Timestamped Embedding-Matching Acoustic-to-Word CTC ASR

Woojay Jeon

Abstract—In this work, we describe a novel method of training an embedding-matching word-level connectionist temporal classification (CTC) automatic speech recognizer (ASR) such that it directly produces word start times and durations, required by many real-world applications, in addition to the transcription. The word timestamps enable the ASR to output word segmentations and word confusion networks without relying on a secondary model or forced alignment process when testing. Our proposed system has similar word segmentation accuracy as a hybrid DNN-HMM (Deep Neural Network-Hidden Markov Model) system, with less than 3ms difference in mean absolute error in word start times on TIMIT data. At the same time, we observed less than 5% relative increase in the word error rate compared to the non-timestamped system when using the same audio training data and nearly identical model size. We also contribute more rigorous analysis of multiple-hypothesis embedding-matching ASR in general.

Index Terms—Automatic speech recognition, CTC, embedding matching.

I. INTRODUCTION

RECENTLY, embedding-matching acoustic-to-word (A2W) ASR [1], [2] has gained interest due to its flexibility in accommodating out-of-vocabulary (OOV) words. The entire vocabulary of the ASR is explicitly represented by a matrix of acoustic word embeddings [3], [4] generated by an external word encoder. The matrix can be treated independently of the rest of the system, and can be extended with more columns (words) with no systemic limits. This modularity is attractive for scenarios where the ASR's vocabulary must be extended with dynamic out-of-vocabulary (OOV) words – especially user-dependent named entities – on-the-fly for every utterance. Named entities can include contact names, on-device song titles, and location-dependent points of interest, all of which can change over time for a given user, and vastly differ from user to user. The system is also scalable because it distills continuous word recognition into a series of nearest neighbor searches over a vector space, which is highly parallelizable in modern hardware [5].

In this paper, a *timestamped* embedding-matching A2W system is proposed, where the underlying connectionist temporal classification (CTC) model is trained such that for every hypothesized word, the model also estimates a word timestamp – the start time and duration of the word. The model learns this capability from ground truth segmentations provided during training. The timestamps allow the decoder to directly produce the word segmentations of recognition results, as well as

Woojay Jeon is with Apple in Cupertino, California, U.S.A. (e-mail: woojay@apple.com)

word confusion networks [6], [7], which are vital to many speech-based applications. Audio segmentations are used for keyword search [8], as well as the "intercom" feature of digital assistants. Word confusion networks are needed to rank and display word alternatives in a mobile user interface [9], semantically parse the utterance [10], and mitigate errors in spoken language translation [11].

A "pure" end-to-end ASR where a static monolithic model produces only the word transcription is of limited use in practical applications [12]. The ability to dynamically modify or augment the vocabulary, as well as produce accurate word segmentations is also needed. The timings naturally produced by conventional CTC models are generally too inaccurate [13] to be relied on. Furthermore, the interval between word onsets are poor estimates of word duration when there is significant silence between the words. A common remedy is to run a second-pass forced alignment using a hybrid DNN-HMM acoustic model [14] to obtain the segmentation, but this is inefficient, and not scalable for multiple hypotheses. Other works used a constrained attention head in a second-pass rescoring model to emit timings from an RNN transducer model [15], combined a CTC model with a separate framewise phoneme classifier [16], or used a separate timestamp estimator model with a CTC-attention model [13]. Slightly related to the timestamp problem is the streaming latency problem, where E2E models are trained to reduce the latency of word outputs [17]. Such methods cannot be a substitute for explicit timestamp estimation, since they are essentially dealing with word *end* times, not start times and durations, and furthermore it is likely impossible to achieve zero streaming latency.

In our proposed system, the word-level CTC model directly provides a timestamp with each word, with no additional model or alignment process at runtime. A DNN-HMM ASR is needed to force-align the audio to the transcriptions to obtain ground truth segmentations used to train the CTC model, but this force-alignment only needs to be done once at *data collection time*, meaning that the ground truth can be permanently stored and reused for the lifetime of the training data. To the best of our knowledge, no CTC model with such timestamping capability exists in the literature. Our system also produces timestamps for every hypothesized word, not just the 1-best words. If a CTC model can directly produce timestamps without requiring an additional system at run-time, the system becomes simpler and faster while consuming less power and computational resources.

We will begin with some theoretical analysis that was absent in our previous study [2] on embedding-matching word CTC ASR. In particular, the multiple-hypothesis system we

proposed in [2] left open the question of why a simple summation of scores would not result in the multiple embeddings collapsing into the same embedding. In this paper, we provide some theoretical developments that address this question. We will then describe and analyze the proposed timestamped system, followed by experimental results on its word segmentation accuracy and its impact on recognition accuracy.

II. A MORE RIGOROUS ANALYSIS OF EMBEDDING-MATCHING WORD CTC

A. Review of Conventional Word CTC

We begin with a brief review of conventional word CTC ASR. The word CTC loss [18], [19] is the negative sum of the log posterior over all training samples $\mathcal{D}$ where each sample consists of an audio input X and a reference word sequence W:

$$\mathcal{L} = - \sum_{(X,W)\in\mathcal{D}} \ln \hat{P}(W|X). \quad (1)$$

Each $\hat{P}(W|X)$ is the sum of the individual posteriors of all possible word sequences π in a set $\mathcal{B}^{-1}(W)$ that map to W, where the "words" in π include a "blank" word:

$$\hat{P}(W|X) = \sum_{\pi\in\mathcal{B}^{-1}(W)} \hat{P}(\pi|X). \quad (2)$$

Each sequence posterior $\hat{P}(\pi|X)$ is in turn the product of the frame-wise scores for every word in the sequence:

$$\hat{P}(\pi|X) = \prod_{t=1}^{T} p_{t,\pi_t}, \quad (3)$$

where T is the number of frames and the score $p_{t,i}$ for every word w_i in a vocabulary of n words at time t is obtained by applying a softmax to the output of a neural network:

$$p_{t,i} = \frac{\exp(s_{t,i})}{\sum_{j=1}^{n}\exp(s_{t,j})}, \ 1 \le t \le T, \ 1 \le i \le n \quad (4)$$

B. Analysis of Single-Hypothesis System

An "embedding-matching" word CTC [1], [2] differs from conventional word CTC in how each $s_{t,i}$ in Equation (4) is computed. Instead of training a neural network to freely and directly produce $s_{t,i}$, we train the network to produce an audio embedding vector $\mathbf{f}_t$. Each $s_{t,i}$ is explicitly constrained to be the negative $\mathcal{L}_2$ distance as in our previous study [2] (or inner product [1]) between $\mathbf{f}_t$ and a text embedding vector $\mathbf{g}_i$ precomputed by an external acoustic text encoder [3], [4] for word w_i. Only the blank word's score $s_{t,0}$ is produced directly:

$$s_{t,i} = \begin{cases} -b_t^2 & (i = 0) \\ -\|\mathbf{f}_t - \mathbf{g}_i\|^2 & (i = 1, \cdots, n) \end{cases} \quad (5)$$

In practice, as we noted previously [2], all $\mathbf{g}_i$ are stored as the columns of matrix G, and a sequence of matrix operations result in a vector $\mathbf{s}_t$ containing the $s_{t,i}$'s.

Although the *pre-softmax* score $s_{t,i}$ is maximized when $\mathbf{f}_t = \mathbf{g}_i$, the *post-softmax* score $p_{t,i}$ is not necessarily maximized,

due to the effect of the competing scores in the denominator in (4). This would imply that the CTC criterion – which essentially attempts to maximize $p_{t,i}$ for reference word i at some time t – may not *strictly* try to make $\mathbf{f}_t$ match $\mathbf{g}_i$. For example, consider a vocabulary of 3 words with 1-dimensional embedding vectors $\mathbf{g}_1 = [1.0]$, $\mathbf{g}_2 = [2.0]$, and $\mathbf{g}_3 = [4.0]$. $s_{t,2}$ is maximized when $\mathbf{f}_t = [2.0]$, but $p_{t,2}$ is maximized when $\mathbf{f}_t = [2.385]$.

We can argue, however, that if the embedding dimensions are sufficiently higher than 1, we can expect $\mathbf{f}_t \approx \mathbf{g}_i$ when $p_{t,i}$ is maximized. The log of the softmax in Equation (4) is

$$\ln p_{t,i} = -\ln\left[1 + \sum_{j\neq i}^{n}\exp(s_{t,j} - s_{t,i})\right]. \quad (6)$$

A well-known interpretation of the softmax [20] is that if $s_{t,i}$ is much higher than all other $s_{t,j}$, then the summation term above will be close to 0, resulting in $p_{t,i} \approx 1$ while $p_{t,j} \approx 0$ ($j \neq i$). In our case, if $\mathbf{f}_t = \mathbf{g}_i$, we will have $s_{t,i} = 0$, while $s_{t,j}$ will be strongly negative if $\mathbf{g}_i$ is sufficiently far away from the rest of the vectors. This effect is more apparent for higher embedding dimensions, since more negative terms are added to $s_{t,j}$. Hence, we conclude that $\mathbf{f}_t = \mathbf{g}_i$ asymptotically maximizes $p_{t,i}$.

Other studies [1], [21] used an inner product instead of negative Euclidean distance for $s_{t,i}$. In Appendix A, we show that a similar argument as above can be made for the inner product that for sufficiently high dimensions, one can expect that $\mathbf{f}_t = \mathbf{g}_i$ will asymptotically maximize $p_{t,i}$.

