Whole-Word Segmental Speech Recognition with Acoustic Word Embeddings

Bowen Shi, Shane Settle, Karen Livescu
TIT-Chicago
{bshi, settle.shane, klivescu}@ttic.edu

Abstract

Segmental models are sequence prediction models in which scores of hypotheses are based on entire variable-length segments of frames. We consider segmental models for whole-word (“acoustic-to-word”) speech recognition, with the segment features vectors defined using acoustic word embeddings. Such models are computationally challenging as the number of paths is proportional to the vocabulary size, which can be orders of magnitude larger than when using subword units like phones. We describe an efficient approach for end-to-end whole-word segmental models, where the sequence probability is computed based on soft spans of the acoustics via an attention mechanism [9, 10]. In segmental modeling, however, these frame features are passed through a neural network and encoded into frame features on input acoustic frames directly to words. Unlike conventional subword-based automatic speech recognition (ASR) systems, A2W models do not require an external lexicon, thus simplifying training and decoding. Recent work has shown that A2W models can achieve performance competitive with state-of-the-art subword-based systems either with large amounts of training data [1] or with careful training techniques [2–5].

Most work on A2W models [1–7] is based on connectionist temporal classification (CTC) [8], where the word sequence probability is defined as the product of frame-level word probabilities. In such approaches there is no explicit modeling of spans of frames corresponding to words. There has also been recent work on encoder-decoder A2W models, which can focus on soft spans of the acoustics via an attention mechanism [9, 10].

In this paper we propose an approach using whole-word segmental models, where the sequence probability is computed based on segment scores instead of frame probabilities. Segmental models have a long history in speech recognition research, but they have been explored primarily for phonetic recognition or used as phone-level acoustic models [11–18]. There has also been work on whole-word segmental models for second-pass rescoring [13, 19, 20], but to our knowledge our approach is the first to address end-to-end A2W segmental models.

The key ingredient in our approach is to define the segment scores in terms of dot products between acoustic word embeddings (AWEs) [21–24] and a weight layer of acoustically grounded word embeddings (AGWEs) [5, 25] corresponding to the word labels. This form of the model allows for (1) efficient re-use of feature functions and therefore reduced memory cost and (2) pre-training of the AWEs and AGWEs, following the successful use of such pre-training in prior work on speech recognition [5] and search [26, 27]. We also obtain speed-ups via GPU implementations of the forward-backward and Viterbi algorithms. We find that pre-trained AWEs provide large gains, and result in segmental models competitive with the best similar-sized models on conversational telephone speech recognition.

2. Segmental Model Formulation

Segmental models compute the score of a hypothesized label sequence as a combination of scores of multi-frame segments of speech in the sequence, rather than using individual frame scores (see Figure 1). Let \( X = \{x_1, x_2, \ldots, x_T\} \) be a sequence of input acoustic frames and \( L = \{l_1, l_2, \ldots, l_K\} \) be the output label sequence. A segmentation \( \pi \) with respect to \( X \) and \( L \) is defined as a sequence of tuples \( \{(t_1, s_1, l_1), (t_2, s_2, l_2), \ldots, (t_K, s_K, l_K)\} \). Each tuple defines a segment \( e_i \) consisting of a start frame \( t_i \), an end frame \( l_i + s_i \), and a label \( l_i \), such that \( t_1 = 0, t_K + s_K = T, t_k + s_k = t_{k+1} \), and \( s_k > 0 \) for all \( 1 \leq k \leq K \). A segmental model assigns a score \( w_{t,s,v} \) to each segment \( (t, s, v) \). The score of a segmentation is then defined as \( w(\pi) = \sum_{(t,s,v) \in \pi} w_{t,s,v} \).

