LEARNING FROM FLAWED DATA: WEAKLY SUPERVISED AUTOMATIC SPEECH RECOGNITION

Dongji Gao[1], Hainan Xu[3], Desh Raj[1], Leibny Paola Garcia Perera[1,2], Daniel Povey[4], Sanjeev Khudanpur[1,2]

[1]CLSP & [2]HLTCOE, Johns Hopkins University, USA
[3]NVIDIA Corp., USA
[4]Xiaomi Corp., China

ABSTRACT

Training automatic speech recognition (ASR) systems requires large amounts of well-curated paired data. However, human annotators usually perform "non-verbatim" transcription, which can result in poorly trained models. In this paper, we propose *Omni-temporal Classification* (OTC), a novel training criterion that explicitly incorporates label uncertainties originating from such weak supervision. This allows the model to effectively learn speech-text alignments while accommodating errors present in the training transcripts. OTC extends the conventional CTC objective for imperfect transcripts by leveraging weighted finite state transducers. Through experiments conducted on the LibriSpeech and LibriVox datasets, we demonstrate that training ASR models with OTC avoids performance degradation even with transcripts containing up to 70% errors, a scenario where CTC models fail completely. Our implementation is available at `https://github.com/k2-fsa/icefall`.

Index Terms— weakly supervised learning, automatic speech recognition, weighted finite-state transducer, omni-temporal classification.

1. INTRODUCTION

The development of automatic speech recognition (ASR) systems based on neural networks typically requires acquiring vast amounts of accurately transcribed speech [1, 2, 3]. However, with the exception of a few very carefully human-annotated or read speech corpora — the creation of which is highly time-consuming and labor-intensive — most datasets contain errors that might degrade the ASR performance of trained models, especially when the amount of data is relatively limited. This poses a major challenge to the development of accurate ASR systems outside large industrial groups.

Existing methods to address this issue may be categorized into two perspectives: *data* and *model*. Data cleaning is a broadly used technique wherein errors in datasets are detected and discarded, usually through multiple stages of model training and alignment. However, data cleaning is usually done at the utterance level [4, 5, 6, 7, 8]; i.e., utterances in long recordings that do not align perfectly with the annotations are often discarded. Such a process can potentially filter out a large number of imperfect utterances that may still contain a lot of useful information for the model. Table 1 presents the summary statistics of some popular ASR datasets that were created using such a data-cleaning process from sources containing non-verbatim transcripts. Unsupervised training [9, 10, 11, 12, 13], on the other hand, leverages non-parallel speech and text, where models are designed to

Table 1: Data filtering statistics for popular ASR corpora. Raw and filtered refer to the amount (in hours) of source audio and the final prepared data, respectively.

Dataset	Raw (h)	Filtered (h)	Ratio (%)
TED-LIUM [18]	216	118	54.6
TED-LIUM 2 [19]	351	207	58.9
TED-LIUM 3 [20]	540	452	83.7
Europarl-ST [21]	816	255	31.3
HK-LegiCoST [22]	1,400	609	43.5
Europarl-ASR [23]	4,900	1,300	26.5

learn high-level features from audio. However, training such models is often challenging due to the lack of supervision, and the improvements achieved by unsupervised training tend to be limited.

From a model perspective, *weakly supervised* learning tackles data errors by designing models that can detect and learn from useful portions of the data while avoiding contamination from errors. This class of methods includes the recently proposed star temporal classification (STC) [14], wild-card CTC (W-CTC) [15], alternative pseudo-labeling (APL) [16], and bypass temporal classification (BTC) [17]. These methods enable the use of audio data combined with non-verbatim transcripts, such as videos with closed captions, which are widely available and accessible on the internet. Our proposed OTC training objective falls into this model perspective, and we will describe it in Section 3.

Data and model based methods may also be used in combination. For instance, Wav2Vec-U [10] used self-supervised training on unlabeled speech. Meta's MMS [24] used multiple rounds of data alignment using a model trained with the STC objective. The recently released Whisper [25] leveraged 680k hours of weakly filtered web data to achieve impressive ASR performance.

In this paper, we focus on the model perspective of weakly supervised ASR. We argue that in order to train models on reasonably-sized corpora with imperfect transcripts (e.g. non-verbatim annotations), the model needs to explicitly account for errors (i.e., substitution, insertion, and deletion errors) that exist within the transcripts during the training process.[1] Our observation is inspired by numerous recent studies such as STC, W-CTC, APL, and BTC, as mentioned earlier. These existing methods focus on specific categories of transcript errors: STC and W-CTC tackle deletion errors, APL addresses substitution errors, and BTC handles both substitu-

[1]Such label errors may be relatively less hazardous in the very-large-data regime (such as Whisper), but most academic (and industrial) systems trained on medium scale data, often for copyright/privacy reasons. At this scale, erroneous transcripts may be devastating (§ 4).

tion and insertion errors. However, the specific type of error present in real-world data sources cannot be predetermined and may include all of these. Consequently, these are only partial solutions, and it is imperative to develop a method capable of effectively handling and accommodating all types of errors.

