded devices. In *Advances in Neural Information Processing Systems*, pp. 10620–10630, 2018.

[40] Parthasarathi, S. H. K. and Strom, N. Lessons from building acoustic models with a million hours of speech. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6670–6674. IEEE, 2019.

[41] Prabhavalkar, R., Rao, K., Sainath, T. N., et al. A comparison of sequence-to-sequence models for speech recognition. In *Interspeech*, pp. 939–943, 2017.

[42] Pratap, V., Hannun, A., Xu, Q., et al. wav2letter++: The fastest open-source speech recognition system. *arXiv preprint arXiv:1812.07625*, 2018.

[43] Saon, G., Kurata, G., Sercu, T., Audhkhasi, K., Thomas, S., Dimitriadis, D., Cui, X., Ramabhadran, B., Picheny, M., Lim, L.-L., Roomi, B., and Hall, P. English conversational telephone speech recognition by humans and machines. In *Interspeech*, pp. 132–136, 2017.

[44] Schuster, M. and Nakajima, K. Japanese and korean voice search. In *International Conference on Acoustics, Speech and Signal Processing*, pp. 5149–5152, 2012.

[45] Soltau, H., Liao, H., and Sak, H. Neural speech recognizer: Acoustic-to-word lstm model for large vocabulary speech recognition. *arXiv preprint arXiv:1610.09975*, 2016.

[46] Sriram, A., Jun, H., Satheesh, S., and Coates, A. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*, 2017.

[47] Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[48] Vaswani, A., Shazeer, N., Parmar, N., et al. Attention is all you need. In *Adv. NIPS*, 2017.

[49] Vesel`y, K., Burget, L., and Cernock`y, J. Semi-supervised DNN training with word selection for ASR. In *Interspeech*, pp. 3687–3691, 2017.

[50] Wang, Y., Deng, X., Pu, S., and Huang, Z. Residual convolutional ctc networks for automatic speech recognition. *arXiv preprint arXiv:1702.07793*, 2017.

[51] Wang, Y., Mohamed, A., Le, D., Liu, C., Xiao, A., Mahadeokar, J., Huang, H., Tjandra, A., Zhang, X., Zhang, F., Fuegen, C., Zweig, G., and Seltzer, M. L. Transformer-based acoustic modeling for hybrid speech recognition, 2019.

[52] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[53] Xiong, W., Droppo, J., Huang, X., Seide, F., Seltzer, M., Stolcke, A., Yu, D., and G., Z. The microsoft 2016 conversational speech recognition system. In *ICASSP*, 2017.

[54] Zeghidour, N., Xu, Q., Liptchinsky, V., Usunier, N., et al. Fully convolutional speech recognition. *CoRR*, abs/1812.06864, 2018. URL `http://arxiv.org/abs/1812.06864`.

[55] Zhou, S., Dong, L., Xu, S., and Xu, B. A comparison of modeling units in sequence-to-sequence speech recognition with the transformer on mandarin chinese. In *International Conference on Neural Information Processing*, pp. 210–220. Springer, 2018.

Table 5: Word error rates on LibriSpeech's development and test sets. Our models listed in the middle and bottom blocks are trained with CTC and Seq2seq losses respectively.