C. Analysis of Multiple-Hypothesis System

In the multiple-embeddings approach we proposed in [2], the underlying model produces L embeddings $\mathbf{f}_t^{(1)}, \cdots, \mathbf{f}_t^{(k)}$, and their pre-softmax scores are summed:

$$s_{t,i} = \begin{cases} -b_t^2 & (i = 0) \\ -\sum_{k=1}^{L}\|\mathbf{f}_t^{(k)} - \mathbf{g}_i\|^2 & (i = 1, \cdots, n) \end{cases} \quad (7)$$

The training can be best understood with a simple example, with $L = 2$. Consider a training dataset of just two utterances that have almost the same audio but different transcriptions: "Directions to First and Park" and "Directions to Fursten Park." We can further simplify this as the *same* audio producing two different outputs, as shown in Fig. 1.

Assuming the language model does not contain the compound word "first_and", all three words "First," "and," and "Fursten" must be separately hypothesized by the acoustic model. Furthermore, to avoid a result like "First Fursten" or "Fursten First," the system will ideally output "First" and "Fursten" simultaneously (Figure 8a shows an example).

Suppose $w_a =$ "First" and $w_b =$ "Fursten", and their text embedding vectors are $\mathbf{g}_a$ and $\mathbf{g}_b$, respectively. The intent of the multiple-hypothesis system is to allow the network, at some point in time t, to output $\mathbf{f}_t^{(1)} = \mathbf{g}_a$ and $\mathbf{f}_t^{(2)} = \mathbf{g}_b$ (or vice versa) for the input audio in Fig. 1.

Since the training criterion is to maximize the posterior for both training samples, we can consider the post-softmax score p_a and p_b at time t. We need not consider the case where p_a
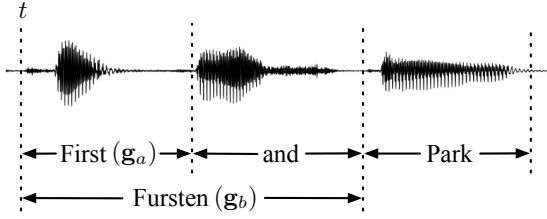
Fig. 1. Example where two training samples have nearly-identical audio but different word transcriptions "First and Park" and "Fursten Park." The multiple-hypothesis embedding-matching system is induced to produce, for the same input audio, two different embeddings $\mathbf{f}^{(1)} = \mathbf{g}_a$ and $\mathbf{f}^{(2)} = \mathbf{g}_b$ at some point in time t (which may be anywhere, not necessarily at the beginning of the words as shown in the diagram), where $\mathbf{g}_a$ is the text embedding for "First" and $\mathbf{g}_b$ is the text embedding for "Fursten."

and p_b are maximized at different times because that would result in the model outputting the two words in sequence, i.e., "First Fursten" or "Fursten First." Since we are dealing with only one specific value of t, we drop t from our notation for simplicity. The CTC training will effectively try to maximize

$$\ln(p) = \ln(p_a p_b) = s_a + s_b - 2\ln\sum_j \exp(s_j). \qquad (8)$$

We make a similar argument as in Sec. II-B that the maximum s_c among all s_j is distinct enough so that

$$s_c \triangleq \max_j s_j \approx \ln\sum_j \exp(s_j). \qquad (9)$$

If we assume for a moment that $\mathbf{g}_j$ can be any value,

$$\frac{\partial s_j}{\partial \mathbf{g}_j} = 2(\mathbf{f}^{(1)} - \mathbf{g}_j) + 2(\mathbf{f}^{(2)} - \mathbf{g}_j) \qquad (10)$$

and $\partial^2 s_j / \partial \mathbf{g}_j^2 = -4$ (elementwise), which means s_j is maximized by $\mathbf{g}_j = \frac{1}{2}(\mathbf{f}^{(1)} + \mathbf{f}^{(2)})$. In reality, $\mathbf{g}_j$ cannot be arbitrary, and can only be chosen from our fixed vocabulary. But if our vocabulary is large, we can assume that such a $\mathbf{g}_j$ exists. Since s_c is the maximum s_j in (9), we have $\mathbf{g}_c = \frac{1}{2}(\mathbf{f}^{(1)} + \mathbf{f}^{(2)})$. Next, we differentiate (8) to obtain

$$\frac{\partial \ln(p)}{\partial \mathbf{f}^{(1)}} = 2(\mathbf{g}_a + \mathbf{g}_b) - 2(\mathbf{f}^{(1)} + \mathbf{f}^{(2)}) \qquad (11)$$

and $\partial^2 \ln(p)/\partial \mathbf{f}^{(1)^2} = -2$. Hence, $\ln(p)$ is maximized by any values for $\mathbf{f}^{(1)}$ and $\mathbf{f}^{(2)}$ that satisfy:

$$\mathbf{f}^{(1)} + \mathbf{f}^{(2)} = \mathbf{g}_a + \mathbf{g}_b, \qquad (12)$$

which also implies $\mathbf{g}_c = \frac{1}{2}(\mathbf{g}_a + \mathbf{g}_b)$.

We now see it is possible to have $\mathbf{f}^{(1)} = \mathbf{f}^{(2)} = \frac{1}{2}(\mathbf{g}_a + \mathbf{g}_b)$, i.e., both hypotheses are always the same, and always point to the "midpoint" between "First" and "Fursten." In such a case, our network is equivalent to the single-embedding case, and it serves no purpose to produce two embeddings.

However, the analysis changes when we assume a high number of dimensions. The vectors become sparse, so we can no longer assume the existence of $\mathbf{g}_c$ in the vocabulary. It is also known that the $\mathcal{L}_2$ distances become more and more similar so that there is little difference between "near" and "far." [22]. To simplify our analysis, in Appendix B we approximate all $\mathcal{L}_2$ distances in high dimensions as *binary*

distances, i.e., 0 if there is an exact match between the vectors, and some positive constant D otherwise:

$$s_j = D\big\{\delta(\mathbf{f}^{(1)} - \mathbf{g}_j) - 1\big\} + D\big\{(\delta(\mathbf{f}^{(2)} - \mathbf{g}_j) - 1\big\} \;\; \forall j, \; (13)$$

where $\delta(.)$ is 1 if the input is 0, and 0 otherwise. Applying Equation (13) to (8), we can identify 8 different cases for $\ln(p)$ after accounting for symmetry, shown in Appendix C, to see that $\ln p$ is maximized when $\mathbf{f}^{(1)} = \mathbf{g}_a$ and $\mathbf{f}^{(2)} = \mathbf{g}_b$, or vice versa. We therefore conclude that for a sufficiently high number of dimensions, the network will be trained so that $\mathbf{f}^{(1)}$ becomes the embedding for "First" and $\mathbf{f}^{(2)}$ becomes the embedding for "Fursten" (or vice versa).

III. PROPOSED METHOD

A. Basic Intuition

The following two arguments are central to our approach:
1) CTC models do not give timely reports, but they can still report accurate timings.
2) We use a vocabulary to train an embedding-matching A2W, but teaching the vocabulary is not our main goal.

It is well-known that CTC [18], by design, operates in a "segmentation-free" manner. The training criterion does not require segmentation information, nor is segmentation outputted by the model. A CTC model's decision about a word can be made at any indeterminate point in time, as long as the correct words are produced in the right order. This, however, does not necessarily mean that the model cannot *know* the segmentation. In an embedding-matching CTC model, we previously argued [2] that the network acts as a word segmenter followed by an audio segment encoder. If this interpretation is indeed true, then the CTC model *must* internally know the precise location and duration of every word it hypothesizes, and we need only to find a way to expose that knowledge.

Consider an example vocabulary of 3 words, $\{I, like, honey\}$. We imagine expanding this to a vocabulary of *timestamped words*, each with the form "`<word>_<start time>_<duration>`". If there are 100 possible start times and 10 possible durations, we would have a vocabulary of 3,000 timestamped words, i.e., $\{I\_1\_1, \cdots, I\_100\_10, like\_1\_1, \cdots, honey\_100\_10\}$. Suppose we define the embedding vector for each timestamped word as the original word's embedding vector with the start time and duration appended as two additional dimensions. If the text embedding for "like" is $\mathbf{g}_{like}$, the embedding for "$like\_\alpha\_\beta$" is $\mathbf{g}_{like\_\alpha\_\beta} = [\mathbf{g}_{like}^T \;\; \alpha \;\; \beta]^T$.

It is then conceivable that we use a training method similar to Section II with the 3-word vocabulary replaced by the expanded 3,000-word vocabulary. Reference start times and durations for the training samples are prepared via forced alignment using a conventional DNN-HMM hybrid ASR [14], [23], [24], and are attached to the words when training the CTC, e.g. "$I\_19\_2 \; like\_31\_5 \; honey\_52\_9$.". Once trained, the CTC model outputs sequences from the timestamped 3,000-word vocabulary, and we have achieved our goal. Obviously, this approach is impractical. The training vocabulary is increased by a factor of 1,000, which is prohibitively large.
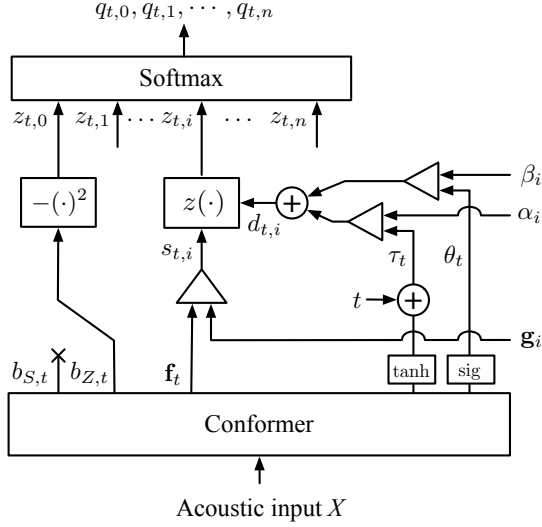
Fig. 2. Training mode of the proposed single-hypothesis system; in particular, the computation of the timestamped-word CTC loss $\mathcal{L}_Z$ at time t. A comparator ($\triangle$) represents the Euclidean distance between two inputs (for $s_{t,i}$, the distance is negated). For clarity, we show only the computation of $z_{t,i}$ for the i'th word in the training vocabulary of size n. The same process is applied to all the other words. Note that $b_{S,t}$ is used for computing $\mathcal{L}_S$, so is unused here. Also note that $b_{Z,t}$, $z_{t,i}$, and $q_{t,i}$ are only used during training, and not during testing (shown in Fig. 3).