To this end, we propose a criterion called Omni-temporal Classification (OTC), an extension of Connectionist Temporal Classification (CTC) [26]. OTC integrates the advancements made in previous works such as STC and BTC [14, 17] and supersedes them by addressing all types of errors encountered during training (hence the name "omni"). We achieve this by directly encoding label uncertainties, encompassing substitution, insertion, and deletion errors, into the training graph by leveraging the weighted finite-state transducer (WFST) framework. Our implementation is built on the k2 toolkit[2], which provides GPU-based implementations of all WFST operations to enable faster training. Through extensive experiments on the LibriSpeech and LibriVox corpora, we demonstrate that OTC-based training maintains the performance of the ASR model despite erroneous transcripts. Notably, on the LibriSpeech dataset, when the proportion of synthetic errors exceeds 50%, OTC-based models still obtain 10-30% word error rates (WERs), whereas CTC training fails completely. Additionally, we show that our method is robust against under-processed data of lower quality, which requires minimal human intervention.

2. PRELIMINARIES

2.1. ASR with CTC

Given an acoustic feature sequence $\mathbf{x} = [x_1, \ldots, x_T]$ of length T, where $x_t \in \mathbb{R}^d$ corresponds to the d-dimensional, real-valued acoustic representation at frame t, an ASR system predicts the most "probable" transcript $\hat{\mathbf{y}}$, i.e.,

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}). \tag{1}$$

Here, $\mathbf{y} = [y_1, \ldots, y_U]$ has length of U, where $y_u \in \mathcal{V}$ represents a discrete unit from a finite vocabulary $\mathcal{V}$. CTC models the posterior probability $P(\mathbf{y}|\mathbf{x})$ by marginalizing over all possible frame-level alignments $\boldsymbol{\pi} = [\pi_1, \ldots, \pi_T]$ between $\mathbf{x}$ and $\mathbf{y}$. Each $\pi_t \in \mathcal{V} \cup \oslash$, where $\oslash$ is a special symbol used in CTC to align extra acoustics, given that $T \geq U$. CTC further assumes that the output units in the frame-level alignment $\boldsymbol{\pi}$ are *conditionally independent* given the acoustic features $\mathbf{x}$, which yields

$$P(\mathbf{y}|\mathbf{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\mathbf{y})} P(\mathbf{y}, \boldsymbol{\pi}|\mathbf{x}) = \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\mathbf{y})} P(\boldsymbol{\pi}|\mathbf{x}) \tag{2}$$

$$= \sum_{\boldsymbol{\pi} \in \mathcal{B}^{-1}(\mathbf{y})} \prod_{t=1}^{T} P(\pi_t|\mathbf{x}), \tag{3}$$

where $\mathcal{B}$ is a deterministic mapping from the alignment sequence $\boldsymbol{\pi}$ to transcript $\mathbf{y}$ by removing $\oslash$ and merging adjacent repetitions.

2.2. CTC within the WFST framework

A weighted finite-state transducer (WFST) is a directed graph that represents a function that maps sequences of input symbols to output symbols. Each edge (or arc), denoted as e, is assigned a weight, $\omega(e)$. Two WFSTs H_1 and H_2 can be composed to cascade mapping

operations, denoted as $H_1 \circ H_2$. In the resulting WFST, the weight of each edge is the $\otimes$-product of the corresponding edge weights from the source WFSTs. When training ASR models, we use WFSTs on the log semiring, such that edge weights may be interpreted as log probabilities; consequently, $\otimes$ is simply the addition operation.[3]

Previous studies [14, 17, 28, 29, 30] have shown that the computation of $P(\mathbf{y}|\mathbf{x})$ can be efficiently implemented within the WFST framework. $\mathcal{B}^{-1}(\mathbf{y})$, which encompasses all possible alignments for a given $\mathbf{y}$, can be represented by a WFST denoted as $S(\mathbf{y})$. This WFST, often referred to as the *training graph*, is illustrated in Figure 1(a). Each path within $S(\mathbf{y})$ corresponds to a distinct alignment $\boldsymbol{\pi}$. By composing $S(\mathbf{y})$ with the emission WFST $E(\mathbf{x})$ that represents $\log P(\boldsymbol{\pi}|\mathbf{x})$ (Fig. 4 (a)), $\log P(\mathbf{y}|\mathbf{x})$ can be expressed as

$$\log P(\mathbf{y}|\mathbf{x}) = \sum_{\boldsymbol{\pi} \in S(\mathbf{y})^{-1}} \underbrace{P(\boldsymbol{\pi}|\mathbf{x})}_{E(\mathbf{x})} = \text{Weight}(E(\mathbf{x}) \circ S(\mathbf{y})). \tag{4}$$

The training graph $S(\mathbf{y})$ can be further factored as $S(\mathbf{y}) = T \circ L \circ G(\mathbf{y})$, where T removes the special symbol $\oslash$ and merges repeated output units, L maps unit sequences (e.g., BPE or phone) to words, and $G(\mathbf{y})$ is a linear WFST representing the transcript $\mathbf{y}$ (Fig. 2(c)). This decomposition yields

$$\log P(\mathbf{y}|\mathbf{x}) = \text{Weight}(E(\mathbf{x}) \circ T \circ L \circ G(\mathbf{y})). \tag{5}$$

In the following sections, we will describe our modifications to $E(\mathbf{x})$ and $G(\mathbf{y})$ that constitute the OTC loss.

3. OMNI-TEMPORAL CLASSIFICATION

3.1. WFST topology

As mentioned earlier, previous CTC variants such as STC [14], W-CTC [15], and BTC [17] partially solved the problem of training with imperfect transcripts. The key insight was to extend the training graph by introducing a special token $\star$ that can handle mismatched acoustics. OTC further enhances the flexibility of the training graph to accommodate all potential errors present in the transcript $\mathbf{y}$, including substitution, insertion, and deletion errors.