2.1. Segment Score Functions

As in other sequence models, the input acoustic frames are first passed through a neural network and encoded into feature frames \( H = \text{Enc}(X) \in \mathbb{R}^{T \times F} \), where \( F \) denotes the feature dimension. In segmental modeling, however, these frame features are then used to produce segment scores \( W \in \mathbb{R}^{T \times S \times V} \), where \( S \) and \( V \) denote the maximum segment size and vocabulary size, respectively, and \( w_{t,s,v} \) is the score of segment \( (t, s, v) \).
Our approach defines segment scores \( \mathbf{W} \) in terms of dot products between learned representation vectors of variable-length segments and word labels:

\[
    w_{t,s,v} = a_{v}^{(2)^T} f_{\text{AVE}}(\mathbf{H}_{t:s}^{(2)}) + b_{v}^{(2)}
\]

where \( f_{\text{AVE}} \) is the AWE function mapping segments \( \mathbf{H}_{t:s} \in \mathbb{R}^{V \times F} \) to fixed-dimensional embeddings \( f_{\text{AVE}}(\mathbf{H}_{t:s}) \in \mathbb{R}^{D} \), \( a_{v}^{(2)} \) is a row from the matrix \( \mathbf{A}^{(2)} \in \mathbb{R}^{V \times D} \) composed of AGWEs for all words \( v \) in the vocabulary and \( b_{v} \) is the bias on word \( v \), which can be interpreted as a log-unigram probability. We compute \( u_{t,s,v} \) using \( f_{\text{AVE}} \) as follows:

\[
    f_{\text{AVE}}(\mathbf{H}_{t:s}) = \text{ReLU}(\mathbf{A}^{(1)} G(\mathbf{H}_{t:s}) + \mathbf{b}^{(1)})
\]

where \( G \) is a pooling function chosen between:

\[
    G(\mathbf{H}_{t:s}) = [\mathbf{h}_{t}; \mathbf{h}_{t:s}]
\]

(3)

\[
    G(\mathbf{H}_{t:s}) = \frac{1}{S} \sum_{i=1}^{S} \mathbf{h}_{t+i}
\]

(4)

\[
    G(\mathbf{H}_{t:s}) = \frac{1}{S} \sum_{i=1}^{S} \text{Softmax}(\mathbf{g}^T \mathbf{H}_{t+i}) \mathbf{h}_{t+i}
\]

(5)

where \( (\cdot)^{\text{concat}} \) is concatenation, \( (\cdot)^{\text{avg}} \) is average pooling, and \( (\cdot)^{\text{attn}} \) is attention pooling (with learnable parameter \( g \)). Equation 1 allows feature sharing, which helps limit the memory overhead of computing segment features to \( O(TS/D) \) and simplifies scoring to matrix multiplication, i.e. \( \mathbf{W}_{t,s} = \mathbf{A}^{(2)^T} f_{\text{AVE}}(\mathbf{H}_{t:s}) + \mathbf{b}^{(2)} \).

Recent work on segmental models has largely used two types of segment score functions: (1) frame classifier-based \([15, 16, 28, 29]\) and (2) segmental recurrent neural network (SRNN) \([18, 29, 30]\). Frame classifier-based score functions use a mapping from input acoustic frames \( \mathbf{X} \) to frame log-probability vectors \( \mathbf{P} \), which are then pooled (via mean, sampling, etc.) to get the segment score \( w_{t,s,v} \). This method introduces a multiplicative memory dependency on \( V \), which is a factor \( V/D \) increase in memory overhead over our approach. In our case \( V \) is the number of words in the vocabulary, which makes this approach prohibitively expensive. The SRNN approach computes the score \( w_{t,s,v} = \phi^TF_{\theta}(\mathbf{h}_{t}; \mathbf{h}_{t:s} ; a_{v}^{(2)}) \), where \( f_{\theta} \) is a learned feature function, \( \phi \) is an embedding of word \( v \) and \( [u; v] \) denotes concatenation of \( u, v \). This method introduces an \( O(TSVD) \) memory overhead, which can again quickly make it infeasible for large-vocabulary word modeling.

In addition to computational savings, our formulation of segment scores as products of AWEs and AGWEs, \( \mathbf{A}^{(2)^T} f_{\text{AVE}}(\mathbf{X}) \), also has the advantage that these two factors can be pre-trained using methods from prior work \([5, 25]\) (see Section 2.4).