This brings us to the second argument above. Note that in our previous work [2], the training vocabulary was less than 1/10th of the testing vocabulary. Our training goal is to make the conformer learn to segment the audio, and map (as an $f(\cdot)$ encoder in [3] or [4] each segment to an embedding. As long as this goal is achieved, it doesn't matter what exact words we put into G (the matrix of $\mathbf{g}_i$'s in Section II-B) during training. We could in fact put a *random* subset of words into G that changes for *every* training sample. [1] In our hypothetical scenario, we need not use all 3,000 timestamped words in G during training. Rather, we can use a small random sample. A static space of 1,000 possible timestamps does not even need to be defined; one could use random start and duration values where needed, as we will describe in greater detail in the next section.

B. Timestamped Single-Hypothesis System

We now formally describe the single-embedding version of our proposed system, illustrated in Fig. 2 and 3. At every time t, the underlying model proposes a word-based blank score $b_{S,t}$, a timestamped-word-based blank score $b_{Z,t}$, an acoustic embedding vector $\mathbf{f}_t$, and a timestamp vector $\mathbf{u}_t = [\tau_t \quad \theta_t]^T$ where τ_t is the start time and θ_t is the duration for $\mathbf{f}_t$.

When producing τ_t, the time t of the current output frame is added first, so the neural network is actually trained to produce *offsets* rather than absolute times. A scaled hyperbolic tangent is used to limit the offset to -1 to $+1$ seconds. When producing θ_t, a sigmoid is used to ensure that the value is

---

[1]Note that the model probably does learn some linguistic information that helps it identify word segments, but our argument still stands that as long as we keep the randomized G reasonably large and fairly sampled, the exact words we include it in are not that important.
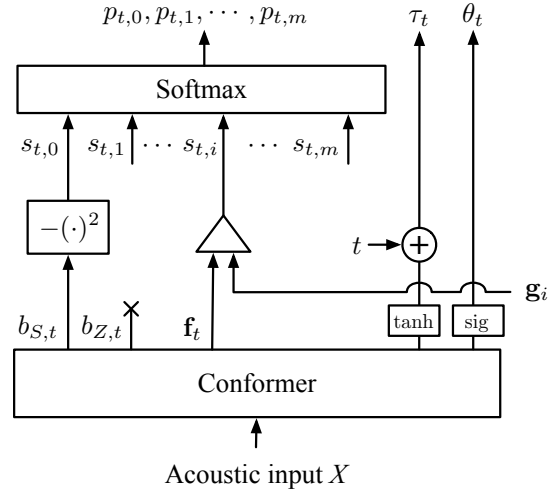

Fig. 3. Testing mode of the proposed single-hypothesis system. We compute the $s_{t,i}$'s and $p_{t,i}$'s the same way as in our original (non-timestamped) embedding-matching system [2] for every word in a testing vocabulary of size m, and also output a timestamp (τ_t, θ_t) at every time t. The blank score $b_{Z,t}$ used to train $\mathcal{L}_Z$ is ignored during testing.

positive, with a scale factor of 2 assuming a maximum word duration of 2 seconds. In Appendix E, we show that both limits are conservatively high, and do not have any effect at testing time. However, when the model was trained without the sigmoid and hyperbolic tangent, some degradation was observed in the recognition and timestamp accuracy, implying that the operations still help at training time.

The model is trained by using the sum of two CTC losses as the optimization criterion: a word CTC loss $\mathcal{L}_S$ and a timestamped-word CTC loss $\mathcal{L}_Z$:

$$\mathcal{L} = \mathcal{L}_S + \mathcal{L}_Z. \tag{14}$$

The word CTC loss $\mathcal{L}_S$ is exactly the same as that we used previously [2], defined in Equation (1), using the scores in Equation (5) where $-b_{S,t}^2$ is the blank score:

$$s_{t,i} = \begin{cases} -b_{S,t}^2 & (i = 0) \\ -\|\mathbf{f}_t - \mathbf{g}_i\|^2 & (i = 1, \cdots, n) \end{cases} \tag{15}$$

The timestamped-word CTC loss $\mathcal{L}_Z$ includes a sequence of word timestamps V corresponding to the word sequence W of every training sample:

$$\mathcal{L}_Z = -\sum_{(X,W,V)\in\mathcal{D}} \ln \hat{P}(W, V|X). \tag{16}$$

For computing $\hat{P}(W, V|X)$, we use the following pre-softmax score:

$$z_{t,i} = \begin{cases} -b_{Z,t}^2 & (i = 0) \\ z(s_{t,i}, d_{t,i}) & (i = 1, \cdots, m) \end{cases} \tag{17}$$

where

$$z(s, d) \triangleq s - d + sd \tag{18}$$

$$d_{t,i} \triangleq \|\mathbf{u}_t - \mathbf{v}_i\|^2 = (\tau_t - \alpha_i)^2 + (\theta_t - \beta_i)^2. \tag{19}$$

Here, we do not use a fixed-vocabulary G. A timestamped-vocabulary of size m is prepared on-the-fly for every training sample. Each i'th entry in the vocabulary is a tuple $(\mathbf{g}_i, \mathbf{v}_i)$ where $\mathbf{v}_i$ is a timestamp $[\alpha_i \quad \beta_i]^T$ for $\mathbf{g}_i$. The m training samples consist of reference samples and negative samples. The reference samples are the reference words and their corresponding reference timestamps (obtained from forced-alignment using a DNN-HMM ASR). The negative samples are generated by taking 1. All the reference words, with random perturbation added to their timestamps and 2. Randomly-chosen non-reference words with random timestamps, where the timestamps are perturbed versions of the timestamps in the reference transcription. We found that such perturbed versions of the reference timestamps – rather than purely random timestamps – give better results.

The post-softmax score is

$$q_{t,i} = \frac{\exp(z_{t,i})}{\sum_j \exp(z_{t,j})}. \tag{20}$$

By inspection, we can see that the same argument can be made regarding $q_{t,i}$ as we did for $p_{t,i}$ in Sec. II-B. Since $d_{t,i} \geq 0$, it always makes $z_{t,i}$ lower, and leaves $z_{t,i}$ unchanged when $d_{t,i} = 0 \leftrightarrow \mathbf{u}_t = \mathbf{v}_i$. When $\mathbf{f}_t = \mathbf{g}_i$, $\tau_t = \alpha_i$, and $\theta_t = \beta_i$, we get the maximum value for $z_{t,i}$, and for a sufficiently large number of dimensions in the embedding vectors, $z_{t,i}$ will be much higher than the other $z_{t,j}$'s, allowing $q_{t,i}$ to be close to the maximum value of 1.

This also implies that $\mathcal{L}_Z$ may be all we need in Equation (14). The reason why we still include $\mathcal{L}_S$ is the blank score. $-b_{Z,t}^2$ from $\mathcal{L}_Z$ is trained to compete with the non-blank scores in Equation (17). However, at testing time, we use the non-blank scores in Equation (5), so must use a compatible blank score, which is $-b_{S,t}^2$ trained using $\mathcal{L}_S$. Also note that the non-blank scores in Equation (17) are generally lower than the non-blank scores in Equation (5) because the former are weighed down by the timestamp estimation errors. We would get many insertion errors if we used $-b_{Z,t}^2$ for testing, because it must compete with non-blank scores that are overall higher than what it had seen at training time.

Testing mode is shown in Fig. 3. When testing, we use $s_{t,i}$ and $p_{t,i}$ in Equation (5) and (4) as in the previous system, and attach a timestamp vector $\mathbf{u}_{t,i} = [\tau_t \quad \theta_t]^T$ to every $s_{t,i}$ $(i > 0)$. Whenever the prefix beam decoder chooses a word w_i at time t with score $s_{t,i}$, it takes $\mathbf{u}_{t,i}$ as the word's timestamp.

C. Timestamped Multiple-Hypothesis System

The multiple-embedding version of our system is shown in Fig. 4 and Fig. 5. For each k'th embedding vector $\mathbf{f}_t^{(k)}$, the conformer outputs a timestamp vector $\mathbf{u}_t^{(k)} = [\tau_t^{(k)} \quad \theta_t^{(k)}]$.

