MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Sequence Transduction with Graph-based Supervision

Moritz, Niko; Hori, Takaaki; Watanabe, Shinji; Le Roux, Jonathan

TR2022-024 March 05, 2022

Abstract

The recurrent neural network transducer (RNN-T) objective plays a major role in building today's best automatic speech recognition (ASR) systems for production. Similarly to the connectionist temporal classification (CTC) objective, the RNN-T loss uses specific rules that define how a set of alignments is generated to form a lattice for the full-sum training. However, it is yet largely unknown if these rules are optimal and do lead to the best possible ASR results. In this work, we present a new transducer objective function that generalizes the RNN-T loss to accept a graph representation of the labels, thus providing a flexible and efficient framework to manipulate training lattices, e.g., for studying different transition rules, implementing different transducer losses, or restricting alignments. We demonstrate that transducer-based ASR with CTC-like lattice achieves better results compared to standard RNN-T, while also ensuring a strictly monotonic alignment, which will allow better optimization of the decoding procedure. For example, the proposed CTC-like transducer achieves an improvement of 4.8% on the testother condition of LibriSpeech relative to an equivalent RNN-T based system.

IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2022

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

SEQUENCE TRANSDUCTION WITH GRAPH-BASED SUPERVISION

Niko Moritz[1], *Takaaki Hori*[1], *Shinji Watanabe*[2], *Jonathan Le Roux*[1]

[1]Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA
[2]Carnegie Mellon University, Pittsburgh, PA, USA

ABSTRACT

The recurrent neural network transducer (RNN-T) objective plays a major role in building today's best automatic speech recognition (ASR) systems for production. Similarly to the connectionist temporal classification (CTC) objective, the RNN-T loss uses specific rules that define how a set of alignments is generated to form a lattice for the full-sum training. However, it is yet largely unknown if these rules are optimal and do lead to the best possible ASR results. In this work, we present a new transducer objective function that generalizes the RNN-T loss to accept a graph representation of the labels, thus providing a flexible and efficient framework to manipulate training lattices, e.g., for studying different transition rules, implementing different transducer losses, or restricting alignments. We demonstrate that transducer-based ASR with CTC-like lattice achieves better results compared to standard RNN-T, while also ensuring a strictly monotonic alignment, which will allow better optimization of the decoding procedure. For example, the proposed CTC-like transducer achieves an improvement of 4.8% on the test-other condition of LibriSpeech relative to an equivalent RNN-T based system.

Index Terms— RNN-T, GTC-T, transducer, CTC, ASR

1. INTRODUCTION

Two of the most popular neural network loss functions in automatic speech recognition (ASR) are the connectionist temporal classification (CTC) [1] and Recurrent Neural Network Transducer (RNN-T) objectives [2]. The CTC and RNN-T losses are designed for an alignment-free training of a neural network model to learn a mapping of a sequence of inputs (e.g., the acoustic features) to a typically shorter sequence of output labels (e.g., words or sub-word units). While the CTC loss requires neural network outputs to be conditionally independent, the RNN-T loss provides an extension to train a neural network whose output frames are conditionally dependent on previous output labels. In order to perform training without knowing the alignment between the input and output sequences, both loss types marginalize over a set of all possible alignments. Such alignments are derived from the supervision information (the sequence of labels) by applying specific instructions that define how the sequence of labels is expanded to adjust to the length of the input sequence. In both cases, such instructions include the usage of an additional blank label and transition rules that are specific to the loss type. For example, in CTC, a blank label can be inserted at the beginning and end of the label sequence as well as between the ASR labels, and must be inserted between two similar ASR labels, and each label can get repeated as many times as necessary to match the input sequence length [1]. The RNN-T loss instead does not allow repetitions of an ASR label but the emission of multiple ASR labels per frame, which is possible because the neural network output has an additional dimensionality corresponding to a decoder state along which the training lattice is expanded [2]. The decoder state of an RNN-T model is obtained by the usage of an internal language model (LM), the predictor, where LM outputs are fused with the output of the encoder neural network using a joiner network [3].

Various prior studies in the literature have investigated using modifications of the CTC label transition rules such as Gram-CTC [4], the automatic segmentation criterion (ASG) [5], and the graph-based temporal classification (GTC) loss [6]. GTC provides a generalization of CTC that accepts a graph representation of the labeling information, allowing for label transitions defined in a graph format. Note that GTC has similarities to other recently proposed works on differentiable weighted finite state transducers such as GTN [7] and k2 [8], with the difference that GTN and k2 rely on automatic differentiation whereas gradients in GTC are manually computed. However, while numerous works have focused on improving training, inference, and neural network architectures for RNN-T [9–13], most studies that investigated altering the training lattice of transducer models focused on achieving a strictly monotonic alignment between the input and output sequences, and left other aspects of RNN-T, such as the emission of ASR labels over a single time frame, unaltered [14–17]. Popular examples of RNN-T variants are the Recurrent Neural Aligner (RNA) [18] and the Monotonic RNN-T (MonoRNN-T) [14] losses, whereby the main motivation for such variants is to better optimize the decoding process by using batching or vectorization techniques and to minimize delays [14, 19].