### 2.2. Training

Segmental models can be trained in a variety of ways \([29]\). One way, which we adopt here, is to interpret segmental models as probabilistic models and optimize the marginal log loss under that model, which is equivalent to viewing our models as segmental conditional random fields \([31]\). Under this view, the model assigns probabilities to paths, conditioned on the input acoustic sequence, by normalizing the path score. Letting \( \mathbf{U} := \exp(\mathbf{W}) \), we define \( p(\pi) := \frac{u^{W}(\pi)}{\sum_{\pi' \in \mathcal{P}_{v:T}} u^{W}(\pi')} \) as the probability of the segmentation, where \( u^{W}(\pi) = \prod_{i \in \{l_i\}} x_{i,v} \) and \( \mathcal{P}_{v:T} \) denotes all segmentations of \( X_{1:T} \). To train our models, we define the loss for a given word sequence \( \mathbf{L} \) and input \( \mathbf{X} \) as the marginal log loss, by marginalizing over all possible segmentations:

\[
    \mathcal{L}(\mathbf{L}, \mathbf{X}) = -\log \sum_{\pi \in \mathcal{P}_{v:T}} u(\pi) + \log \sum_{\pi \in \mathcal{P}_{v:T}} u(\pi)
\]

where \( B(\pi) \) maps \( \pi = \{(t_i, s_i, l_i)\}_{1 \leq i \leq |\pi|} \) to its label sequence \( \{l_i\}_{1 \leq i \leq |\pi|} \). The summations can be efficiently computed with dynamic programming:

\[
    \alpha_t^{(d)} := \sum_{\pi \in \mathcal{P}_{v:T}} u(\pi) = \sum_{s=1}^{V} \sum_{v=1}^{S} u_{t-s,s,v} \alpha_{t-s}^{(d-1)}
\]

(6)

\[
    \beta_t^{(d)} := \sum_{\pi \in \mathcal{P}_{v:T}} u(\pi) = \sum_{s=1}^{S} \sum_{v=1}^{V} u_{t-s,v} \beta_{t-s}^{(d-1)}
\]

(7)

With \( \alpha_t^{(d)} \) and \( \beta_t^{(d)} \) computed, the loss value follows directly from \( \mathcal{L}(\mathbf{L}, \mathbf{X}) = -\log \frac{\alpha_T^{(n)}}{\beta_T^{(n)}} + \log \pi_{G}^{(d)} \). The last summations in Equations 7 can be efficiently implemented on a GPU. In addition, \( \alpha_t^{(n)} \) and \( \beta_t^{(n)} \) can be computed in parallel given \( \alpha_t^{(n)} \) such that the overall time complexity\(^1\) of computing the loss is \( O(T \log(SV) + |\mathbf{L}| \log(|\mathbf{L}|)) \).

To train the model with gradient descent, we need to differentiate \( \mathcal{L}(\mathbf{L}, \mathbf{X}) \) with respect to \( \mathbf{X} \), which can in principle be done with auto-differentiation tools \([\text{e.g.} \text{PyTorch}][32]\). However, in practice using auto-differentiation to compute the gradient is often many times slower than the loss computation. Instead, we explicitly implement the gradient computation \( \frac{\partial \mathcal{L}(\mathbf{L}, \mathbf{X})}{\partial u_{t,s,v}} \) using the backward algorithm. We define two backward variables \( \beta_t^{(n)} \) and \( \alpha_t^{(n)} \) for the denominator and numerator, respectively:

\[
    \beta_t^{(n)} := \sum_{\pi \in \mathcal{P}_{v:T}} u(\pi) = \sum_{s=1}^{S} \sum_{v=1}^{V} u_{t-s,v} \beta_{t-s}^{(n-1)}
\]

(8)

\[
    \alpha_t^{(n)} := \sum_{\pi \in \mathcal{P}_{v:T}} u(\pi) = \sum_{s=1}^{S} \sum_{v=1}^{V} u_{t-s,v} \alpha_{t-s}^{(n-1)}
\]

The gradient \( \frac{\partial \mathcal{L}(\mathbf{L}, \mathbf{X})}{\partial u_{t,s,v}} \) is then given by

\[
    \frac{\partial \mathcal{L}(\mathbf{L}, \mathbf{X})}{\partial u_{t,s,v}} = -\sum_{i \in \{l_i=v\}} \alpha_t^{(n)} \beta_{t+i}^{(n)} + \alpha_t^{(n)} \beta_{t+i}^{(n)}
\]

(9)

where \( \{l_i=v\} \) is the set of indices in \( \mathbf{L} \) where label \( v \) occurs.