For each label i, the pre-softmax score for $\mathcal{L}_Z$ is based on the multiple-embedding score $s_{t,i}$ in Equation (7) and the hypothesized timestamp corresponding to the best-matching hypothesized embedding:

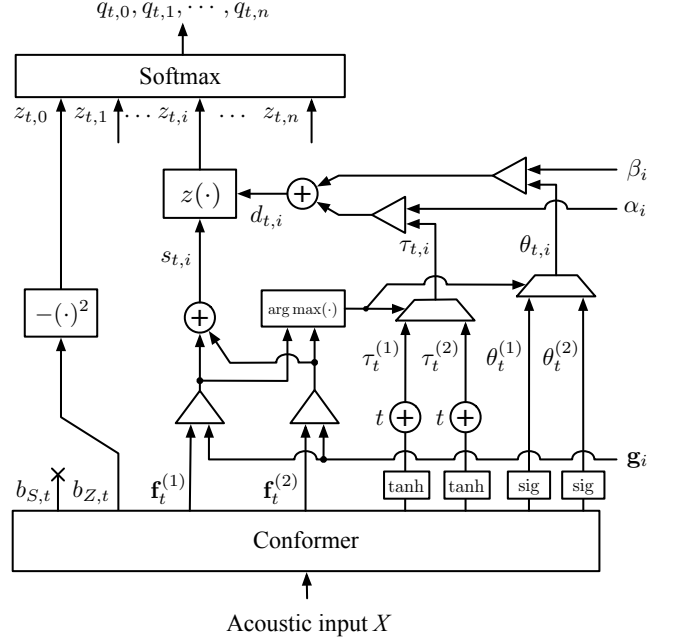$$z_{t,i} = \begin{cases} -b_{Z,t}^2 & (i = 0) \\ z(s_{t,i}, d_{t,i}) & (i = 1, \cdots, n), \end{cases} \tag{21}$$

Fig. 4. Training mode of the proposed system when $L = 2$; in particular, computation of $\mathcal{L}_Z$ at time t. A comparator ($\triangle$) represents the $\mathcal{L}_2$ distance (negated for $s_{t,i}$) between two inputs. A multiplexer ($\triangle$) chooses one of two timestamps based on the hypothesized embedding that best matches the i'th word. For clarity, we show only the computation of $z_{t,0}$ and $z_{t,i}$ in this figure. The other $z_{t,j}$'s are computed the same way, using different α_j, β_j and $\mathbf{g}_j$. Note that $b_{S,t}$ is used for computing $\mathcal{L}_S$, so is unused here.

where $z(\cdot)$ is defined in Equation (18) and

$$s_{t,i} = \sum_{k=1}^{L} s_{t,i}^{(k)} \quad (i = 1, \cdots, n) \tag{22}$$

$$s_{t,i}^{(k)} = -\|\mathbf{f}_t^{(k)} - \mathbf{g}_i\|^2 \quad (i = 1, \cdots, n) \tag{23}$$

$$d_{t,i} = \|\mathbf{u}_{t,i} - \mathbf{v}_i\|^2 = (\tau_{t,i} - \alpha_i)^2 + (\theta_{t,i} - \beta_i)^2 \tag{24}$$

$$\mathbf{u}_{t,i} = \mathbf{u}_t^{(l)} \tag{25}$$

$$l = \arg\max_k s_{t,i}^{(k)} \quad (i = 1, \cdots, n). \tag{26}$$

The reasoning behind Equation (26) is that since each $\mathbf{f}^{(k)}$ corresponds to one segment in the speech, for every word w_i it will be the $\mathbf{f}^{(k)}$ that best matches the word that will also have the most accurate timestamp. We only care about cases when there exists a strong match with w_i. If all $\mathbf{f}^{(k)}$'s have a weak match with w_i, the word wouldn't be chosen by the decoder so it doesn't matter what timestamp we assign to it. If multiple $\mathbf{f}^{(k)}$'s have a strong match with the same w_i, those $\mathbf{f}^{(k)}$'s are also likely to have very similar timestamps (since they are representing the same audio segment), so we could choose any one of them.

D. Analysis of Timestamped Multiple-Hypothesis System

We extend the example in Sec. II-C to assume one input training audio with two different reference *timestamped* word transcriptions. We want to show that the timestamped multiple-hypothesis system with $L = 2$ will be able to separately represent *both* the word embeddings and the timestamps for "First" and "Fursten."
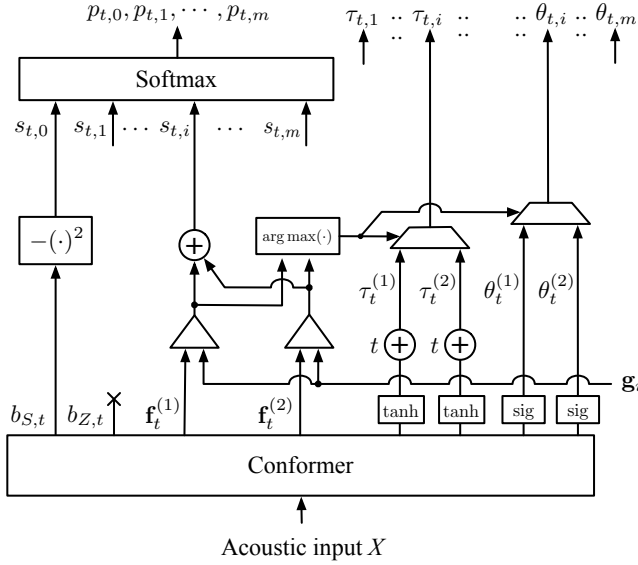
Fig. 5. Testing mode of the proposed multiple-hypothesis ($L = 2$) system. At time t, for every word w_i in the testing vocabulary, the network produces a posterior score $p_{t,i}$, a start time $\tau_{t,i}$, and a duration $\theta_{t,i}$. For clarity, in the figure we only show the score computation for the i'th word. The same process is applied to all the other words. The blank score $b_{Z,t}$ used to train $\mathcal{L}_Z$ is ignored during testing.
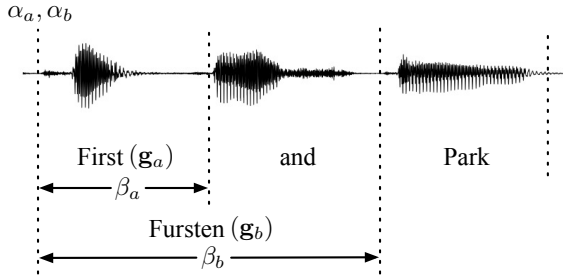

Fig. 6. Extension of the example in Fig. 1 with timestamps $\mathbf{v}_a = (\alpha_a, \beta_a)$ and $\mathbf{v}_b = (\alpha_b, \beta_b)$ included in the reference transcriptions where α_i and β_i are the start time and duration, respectively, of word w_i. This time, for one audio input, the multiple-hypothesis embedding-matching system must learn to produce two different *timestamped* embeddings $(\mathbf{g}_a, \mathbf{v}_a)$ and $(\mathbf{g}_b, \mathbf{v}_b)$.

Word w_a ("First") is described by $(\mathbf{g}_a, \mathbf{v}_a)$ where $\mathbf{v}_a$ stores the reference timestamp $[\alpha_a, \beta_b]^T$. Word w_b ("Fursten") is described by $(\mathbf{g}_b, \mathbf{v}_b)$. The CTC's underlying model produces two embeddings $\mathbf{f}_t^{(1)}$ and $\mathbf{f}_t^{(2)}$, and two corresponding timestamps $\mathbf{u}_t^{(1)} = [\tau_t^{(1)}, \theta_t^{(1)}]$ and $\mathbf{u}_t^{(2)} = [\tau_t^{(2)}, \theta_t^{(2)}]$.

Since our optimization criterion in (14) is the sum of the log of two CTC losses, we can consider the product of the post-softmax scores $p_a \cdot p_b \cdot q_a \cdot q_b$ at some point in time t that CTC training would attempt to maximize. We will show that the following condition is a sufficient condition for maximizing q:

$$\mathbf{f}_t^{(1)} = \mathbf{g}_a, \quad \mathbf{u}_t^{(1)} = \mathbf{v}_a, \quad \mathbf{f}_t^{(2)} = \mathbf{g}_b, \quad \text{and} \quad \mathbf{u}_t^{(2)} = \mathbf{v}_b \quad (27)$$

where 1 and 2 may be swapped.

We already know that (27) maximizes $p_a \cdot p_b$, so we only consider $q = q_a \cdot q_b$. As before, we only need to consider 1 point in time, so we drop t to simplify the notation. Note that $\mathbf{f}^{(1)} = \mathbf{g}_a$ implies $\mathbf{u}_a = \mathbf{u}^{(1)}$, i.e., for word w_a, the first

embedding is the obvious "winner", so w_a is also assigned the first timestamp. Likewise, $\mathbf{f}^{(2)} = \mathbf{g}_b$ implies $\mathbf{u}_b = \mathbf{u}^{(2)}$.

First, one can see that

$$\frac{\partial z_j}{\partial \mathbf{u}_i} = \begin{cases} 2(\mathbf{u}_i - \mathbf{v}_i)(s_i - 1), & \text{for } i = j \\ 0, & \text{for } i \neq j. \end{cases} \quad (28)$$

We wish to maximize

$$\ln(q) = \ln(q_a) + \ln(q_b) = z_a + z_b - 2\ln\sum_j \exp(z_j). \quad (29)$$

We can see that

$$\frac{\partial \ln q}{\partial \mathbf{u}_a} = \frac{\partial z_a}{\partial \mathbf{u}_a}(1 - 2q_a) \quad (30)$$

$$\frac{\partial^2 \ln q}{\partial \mathbf{u}_a^2} = 2(s_a - 1)(1 - 2q_a) - 2\left(\frac{\partial z_a}{\partial \mathbf{u}_a}\right)^2 q_a(1 - q_a). \quad (31)$$

Hence, $q_a < 1/2$ is a sufficient condition for $\ln q$ to be concave with respect to $\mathbf{u}_a$, in which case $\ln q$ will be maximized with respect to $\mathbf{u}_a$ when $\mathbf{u}_a = \mathbf{v}_a$. By symmetry, if $q_b < 1/2$, $\ln q$ will be maximized with respect to $\mathbf{u}_b$ when $\mathbf{u}_b = \mathbf{v}_b$. In Appendix D, we apply the same assumption as in Sec. II-C for high embedding dimensions to show that $q_a < 1/2$ and $q_b < 1/2$ indeed holds, and we also show that $\mathbf{f}^{(1)} = \mathbf{g}_a$ and $\mathbf{f}^{(2)} = \mathbf{g}_b$ maximize q. Hence, Condition (27) is a sufficient condition for maximizing q.

IV. EXPERIMENTS

We trained "previous" embedding-matching models using word-pronunciation embeddings in the same manner as we previously proposed in [2]. The training and development data consisted of utterances spoken by diverse anonymized speakers to a digital assistant. The training data had a total 2M utterances (3,030 hours), and the development data had 400k utterances (534 hours). We then trained the proposed timestamped models using the same audio and embeddings as the previous models, but also with ground truth timestamps provided by a DNN-HMM ASR. We also varied the number of internal embeddings (the value of L in Equations (7) and (22)) from 1 to 3.