To achieve this, we modify $G(\mathbf{y})$ by adding self-loop arcs (Fig. 2(a)) into each state and bypass arcs (Fig. 2(b)) into each arc. Each added arc is associated with the special token $\star$ and penalty λ adopted from [14, 17]. The modified WFST $G_{\text{otc}}(\mathbf{y})$ is shown in Fig. 2(d). This $\star$ symbol acts as a token of uncertainty, offering a preferable choice over aligning with potentially incorrect tokens. Fig. 3 illustrates an example of how the $\star$ token functions.

Bypass arcs were proposed in [17] to address substitution and insertion errors. When encountering an incorrectly substituted word y_u', the bypass arc provides an alternative path parallel to the erroneous word. This allows the model to associate the acoustics with the $\star$ token instead of the erroneous token, effectively reducing the probability $P(y_u'|\boldsymbol{x})$. By doing so, the model avoids learning wrong relationships through back-propagation during training. This strategy also applies to insertion errors, where the transcript $\mathbf{y}$ contains additional words not present in the intended reference. In such cases, the model allocates a minimal amount of acoustics, usually a single frame, from neighboring tokens to the $\star$ token.

Self-loop arcs are proposed to handle deletion errors. These errors occur when there are acoustics present in the input data $\mathbf{x}$ which

(a) CTC training graph.

(b) OTC training graph. The self-loop arcs and bypass arcs are highlighted in green and blue, respectively.
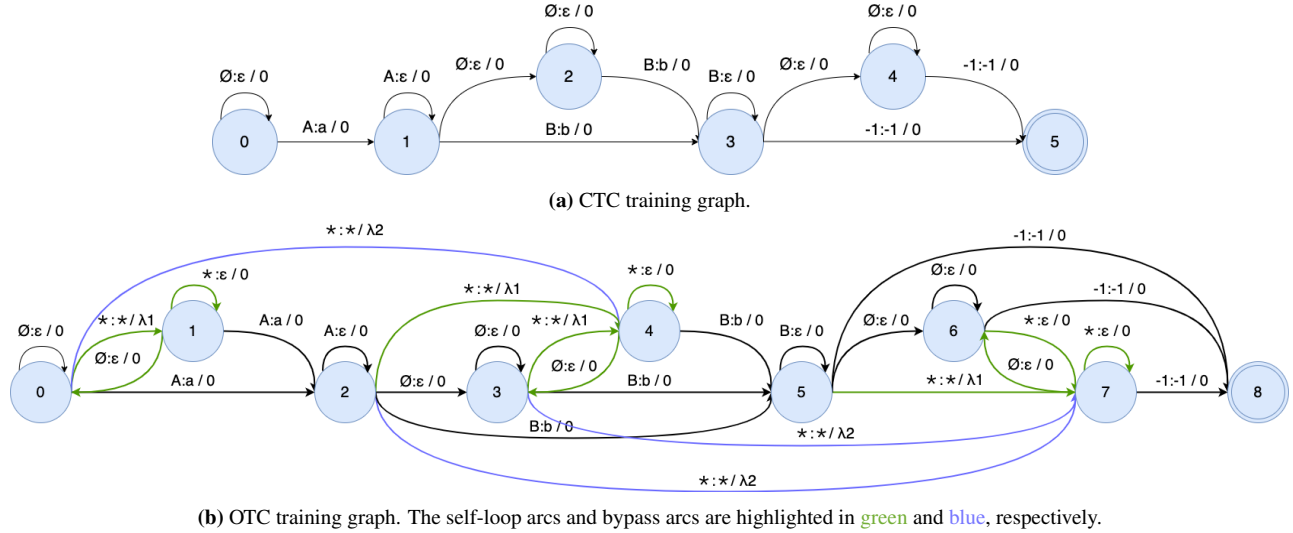
Fig. 1: Training graphs of the transcript "a b" given the lexicon {a:A, b:B}. A state labeled '0' is the starting state. The state with the double circle (state 5) is the final state. States are connected by directed arcs. Each arc has an input and output symbol (separated by a colon) and a weight (after a slash). The arc labeled '-1' is a special arc pointing to the final state.
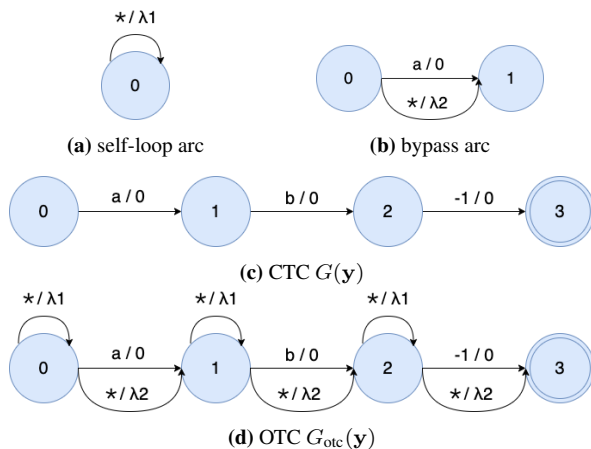

(a) self-loop arc   **(b)** bypass arc

(c) CTC $G(\mathbf{y})$

(d) OTC $G_{\mathrm{otc}}(\mathbf{y})$

Fig. 2: CTC and OTC WFST representations of $\mathbf{y}$: $G(\mathbf{y})$ and $G_{\mathrm{otc}}(\mathbf{y})$. The self-loop arc and bypass arc are associated with penalty λ_1 and λ_2, respectively. An arc with a single symbol indicates identical input and output.

do not have a corresponding token y_u in the transcript $\mathbf{y}$. To handle deletions, the self-loop arc can insert an arbitrary number of $\star$ tokens. These $\star$ tokens can then be aligned with the "orphan" acoustics in the input x, filling in the gaps left by the deleted tokens.