### 2.3. Decoding

Decoding consists of solving \( \pi^* = \arg \max_{\pi \in \mathcal{P}_{v:T}} u(\pi) \). This optimization problem can be solved efficiently via the Viterbi algorithm with the recursive relationship:

\[
    d(t) := \max_{v \in \mathcal{P}_{0:t}} w(\pi) = \max_{1 \leq t \leq S} \left[ u_{t-s,s,v} + d(t-s) \right]
\]

where the last max operation can be parallelized on a GPU such that the overall runtime\(^2\) of decoding is only \( O(T \log(SV)) \).

### 2.4. AWE and AGWE pre-training

One important issue in whole-word models is that many words are typically infrequent or unseen in the training set. In particular, the final weight layer, which corresponds to embeddings of the word labels, can be very poorly learned. Recent work has shown that jointly pre-trained AWEs and corresponding AGWEs of the written words \([25]\) can serve as a good parameter initialization for CTC-based A2W models \([5]\), improving performance on conversational speech recognition. In this prior work, the AGWEs are parametric functions of character sequences, so that word

\(^1\)Number of times \( a + b \) is called

\(^2\)Number of times \( \max(a, b) \) is called
embeddings can be produced for unseen or infrequent words. We follow this idea and jointly pre-train our segmental AWE-based feature function \( f_{AWE} \) and the corresponding weight layer \( A^{(2)} \) in Equation 1. Note that typical pre-trained written word embeddings (such as word2vec [33], GloVe [34], and contextual word embeddings [35, 36]) are not what is needed for the label embedding layer; we are interested in embeddings that represent the way a word sounds rather than what it means.

Our pre-training follows the multi-view AWE+AGWE training approach of [5, 25]. We jointly train an acoustic “view” embedding model \( f \) and a written “view” model \( g \) using a contrastive loss. The written view model takes in a word label \( v \), maps it to a subword (e.g., character/phone) sequence using a lexicon, and uses this sequence to produce an AGWE as output. Embedding models \( f \) and \( g \) are trained to minimize an objective consisting of three contrastive triple loss terms:

\[
\sum_{i=1}^{N} \left[ m + d(f(X_i), g(v_i)) - \min_{v' \in V} d(f(X_i), g(v')) \right] + \\
+ \left[ m + d(g(v_i), f(X_i)) - \min_{X' \in V} d(g(v_i), f(X')) \right] + \\
+ \left[ m + d(g(v_i), f(X_i)) - \min_{v' \in V} d(g(v_i), g(v')) \right]
\tag{11}
\]

where \( X_i \) is a spoken word segment, \( v_i \) is its word label, \( m \) is a margin hyperparameter, \( d \) denotes cosine distance \( d(a,b) = 1 - \frac{a \cdot b}{\| a \| \| b \|} \), and \( N \) is the number of training pairs \((X, v)\). We conduct semi-hard [37] negative sampling w.r.t. each pair:

\[
\mathcal{V}_{1}(X, v) := \{ v' | d(f(X), g(v')) > d(f(X), g(v)), v' \in V/v \} \\
\mathcal{X}(v) := \{ X' | d(g(v), f(X')) > d(g(v), f(X)), X' \in V/v \} \\
\mathcal{V}_{2}(X, v) := \{ v' | d(g(v), g(v')) > d(g(v), g(v)), v' \in V/v \}
\]

where \( V \) is the training vocabulary. This loss aims to map spoken word segments corresponding to the same word label close together and close to their learned label embeddings, while ensuring that segments corresponding to different word labels are mapped farther apart (and nearer to their respective label embeddings). For efficiency, this is performed over the mini-batch such that \( N \) is the batch size and \( V \) consists of words in the mini-batch. Additionally, rather than the single most offending semi-hard negative we use \( M \) and each contrastive loss term inside the sum in Equation 11 is an average over these \( M \) negatives.