All models were conformer [25] networks, with 272 encoder dimensions, 8 attention heads, 18 layers, and a kernel size of 31. A final linear transformation was applied to the conformer's output to obtain the desired number of embeddings and the timestamps (when applicable). For example, for the proposed system with 3 internal embeddings, the final output of the model had $40 \times 3 + 2 + 2 \times 3 = 127$ dimensions (3 embeddings, 2 blank scores, 3 start times, and 3 durations). The final models had the following total number of parameters:

The training schedule follows the 2-phase specification in [25] and [26], but with an additional third phase where the learning rate decays exponentially (rather than following the inverse square root), which we found to improve accuracy. The schedule is specified by 4 variables: the number of steps p_1 in the first phase (warmup phase), the number of steps p_2 in the second phase (inverse square root), the exponential decay

TABLE I
TOTAL NUMBER OF TRAINABLE MODEL PARAMETERS

Model	Internal Embeddings (L)	Parameters
Previous	1	32,403,673
	2	32,414,593
	3	32,425,513
Proposed	1	32,404,492
	2	32,415,958
	3	32,427,424

rate γ in the third phase, and the peak learning rate R. The learning rate at training step s is

$$\begin{cases} \dfrac{R}{p_1}s & 1 \leq s \leq p_1 \\ \dfrac{R\sqrt{p_1}}{\sqrt{s}} & p_1 \leq s \leq p_1 + p_2 \quad (32) \\ R\sqrt{\dfrac{p_1}{p_1 + p_2}} \cdot 2^{(p_1+p_2)/\gamma} \cdot 2^{-s/\gamma} & p_1 + p_2 \leq s \end{cases}$$

We used $R = 0.01/\sqrt{d}$ where d is the encoder dimensions of the conformer. We also set $p_1 = 30k$, $p_2 = 270k$, and $\gamma = 30k$ with a batch size of 256. A convolutional subsampling was applied to the acoustic input before the conformer, resulting in a frame rate of 25 frames/second (or 40ms per frame). For every training job, the model produced at every epoch was evaluated on a small held-out tuning dataset of 4k utterances, and the model that had the lowest WER was chosen as the final model.

All embedding-matching CTC models were word-pronunciation-based models as described in [2], using a pronunciation-based text encoder as described in [3] that outputs text embeddings with 40 dimensions. For the previous models, the training vocabulary size was 39,478. For the timestamped models, the randomized timestamp vocabulary size was fixed to a similar size, 40,000.

A. Segmentation Accuracy on TIMIT Data

The TIMIT [27] dataset includes human-edited reference word timestamps [28]. We force-aligned the TIMIT audio to the reference words using the proposed timestamped model, as well as a hybrid DNN-HMM ASR (the same model that provides the ground truth when training the proposed model) to obtain hypothesized start times and durations. Force alignment using the embedding-matching models is done by constraining the prefix beam decoder (e.g. [29]) to only follow the reference words. We compute the word start error as the absolute difference between the hypothesized start and the reference start for every word, then compute the mean of this error over all the words. In another study [13], the same metric is called "word absolute start time delta (WASTD)." The same mean absolute error can be computed for word duration.

We will now describe how the decoder extracts word timestamps with the proposed CTC model. The *first* occurrence of each word in the 1-best alignment is used to determine the timestamp. For example, assume the optimal label sequence found by the beam decoder has three words w_{41}="how's",

w_{25}="the", and w_{73} ="weather" as follows ("_" indicates blank):

Time	0	1	2	3	4	5	6	7	8	9	10
Word	_	_	how's	how's	_	_	the	weather	_	_	_

The timestamps are $\mathbf{u}_{2,41}$, $\mathbf{u}_{6,25}$, $\mathbf{u}_{7,73}$, or $(\tau_{2,41}, \theta_{2,41})$, $(\tau_{6,25}, \theta_{6,25})$, $(\tau_{7,73}, \theta_{7,73})$. Hence, the timestamp $\mathbf{u}_{3,41}$ associated with the second *how's* is ignored. Note, however, that $\mathbf{u}_{3,41}$ is most likely very similar to $\mathbf{u}_{2,41}$, because the timestamp is part of the word identity during CTC model training (the training wants *both* $\mathbf{u}_t = \mathbf{v}_i$ *and* $\mathbf{f}_t = \mathbf{g}_i$ in Section III-B).

Label repetitions actually occur rarely, but we show a real example from the TIMIT data that demonstrates the aforementioned effect. For the dr8/mres0/si1847 utterance, our decoder using the proposed timestamped CTC model produced the sequence of labels shown in Table II, where the label for "radiated" was repeated in frames 12 and 13.

The word start and durations (in milliseconds) produced by the neural network for the corresponding labels are shown in Tables III and IV, respectively. For the repeated label, we can see that the start times and end times are near-identical for both frames 12 and 13.

A total 6,300 utterances containing a total 54,387 words were tested. Table V shows the results for the baseline hybrid DNN-HMM ASR, and the proposed timestamped embedding-matching CTC ASR using a variable number of internal embeddings. The mean word start accuracy μ_α (18.9ms for the baseline, 21.8ms for the proposed with 3 embeddings) is similar to the same measurement made for a GMM-HMM system in [13], which was 21.3ms. Compared to the baseline hybrid DNN-HMM ASR, the proposed system has less than 3ms difference in average word start time accuracy, and less than 7ms difference in average word duration accuracy. Note that the accuracy of the proposed model is bounded by the accuracy of the DNN-HMM, since the latter provides the ground truth timestamps when the former is trained. The segmentation accuracy of the proposed system also improves with increasing number of internal embeddings. This is probably for similar reasons as the improved recognition accuracy observed in [2], i.e., the ability of the embedding-matching model to generate more diverse hypotheses over more diverse word segments allow it to better represent the reference word sequences during forced alignment.

In Table VI, we ran the same experiment on Noisy TIMIT [30] for three different types of noise with varying SNR. For both the baseline DNN-HMM and the proposed systems, the accuracy degraded as the SNR decreased. The accuracy gap between the baseline and the proposed systems also seemed to get wider for lower SNR, but in most cases, the gap was not larger than 10ms.

B. Impact on Recognition Accuracy

To test for any negative impact on recognition accuracy by the addition of $\mathcal{L}_Z$ in Equation (14), we tested the system on a dataset of 35,907 utterances spoken by diverse anonymized speakers to a digital assistant. For every utterance, a user-dependent list of contact names was added to the base vo-

TABLE II
LABELS OUTPUTTED BY THE PROPOSED MODEL AT EVERY 40MS FRAME FOR dr8/mres0/si1847, STARTING AT FRAME 0 (THE TOP LEFT CELL), AND ENDING AT FRAME 58. EACH ROW IN THE TABLE SHOWS 10 FRAMES (THE FIRST ROW SHOWS FRAMES 0 ∼ 9, THE SECOND ROW SHOWS 10 ∼ 19, AND SO ON).

Frame No.	0	1	2	3	4	5	6	7	8	9
0	–	–	–	–	–	she	–	–	–	–
10	–	–	radiated	radiated	–	–	–	–	–	–
20	–	–	–	–	–	–	warmth	–	–	–
30	–	–	–	–	–	–	–	and	–	–
40	–	good	–	–	–	–	–	–	–	–
50	fellowship	–	–	–	–	–	–	–	–	–

TABLE III
WORD START TIMES (IN MS) PRODUCED BY PROPOSED MODEL FOR THE LABEL SEQUENCE IN TABLE. II. FOR THE REPEATED LABEL AT FRAMES 12 AND 13, THE WORD START TIMES ARE NEAR-IDENTICAL. THE CORRESPONDING REFERENCE START TIMES (INCLUDED IN THE TIMIT DATA) ARE 141, 362, 992, 1512, 1649, AND 1864.

Frame No.	0	1	2	3	4	5	6	7	8	9
0	–	–	–	–	–	84	–	–	–	–
10	–	–	388	389	–	–	–	–	–	–
20	–	–	–	–	–	–	1024	–	–	–
30	–	–	–	–	–	–	–	1469	–	–
40	–	1573	–	–	–	–	–	–	–	–
50	1875	–	–	–	–	–	–	–	–	–

TABLE IV
DURATIONS (IN MS) PRODUCED BY PROPOSED MODEL FOR THE LABEL SEQUENCE IN TABLE. II. FOR THE REPEATED LABEL AT FRAMES 12 AND 13, THE DURATIONS ARE NEAR-IDENTICAL. THE CORRESPONDING REFERENCE DURATIONS (INCLUDED IN THE TIMIT DATA) ARE 221, 630, 520, 137, 215, AND 587.

Frame No.	0	1	2	3	4	5	6	7	8	9
0	–	–	–	–	–	270	–	–	–	–
10	–	–	597	599	–	–	–	–	–	–
20	–	–	–	–	–	–	423	–	–	–
30	–	–	–	–	–	–	–	73	–	–
40	–	256	–	–	–	–	–	–	–	–
50	600	–	–	–	–	–	–	–	–	–

TABLE V
WORD SEGMENTATION ACCURACY ON TIMIT DATA (LOWER VALUES ARE BETTER). THE MEAN μ IN MILLISECONDS OF THE ABSOLUTE DIFFERENCE BETWEEN THE HYPOTHESIZED AND REFERENCE START TIMES (α) AND DURATIONS (β) ARE SHOWN FOR A HYBRID DNN-HMM ASR AND THE PROPOSED TIMESTAMPED SYSTEM.