AM		LM		Dev		Test	
type	lexicon	type	lexicon	clean	other	clean	other
CTC							
ResNet (306M)	10k WP	-	-	3.93	10.13	4.08	10.03
Decoding		ZeroLM	lex	3.76	9.7	4.07	9.77
Decoding		4gram	word	3.29	8.56	3.68	8.69
Decoding		GCNN	word	2.99	7.50	3.28	7.53
ResNet (500M) LibriVox	10k WP	-	-	2.34	5.54	2.55	5.99
Decoding		ZeroLM	lex	2.37	5.45	2.73	5.96
Decoding		4gram	word	2.34	5.23	2.68	5.75
Decoding		GCNN	word	2.19	4.64	2.45	5.13
TDS (200M)	10k WP	-	-	4.22	11.16	4.63	11.16
Decoding		ZeroLM	lex	3.93	10.61	4.44	10.67
Decoding		4gram	word	3.49	9.18	3.98	9.53
Decoding		GCNN	word	2.92	7.52	3.40	8.05
TDS (500M) LibriVox	10k WP	-	-	2.44	5.70	2.66	6.11
Decoding		ZeroLM	lex	2.47	5.61	2.86	6.18
Decoding		4gram	word	2.44	5.33	2.81	5.91
Decoding		GCNN	word	2.26	4.71	2.55	5.24
Transf. (322M)	10k WP	-	-	2.99	7.31	3.09	7.40
Decoding		ZeroLM	lex	2.85	6.98	3.14	7.23
Decoding		4gram	word	2.63	6.20	2.86	6.72
+ Rescoring		GCNN + Transf.	word	2.18	4.90	2.44	5.36
Decoding		GCNN	word	2.35	5.29	2.57	5.85
+ Rescoring		GCNN + Transf.	word	2.20	4.94	2.47	5.45
Transf. (299M) LibriVox	10k WP	-	-	2.28	5.00	2.39	5.35
Decoding		ZeroLM	lex	2.31	4.94	2.58	5.42
Decoding		4gram	word	2.24	4.59	2.52	5.22
+ Rescoring		GCNN + Transf.	word	1.99	3.91	2.28	4.50
Decoding		GCNN	word	2.09	4.27	2.41	4.79
+ Rescoring		GCNN + Transf.	word	2.01	3.95	2.31	4.54
Seq2Seq							
ResNet (389M)	10k WP	-	-	3.51	9.89	4.92	10.33
Decoding		ZeroLM	lexfree	3.42	9.60	4.31	9.59
Decoding		6gram	10k WP	3.05	8.69	3.88	8.88
Decoding		GCNN	10k WP	2.78	7.86	3.79	8.21
ResNet (500M) LibriVox	10k WP	-	-	2.27	5.29	2.86	5.88
Decoding		ZeroLM	lexfree	2.26	5.28	2.67	5.54
Decoding		6gram	10k WP	2.29	5.25	2.69	5.62
Decoding		GCNN	10k WP	2.26	4.91	2.66	5.31
TDS (190M)	10k WP	-	-	3.20	8.20	3.43	8.30
Decoding		ZeroLM	lexfree	2.89	8.00	3.24	7.99
Decoding		6gram	10k WP	2.76	7.01	3.18	7.16
Decoding		GCNN	10k WP	2.54	6.30	2.93	6.43
TDS (500M) LibriVox	10k WP	-	-	2.17	4.78	2.37	5.15
Decoding		ZeroLM	lexfree	2.20	4.80	2.38	5.11
Decoding		6gram	10k WP	2.18	4.61	2.35	5.02
Decoding		GCNN	10k WP	2.08	4.21	2.24	4.61
Transf. (270M)	10k WP	-	-	2.54	6.67	2.89	6.98
Decoding		ZeroLM	lexfree	2.49	6.32	2.75	6.58
Decoding		6gram	10k WP	2.29	5.81	2.72	6.23
+ Rescoring		GCNN + Transf.	word	2.13	5.00	2.51	5.47
Decoding		GCNN	10k WP	2.12	5.20	2.40	5.70
+ Rescoring		GCNN + Transf.	word	2.10	4.79	2.33	5.17
Transf. (296M) LibriVox	10k WP	-	-	2.12	4.59	2.28	4.88
Decoding		ZeroLM	lexfree	2.10	4.53	2.27	4.80
Decoding		6gram	10k WP	2.06	4.32	2.25	4.70
+ Rescoring		GCNN + Transf.	word	1.91	3.76	2.10	4.20
Decoding		GCNN	10k WP	1.97	3.95	2.17	4.37
+ Rescoring		GCNN + Transf.	word	2.00	3.65	2.09	4.11

A Experiment Details

Comprehensive WER results for LIBRISPEECH and LIBRIVOX acoustic models, including with greedy and beam-search decoding with different LMs and beam rescoring can be found in Table 5. This section mainly focus on providing details of how we optimize the beam-search decoding and rescoring procedures for our acoustic models.

A.1 Beam-Search Decoding

When beam-search decoding, we use the `dev-clean` and `dev-other` sets as validation sets and use random search to optimize decoding hyper-parameters. The search ranges of those hyper-parameters are listed in Table 6. We use between 64 and 128 runs in each random search with hyper-parameter values uniformly sampled from the given ranges. It is worth noting that the optimal ranges for language model weight for models trained on LIBRISPEECH are higher than ones found for LIBRIVOX models as shown in Table 7. This is conceivably additional evidence that models trained with additional audio rely less on language models.

Table 6: Hyper-parameter values and ranges used in a random search for beam-search decoding with n-gram (top block) and GCNN (bottom block) LMs.

	LIBRISPEECH		LIBRIVOX	
PARAMETERS	CTC	S2S	CTC	S2S
BEAM	500	50, 100	500	20, 50, 100
TOKEN BEAM	100	10, 50	100	3, 5, 10
LM WEIGHT	[0, 3]	[0, 2]	[0, 1.5]	[0, 1]
THRESHOLD	100	10, 50	100	5, 10, 50
WORD INSERT.	[−3, 3]	-	[−3, 3]	-
EOS-PENALTY	-	[−10, 0]	-	[−10, 0]
BEAM	250	50	250	20, 50, 100
TOKEN BEAM	100	10, 18	100	3, 5, 10
LM WEIGHT	[0, 3]	[0, 2]	[0, 1.5]	[0, 0.8]
THRESHOLD	20	10, 15	20	5, 10, 50
WORD INSERT.	[−3, 3]	-	[−3, 3]	-
EOS-PENALTY	-	[−10, 0]	-	[−10, 0]

Table 7: Optimal LM weight ranges (based on WER) for beam-search decoding with n-gram (top block) and GCNN (bottom block) LMs found via random search.

	LIBRISPEECH		LIBRIVOX	
DATA	CTC	S2S	CTC	S2S
CLEAN	[0.8, 1.4]	[0.6, 1.1]	[0.2, 0.4]	[0.0, 0.2]
OTHER	[1.1, 1.9]	[0.6, 1.2]	[0.5, 0.7]	[0.1, 0.5]
CLEAN	[0.4, 0.8]	[0.2, 0.5]	[0.2, 0.5]	[0.0, 0.4]
OTHER	[0.5, 1.1]	[0.3, 0.7]	[0.3, 0.6]	[0.2, 0.4]

A.2 Rescoring

To perform rescoring, we first dump all hypotheses proposed during beam-search decoding using the optimal hyper-parameters found with random search. When dumping candidates, beam size, token beam size, and beam threshold are increased so as to increase the number of proposed hypotheses on which to run rescoring. Further details are listed in Table 8. We find optimal values of rescoring hyper-parameters α_1, α_2 and β (see Section 4.2) via a grid search for CTC models ($\alpha_1, \beta \in [0, 1]$ and $\alpha_2 \in [-0.3, 0.3]$ where the grid step is set to 0.1), and a random search for sequence-to-sequence models ($\alpha_1, \in [0, 2.5]$, $\alpha_2 \in [-1, 1]$, $\beta \in [-3, 3]$ with 1000 attempts).

B Generating Shuffled Audio

This section provides details of how we generated shuffled utterances used in the experiments in Section 5.4. Each experiment could introduce systematic error. Therefore, we propose several experiments to conclude. For the two

Table 8: Parameters values used when dumping beam candidates for rescoring with n-gram (top block) and GCNN (bottom block) LMs.

PARAMETERS	CTC	S2S
BEAM	2500	250
TOKEN BEAM	1500	150
THRESHOLD	5000	150
BEAM	250	250
TOKEN BEAM	100	100
THRESHOLD	20	100

methods generating existing or using new audio (**TTS** and **Segmentation**), we shuffle dev-other five times and report the mean and standard deviation (as error bars) in Figure 2.

B.1 TTS

For each sentence in dev-other, we randomly shuffle its words to form a new sentence. We run the resulting text through a TTS model as outlined in Section 5.3 to create synthetic audio for the scrambled sentences. While simple and easy to implement, this method introduces and amplifies intrinsic errors in the TTS model into the ablation. In particular, the model struggles to handle many of the rare words present in dev-other. Also TTS approach is still away from the human speech.

B.2 Segmentation

With this method, we first force-align the transcriptions of dev-other to the existing audio using a letter-based stride-two ASG model as outlined in Section 5.3 and collecting the beginning timestamp and duration of each word. Then, to avoid splicing words that are ready closely together, audio samples are only split when silence of longer than 130 milliseconds is detected (split is done in the middle of silence segment). Finally, audio chunks are randomly shuffled and re-assembled into new utterances. Since this ablation aims to remove LM-friendly context from audio, we filter the resulting recombined audio samples. In particular, we filter all utterances that have only one segment, or have at least one segment with more than 6 words in it. After filtering, 1969 out of 2864 samples in dev-other remain. The distribution of the number of words in each of the resulting segments is shown in Figure 4.

Unlike the TTS method described above, the segmentation method reuses audio as much as possible from dev-other. That said, neither the force alignment nor the segmentation techniques handle all the word boundaries. As such, there may be incomplete words in the resulting audio and LM-friendly context.

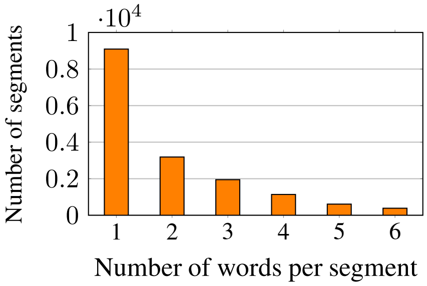


Figure 4: Distribution of all n-grams in the obtained segments of filtered dev-other (1969 samples with 16,362 segments in total).

B.3 Recording

The paper's authors recorded 184 randomly selected sentences from dev-other as well as a single set of shuffled utterances. The unshuffled recorded audio has the lowest WER among all the three methods. We plan to complete a collection of unshuffled and shuffled audio for dev-other in future work.

Table 9: Performance of word-level 4-gram and Transformer LMs from Table 1 on original and shuffled audio transcriptions generated from LIBRISPEECH `dev-other`.

SETTING	SHUFFLED	4-GRAM LM	TRANSF. LM
TTS	NO	147	50
TTS	YES	749 ± 2	389 ± 2
SEGMENT.	NO	167	56
SEGMENT.	YES	827 ± 5	743 ± 9
RECORDING	NO	162	49
RECORDING	YES	3807	2995

B.4 Perplexity

As shown in Table 9, there are large gaps between the perplexity of transcriptions in the original and shuffled sets across all settings. Our shuffling strategy thus removes important word context and breaks the alignment of the audio words distribution with the LM. The WER gap between the two sets is thus a proxy for the amount of language modeling an acoustic model may implicitly perform.