Our pre-training approach is the same as that of [5, 25] except for the addition of semi-hard negative sampling (replacing hard negative sampling in [5]), the inclusion of a third contrastive term \((obj_3)\) of [25], and an extra convolutional layer and pooling in the AWE encoder (see Section 3). The first two changes were found to improve performance on a small subset of Switchboard-300h explored in prior work on AWEs [22, 23, 25], and the third change was adopted for efficiency in the segmental model.

The pre-trained AWE/AGWE models are tuned using a cross-view word discrimination task as in [5, 25], applied to word segments from the development set and word labels from the vocabulary. The task is to determine whether a given acoustic word segment and word label match. We compute the embeddings of the acoustic segment and character sequence by forwarding them through \( f \) and \( g \) respectively, and then compute their cosine distance. If this distance is below a threshold, then the pair is labeled a match. The quality of the embeddings is measured by the average precision (AP) over all thresholds over the dev set.

![Figure 2](image)

### Figure 2: Dev WER vs. epoch for A2W CTC and segmental models with different initialization. Numbers: lowest dev WER.

Similarly to [5], in addition to initializing with the pre-trained AGWEs, we also consider L2 regularization toward the pre-trained AGWEs. We add a term to the recognizer loss (Equation 6) corresponding to the distance between the rows \( a^{(2)} \) of \( A^{(2)} \) and the pre-trained AGWEs \( g(v) \) after initialization:

\[
\mathcal{L}_{reg}(L, X) = (1 - \lambda) \mathcal{L}_{reg}(L, X) + \lambda \sum_{v \in V} \| a^{(2)} - g(v) \|^2
\tag{12}
\]

where \( \lambda \) is a hyperparameter.

### 3. Experiments

Experiments are conducted on the standard Switchboard-300h dataset and data division [38]. We use Kaldi [39] to extract 40-dimensional log-Mel spectra +Δ+ΔΔs. Every two successive frames are stacked and alternate frames dropped, resulting in 240-dimensional features. We explore 5K, 10K, and 20K vocabularies based on word occurrence thresholds of 18, 6, and 2, respectively. The backbone network for the segmental model is a 6-layer bidirectional long short-term memory network (BiLSTM) with 512 hidden units per direction per layer with dropout added between layers (0.25 by default). To speed up training, we add a convolutional layer with kernel size 5 followed by average pooling with stride 4 on top of the BiLSTM. The maximum segment length is set to 32, corresponding to a maximum word duration of \( \sim 2.4s \). Training utterances are sorted by input length such that similar length utterances are batched together. To further speed up training, we reduce the maximum segment size per batch (batch size: 16) to \( \min(2 \times \text{max input length}) \), 32. The model is trained with the Adam optimizer [40] at an initial learning rate of 0.001, which is decreased by a factor of 2 when the dev WER stops decreasing. No language model is used for decoding. Training takes 3 days on one GTX 1080ti GPU. As a baseline, we also train an A2W CTC model using the same network structure (6-layer BiLSTM + convolutional + pooling).

#### 3.1. Phone CTC pre-training

As an initial experiment on the 5K vocabulary, we initialize the backbone BiLSTM network by pre-training with a phone CTC objective, as in prior work on CTC-based A2W models [2–5]. The phone error rate (PER) of this phone CTC model is \( \sim 11.0% \). On top of the pre-trained BiLSTM, the convolutional layer and word embedding parameters \((A^{(1)}, a^{(1)}, b^{(1)}, b^{(2)})\) in our segmental model are randomly initialized. Compared to random initialization, phone CTC pre-training reduces WER by 1.9% (Figure 2), which is consistent with prior work [2]. When both our segmental model and the baseline A2W CTC model are pre-trained with phone CTC, our model achieves \( \sim 1\% \) lower word error rate (WER) (Figure 2).
The pooling operation $G$ in Equation 2 is tuned among average pooling (18.5%), attention pooling (19.0%) and concatenation (18.2%). The best performance is obtained with concatenation, suggesting the context information provided by the BiLSTM, topped with a convolutional layer, and trained end to end, is enough to represent the acoustic information in a segment. In addition, concatenation increases the feature dimension of $A^{(1)}$, while consuming a factor of $S/2$ less memory computing segment features.