System	Internal Embeddings	μ_α (ms)	μ_β
Baseline DNN-HMM	(Not applicable)	18.9	30.4
Proposed (trained on 3k hours)	1	23.6	38.7
	2	22.6	37.4
	3	21.8	37.1

cabulary. A language model (LM) that supports a *$CONTACT* variable was used with the decoder, where *$CONTACT* appears in contexts such as "*call $CONTACT office*" or "*text $CONTACT I'll be five minutes late.*". Table VII shows the ASR accuracy. In addition to the word error rate (WER), we also report a named entity error rate (NEER), which is the word error rate computed on only the named entity portions of the utterances. When comparing the proposed system with the previous system in [2] – both using 3 internal embeddings – we can notice a 4.5% relative degradation of the WER and 5.5% degradation of the NEER. This is understandable given that both systems are using the exact same training data and nearly identical model size, but the proposed system has an additional term $\mathcal{L}_Z$ in the loss function that increases the complexity of the learning criterion and requires more information (the timestamps) to be modeled compared to the previous system.

For both the previous and proposed systems, there were 3 tunable parameters for each recognizer: a *named entity weight* that is applied to the acoustic score of any named entity hypothesized by the acoustic model, a *language model weight* applied to the language model scores, and a constant *blank score divisor* [31] applied to the acoustic model's blank score. The prefix beam decoder had 3 beam widths: an *input beam* that limits the number of words emitted by the acoustic model at every frame, a *label beam* that limits the total number of label sequences (including blanks) hypothesized by the decoder for each unique word sequence, and a *word beam* that limits the total number of unique word sequences hypothesized by the decoder. The label beam is always set to 1. On a separate tuning dataset of 12k utterances, we set the input beam to 40 and the word beam to 100 and did a coarse parameter sweep to find the named entity weight, language model weight, and blank score divisor that minimized the word error rate. These tuning parameters were then applied to the 36k evaluation data, using an input beam of 200 and a word beam of 500, to produce the results in Table VII.

In Appendix F, we also discuss some (infrequent) cases where the timestamps allow the decoder to avoid insertion errors when the acoustic model hypothesizes words whose

TABLE VI
WORD SEGMENTATION ACCURACY ON NOISY TIMIT DATA (LOWER VALUES ARE BETTER). THE MEAN μ IN MILLISECONDS OF THE ABSOLUTE DIFFERENCE BETWEEN THE HYPOTHESIZED AND REFERENCE START TIMES (α) AND DURATIONS (β) ARE SHOWN FOR A HYBRID DNN-HMM ASR AND THE PROPOSED TIMESTAMPED SYSTEM USING 3 INTERNAL EMBEDDINGS.

System	Noise Type	SNR	μ_α (ms)	μ_β (ms)
Baseline DNN-HMM	Babble	20	20.5	32.2
		15	21.6	34.6
		10	24.0	40.9
		5	31.4	57.3
	Pink	20	18.5	31.4
		15	19.8	35.7
		10	23.6	46.3
		5	33.7	69.1
	White	20	19.4	34.6
		15	21.8	42.4
		10	27.2	56.6
		5	39.1	81.5
Proposed ($L = 3$)	Babble	20	23.3	37.8
		15	24.8	40.1
		10	28.3	47.9
		5	41.4	70.4
	Pink	20	21.5	36.4
		15	22.9	39.7
		10	26.9	49.5
		5	39.2	74.8
	White	20	22.2	38.0
		15	24.4	43.6
		10	30.3	57.0
		5	48.6	89.1

TABLE VII
RECOGNITION ERROR RATE (% WORD ERROR RATE AND NAMED ENTITY ERROR RATE) ON DIGITAL ASSISTANT DATA WITH DYNAMIC OUT-OF-VOCABULARY WORDS (CONTACT NAMES) FOR PREVIOUS METHOD [2] AND THE PROPOSED METHOD.

System	Internal Embeddings (L)	WER	NEER
Previous	1	9.5	9.7
	2	9.2	9.3
	3	8.9	9.0
Proposed	1	9.9	10.0
	2	9.8	10.0
	3	9.3	9.5

pronunciations partially overlap with each other. The timestamps also enable the decoder to generate heterogeneous word confusion networks (HWCNs) [7] for downstream processing. The prefix tree built during decoding directly becomes the ASR's word hypothesis lattice, which can then be converted to an HWCN. We show examples in Figure 9.

V. CONCLUSION AND FUTURE WORK

In this paper, a novel timestamped embedding-matching acoustic-to-word (A2W) model has been proposed, where a word-level embedding-matching CTC model is trained to directly estimate a start time and duration of every hypothesized word, allowing simpler, faster, and more efficient ASR. Experiments show that the word segmentation accuracy can approach that of a hybrid DNN-HMM ASR, with small impact on word error rate. We also contributed more rigorous analysis on multiple-hypothesis embedding-matching A2W to provide further insight into how the model can learn to produce multiple embeddings, and extended the analysis to show how the model can also produce word timestamps.

APPENDIX A
MAXIMIZATION OF THE SOFTMAX WHEN USING THE INNER PRODUCT

In the inner product case [1], [21], we have

$$s_{i,t} = \mathbf{f}_t^T \mathbf{g}_i. \tag{33}$$

For embedding-matching to work, we need $s_{i,t}$ to be maximized if and only if $\mathbf{f}_t = \mathbf{g}_i$. Such a property already always holds for the negative $\mathcal{L}_2$ distance in Equation (5), but for the inner product, we also require the normalization

$$\mathbf{f}_t^T \mathbf{f}_t = \mathbf{g}_i^T \mathbf{g}_i = c \quad \forall t, i, \tag{34}$$

for some constant $c > 0$, so that by the Cauchy-Schwarz Inequality we would have

$$(\mathbf{f}_t^T \mathbf{g}_i)^2 \leq (\mathbf{f}_t^T \mathbf{f}_t)(\mathbf{g}_i^T \mathbf{g}_i) = c^2, \tag{35}$$

where equality holds if and only if $\mathbf{f}_t = \mathbf{g}_i$.

If we were to take the approach of Sec. II-B, we have

$$\ln p_{t,i} = -\ln \left[1 + \sum_{j \neq i}^{n} \exp \left(\mathbf{f}_t^T \mathbf{g}_j - \mathbf{f}_t^T \mathbf{g}_i \right) \right]. \tag{36}$$

We can apply the fact that any two isotropic random vectors on a d-dimensional sphere tend to be orthogonal as $d \to \infty$ [32]. Setting $\mathbf{f}_t^T \mathbf{g}_j = 0$ and $\mathbf{f}_t^T \mathbf{g}_i = c$ (since $\mathbf{f}_t = \mathbf{g}_i$), we obtain:

$$\lim_{d \to \infty} \ln p_{t,i} = -\ln \left[1 + (n-1)e^{-c} \right]. \tag{37}$$

Hence, for a sufficiently large number of dimensions d, and a sufficiently large c, we will have $p_{t,i} \approx 1$ when $\mathbf{f}_t = \mathbf{g}_i$.

APPENDIX B
APPROXIMATE $\mathcal{L}_2$ DISTANCES IN HIGH DIMENSIONS

High dimensional vectors are known to exhibit the "distance concentration phenomenon" [32]–[34] where d-dimensional vectors with independent and identically-distributed elements tend to be concentrated on a $\mathcal{L}_p$ sphere of constant radius as d grows larger. In [34], this is expressed as

$$\forall \epsilon > 0, \lim_{d \to \infty} P \left\{ \left| \frac{||\mathbf{x}||_p}{E||\mathbf{x}||_p} - 1 \right| \geq \epsilon \right\} = 0 \tag{38}$$

where $\mathbf{x} \in \mathcal{R}^d$.

It is also known that isotropic vectors tend to be orthogonal in high dimensions [32], i.e.,

$$|\mathbf{x}^T \mathbf{y}| \approx \frac{1}{\sqrt{d}} \tag{39}$$

which approaches 0 for large d. Acoustic neighbor embeddings are trained by modeling the "induced" probabilities in the embedding space as isotropic Gaussian distributions [3], [35].

Hence, for the purpose of making our analysis easier in Sections II-C and III-C, we make the first-order approximation that the distance between any two distances will tend to be some constant D in a high-dimensional space when $\mathbf{x} \neq \mathbf{y}$:

$$||\mathbf{x} - \mathbf{y}||^2 = \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{y} + \mathbf{y}^T\mathbf{y} \approx D \qquad (40)$$

This approximation is also a reflection of [22] where it is shown that $\mathcal{L}_2$ distances become more and more similar so that there is little difference between "near" and "far" in high dimensions.

APPENDIX C
MAXIMIZATION OF POSTERIOR FOR HIGH DIMENSIONS IN SECTION II-C

Defining sets $\mathcal{G} = \{\mathbf{g}_1, \cdots, \mathbf{g}_n\}$ and $\mathcal{G}' = \mathcal{G} \cap \{\mathbf{g}_a, \mathbf{g}_b\}^C$, we can exhaustively identify – without using Condition (12) – 8 different cases for the value of $\ln p$ after accounting for symmetry

$(i) - 4D - 2\ln(ne^{-2D}), \mathbf{f}^{(1)} \notin \mathcal{G}, \mathbf{f}^{(2)} \notin \mathcal{G}$

$(ii) - 3D - 2\ln(e^{-D} + (n-1)e^{-2D}),$
$\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_a, \mathbf{f}^{(2)} \notin \mathcal{G}$

$(iii) - 3D - 2\ln(2e^{-D} + (n-2)e^{-2D}),$
$\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_a, \mathbf{f}^{(2)} \in \mathcal{G}'$

$(iv) - 2D - 2\ln(1 + (n-1)e^{-2D}), \mathbf{f}^{(1)} = \mathbf{f}^{(2)} = \mathbf{g}_a$

$(v) - 2D - 2\ln(2e^{-D} + (n-2)e^{-2D}),$
$\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_a, \mathbf{f}^{(2)} = \mathbf{g}_b$