In this work, we propose the GTC-Transducer (GTC-T) objective, which extends GTC to conditional dependent neural network outputs similar to RNN-T. GTC-T allows the user to define the label transitions in a graph format and by that to easily explore new lattice structures for transducer-based ASR. Here, we propose to use a CTC-like lattice for training a GTC-T based ASR system, and compare results to a MonoRNN-T lattice type (also realized using GTC-T), standard RNN-T, as well as CTC. ASR results demonstrate that transducer-based ASR with a CTC-like graph outperforms all other loss types in terms of word error rates (WERs), especially when also using an external LM via shallow fusion.

2. GTC-TRANSDUCER

Let us consider a feature sequence X of length T' derived from a speech utterance, processed by a neural network to produce an output sequence of length T, potentially different from T' due to downsampling. This output sequence contains a set of posterior probability distributions at every point, since the neural network is conditionally dependent on previous label outputs generated by the ASR system and therefore has different states producing multiple posterior probability distributions for the labels. For example, $\boldsymbol{v}^{t,i}$ denotes the posterior probabilities for neural network state i at time step t and $v_k^{t,i}$ the posterior probability of output label k for state i at time t. The Graph-based Temporal Classification-Transducer (GTC-

T) objective function marginalizes over all possible label alignment sequences that are represented by the graph $\mathcal{G}$. Thus, the conditional probability for a given graph $\mathcal{G}$ is defined by the sum over all sequences of nodes in $\mathcal{G}$ of length T, which can be written as:

$$p(\mathcal{G}|X) = \sum_{\pi \in \mathcal{S}(\mathcal{G},T)} p(\pi|X), \tag{1}$$

where $\mathcal{S}$ represents a search function that expands $\mathcal{G}$ to a lattice of length T (not counting non-emitting start and end nodes), π denotes a single node sequence and alignment path, and $p(\pi|X)$ is the posterior probability for the path π given feature sequence X.

We introduce a few more notations that will be useful to derive $p(\mathcal{G}|X)$. The nodes are sorted in a breadth-first search manner and indexed using $g = 0, \ldots, G+1$, where 0 corresponds to the non-emitting start node and $G+1$ to the non-emitting end node. We denote by $l(g)$ the output symbol observed at node g, and by $W_{g,g'}$ and $I(g,g')$ the transition weight and the decoder state index on the edge connecting the nodes g and g'. Finally, we denote by $\pi_{t:t'} = (\pi_t, \ldots, \pi_{t'})$ the node sub-sequence of π from time index t to t'. Note that π_0 and π_{T+1} correspond to the non-emitting start and end nodes 0 and $G+1$.

In RNN-T, the conditional probabilities $p(\boldsymbol{y}|X)$ for a given label sequence $\boldsymbol{y}$ are computed efficiently by a dynamic programming algorithm, which is based on computing the forward and backward variables and combining them to compute $p(\boldsymbol{y}|X)$ at any given time t [2]. In a similar fashion, the GTC-T forward probability can be computed for $g = 1, \ldots, G$ using

$$\alpha_t(g) = \sum_{\substack{\pi \in \mathcal{S}(\mathcal{G},T):\\ \pi_{0:t} \in \mathcal{S}(\mathcal{G}_{0:g},t)}} \prod_{\tau=1}^{t} W_{\pi_{\tau-1},\pi_\tau} v_{l(\pi_\tau)}^{\tau,I(\pi_{\tau-1},\pi_\tau)}, \tag{2}$$

where $\mathcal{G}_{0:g}$ denotes the sub-graph of $\mathcal{G}$ containing all paths from node 0 to node g. The sum is taken over all possible π whose subsequence up to time index t can be generated in t steps from the sub-graph $\mathcal{G}_{0:g}$. Note that $\alpha_0(g)$ equals 1 if g corresponds to the start node and it equals 0 otherwise. The backward variable β is computed similarly for $g = 1, \ldots, G$ using

$$\beta_t(g) = \sum_{\substack{\pi \in \mathcal{S}(\mathcal{G},T):\\ \pi_{t:T+1} \in \mathcal{S}(\mathcal{G}_{g:G+1},T-t+1)}} W_{\pi_T,\pi_{T+1}} \prod_{\tau=t}^{T-1} W_{\pi_\tau,\pi_{\tau+1}} v_{l(\pi_{\tau+1})}^{\tau+1,I(\pi_\tau,\pi_{\tau+1})}, \tag{3}$$

where $\mathcal{G}_{g:G+1}$ denotes the sub-graph of $\mathcal{G}$ containing all paths from node g to node $G+1$. From the forward and backward variables at any t, the probability function $p(\mathcal{G}|X)$ can be computed using

$$p(\mathcal{G}|X) = \sum_{(g,g') \in \mathcal{G}} \alpha_{t-1}(g) W_{g,g'} v_{l(g')}^{t,I(g,g')} \beta_t(g'). \tag{4}$$