### 3.2. Vocabulary size

We find that, unlike our A2W CTC models and those of prior work [4,5], the segmental models do not improve significantly (or at all) with larger vocabulary. One possible reason is that word representations in segmental models, especially for rare words, are harder to learn as they must be robust to variations in segment duration and content. Segmental models may require more data and overfit when many rare words are included. We find that it is important to set a larger dropout value as the vocabulary size increases. The best dropout values for $5K$, $10K$ and $20K$ are 0.25, 0.35, and 0.45, respectively, with results in Table 1.

### 3.3. AWE + AGWE pre-training

We now investigate whether pre-trained AWE and AGWE models can provide a better starting point for the segmental model. Following [5], we train a multi-view AWE model on the Switchboard-300h training set as described in Section 2.4 ($m = 0.45, M = 64$ reduced by 1 per batch until $M = 6$, variable batch size with 20000 frames per batch). The acoustic view model $f$ has the same structure as the segmental model without the prediction layer ($A^{(1)}, b^{(1)}$), and the written view model $g$ is composed of an input embedding layer mapping one of 37 characters to 32-dimensional embeddings followed by a 1-layer BiLSTM with 256 hidden units per direction. The Adam optimizer [40] is used with an initial learning rate of 0.0005, and the learning rate is reduced by a factor of 10 when dev cross-view average precision (AP) does not improve for 3000 steps. Training is stopped when the learning rate is dropped below $10^{-9}$. The resulting embeddings produce a cross-view AP of 83.3% when evaluated on dev set word segments and a $20K$ vocabulary. After multi-view training, the acoustic view $f$ (our AWE function) and the written view $g$ (our AGWE function) can be used to initialize our segmental feature function $f_{AWE}$ and $A^{(2)}$, respectively, in Equation 1.

Table 1 compares an A2W CTC model with segmental models using different initializations. Initialization with the pre-trained AWE model reduces WER by 1–2% over phone CTC initialization. Initialization of $A^{(2)}$ with pre-trained AGWE models alone does not help, but initializing with AWE and AGWE while regularizing toward the pre-trained AGWEs (see Section 2.4) is helpful, especially for larger vocabularies. This is consistent with our expectation: Since the AGWEs are composed from character sequences, they are less impacted by vocabulary size, helping with recognition of rare words. In addition, we notice that the optimal $\lambda$ in Equation 12 tends to be larger as the vocabulary size increases, reinforcing the need for more regularization when there are many rare words.

Table 2 shows the final evaluation results on the Switchboard (SWB) and CallHome (CH) test sets, compared to prior work with A2W models. For the smaller vocabulary sizes ($5K$, $10K$), the segmental model improves WER over CTC by around 1% (absolute). Training with SpecAugment [41] produces an additional gain of 3% across all vocabulary sizes. On the SWB test set, our model also improves over all previous models except for the one of [4], which is larger and trained with speed perturbation. On the CH test set performance also improves over CTC except at the largest vocabulary size. We hypothesize that CH has even larger variability within segments, but this remains to be investigated. We leave as future work the incorporation of larger models and speed perturbation.

### 4. Conclusions

We have introduced an end-to-end whole-word segmental model, which to our knowledge is the first to perform large-vocabulary speech recognition competitively and efficiently. Our model uses a simple segment score function based on a dot product between written word embeddings and acoustic word embeddings, which both improves efficiency and enables us to use jointly pre-trained acoustic and written word embeddings. We find that the proposed model usually outperforms A2W CTC models. Future work includes adding other orthogonal improvements, like using a larger network, to the current model. Given the good performance of segmental models especially when the label set is relatively small, it will also be interesting to apply the approach to recognition based on subwords like byte pair encodings [42].