$(vi) - 4D - 2\ln(2e^{-D} + (n-2)e^{-2D}),$
$\qquad\qquad\qquad \mathbf{f}^{(1)} \in \mathcal{G}', \mathbf{f}^{(2)} \in \mathcal{G}', \mathbf{f}^{(1)} \neq \mathbf{f}^{(2)}$

$(vii) - 4D - 2\ln(1 + (n-1)e^{-2D}), \mathbf{f}^{(1)} = \mathbf{f}^{(2)} \in \mathcal{G}'$

$(viii) - 4D - 2\ln(e^{-D} + (n-1)e^{-2D}),$
$\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} \in \mathcal{G}', \mathbf{f}^{(2)} \notin \mathcal{G}$

Since $D > 0$, it is obvious that $(iii) < (v)$ and $(vi) < (v)$. For (i), we have

$$
\begin{aligned}
(i) &= -2\ln(n) = -2\ln(2 + (n-2)) \qquad (41) \\
&< -2\ln(2 + (n-2)e^{-D}) = (v). \qquad (42)
\end{aligned}
$$

For (ii), we have

$$
\begin{aligned}
(ii) &= -2\ln(2e^{D/2} + (n-1)e^{-D/2}) \qquad (43) \\
&< -2\ln(2e^{D/2} + (n-1)e^{-D/2} - e^{-D/2}) \qquad (44) \\
&= (iii) < (v). \qquad (45)
\end{aligned}
$$

For (iv), we have

$$
\begin{aligned}
(iv) &= -2\ln(e^D + e^{-D} + (n-2)e^{-D}) \qquad (46) \\
&< -2\ln(2 + (n-2)e^{-D}) = (v). \qquad (47)
\end{aligned}
$$

where we used the fact that $e^{-D} + e^D > 2$.

It is also obvious that $(vii) < (iv)$, so $(vii) < (v)$.
It is also obvious that $(viii) < (ii)$, so $(viii) < (v)$.
Hence, (v) is the maximum among the 8 cases shown above.

APPENDIX D
MAXIMIZATION OF POSTERIOR FOR HIGH DIMENSIONS IN SECTION III-D

Assuming high dimensionality, we can approximate as follows:

$$
z_i = \begin{cases}
-d_i & \text{if } \mathbf{f}^{(1)} = \mathbf{f}^{(2)} = \mathbf{g}_i \\
-D - d_i(D+1) & \text{if } \mathbf{f}^{(1)} = \mathbf{g}_i, \mathbf{f}^{(2)} \neq \mathbf{g}_i \\
-2D - d_i(2D+1) & \text{if } \mathbf{f}^{(1)} \neq \mathbf{g}_i, \mathbf{f}^{(2)} \neq \mathbf{g}_i.
\end{cases} \qquad (48)
$$

By Condition (27), we have $d_a = d_b = 0$, so $z_a = z_b = -D$, which means $q_a < 1/2$ and $q_b < 1/2$. Hence, $\mathbf{u}^{(1)} = \mathbf{v}_a$ and $\mathbf{u}^{(2)} = \mathbf{v}_b$ maximize q.

Under this constraint, we can identify 8 different cases for $\ln(q) = \ln(q_a) + \ln(q_b)$ after accounting for symmetry:

$(i) - 2D - 2\ln\left(2e^{-D} + e^D \sum_{j \neq a,b} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} \notin \mathcal{G}, \mathbf{f}^{(2)} \notin \mathcal{G}$

$(ii) - 2D - 2\ln\left(e^{-D/2} + e^{-3D/2} + e^{D/2} \sum_{j \neq a,b} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_a, \mathbf{f}^{(2)} \notin \mathcal{G}$

$(iii) - 2D - 2\ln\left(e^{-D/2} + e^{-3D/2} + e^{D/2}m_c + e^{D/2}\sum_{j \neq a,b,c} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_a, \mathbf{f}^{(2)} = \mathbf{g}_c$

$(iv) - 2D - 2\ln\left(1 + e^{-2D} + \sum_{j \neq a,b} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{f}^{(2)} = \mathbf{g}_a$

$(v) - 2D - 2\ln\left(2e^{-D} + \sum_{j \neq a,b} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_a, \mathbf{f}^{(2)} = \mathbf{g}_b$

$(vi) - 2D - 2\ln\left(2e^{-D} + e^D m_c + e^D m_d + e^D \sum_{j \neq a,b,c,d} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_c, \mathbf{f}^{(2)} = \mathbf{g}_d$

$(vii) - 2D - 2\ln\left(2e^{-D} + e^D e^{-d_c} + e^D \sum_{j \neq a,b,c} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{f}^{(2)} = \mathbf{g}_c$

$(viii) - 2D - 2\ln\left(2e^{-D} + e^D m_c + e^D \sum_{j \neq a,b,c} h_j\right),$
$\qquad\qquad\qquad\qquad\qquad \mathbf{f}^{(1)} = \mathbf{g}_c, \mathbf{f}^{(2)} \notin \mathcal{G},$

where w_a, w_b, w_c, w_d are all distinct words in the training vocabulary, with corresponding distinct embeddings $\mathbf{g}_a, \mathbf{g}_b, \mathbf{g}_c, \mathbf{g}_d \in \mathcal{G}$, and we have defined

$$
\begin{aligned}
h_i &\triangleq e^{-2D - d_i(2D+1)} \qquad (49) \\
m_i &\triangleq e^{-D - d_i(D+1)}, \qquad (50)
\end{aligned}
$$

for $i \geq 1$.

Since $D > 0$, it is obvious that $(i) < (v)$.

We know $e^{D/2} + e^{-D/2} > 2$, which implies $e^{-D/2} + e^{-3D/2} > 2e^{-D}$, and therefore

$$(ii) < -2D - 2\ln\left(2e^{-D} + e^{D/2}\sum_{j\neq a,b} h_j\right) < (v). \quad (51)$$

Since $m_c > h_c$,

$$(iii) < -2D - 2\ln\left(e^{-D/2} + e^{-3D/2} + e^{D/2}\sum_{j\neq a,b} h_j\right)$$
$$= (ii) < (v) \quad (52)$$

We know $e^D + e^{-D} > 2$, which implies $1 + e^{-2D} > 2e^{-D}$, so

$$(iv) < -2D - 2\ln\left(2e^{-D} + \sum_{j\neq a,b} h_j\right) = (v). \quad (53)$$

Since $m_c > h_c$ and $m_d > h_d$, we have

$$(vi) < -2D - 2\ln\left(2e^{-D} + e^{D}\sum_{j\neq a,b} h_j\right) < (v). \quad (54)$$

Since $h_c < e^{-d_c}$, we have

$$(vii) < -2D - 2\ln\left(2e^{-D} + e^{D}h_c + e^{D}\sum_{j\neq a,b} h_j\right) = (i) < (v). \quad (55)$$

Since $m_c > h_c$, we have

$$(viii) < -2D - 2\ln\left(2e^{-D} + e^{D}\sum_{j\neq a,b} h_j\right) = (i) < (v). \quad (56)$$

Hence, (v) is the maximum of $\ln(q)$.

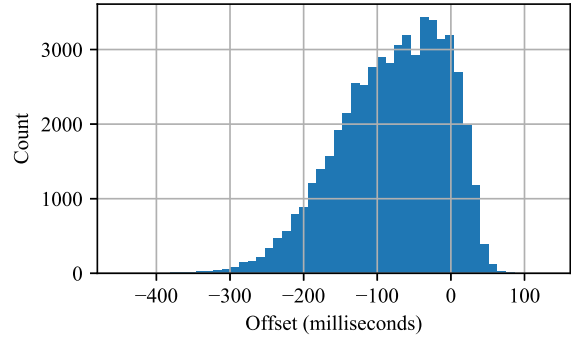## APPENDIX E
WORD OFFSET AND DURATION LIMITS

In Figure 7, we show histograms of the word offsets and durations produced by the proposed system ($L = 3$) system for the TIMIT data. The minimum observed word offset was -451ms, and the maximum was 133ms. The minimum observed word duration was 17ms, and the maximum was 1684ms. All the offsets and durations are well within the -1s $\sim +1$s and 2s limits imposed by the hyperbolic tangent and sigmoid operators in the system, implying that the limits do not have an actual effect during evaluation.

When we removed the sigmoid and hyperbolic tangent operators and retrained the models, however, we noticed a small degradation in the word segmentation accuracy and speech recognition accuracy, implying that the operators may still help train the system more accurately.
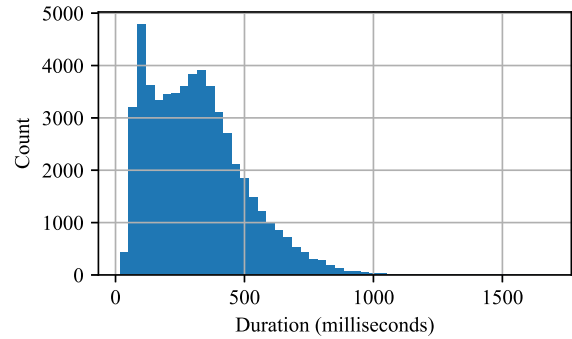
APPENDIX F
MITIGATION OF INSERTION ERRORS USING TIMESTAMPS

Fig. 8a shows a simplified set of word posterior scores produced by the embedding-matching word CTC model for the speech utterance "directions to Fursten Park." Each bar "I" represents a spike in the score for the corresponding word and time. It is possible for the decoder to produce "directions to Fursten and Park" as the final recognition result, even though "and" is clearly the last part of "Fursten" erroneously getting recognized a second time.

Fig. 7. Histograms of word offsets and durations produced by the proposed system for TIMIT data.

In a hybrid DNN-HMM ASR system, this sort of insertion typically does not occur because every speech frame is explicitly assigned a position in a state sequence, and a speech frame that is already consumed by "Fursten" cannot be reused by any succeeding word. In our embedding-matching word CTC system, however, such an insertion can occur because there is no built-in systemic notion of word-frame ownership. This is exacerbated when we aggressively induce the system to hypothesize over multiple segment lengths at every point in time [2] as described in Sec. II-C.