For gradient descent training, the loss function

$$\mathcal{L} = -\ln p(\mathcal{G}|X) \tag{5}$$

must be differentiated with respect to the network outputs, which can be written as

$$-\frac{\partial \ln p(\mathcal{G}|X)}{\partial v_k^{t,i}} = -\frac{1}{p(\mathcal{G}|X)} \frac{\partial p(\mathcal{G}|X)}{\partial v_k^{t,i}}, \tag{6}$$

for any symbol $k \in \mathcal{U}$ and any decoder state $i \in \mathcal{I}$, where $\mathcal{U}$ denotes a set of all possible output symbols and $\mathcal{I}$ a set of all possible decoder state indices. The derivative of $p(\mathcal{G}|X)$ with respect to $v_k^{t,i}$ can be written as

$$\frac{\partial p(\mathcal{G}|X)}{\partial v_k^{t,i}} = \sum_{(g,g') \in \Phi(\mathcal{G},k,i)} \alpha_{t-1}(g) W_{g,g'} \beta_t(g'), \tag{7}$$

where $\Phi(\mathcal{G},k,i) = \{(g,g') \in \mathcal{G} : l(g') = k \wedge I(g,g') = i\}$ denotes the set of edges in $\mathcal{G}$ that correspond to decoder state i and where label k is observed at node g'. To backpropagate the gradients through the softmax function of $v_k^{t,i}$, we need the derivative with respect to the unnormalized network outputs $h_k^{t,i}$ before the softmax is applied, which is

$$-\frac{\partial \ln p(\mathcal{G}|X)}{\partial h_k^{t,i}} = -\sum_{k' \in \mathcal{U}} \frac{\partial \ln p(\mathcal{G}|X)}{\partial v_{k'}^{t,i}} \frac{\partial v_{k'}^{t,i}}{\partial h_k^{t,i}}, . \tag{8}$$

Finally, the gradients for the neural network outputs are

$$-\frac{\partial \ln p(\mathcal{G}|X)}{\partial h_k^{i,t}} = \frac{v_k^{t,i}}{p(\mathcal{G}|X)} \Big(\sum_{(g,g') \in \Psi(\mathcal{G},i)} \alpha_{t-1}(g) W_{g,g'} v_{l(g')}^{t,i} \beta_t(g') - \sum_{(g,g') \in \Phi(\mathcal{G},k,i)} \alpha_{t-1}(g) W_{g,g'} \beta_t(g') \Big), \tag{9}$$

where $\Psi(\mathcal{G},i) = \{(g,g') \in \mathcal{G} : I(g,g') = i\}$. Eq. (9) is derived by substituting (7) and the derivative of the softmax function $\partial v_{k'}^{t,i} / \partial h_k^{t,i} = v_{k'}^{t,i} \delta_{kk'} - v_{k'}^{t,i} v_k^{t,i}$ into (8) and by using the fact that

$$\begin{aligned} -\sum_{k' \in \mathcal{U}} \frac{\partial \ln p(\mathcal{G}|X)}{\partial v_{k'}^{t,i}} v_{k'}^{t,i} \delta_{kk'} &= -\frac{\partial \ln p(\mathcal{G}|X)}{\partial v_k^{t,i}} v_k^{t,i} \\ &= -\frac{v_k^{t,i}}{p(\mathcal{G}|X)} \sum_{(g,g') \in \Phi(\mathcal{G},k,i)} \alpha_{t-1}(g) W_{g,g'} \beta_t(g'), \end{aligned} \tag{10}$$

and that

$$\begin{aligned} &\sum_{k' \in \mathcal{U}} \frac{\partial \ln p(\mathcal{G}|X)}{\partial v_{k'}^{t,i}} v_{k'}^{t,i} v_k^{t,i} \\ &= \sum_{k' \in \mathcal{U}} \frac{v_{k'}^{t,i} v_k^{t,i}}{p(\mathcal{G}|X)} \sum_{(g,g') \in \Phi(\mathcal{G},k',i)} \alpha_{t-1}(g) W_{g,g'} \beta_t(g'), \\ &= \frac{v_k^{t,i}}{p(\mathcal{G}|X)} \sum_{k' \in \mathcal{U}} \sum_{(g,g') \in \Phi(\mathcal{G},k',i)} \alpha_{t-1}(g) W_{g,g'} v_{k'}^{t,i} \beta_t(g'), \\ &= \frac{v_k^{t,i}}{p(\mathcal{G}|X)} \sum_{(g,g') \in \Psi(\mathcal{G},i)} \alpha_{t-1}(g) W_{g,g'} v_{l(g')}^{t,i} \beta_t(g'). \end{aligned} \tag{11}$$

The GTC-T loss is implemented in CUDA as an extension for pytorch to make it efficient.

3. GRAPH TOPOLOGY

The GTC-T objective allows the usage of different graph topologies for constructing the training lattice. In this work, we test two different graph types as shown in Fig. 1, where arrows correspond to edges and circles to nodes at which either a blank label, denoted by $\varnothing$, or an ASR label (empty circles) is emitted. Neural network states are indicated using i and reside on the edges of the graph.

Graph type a) of Fig. 1 corresponds to a CTC-like topology, i.e., the graph can insert blanks between ASR labels following the CTC transition rules [1] and each label can get repeated as many times as necessary to match the target sequence length. Dashed lines correspond to optional edges, which are used to account for the fact that blank nodes can be skipped unless the ASR labels of two consecutive nodes are the same, see CTC rules [1].

Graph type b) of Fig. 1 corresponds to a MonoRNN-T loss type [14]. The main difference of MonoRNN-T to standard RNN-T is that multiple ASR outputs per time frame are not permitted, i.e., a strictly monotonic alignment between the input sequence and the output sequence is enforced.

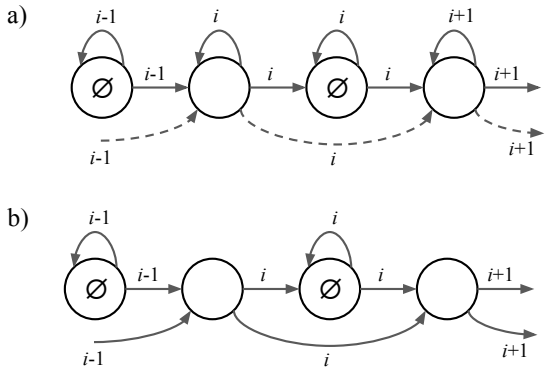

Fig. 1. Graph topologies for GTC-T training: a) CTC-like graph, b) MonoRNN-T graph. The neural network state are indicated by i, $\varnothing$ denotes the blank label, and empty nodes (circles) represent an ASR label.

Note that in order to ensure that the probabilities of all alignments in training will sum to one at most, the graph topology for GTC-T training must be carefully selected. In general, this means that the graph should be deterministic and that the posterior probabilities of all outgoing edges of a node should be derived from the same distribution, i.e., generated by using the same neural network state. Also note that all weights $W_{g,g'}$ of the graphs are set to one in this work.

4. DECODING ALGORITHM

The beam search algorithm for GTC-T with a CTC-like graph is shown as pseudo-code in Algorithm 1. The algorithm is inspired by a frame-synchronous CTC prefix beam search algorithm [20]. In this notation, ℓ corresponds to a prefix sequence, the prefix probability is separated into $p^t_{\text{nb}}(\ell)$ and $p^t_{\text{b}}(\ell)$ for ending with in blank (b) or not ending in blank (nb) at time step t, and θ_1 and θ_2 are used as thresholds for pruning the set of posterior probabilities locally and for score-based pruning of the set of prefixes/hypotheses. More specifically, function $\textsc{Prune}(\Omega_{\text{next}}, p_{\text{asr}}, P, \theta_2)$ performs two pruning steps. First, the set of hypotheses residing in Ω_{next} is limited to the P best hypotheses using the ASR scores p_{asr}, then any ASR hypothesis whose ASR score is less than $\log p_{\text{best}} - \theta_2$ is also removed from the set, where p_{best} denotes the best prefix ASR score in the set. The posterior probabilities $\boldsymbol{\upsilon}^{t,i}$ are generated by the neural network using $\text{NN}_{\text{ET}}(X, \ell, t)$, where X represents the input feature sequence, and i denotes the neural network state that depends on prefix ℓ. The posterior probability of ASR label k is denoted by $\upsilon^{t,i}_k$ and the neural network output $\upsilon^{t,j}_k$ for state j, cf. line 24 to 26, is associated with the prefix ℓ_+. Furthermore, α and β are the LM and label insertion bonus weights [20–22] and $|\ell|$ denotes the sequence length of prefix ℓ. The $\varnothing$ symbol represents the blank label and $\langle\text{sos}\rangle$ a start of sentence symbol.