### 5. Acknowledgements

We are grateful to Hao Tang for helpful suggestions and discussions. This research was funded by NSF award IIS-1816627, and by an AWS Machine Learning Research Award.

---

Table 1: Comparison of initialization with phone CTC vs. AWE/AGWE, in terms of SWB dev WER (%). “AWE init” refers to initialization of the parameters of $f_{AWE}$. “AGWE init” refers to initialization of the rows of $A^{(2)}$.

| System                  | 5K  | 10K | 20K |
|-------------------------|-----|-----|-----|
| A2W CTC with phone CTC init | 19.3 | 18.0 | 17.7 |
| A2W Segmental with phone CTC init | 18.2 | 17.9 | 18.0 |
| A2W Segmental with AWE init | 17.1 | 16.0 | 16.4 |
| A2W Segmental with AGWE init | 17.1 | 15.8 | 16.5 |
| + AGWE L2 reg            | 17.0 | 15.5 | 15.6 |

Table 2: WER (%) results on SWB/CH evaluation sets.

| System                  | 4K/5K | 10K | 20K |
|-------------------------|-------|-----|-----|
| Seg. AWE+AGWE init      | 14.0/24.9 | 12.8/23.5 | 12.5/24.5 |
| +SpecAugment            | 12.8/22.9 | 11.9/21.2 | 12.1/22.5 |
| CTC, phone init [5]     | 16.4/25.7 | 14.8/24.9 | 14.7/24.3 |
| CTC, AWE+AGWE init [5]  | 15.6/25.3 | 14.2/24.2 | 13.8/24.0 |
| +reg [5]                | 15.5/25.4 | 14.0/24.5 | 13.7/23.8 |
| CTC, AWE+AGWE rescore [5] | 15.0/25.3 | 14.4/24.5 | 14.2/24.7 |
| S2S [10]                | -     | 22.4/36.1 | 22.4/36.2 |
| Curriculum [4]          | -     | 13.4/24.2 | - |
| +Joint CTC/CE [4]       | -     | 13.0/23.4 | - |
| +Speed Perturbation [4]  | -     | 11.4/20.8 | - |

We do not compare with prior segmental models [17, 18] as we are unable to train them for A2W recognition with the same network architecture using a typical GPU (e.g., 12GB).
6. References

[1] H. Soltau, H. Liao, and H. Sak, “Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition,” in Proc. Interspeech, 2016.

[2] K. Audhkhasi, B. Ramabhadran, G. Saon, M. Picheny, and D. Na-hamoo, “Direct acoustic-to-word models for English conversational speech recognition,” in Proc. Interspeech, 2017.

[3] K. Audhkhasi, B. Kingsbury, B. Ramabhadran, G. Saon, and M. Picheny, “Building competitive direct acoustic-to-word models for English conversational speech recognition,” in Proc. Interspeech, 2018.

[4] C. Yu, C. Zhang, C. Weng, J. Cui, and D. Yu, “A multistage training framework for acoustic-to-word model,” in Proc. Interspeech, 2018.

[5] S. Settle, K. Audhkhasi, K. Livescu, and M. Picheny, “Acoustically grounded word embeddings for improved acoustic-to-word speech recognition,” in Proc. ICASSP, 2019.

[6] J. Li, G. Ye, A. Das, R. Zhao, and Y. Gong, “Advancing acoustic-to-word CTC model,” in Proc. ICASSP, 2018.

[7] Y. Gaur, J. Li, Z. Meng, and Y. Gong, “Acoustic-to-phrase models for speech recognition,” in Proc. Interspeech, 2019.

[8] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in Proc. Int. Conf. on Machine Learning (ICML), 2006.

[9] R. Collobert, A. Hamm, and G. Synnaeve, “Word-level speech recognition with a dynamic lexicon,” arXiv preprint arXiv:1906.04525, 2019.

[10] S. Palaskar and F. Metze, “Acoustic-to-word recognition with sequence-to-sequence models,” in Proc. IEEE Workshop on Spoken Language Technology (SLT), 2018.