After composing the modified WFST $G_{\mathrm{otc}}(\mathbf{y})$ with L and T, the OTC training graph is shown in Fig. 1(b). We incorporate the penalty strategy introduced in [17] and apply different configurations for the self-loop arc and bypass arc. The penalties are set as

$$\lambda_{1_i} = \beta_1 * \tau_1^i, \quad \lambda_{2_i} = \beta_2 * \tau_2^i \qquad (6)$$

for the i-th training epoch. β is the initial penalty that encourages the model to rely more on the given transcript at the start of training. It decays exponentially by a factor of $\tau \in (0, 1)$, gradually encouraging the model to align speech with $\star$ when getting confused.
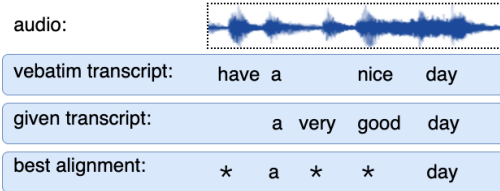

Fig. 3: Example demonstrates how OTC tackles errors in the given transcript "a very good day". The errors include one deletion (missing "have"), one insertion ("very"), and one substitution (replacing "nice" with "good") error compared to the verbatim transcript. During training, OTC automatically aligns the audio with the pattern " $\star$ a $\star$ $\star$ day", by assigning a higher probability to this alignment.

3.2. Modeling $\star$ token

In [17], the symbol $\star$ represented an individual token with its own distribution. This approach works for substitution and deletion errors, as the bypass arc offers a non-$\star$ token as the reference. In effect, it guides the model to treat $\star$ as a "garbage" token capable of matching any non-blank tokens by distinguishing between $\star$ and reference token given the acoustic context.

However, in our preliminary experiments, we found that such modeling of $\star$ is not effective for insertion (self-loop) case, as the model may confound the roles of $\star$ and $\oslash$ symbols as there is no token to be compared within the self-loop arc. To remedy this, we adopt the strategy used in STC [14], where $\star$ is represented as the average probability of all non-blank tokens. In our WFST-based implementation, this is done by modifying the emission WFST $E(\mathbf{x})$, as shown in Fig. 4 (b).

4. DATA PREPARATION

We use the publicly available LibriSpeech and LibriVox corpora for demonstrating OTC.

LibriSpeech [31] (LS) train set consists of 960 hours of read

(a) CTC emission WFST $E(\mathbf{x})$
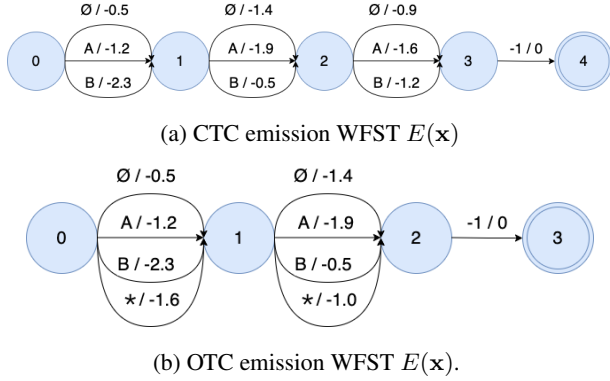

(b) OTC emission WFST $E(\mathbf{x})$.

Fig. 4: OTC and CTC emission WFST $E(\mathbf{x})$ over 2 frames. For OTC, the weight associated with each arc is its log probability. The weight of $\star$ is the log average probability of 'A' and 'B': $\log \frac{e^{-1.2}+e^{-2.3}}{2} = -1.6$ and $\log \frac{e^{-1.9}+e^{-0.5}}{2} = -1.0$ for 2 frames.
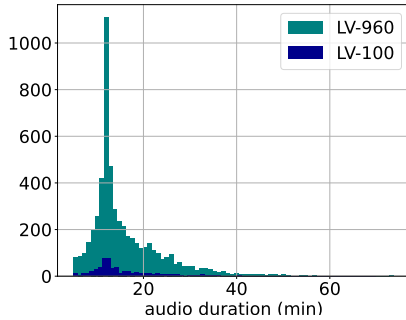

Fig. 5: Distribution of chapter audio length of LibriVox 100 hours (green) and LibriVox 960 hours (blue).

speech divided into three subsets: 100 + 360 (clean) and 500 (other). Additionally, it includes two development subsets and two test subsets, each containing 5 hours of speech.

LibriVox[4] (LV) is a set of audiobooks, from which the LS dataset is derived. Each audiobook consists of multiple chapters, and the audio is available at the chapter level without any segmentation. For our investigation, we retrieved the exact portions of the audiobooks that were used to create LS. We refer to the retrieved data as **LV-100** and **LV-960**, respectively, which correspond to the 100h subset and the full LS, respectively. The duration of each chapter varies between 5 minutes and 70 minutes as shown in Fig. 5. It should be noted that the accompanying text derived from the books is not perfect, as it contains errors from human reading.[5]

4.1. LibriSpeech processing

As described in [31], significant human efforts were dedicated to creating the LS dataset from the LV source.