5. EXPERIMENTS

5.1. Setup

We use the HKUST [23] and the LibriSpeech [24] ASR benchmarks for evaluation. HKUST is a corpus of Mandarin telephone speech recordings with more than 180 hours of transcribed speech data, and LibriSpeech comprises nearly 1k hours of read English audio books.

The ASR systems of this work are configured to first extract 80-dimensional log-mel spectral energies plus 3 extra features for pitch information [25]. The derived feature sequence is processed by a 2-layer VGG neural network [26], which downsamples the sequence of features to a frame rate of 60 ms, before being fed into a Conformer encoder architecture [27]. The encoder neural network is composed of 12 Conformer blocks, where each block includes a self-attention layer, a convolution module, and two Macaron-like feed-forward neural network modules [27]. In addition, the input to each component of the Conformer block is layer normalized and dropout is applied to the output of several neural network layers similar to [28]. Hyperparameters of the Conformer encoder are similar to [28], i.e., $d_{\text{model}} = 256$, $d_{\text{ff}} = 2048$, $d_h = 4$, and $E = 12$ for HKUST, while d_{model} and d_h are increased to 512 and 8 for LibriSpeech. For the CTC model, the output of the encoder neural network is projected to the number of output labels (including the blank label) using a linear layer and a softmax function to derive a probability distribution over the labels. For the GTC-T and RNN-T loss types, two additional neural network components are used, the prediction network and the joiner network. The prediction network, which consists of a single long short-term memory (LSTM) neural network and a dropout layer, acts like a language model and receives as an input the previously emitted ASR labels (ignoring the blank label), which are converted into an embedding space. The joiner network combines the sequence of encoder frames and the prediction neural network outputs using a few linear layers and a tanh activation function. Finally, a softmax is used to produce a probability distribution for the labels.

The Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 10^{-9}$, and

Algorithm 1 Beam search for GTC-T with a CTC-like graph.

1: $\ell \leftarrow (\langle\text{sos}\rangle,)$
2: $p^0_{\text{nb}}(\ell) \leftarrow 0$, $p^0_{\text{b}}(\ell) \leftarrow 1$
3: $\Omega_{\text{pruned}} \leftarrow \{\ell\}$, $\Omega_{\text{prev}} \leftarrow \{\}$
4: **for** $t = 1, \ldots, T$ **do**
5: $\quad\Omega_{\text{next}} \leftarrow \{\}$
6: $\quad$**for** ℓ **in** Ω_{pruned} **do**
7: $\quad\quad\boldsymbol{\upsilon}^{t,i} \leftarrow \text{NN}_{\text{ET}}(X, \ell, t)$
8: $\quad\quad C \leftarrow \{k$ **for** k **in** $\mathcal{U}$ **if** $\upsilon^{t,i}_k > \theta_1\}$
9: $\quad\quad$add $\varnothing$ to C
10: $\quad\quad$**for** k **in** C **do**
11: $\quad\quad\quad$**if** $k = \varnothing$ **then**
12: $\quad\quad\quad\quad p^t_{\text{b}}(\ell) \leftarrow \upsilon^{t,i}_k (p^{t-1}_{\text{b}}(\ell) + p^{t-1}_{\text{nb}}(\ell))$
13: $\quad\quad\quad\quad$**if** $|\ell| > 1$ **and** ℓ_{end} **not in** C **then**
14: $\quad\quad\quad\quad\quad p^t_{\text{nb}}(\ell) \leftarrow \upsilon^{t,i}_{\ell_{\text{end}}} p^{t-1}_{\text{nb}}(\ell)$
15: $\quad\quad\quad\quad$add ℓ to Ω_{next}
16: $\quad\quad\quad$**else**
17: $\quad\quad\quad\quad\ell_+ \leftarrow \ell + (k,)$
18: $\quad\quad\quad\quad$**if** $k = \ell_{\text{end}}$ **then**
19: $\quad\quad\quad\quad\quad p^t_{\text{nb}}(\ell_+) \leftarrow \upsilon^{t,i}_k p^{t-1}_{\text{b}}(\ell)$
20: $\quad\quad\quad\quad\quad p^t_{\text{nb}}(\ell) \leftarrow \upsilon^{t,i}_k p^{t-1}_{\text{nb}}(\ell)$
21: $\quad\quad\quad\quad$**else**
22: $\quad\quad\quad\quad\quad p^t_{\text{nb}}(\ell_+) \leftarrow \upsilon^{t,i}_k (p^{t-1}_{\text{b}}(\ell) + p^{t-1}_{\text{nb}}(\ell))$
23: $\quad\quad\quad\quad$**if** ℓ_+ **not in** Ω_{pruned} **and** ℓ_+ **in** Ω_{prev} **then**
24: $\quad\quad\quad\quad\quad\boldsymbol{\upsilon}^{t,j} \leftarrow \text{NN}_{\text{ET}}(X, \ell_+, t)$
25: $\quad\quad\quad\quad\quad p^t_{\text{b}}(\ell_+) \leftarrow \upsilon^{t,j}_{\varnothing} (p^{t-1}_{\text{b}}(\ell_+) + p^{t-1}_{\text{nb}}(\ell_+))$
26: $\quad\quad\quad\quad\quad p^t_{\text{nb}}(\ell_+) \leftarrow \upsilon^{t,j}_k p^{t-1}_{\text{nb}}(\ell_+)$
27: $\quad\quad\quad\quad\quad$add ℓ_+ to Ω_{next}
28: $\quad$**for** ℓ **in** Ω_{next} **do**
29: $\quad\quad p_{\text{asr}}(\ell) \leftarrow (p^t_{\text{b}}(\ell) + p^t_{\text{nb}}(\ell)) p_{\text{LM}}(\ell)^\alpha |\ell|^\beta$
30: $\quad\Omega_{\text{pruned}} \leftarrow \textsc{Prune}(\Omega_{\text{next}}, p_{\text{asr}}, P, \theta_2)$
31: $\quad\Omega_{\text{prev}} \leftarrow \Omega_{\text{next}}$

Table 1. HKUST ASR results. CTC15 denotes parameter initialization using the snapshot after 15 epochs of CTC training, BS10 denotes beam search with beam size 10, and joint indicates multi-objective training using RNN-T and CTC.

Loss	Training Graph	Init	Decoding Search	Decoding LM	WER [%] train-dev	WER [%] dev
CTC	-	-	greedy	-	21.3	21.6
CTC	-	-	BS10	LSTM	20.3	20.9
RNN-T	-	-	BS10	-	22.7	23.1
RNN-T	-	joint	BS10	-	21.5	22.2
RNN-T	-	joint	BS10	LSTM	21.3	22.2
GTC-T	CTC-like	-	greedy	-	24.2	24.4
GTC-T	CTC-like	CTC15	greedy	-	21.4	22.1
GTC-T	MonoRNN-T	CTC15	greedy	-	21.9	22.7
GTC-T	CTC-like	CTC15	BS10	-	20.9	21.8
GTC-T	CTC-like	CTC15	BS10	LSTM	20.8	21.7

learning rate scheduling similar to [29] with 25000 warmup steps is applied for training. The learning rate factor and the maximum number of training epochs are set to 1.0 and 50 for HKUST and to 5.0 and 100 for LibriSpeech. SpecAugment is used for all experiments [30]. A task-specific LSTM-based language model (LM) is trained using the official training text data of each ASR task [23, 24] and employed via shallow fusion during decoding whenever indicated. For HKUST, the LM consists of 2 LSTM layers with 650 units each. For LibriSpeech, 4 LSTM layers with 2048 units each are used instead. For LibriSpeech, we also test the effect of a strong Transformer-based LM (Tr-LM) with 16 layers. ASR output labels consist of a blank token plus 5,000 subword units obtained by the SentencePiece method [31] for LibriSpeech or of a blank token plus 3,653 character-based symbols for the Mandarin HKUST task.

Note that, in the following results section, greedy search is taking the underlying loss types into account, i.e., label sequences are collapsed according to the lattice topologies. The beam search method for RNN-T is based on the standard decoding algorithm proposed by Graves [2], and the GTC-T beam search with a CTC-like graph is explained in Section 4.

5.2. Results

ASR results for the CTC, RNN-T, and GTC-T losses on the HKUST benchmark are shown in Table 1. Joint CTC / RNN-T training [13] as well as parameter initialization for GTC-T training via CTC pre-training greatly improves ASR results for both RNN-T and GTC-T based models. Note that CTC-based initialization only affects parameters of the encoder neural network, while parameters of the prediction and joiner network remain randomly initialized. We leave better initialization of such model components to future work. The ASR results demonstrate that for GTC-T training the usage of a CTC-like graph performs better compared to a MonoRNN-T graph. In addition, the GTC-T model outperforms the results of the RNN-T model by 0.5% on the HKUST dev test set. While the usage of an LM via shallow fusion did not help to improve word error rates (WERs) for HKUST much in general, the CTC-based ASR results benefit the most with improvements between 0.7% and 1.0%. For HKUST, the CTC system also outperformed both the RNN-T as well as the GTC-T systems. We suspect the reasons for it is that RNN-T models are known to be data hungry [32] and the training data size is probably too small to show the full potential of transducer-based ASR systems.

ASR results on the larger LibriSpeech dataset are shown in Table 2, where RNN-T as well as GTC-T clearly outperform CTC

Table 2. WERs [%] for LibriSpeech. CTC20 indicates parameter initialization (Init) from epoch 20 of CTC training and CTC under Init denotes parameter initialization from a fully trained CTC model.

Loss	Training Graph	Init	Decoding Search	Decoding LM	dev clean	dev other	test clean	test other
CTC	-	-	greedy	-	4.9	12.0	5.0	11.7
CTC	-	-	BS10	LSTM	2.8	7.2	2.9	7.3
CTC	-	-	BS30	LSTM	2.7	7.1	2.9	7.1
CTC	-	-	BS10	Tr.	2.6	6.9	2.8	6.9
CTC	-	-	BS30	Tr.	2.5	6.8	2.5	6.8
RNN-T	-	-	greedy	-	3.2	8.0	3.3	8.2
RNN-T	-	-	BS10	-	3.0	7.8	3.1	8.0
RNN-T	-	joint	BS10	-	2.9	7.8	3.1	7.8
RNN-T	-	joint	BS10	LSTM	2.4	6.6	2.7	6.7
RNN-T	-	joint	BS30	LSTM	2.4	6.3	2.6	6.4
RNN-T	-	joint	BS10	Tr.	2.4	6.8	2.7	6.5
RNN-T	-	joint	BS30	Tr.	2.3	6.2	2.5	6.2
GTC-T	MonoRNN-T	-	greedy	-	5.3	13.2	5.4	13.5
GTC-T	MonoRNN-T	CTC20	greedy	-	4.1	10.3	4.2	10.5
GTC-T	CTC-like	-	greedy	-	4.3	11.0	4.5	11.2
GTC-T	CTC-like	-	BS10	-	4.2	10.4	4.3	10.6
GTC-T	CTC-like	CTC20	BS10	-	3.4	8.8	3.6	9.0
GTC-T	CTC-like	CTC	BS10	-	3.2	8.4	3.4	8.5
GTC-T	CTC-like	CTC	BS10	LSTM	2.4	6.1	2.7	6.2
GTC-T	CTC-like	CTC	BS30	LSTM	2.4	6.0	2.6	6.2
GTC-T	CTC-like	CTC	BS10	Tr.	2.3	6.0	2.5	6.0
GTC-T	CTC-like	CTC	BS30	Tr.	2.3	5.8	2.5	5.9

results. For example, GTC-T with a CTC-like graph, CTC-based initialization, a Transformer-based LM, and a beam size of 30 for decoding achieves a WERs of 5.9% for the test-other conditions of LibriSpeech. This is 0.9% better compared to the best CTC results despite using a strong LM and a generous beam size. The GTC-T results are also 0.3% better compared to the best RNN-T results of this work. In addition, similar to the HKUST experiments, it can be noticed that GTC-T with a CTC-like graph obtains better results than using the MonoRNN-T graph. However, the results of Table 2 also demonstrate that parameter initialization of the encoder neural network is particularly important for GTC-T training, and without initialization the training converges more slowly. We also conducted extra experiments by training the GTC-T model for more epochs when model parameters are not initialized (results are not shown in the table), which further improved the ASR results. In comparison, RNN-T model training converged faster in our experiments and extra training did not help improve results any further, which supports the assumption that initialization is particularly important for the GTC-T based training. We can also notice from the results that, for LibriSpeech, the RNN-T model performs better than GTC-T when no external LM is used. Conversely, this also indicates that the GTC-T system can make better use of an extra LM via shallow fusion, but the investigation of this finding, especially with respect to personalization issues of transducer-based ASR systems, remains for future work.

6. CONCLUSIONS

The proposed GTC-T loss provides a general framework for training transducer-based ASR models, where instructions for generating the training lattice are defined in graph format. We found that GTC-T with a CTC-like lattice outperforms standard RNN-T in terms of WERs, while also omitting a practical issue of RNN-T by not permitting repeated ASR outputs per time frame, which allows for better optimization of the decoding procedure. On LibriSpeech, the proposed CTC-like transducer ASR system achieved WERs of 2.5% (test-clean) and 5.9% (test-other), which is a relative improvement of almost 5% compared to standard RNN-T for the test-other condition.

7. REFERENCES

[1] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, vol. 148, Jun. 2006, pp. 369–376.

[2] A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[3] A. Graves, A. rahman Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, May 2013.

[4] H. Liu, Z. Zhu, X. Li, and S. Satheesh, "Gram-CTC: Automatic unit selection and target decomposition for sequence labelling," in *Proc. ICML*, Aug. 2017, p. 2188–2197.

[5] R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2Letter: an end-to-end ConvNet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[6] N. Moritz, T. Hori, and J. L. Roux, "Semi-supervised speech recognition via graph-based temporal classification," in *Proc. ICASSP*, Jun. 2021, pp. 6548–6552.

[7] A. Hannun, V. Pratap, J. Kahn, and W.-N. Hsu, "Differentiable weighted finite-state transducers," *arXiv preprint arXiv:2010.01003*, 2020.

[8] D. Povey *et al.*, "k2," https://github.com/k2-fsa/k2.

[9] J. Li, R. Zhao, H. Hu, and Y. Gong, "Improving RNN transducer modeling for end-to-end speech recognition," in *Proc. ASRU*, 2019, pp. 114–121.

[10] E. Weinstein, J. Apfel, M. Ghodsi, R. Cabrera, and X. Liu, "RNN-transducer with stateless prediction network," in *Proc. ICASSP*, 2020, pp. 7049–7053.

[11] G. Saon, Z. Tüske, and K. Audhkhasi, "Alignment-length synchronous decoding for rnn transducer," in *Proc. ICASSP*, 2020, pp. 7804–7808.

[12] C. Liu, F. Zhang, D. Le, S. Kim, Y. Saraf, and G. Zweig, "Improving RNN transducer based ASR with auxiliary tasks," *Proc. SLT*, pp. 172–179, 2021.

[13] F. Boyer, Y. Shinohara, T. Ishii, H. Inaguma, and S. Watanabe, "A study of transducer based end-to-end ASR with ESPnet: Architecture, auxiliary loss and decoding strategies," in *Proc. ASRU*, 2021, pp. 16–23.

[14] A. Tripathi, H. Lu, H. Sak, and H. Soltau, "Monotonic recurrent neural network transducer and decoding strategies," in *Proc. ASRU*, 2019, pp. 944–948.

[15] J. Mahadeokar, Y. Shangguan, D. Le, G. Keren, H. Su, T. Le, C.-F. Yeh, C. Fuegen, and M. L. Seltzer, "Alignment restricted streaming recurrent neural network transducer," in *Proc. SLT*, 2021, pp. 52–59.

[16] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, "A new training pipeline for an improved neural transducer," in *Proc. Interspeech*, Oct. 2020, pp. 2812–2816.

[17] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid autoregressive transducer (HAT)," in *Proc. ICASSP*, 2020, pp. 6139–6143.

[18] H. Sak, M. Shannon, K. Rao, and F. Beaufays, "Recurrent neural aligner: An encoder-decoder neural network model for sequence to sequence mapping," in *Proc. Interspeech*, 2017.

[19] H. Seki, T. Hori, S. Watanabe, N. Moritz, and J. L. Roux, "Vectorized beam search for CTC-attention-based speech recognition," in *Proc. Interspeech*, 2019, pp. 3825–3829.

[20] A. Y. Hannun, A. L. Maas, D. Jurafsky, and A. Y. Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs," *arXiv preprint arXiv:1408.2873*, 2014.

[21] N. Moritz, T. Hori, and J. Le Roux, "Streaming automatic speech recognition with the transformer model," in *Proc. ICASSP*, May 2020, pp. 6074–6078.

[22] N. Moritz, T. Hori, and J. Le Roux, "Streaming end-to-end speech recognition with joint CTC-attention based models," in *Proc. ASRU*, Dec. 2019, pp. 936–943.

[23] Y. Liu, P. Fung, Y. Yang, C. Cieri, S. Huang, and D. Graff, "HKUST/MTS: A very large scale mandarin telephone speech corpus," in *Proc. ISCSLP*, vol. 4274, 2006, pp. 724–735.

[24] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "LibriSpeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, Apr. 2015.

[25] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. Enrique Yalta Soplin, J. Heymann, M. Wiesner, N. Chen, A. Renduchintala, and T. Ochiai, "ESPnet: End-to-end speech processing toolkit," in *Proc. Interspeech*, Sep. 2018, pp. 2207–2211.

[26] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *Proc. Interspeech*, Aug. 2017, pp. 949–953.

[27] A. Gulati, J. Qin, C.-C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu *et al.*, "Conformer: Convolution-augmented transformer for speech recognition," in *Proc. Interspeech*, Oct. 2020.

[28] N. Moritz, T. Hori, and J. Le Roux, "Dual causal/non-causal self-attention for streaming end-to-end speech recognition," in *Proc. Interspeech*, 2021, pp. 1822–1826.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, Dec. 2017, pp. 6000–6010.

[30] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

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

[32] J. Li, Y. Wu, Y. Gaur, C. Wang, R. Zhao, and S. Liu, "On the comparison of popular end-to-end models for large scale speech recognition," in *Proc. Interspeech*, 2020.