[11] M. Ostendorf, V. Digalakis, and O. Kimball, “From HMM’s to random fields: A summary of the JHU CLSP 2010 summer workshop,” in Proc. ICASSP, 2011.

[12] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz et al., “The Kaldi speech recognition toolkit,” in Proc. Interspeech, 2011.

[13] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” CoRR, abs/1301.3781, 2013.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[15] N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz et al., “The Kaldi speech recognition toolkit,” in Proc. IEEE Int. Workshop on Automatic Speech Recognition and Understanding (ASRU), 2011.

[16] H. Tang, L. Lu, G. Zweig, P. Nguyen, and A. Smith, “End-to-end neural segmental models for speech recognition,” in Proc. IEEE Int. Workshop on Automatic Speech Recognition and Understanding (ASRU), 2019.

[17] H. Kamper, W. Wang, and K. Livescu, “Deep convolutional acoustic word embeddings using word-pair side information,” in Proc. ICASSP, 2016.

[18] S. Settle and K. Livescu, “Discriminative acoustic word embeddings: Recurrent neural network-based approaches,” in Proc. IEEE Workshop on Spoken Language Technology (SLT), 2016.

[19] Y.-A. Chung, C.-C. Wu, C.-H. Shen, H.-Y. Lee, and L.-S. Lee, “Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoders,” in Proc. Interspeech, 2016.

[20] W. He, W. Wang, and K. Livescu, “Multi-view recurrent neural acoustic word embeddings,” in Proc. Int. Conf. on Learning Representations (ICL), 2017.

[21] S. Settle, K. Levin, H. Kamper, and K. Livescu, “Query-by-example search with discriminative neural acoustic word embeddings,” in Proc. Interspeech, 2017.

[22] K. Audhkhasi, A. Rosenberg, A. Sethy, B. Ramabhadran, and B. Kingsbury, “End-to-end ASR-free keyword search from speech,” IEEE Journal of Selected Topics in Signal Processing, vol. 11, no. 8, pp. 1351–1359, 2017.

[23] H. Tang, W. Wang, K. Gimpel, and K. Livescu, “Discriminative segmental cascades for feature-rich phone recognition,” in Proc. IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), 2015.

[24] H. Tang, L. Lu, K. Gimpel, C. Dyer, and A. S. Smith, “End-to-end neural segmental models for speech recognition,” IEEE Journal of Selected Topics in Signal Processing, 2017.

[25] L. Kong, C. Dyer, and N. A. Smith, “Segmental recurrent neural networks,” in Proc. Int. Conf. on Learning Representations (ICLR), 2016.

[26] G. Zweig et al., “Speech recognition with segmental conditional random fields: A summary of the IJU CLSP 2010 summer workshop,” in Proc. ICASSP, 2011.

[27] A. Paszke et al., “PyTorch: An imperative style, high-performance deep learning library,” in Advances in Neural Information Processing Systems (NIPS), 2019.

[28] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” CoRR, abs/1301.3781, 2013.

[29] J. Pennington, R. Socher, and C. Manning, “GloVe: Global vectors for word representation,” in Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[30] E. J. Goldberg, J. Wang, and C. Dyer, “Deep convolutional acoustic word embeddings,” in Proc. NAACL, 2018.

[31] H. Kamper, W. Wang, K. Livescu, and M. Picheny, “Building competitive direct acoustics-to-word models for English conversational speech recognition,” in Proc. Int. Conf. on Learning Representations (ICLR), 2017.

[32] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. NAACL, 2019.

[33] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2015.

[34] J. J. Godfrey, E. C. Holiman, and J. McDaniel, “SWITCHBOARD: Telephone speech corpus for research and development,” in Proc. ICASSP, 1992.

[35] V. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz et al., “The Kaldi speech recognition toolkit,” in Proc. ICASSP, 2014.

[36] V. Vinyals, O. Vinyals, J. Dean, and Q. V. Le, “SpecAugment: A simple data augmentation method for automatic speech recognition,” in Proc. Interspeech, 2019.

[37] S. Ren, M. Long, and A. Birch, “Neural machine translation of rare words with subword units,” in Proc. Association for Computational Linguistics, 2016.