Accurate word timestamps allow us to easily address this problem. As shown in Figure 8b, the beam decoder can see that "Fursten" significantly overlaps with "and" and therefore refuse to follow such a path. Hence, in the decoder for the proposed timestamped system, we add a condition that when a preceding word's end time is t_e and a candidate succeeding word's start time is t_s, the prefix decoder refuses to join the two words if $t_e > t_s + \gamma$ where $\gamma \geq 0$ is a constant tolerance. For the results in Table VII, we used $\gamma = 200$ms.

REFERENCES

[1] S. Settle, K. Audhkhasi, K. Livescu, and M. Picheny, "Acoustically grounded word embeddings for improved acoustics-to-word speech recognition," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 5641–5645.

[2] H. Yen and W. Jeon, "Improvements to embedding-matching acoustic-to-word ASR using multiple-hypothesis pronunciation-based embeddings," in *2023 IEEE International Conference on Acoustics,*
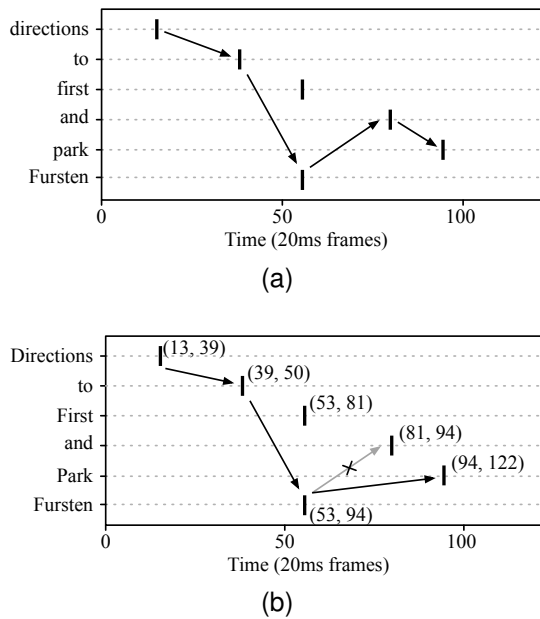
(a)

(b)

Fig. 8. Example word posterior scores on (*time*×*word*) planes. The arrows represent example paths hypothesized by the decoder for the audio "directions to Fursten Park." In (a), it is possible for the decoder to choose an erroneous path representing "directions to Fursten and Park." When timestamps are present as in (b), however, the decoder can avoid the erroneous path and choose the correct one.
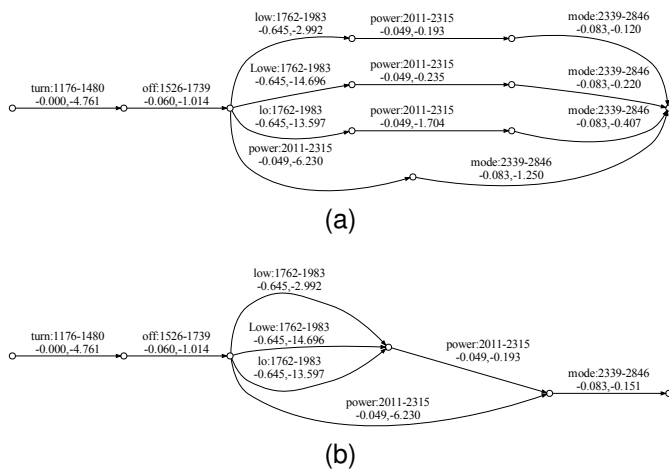

(a)

(b)

Fig. 9. (a) Word hypothesis lattice outputted by the decoder for the utterance, "*turn off low power mode,*" and (b) the corresponding heterogeneous word confusion network [7]. Each arc is labeled as "`<word>`:`<start time in ms>`-`<end time in ms> <acoustic score>`, `<LM score>`".

Speech and Signal Processing (ICASSP), 2023. [Online]. Available: https://arxiv.org/abs/2210.16726

[3] W. Jeon, "Acoustic neighbor embeddings," 2020. [Online]. Available: https://arxiv.org/abs/2007.10329

[4] W. He, W. Wang, and K. Livescu, "Multi-view recurrent neural acoustic word embeddings," in *International Conference on Learning Representations (ICLR)*, 2017.

[5] V. Garcia, E. Debreuve, F. Nielsen, and M. Barlaud, "K-nearest neighbor search: Fast gpu-based implementations and application to high-dimensional feature matching," in *2010 IEEE International Conference on Image Processing*, 2010, pp. 3757–3760.

[6] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus among words: lattice-based word error minimisation," *Computer Speech and Language*, pp. 373–400, 2000.

[7] W. Jeon, M. Jordan, and M. Krishnamoorthy, "On modeling ASR word confidence," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6324–

6328.

[8] G.-X. Shi, W.-Q. Zhang, G.-B. Wang, J. Zhao, S.-Z. Chai, and Z.-Y. Zhao, "Timestamp-aligning and keyword-biasing end-to-end asr front-end for a kws system," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2021, no. 1, p. 27, 2021. [Online]. Available: https://doi.org/10.1186/s13636-021-00212-9

[9] K. Larson and D. Mowatt, "Speech error correction: The story of the alternates list," *International Journal of Speech Technology*, vol. 6, no. 2, pp. 183–194, 2003. [Online]. Available: https://doi.org/10.1023/A:1022342732234

[10] G. Tur, A. Deoras, and D. Hakkani-Tür, "Semantic parsing using word confusion networks with conditional random fields," in *Proc. Interspeech 2013*, 2013, pp. 2579–2583.

[11] R. W. M. Ng, K. Shah, W. Aziz, L. Specia, and T. Hain, "Quality estimation for asr k-best list rescoring in spoken language translation," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 5226–5230.

[12] J. Li, "Recent advances in end-to-end automatic speech recognition," *APSIPA Transactions on Signal and Information Processing*, vol. 11, no. 1, pp. –, 2022. [Online]. Available: http://dx.doi.org/10.1561/116.00000050

[13] R. Yang, G. Cheng, P. Zhang, and Y. Yan, "An e2e-asr-based iteratively-trained timestamp estimator," *IEEE Signal Processing Letters*, vol. 29, pp. 1654–1658, 2022.

[14] A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012.

[15] T. N. Sainath, R. Pang, D. Rybach, B. García, and T. Strohman, "Emitting Word Timings with End-to-End Models," in *Proc. Interspeech 2020*, 2020, pp. 3615–3619. [Online]. Available: http://dx.doi.org/10.21437/Interspeech.2020-1059

[16] R. Yang, G. Cheng, H. Miao, T. Li, P. Zhang, and Y. Yan, "Keyword search using attention-based end-to-end asr and frame-synchronous phoneme alignments," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3202–3215, 2021.

[17] Y. Shinohara and S. Watanabe, "Minimum latency training of sequence transducers for streaming end-to-end speech recognition," in *Proc. Interspeech 2022*, 2022, pp. 2098–2102.

[18] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning*, ser. ICML '06. New York, NY, USA: Association for Computing Machinery, 2006, p. 369–376. [Online]. Available: https://doi.org/10.1145/1143844.1143891

[19] H. Soltau, H. Liao, and H. Sak, "Neural Speech Recognizer: Acoustic-to-Word LSTM Model for Large Vocabulary Speech Recognition," in *Proc. Interspeech 2017*, 2017, pp. 3707–3711.

[20] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[21] R. Collobert, A. Hannun, and G. Synnaeve, "Word-level speech recognition with a letter to word encoder," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. JMLR.org, 2020.

[22] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is "nearest neighbor" meaningful?" in *Database Theory — ICDT'99*, C. Beeri and P. Buneman, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 217–235.

[23] M. Mohri, F. Pereira, and M. Riley, *Speech Recognition with Weighted Finite-State Transducers*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 559–584. [Online]. Available: https://doi.org/10.1007/978-3-540-49127-9_28

[24] Z. Huang, T. Ng, L. Liu, H. Mason, X. Zhuang, and D. Liu, "Snd-cnn: Self-normalizing deep cnns with scaled exponential linear units for speech recognition," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 6854–6858.

[25] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech 2020*, 2020, pp. 5036–5040.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[27] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, N. Dahlgren, and V. Zue, "Timit acoustic-phonetic continuous speech corpus," *Linguistic Data Consortium*, 11 1992.

[28] V. W. Zue and S. Seneff, "Transcription and alignment of the timit database," in *Recent Research Towards Advanced Man-Machine Interface Through Spoken Language*, H. Fujisaki, Ed. Amsterdam: Elsevier Science B.V., 1996, pp. 515–525. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780444816078500888

[29] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," 2014. [Online]. Available: https://arxiv.org/abs/1408.2873

[30] A. Abdulaziz and V. Kepuska, "Noisy timit speech ldc2017s04," *Linguistic Data Consortium*, 2017.

[31] H. Sak, A. Senior, K. Rao, O. İrsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 4280–4284.

[32] R. Vershynin, *High-Dimensional Probability: An Introduction with Applications in Data Science*, ser. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2018. [Online]. Available: https://books.google.com/books?id=NDdqDwAAQBAJ

[33] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional spaces," in *Proceedings of the 8th International Conference on Database Theory*, ser. ICDT '01. Berlin, Heidelberg: Springer-Verlag, 2001, p. 420–434.

[34] G. Biau and D. M. Mason, *High-Dimensional $$p$$-Norms*. Cham: Springer International Publishing, 2015, pp. 21–40. [Online]. Available: https://doi.org/10.1007/978-3-319-12442-1_3

[35] G. E. Hinton and S. Roweis, "Stochastic neighbor embedding," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds., vol. 15. MIT Press, 2002. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf