INVESTIGATING SEQUENCE-LEVEL NORMALISATION FOR CTC-LIKE END-TO-END ASR

Zeyu Zhao, Peter Bell

Centre for Speech Technology Research, University of Edinburgh, UK

ABSTRACT

End-to-end Automatic Speech Recognition (E2E ASR) significantly simplifies the training process of an ASR model. Connectionist Temporal Classification (CTC) is one of the most popular methods for E2E ASR training. Implicitly, CTC has a unique topology which is very useful for sequence modelling. However, we find that by changing to another topology, we can make it even more effective. In this paper, we propose a new CTC-like method, for E2E ASR training, by modifying the topology of original CTC, so that the well-known abuse of the blank label in CTC can be resolved theoretically. As we change the topology, a normalisation term is necessary, which makes the form of the final loss function similar to Maximum Mutual Information (MMI); we hence name our method MMI-CTC. In addition to maximising the posterior probability of the target sequence, the normalisation enables models to explicitly minimise the probability of competing hypothesis at the word sequence level. Our experimental results show that MMI-CTC is more efficient than CTC, and that the normalisation is essential for sequence training.

Index Terms— ASR, E2E ASR, CTC, MMI, Sequence Training

1. INTRODUCTION

An Automatic Speech Recognition (ASR) system transcribes input speech into corresponding word sequences. Conventional ASR systems are based on Hidden Markov Model (HMM) [1, 2, 3]. To develop a conventional ASR model, we need to train several components separately, including an acoustic model, a language model, etc. Several loss functions were applied to train conventional ASR models [2, 3, 4, 5]. After obtaining all components, a decoding algorithm is applied to get the final best hypothesis.

To simplify the training procedure of ASR models, in the recent decade, many researchers focused on end-to-end (E2E) ASR modelling [6, 7, 8]. Instead of training several components separately, thanks to deep neural networks (DNN), E2E modelling makes it possible to develop an ASR model by simply training one DNN based on a loss function or a specific neural network structure [6, 4, 8, 7], where one only needs to pay attention to the inputs and the outputs of DNNs.

Inspired by both conventional and E2E loss functions, in this paper, we propose a new loss function, MMI-CTC, for E2E ASR training. We will see that MMI-CTC has a better than CTC and a faster convergence speed.

In the rest of this paper, we will first go through some related work in section 2. Then we will introduce the proposed loss function in depth in section 3. After that, we will show and analyse the experiment results in section 4. Finally, we conclude in section 5.

2. RELATED WORK

2.1. Connectionist Temporal Classification

The main problem for sequence training is that there is usually no alignment between input and target sequences available. In CTC [6], a blank label was introduced to deal with the absence of alignment. Given an input sequence $X = X_{1:T}$ and an target sequence $Y = Y_{1:L}$, CTC [6] marginalises alignments by

$$p(Y|X) = \sum_{\pi \in B^{-1}(Y)} p(\pi|X) = \sum_{\pi \in B^{-1}(Y)} \prod_{t'=1}^{T} p_{t'}(\pi_{t'}|X), \tag{1}$$

where $B(*)$ is the mapping that maps an alignment sequence to a target sequence by merging consecutive identical labels and removing blank labels. One essential assumption in CTC is conditional independence, i.e., $p(\pi|X) = \prod_{t'=1}^{T} p_{t'}(\pi_{t'}|X)$.

2.2. Maximum Mutual Information

Originally, in conventional ASR training, where there are separate acoustic and language models, the maximum likelihood (ML) loss function

$$\mathcal{L}_{ML} = -\log p(X|Y), \tag{2}$$

was applied to train acoustic models [2, 1]. In addition to maximising the likelihood $p(X|Y)$ for the corresponding

This work was supported by a Consolidated Studentships Award funded by Huawei Technologies Co., Ltd.

transcriptions, Maximum Mutual Information,

$$\mathcal{L}_{MMI} = -\log \frac{p(X|Y)p(Y)}{\sum_{Y'} p(X|Y')p(Y')}, \quad (3)$$

where $p(X|Y)$ and $p(Y)$ are modelled by the acoustic and the language model respectively, minimises the summation of the probabilities of other word sequences Y' at the same time, which has been proved to be more effective [5]. In order to make use of GPUs and to avoid lattice generation, Lattice Free MMI (LF-MMI) was also introduced, achieving state-of-the-art performance [9, 10].