1. **Preprocessing**: The audio of each chapter was divided into segments, with a maximum duration of 30 minutes. These segments were decoded using an external ASR model trained on the Vox-Forge dataset, using the Kaldi toolkit [32].

---

[4]https://librivox.org
[5]Common errors include missing chapter titles and reader information in the text, repetitions in the audio, and reading incorrect words.
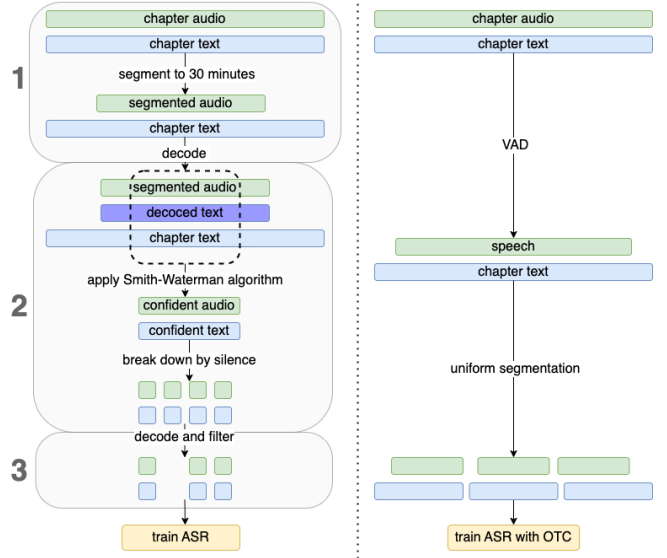

Fig. 6: Comparison between data processing of LibriSpeech (left) and LibriVox (right). Audio, reference text, and decoded text are represented as blocks of color: green, light blue, and dark blue. The 3 big grey blocks on the left represent the following stages: preprocessing, first alignment, and second alignment.

2. **First alignment (coarse filtering)**: The Smith-Waterman algorithm [33] was applied to compare the decoded text with the original chapter text. This process aimed to identify the region that exhibited the closest text match to retrieve the corresponding speech fragment. The retrieved fragments were subsequently segmented into smaller chunks, ensuring that each chunk was shorter than 35 seconds and separated by periods of silence.

3. **Second alignment (fine filtering)**: A specialized decoding graph was constructed as a refined filter to only allow arbitrary phone insertions between words in the transcript, or replacement of words in the transcript. This graph was applied to the smaller audio chunks obtained from the previous stage, in order to filter out chunks whose decoded text significantly differed from the expected transcript, thereby providing a more accurate selection of suitable speech fragments.

4.2. LibriVox processing: weak supervision

To demonstrate the efficiency of training an ASR system using OTC on weakly supervised speech data, we use the LibriVox dataset. Our goal is to highlight the minimal effort required for preparing training data in this approach. For a visual representation of the comparison between data preparation for LS and LV, please refer to Fig. 6.

As described in Section 4, we extract the chapter-level audio and text from the original audiobooks corresponding to the 100h and 960h sets from LS, and these are referred to as LV-100 and LV-960, respectively. By employing OTC, the data processing workflow can be simplified into two steps.

1. First, a voice activity detection (VAD) technique is applied to remove non-speech audio.

2. Following that, both the audio and text data are uniformly segmented, with each speech segment restricted to a duration of 60 seconds. In Section 7.2, we present an analysis of different choices for this uniform duration.
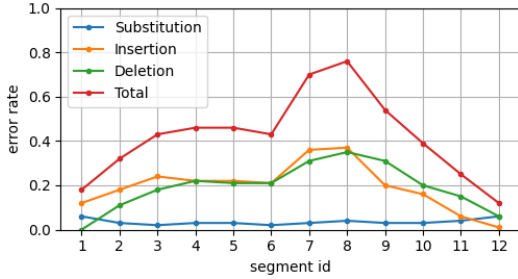
Fig. 7: Text error rates of 12 segments within the chapter (id 103-1240), span from 0.2 to 0.8. The text error rates are dominated by shift errors (insertion and deletion), particularly for the segments located in the middle of the chapter.

The processed LV corpus contains errors from two distinct sources. The first one, which we label as a "shift error," is a product of uniform segmentation, which can cause a misalignment between the speech and the corresponding text segments. Such a mismatch may occur if the segmented portions do not exactly match the natural boundaries of speech. The second error source, which we refer to as an "internal error," originates from the potential imperfections within the original chapter text itself. Such errors are inherent due to human error in the reading process.

In order to gain a clearer understanding of the text mismatch, we manually aligned the text for each segment within a specific chapter (id 103-1240), which we then used as the ground truth. We compared this ground truth text to the text acquired via uniform segmentation and calculated the word error rates. This was done to numerically express the extent of misalignment, as shown in Fig. 7.

5. EXPERIMENTAL SETUP

5.1. Synthetic data generation

For a controlled experiment, we first generated synthetic data based on LS `train-clean-100`. This allows us to test whether OTC performs as well as tuning hyper-parameters. We introduced substitution, insertion, and deletion, errors by randomly replacing, inserting, and removing each token in the transcript with probabilities $p_{sub}, p_{ins}, p_{del}$, respectively. We trained models on transcripts containing different degrees of each of these errors, by setting the corresponding probabilities to one of $\{0.1, 0.3, 0.5, 0.7\}$. Additionally, we train on transcripts containing all the errors combined. For this, we consider settings where $p_{sub} + p_{ins} + p_{del}$ falls within the same set of values, such that $p_{sub} = p_{ins} = p_{del}$.

5.2. Model

We used the wav2vec 2.0 (base) model [34] to extract 768-dimensional features with a stride of 20 ms, from audio recordings originally sampled at 16 kHz. For this, we used the extractor available in the S3PRL toolkit [35].[6] These features were fed to an acoustic model based on a 12-layer conformer [36] network. Each conformer block consists of two feed-forward layers, with half-step residual connections. These layers encapsulate multi-headed self-attention and convolution modules, followed by layer normalization. The decoder is simply a linear layer with a softmax, which converts encoder representations into a probability distribution over the extended vocabulary. For CTC-based training, the output units comprise BPE
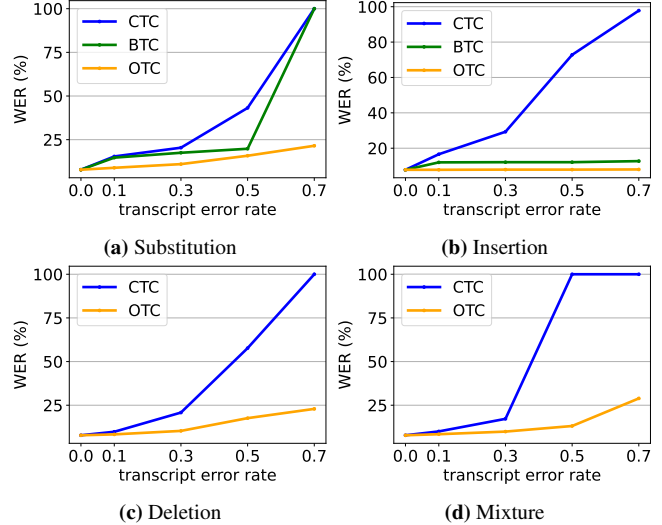

(a) Substitution   **(b)** Insertion

(c) Deletion   **(d)** Mixture

Fig. 8: WERs of models trained on LS `train-clean-100` with synthetic transcript errors: substitution, insertion, deletion, and a mixture of these at error rates: $\{0.0, 0.1, 0.3, 0.5, 0.7\}$. BTC results are compared in (a) substitution and (b) insertion. Results of CTC, BTC, and OTC are depicted in blue, green, and orange, respectively.

units and a blank token. For OTC, we additionally include the $\star$ token in the output, as described earlier.

5.3. Tokenization

We used byte pair encoding (BPE) [37] for sub-word tokenization. For this, we used the implementation available in the `sentencepiece` toolkit[7]. We explored vocabulary sizes of 100, 200, and 500, and determined that a size of 200 yields the optimal performance. Further analysis and details can be found in Section 7.3.

6. RESULTS & DISCUSSION

6.1. LibriSpeech 100h

As mentioned in Section 5.1, we first experimented with OTC-based training for LS `train-clean-100` by incorporating synthetic errors. We performed experiments under four scenarios: substitution, insertion, deletion, and a mix of all three errors at different error rates. For each of these weakly supervised settings, we trained models using both CTC and OTC objectives and evaluated their word error rate (WER) performance on LS `test-clean` using greedy decoding. For substitution and insertion cases, we also compared the results with BTC. The comparisons are shown in Fig. 8.

As the error rates increase (especially beyond 0.5), the performance of the ASR system using regular CTC worsens, and it fails to converge entirely beyond error rates of 0.7. On the other hand, models trained with OTC maintain their performance with only small degradation. OTC also outperforms BTC in scenarios involving substitution and insertion errors, due to the strategy of modeling the $\star$ token described in Section 3.2. Importantly, OTC performs **on par** with CTC in the case of verbatim transcripts, as shown in Table 2.

---

[6]`https://github.com/s3prl/s3prl`   [7]`https://github.com/google/sentencepiece`

Table 2: WER (%) of CTC/OTC on LS `train-clean-100` with verbatim transcript (transcript error rate = 0.0).

Transcript Error Rate	CTC	OTC
0.0	7.8	7.8

Table 3: Comparison of performance (WER (%)) of ASR systems trained on LS and LV using CTC/OTC.

	Training Data	Dev		Test	
		Clean	Other	Clean	Other
CTC	LS train-clean-100	4.9	13.6	5.2	13.5
	LV-100	98.9	98.8	98.8	98.7
OTC	LV-100	7.4	18.4	7.9	18.9
CTC	LS train-960	3.1	8.1	3.3	8.2
	LV-960	99.2	99.8	99.6	99.7
OTC	LV-960	5.9	12.6	6.1	12.5

6.2. LibriVox

We also evaluated OTC on LV-100 and LV-960 datasets. To establish a baseline for comparison, we trained the ASR system using the CTC on the LibriVox dataset. We define a "topline" as the result obtained by training the ASR system on the corresponding LS subset, which is clean and well-segmented as outlined in Section 4.1. We decoded using a 3-gram language model, and the results are shown in Table 3. The results obtained using CTC are emphasized in red.

These experiments highlight the ineffectiveness of direct training on the LV dataset using the CTC approach due to transcript errors. However, when utilizing OTC for training, we observed a significant reduction in the performance gap between the LS and LV datasets. Specifically, when considering the 100-hour subset, OTC only incurs a loss of 2.7% absolute WER on `test-clean` (from 5.2% to 7.9%) and 4.6% absolute WER `test-other` (from 13.5% to 18.9%). Similarly, on the 960-hour set, OTC experiences a loss of 2.8% on `test-clean` (from 3.3% to 6.1%) and 4.3% on `test-other` (from 8.2% to 12.5%).

7. ANALYSIS

The analyses are carried out using the LS `train-clean-100` with synthetic errors in the transcripts. Here, "error type" refers to the specific error made in the transcripts, while "error rate" indicates the ratio of that error to the reference transcripts.

7.1. Impact of penalty

The initial penalty β_1 and β_2 on the self-loop arc and bypass arc are determined empirically. We experimented with a range of different values and evaluated their impact on the performance of the ASR system. Parameter β_1 was selected for cases involving deletion errors only, where the error rate was set to 0.5 and the system was configured to only enable self-loop. β_2 was chosen for cases of substitution and insertion errors separately, with an error rate again set to 0.5, enabling only the bypass arc. The decay factors are set to $\tau_1 = 0.999, \tau_2 = 0.975$. The comparison is shown in Fig. 9

In the deletion case, when λ_1 is set to 3.75, the system achieves its lowest WER of 15.4. In the substitution and insertion case, the system achieves its best performance with a WER of 15.8 for substitution and 8.1 for insertion when β_2 is set to -19.
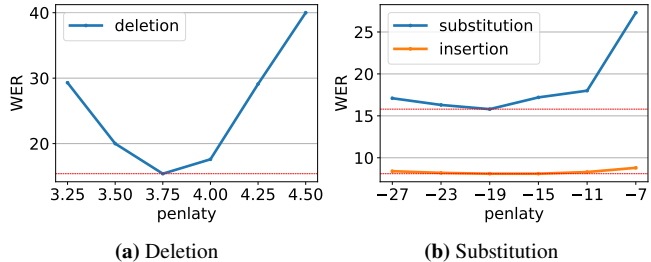

(a) Deletion **(b)** Substitution
Fig. 9: Self-loop and bypass penalty impact on WER (%).

Table 4: WER (%) for substitution/deletion at rate 0.5 across subword vocabulary size. The best result is highlighted in **bold**.

Error Type	Error Rate	Vocabulary Size		
		100	200	500
substitution	0.5	15.8	**15.2**	16.4
deletion	0.5	16.6	**15.4**	18.9

7.2. Segmentation length choices

Traditional ASR training typically utilizes utterances with lengths under 30 seconds, achieving optimal performance within the 10 to 15-second range [38]. However, in the context of OTC, uniform segmentation with short lengths introduces significant "shift errors" that misalign speech and text, while excessively long segments can overwhelm GPU memory in the conformer architecture. We conducted an exploration of different segment durations, including 15, 30, and 60 seconds. Our findings revealed that the model fails to converge when using 15 or 30-second segments. However, we observed that utilizing 60-second segments strikes the optimal balance.

7.3. Subword vocabulary size

We conducted experiments using subword vocabularies of sizes 100, 200, and 500, focusing on cases of substitution and deletion errors with an error rate of 0.5. The results, shown in Table 4, demonstrate that a vocabulary size of 200 consistently yielded the best performance in terms of WER for both substitution and deletion errors (namely, 15.2% for substitution and 15.4% for deletion).

8. CONCLUSION

We proposed a novel training criterion, Omni-Temporal Classification (OTC), for training ASR systems using weakly supervised data, such as non-verbatim transcripts. OTC allows the model to learn speech-text alignment while effectively addressing errors present in the transcripts within the WFST framework. Experimental results on the LibriSpeech and LibriVox datasets demonstrated that models trained with OTC maintain reasonable ASR performance even when the transcripts contain up to 70% errors of different types. From the LibriVox experiments, it is evident that our method can significantly reduce the human effort required for data preparation in training ASR systems.

9. ACKNOWLEDGEMENT

10. REFERENCES

[1] Jinyu Li, Yu Wu, Yashesh Gaur, et al., "On the comparison of popular end-to-end models for large scale speech recognition," in *INTERSPEECH*, 2020.

[2] Liang Lu, Changliang Liu, Jinyu Li, et al., "Exploring transformers for large-scale speech recognition," in *INTERSPEECH*, 2020.

[3] Yongqiang Wang, Yangyang Shi, Frank Zhang, et al., "Transformer in action: A comparative study of transformer-based acoustic models for large scale speech recognition applications," in *IEEE ICASSP*, 2020.

[4] Joris Driesen and Steve Renals, "Lightly supervised automatic subtitling of weather forecasts," *IEEE ASRU*, 2013.

[5] Yanhua Long, Mark John Francis Gales, Pierre Lanchantin, et al., "Improving lightly supervised training for broadcast transcription," in *INTERSPEECH*, 2013.

[6] Pierre Lanchantin, Mark John Francis Gales, Panagiota Karanasou, et al., "Selection of multi-genre broadcast data for the training of automatic speech recognition systems," in *INTERSPEECH*, 2016.

[7] Matthew Wiesner, Mousmita Sarma, Ashish Arora, et al., "Training hybrid models on noisy transliterated transcripts for code-switched speech recognition.," in *INTERSPEECH*, 2021.

[8] Hainan Xu and Philipp Koehn, "Zipporah: a fast and scalable data cleaning system for noisy web-crawled parallel corpora," in *EMNLP*, 2017.

[9] CK Yeh, J. Chen, C. Yu, et al., "Unsupervised speech recognition via segmental empirical output distribution matching," in *ICLR*, 2019.

[10] A. Baevski, WN Hsu, A. Conneau, et al., "Unsupervised speech recognition," in *NeurIPS*, 2021.

[11] Alexander H. Liu, Wei-Ning Hsu, Michael Auli, et al., "Towards end-to-end unsupervised speech recognition," in *IEEE SLT*, 2022.

[12] Ondřej Klejch, Electra Wallington, and Peter Bell, "Deciphering speech: a zero-resource approach to cross-lingual transfer in asr," in *INTERSPEECH*, 2022.

[13] Dongji Gao, Jiatong Shi, Shun-Po Chuang, et al., "EURO: ESPnet Unsupervised ASR Open-source Toolkit," in *IEEE ICASSP*, 2023.

[14] Vineel Pratap, Awni Hannun, Gabriel Synnaeve, et al., "Star temporal classification: Sequence classification with partially labeled data," in *NeurIPS*, 2022.

[15] Xingyu Cai, Jiahong Yuan, Yuchen Bian, et al., "W-CTC: a Connectionist Temporal Classification Loss with Wild Cards," in *ICLR*, 2022.

[16] Han Zhu, Dongji Gao, Gaofeng Cheng, et al., "Alternative pseudo-labeling for semi-supervised automatic speech recognition," *IEEE/ACM TASLP*, 2023.

[17] Dongji Gao, Matthew Wiesner, Hainan Xu, et al., "Bypass temporal classification: Weakly supervised automatic speech recognition with imperfect transcripts," in *INTERSPEECH*, 2023.

[18] Anthony Rousseau, Paul Deléglise, and Yannick Estève, "TED-LIUM: an automatic speech recognition dedicated corpus," in *LREC*, 2012.

[19] Anthony Rousseau, Paul Deléglise, and Yannick Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *IREC*, 2014.

[20] François Hernandez, Vincent Nguyen, Sahar Ghannay, et al., "Ted-lium 3: Twice as much data and corpus repartition for experiments on speaker adaptation," in *SPECOM*, 2018.

[21] Javier Iranzo-Sánchez, Joan Albert Silvestre-Cerdà, Javier Jorge, et al., "Europarl-st: A multilingual corpus for speech translation of parliamentary debates," in *IEEE ICASSP*, 2020.

[22] Cihan Xiao, Henry Li Xinyuan, Jinyi Yang, et al., "HK-LegiCoST: Leveraging Non-Verbatim Transcripts for Speech Translation," in *INTERSPEECH*, 2023.

[23] Gonçal V. Garcés Díaz-Munío, Joan-Albert Silvestre-Cerdà, Javier Jorge, et al., "Europarl-ASR: A Large Corpus of Parliamentary Debates for Streaming ASR Benchmarking and Speech Data Filtering/Verbatimization," in *INTERSPEECH*, 2021.

[24] Vineel Pratap, Andros Tjandra, Bowen Shi, et al., "Scaling speech technology to 1, 000+ languages," *ArXiv*, vol. abs/2305.13516, 2023.

[25] Alec Radford, Jong Wook Kim, Tao Xu, et al., "Robust speech recognition via large-scale weak supervision," *ArXiv*, 2022.

[26] Alex Graves, Santiago Fernández, Faustino Gomez, et al., "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[27] Mehryar Mohri, Fernando Carlos Neves Pereira, and Michael Riley, "Speech recognition with weighted finite-state transducers," 2008.

[28] Yajie Miao, Mohammad Abdelaziz Gowayyed, and Florian Metze, "Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding," in *IEEE ASRU*, 2015.

[29] Hasim Sak, Andrew W. Senior, Kanishka Rao, et al., "Learning acoustic frame labeling for speech recognition with recurrent neural networks," *IEEE ICASSP*, 2015.

[30] Aleksandr Laptev, Somshubra Majumdar, and Boris Ginsburg, "CTC Variations Through New WFST Topologies," in *INTERSPEECH*, 2021.

[31] V. Panayotov, G. Chen, D. Povey, et al., "Librispeech: an ASR corpus based on public domain audio books," in *IEEE ICASSP*, 2015.

[32] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, et al., "The kaldi speech recognition toolkit," in *IEEE ASRU*, 2011.

[33] Temple F Smith, Michael S Waterman, et al., "Identification of common molecular subsequences," *Journal of molecular biology*, 1981.

[34] A. Baevski, Y. Zhou, A. Mohamed, et al., "wav2vec 2.0: A framework for self-supervised learning of speech representations," in *NeurIPS*, 2020.

[35] Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, et al., "SUPERB: Speech Processing Universal PERformance Benchmark," in *INTERSPEECH*, 2021.

[36] A. Gulati, J. Qin, CC Chiu, et al., "Conformer: Convolution-augmented transformer for speech recognition," in *INTERSPEECH*, 2020.

[37] Rico Sennrich, Barry Haddow, and Alexandra Birch, "Neural machine translation of rare words with subword units," in *ACL*, 2015.

[38] Tomás Valenta and Lubos Smídl, "On the impact of sentence length on recognition accuracy," in *ISCP*, 2014.