2.3. Discussion

One of the well known issues with CTC is the so-called "peaky" issue, whereby particular labels tend to dominate the posterior probability at each time frame. This can be sub-optimal [11]. Regarding the reasons for the peaky issue, we believe that, when we use only one label to model a specific phoneme, grapheme or sentence piece [12], we strongly assume that the acoustics should keep the same over its whole process. However, this does not hold all the time. For instance, a diphthong in English has two sub-vowel sounds, so the acoustics of the beginning and the ending parts cannot be the same but we force our classifiers to classify them as the same class. As a result, only a small number of frames of acoustic features can be recognised as the corresponding label in CTC with the rest of them labelled as blank labels. On the other hand, in CTC [6], a blank label can actually represent several different things: firstly, a blank label can represent the silent parts of utterances; secondly, a blank label is compulsory between a pair of repeated labels, which is an imposed demand but it literally needs to absorb at least several frames of input features; and last, because of peaky issue [11], a blank token can also represent some parts of non-blank tokens. It is obviously incorrect or undesired that the blank label has so many roles. Thus, we would like to change the topology that allows us to use two output labels to model one output token and the above abuse of the blank label can be avoided. However, after changing the topology, we observe that of course, not all the alignments are valid. As a result, we need a normalisation procedure, which makes the final loss function similar to MMI. There have been some other work of modifying CTC [13, 14, 15] but we have not work looking at the aspect of topology.

3. METHOD

3.1. Tokens and Topology

As we have discussed in section 2.3, the shared blank label and the topology in CTC have some disadvantages. Thus, we propose individual blank tokens for different character tokens in our alphabet to let them inherit the third functions of the blank label in section 2.3. Besides, we also introduce a space token $\langle space \rangle$ to represent silent parts of utterances to deal with the first point in section 2.3. In summary, we denote A as the set of all character tokens, B as the set of all blank tokens, and $\langle space \rangle$ as the space token, where $|A| = |B|$, meaning each character token $a \in A$ has an individual blank token $\epsilon_a \in B$, and there is no blank token for $\langle space \rangle$. For the following discussion, let $C = A \cup B \cup \langle space \rangle\}$ denote the whole token set.

The topology of CTC and the proposed loss function is shown in fig. 1. By removing the self-loop on character token, a blank token is not mandatory between repeated tokens any more in our loss function, which tackles the second point in section 2.3.

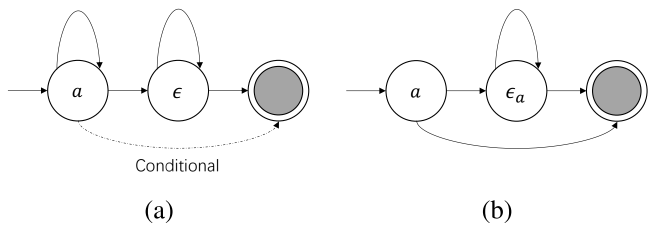

Fig. 1. Different topology of CTC (a) and the proposed one (b), where ϵ represents the shared blank label in CTC, and ϵ_a denotes individual blank token for token a in the proposed loss topology.

3.2. Objective

Define the input speech features as $X = X_{1:T}$, the target token sequence as $Y = Y_{1:L}$, where $X_t \in \mathbb{R}^D$ is the input feature vector at time step t, and $Y_l \in A$ is the l-th token in character token set A. Similar to CTC [6], because there is no alignment information between input speech and transcriptions, the posterior probability $p(Y|X)$ can be expressed as

$$p(Y|X) = \sum_{\pi \in B^{-1}(Y)} p(\pi|X) = \sum_{\pi \in B^{-1}(Y)} \prod_{t=1}^{T} p_t(\pi_t|X), \quad (4)$$

where $B : \pi \to Y$ is the mapping from an alignment sequence π to the target sequence Y by simply removing all space tokens and blank tokens, and $\pi = \pi_{1:T}$ is an alignment sequence with the length of T and $\pi_t \in A'$, and, finally, $p_t(\pi_t|X)$ is the posterior probability for token π_t at time step t. Besides, there is also a conditional independence assumption in our case as in CTC [6], i.e, $p(\pi|X) = p(\pi_1, \pi_2, \ldots, \pi_T|X) = \prod_{t=1}^{T} p(\pi_t|X)$.

As we know that the summation of the probabilities of all alignment sequences equals to one. However, there are many invalid alignments, as we introduce individual blank tokens and a character token can not be followed by other blank tokens expect the corresponding one. Thus, regarding the definition in eq. (4), the summation over all possible target sequences is smaller than one. One may be curious about the whether this does not happen in CTC. In fact, due to the CTC topology, an arbitrary alignment sequence can be mapped to a target sequence, so there is no invalid alignment sequence in CTC at all.

To normalise eq. (4), we need to introduce a denominator,

$$p(Y|X) = \frac{\sum_{\pi \in B^{-1}(Y)} \prod_{t=1}^{T} p_t(\pi_t|X)}{\sum_{Y'} \sum_{\pi \in B^{-1}(Y')} \prod_{t=1}^{T} p_t(\pi_t|X)}, \tag{5}$$

where the summation over Y' is calculated over all possible target sequences. Finally, the objective loss function is

$$\mathcal{L}_{MMI-CTC} = -\log p(Y|X) = \log D - \log N, \tag{6}$$

where D and N denote the denominator and the numerator respectively in eq. (5). As the structure of this loss function is similar with that of MMI, we name it *MMI-CTC*.

Besides, the main difference from LF-MMI [9, 10] is that there is no phoneme language model needed in MMI-CTC.

3.3. Forward-Backward Variables

The difficulty of computing the numerator and the denominator parts of eq. (6) is the summation over all corresponding alignments. Similar to CTC [6], where the authors proposed a forward-backward algorithm [16], for the numerator and the denominator, we define two pairs of forward and backward variables $\alpha_t^{(N)}, \beta_t^{(N)}$ and $\alpha_t^{(D)}, \beta_t^{(D)}$, which can be written as

$$\alpha_t^{(N)}(s) = \sum_{\substack{\pi \in B^{-1}(Y) \\ \pi_t = l_s}} p_t^{\alpha}(\pi), \alpha_t^{(D)}(k) = \sum_{\substack{\pi \in \Pi \\ \pi_t = c_k}} p_t^{\alpha}(\pi) \tag{7}$$

$$\beta_t^{(N)}(s) = \sum_{\substack{\pi \in B^{-1}(Y) \\ \pi_t = l_s}} p_t^{\beta}(\pi), \beta_t^{(D)}(k) = \sum_{\substack{\pi \in \Pi \\ \pi_t = c_k}} p_t^{\beta}(\pi) \tag{8}$$

where $p_t^{\alpha}(\pi) = \prod_{t'=1}^{t} p_{t'}(\pi_{t'}|X)$ is the forward probability of an alignment π at time step t and $p_t^{\beta}(\pi) = \prod_{t'=t}^{T} p_{t'}(\pi_{t'}|X)$ is the backward probability. In eqs. (7) and (8), for the numerator, s denotes the s-th token l_s in the auxiliary sequence $Y' = \{l_s\}_{s=1}^{S}$. For instance, considering a target sequence $Y = \{I\ am\}$, its auxiliary sequence is $Y' = \{\langle space \rangle, i, \epsilon_i, \langle space \rangle, a, \epsilon_a, m, \epsilon_m, \langle space \rangle\}$, where the beginning and the ending space tokens and all blank tokens are optional. For the denominator, k denotes the k-th token c_k in the complete token set C, and Π denotes all valid alignment sequences. The computation of all these variables, including initialisation, recursion and ending, is similar to CTC and due to space limitations we do not introduce it in detail.

3.4. Gradient

To train a neural network model, we need to calculate the gradients of the loss function with respect to the output of the neural network. For simplicity reason, we first define two variables,

$$\gamma_t^{(N)}(s) = \sum_{\substack{\pi \in B^{-1}(Y) \\ \pi_t = l_s}} p(\pi|X), \gamma_t^{(D)}(k) = \sum_{\substack{\pi \in \Pi \\ \pi_t = c_k}} p(\pi|X) \tag{9}$$

for the numerator and the denominator respectively. Obviously, substitute eqs. (7) and (8) into eq. (9) and we have,

$$\gamma_t^{(N)}(s) = \frac{\alpha_t^{(N)}(s)\beta_t^{(N)}(s)}{p_t(l_s|X)}, \gamma_t^{(D)}(k) = \frac{\alpha_t^{(D)}(s)\beta_t^{(D)}(s)}{p_t(c_k|X)}. \tag{10}$$

Besides, according to eqs. (5) and (9), we have

$$N_t = \sum_s \gamma_t^{(N)}(s), D_t = \sum_k \gamma_t^{(D)}(k), \tag{11}$$

where N_t and D_t are the numerator and the denominator of eq. (5) at time step t, respectively and they are certainly identical over all the time steps from 1 to T. Thus, we just denote them as N and D in the following, respectively. The derivative of loss function $\mathcal{L}$ with respect to the output of the neural network $\log p_t(c_k|X)$

$$\frac{\partial \mathcal{L}}{\partial \log p_t(c_k|X)} = \frac{1}{D} \frac{\partial D}{\partial \log p_t(c_k|X)} - \frac{1}{N} \frac{\partial N}{\partial \log p_t(c_k|X)}. \tag{12}$$

Substitute eq. (11) and consider eq. (10) and the chain rule, and we have,

$$\frac{\partial \mathcal{L}}{\partial \log p_t(c_k|X)} = \frac{\gamma_t^{(D)}(k)}{\sum_k \gamma_t^{(D)}(k)} - \sum_s \frac{\gamma_t^{(N)}(s)}{\sum_{s'} \gamma_t^{(N)}(s')} \frac{\partial p_t(l_s|X)}{\partial p_t(c_k|X)}, \tag{13}$$

where $\frac{\partial p_t(l_s|X)}{\partial p_t(c_k|X)} = 1$ when $l_s = c_k$, else 0.

3.5. Implementation

To calculate the forward and the backward variables $\alpha_t^{(N)}(s)$, $\beta_t^{(N)}(s)$, $\alpha_t^{(D)}(k)$ and $\beta_t^{(D)}(k)$ efficiently, we generate transition matrix A and transform matrix B according to the auxiliary sequence Y', where $A = [a_{ij}]_{S \times S}$ and $a_{ij} = 1$ if l_i can be arrived from l_j otherwise $a_{ij} = 0$ [16]. The transform matrix $B = B_{S \times K}$, whose row vectors are one-hot, transforms $p(c_k|X)$ to $p(l_s|X)$ by matrix multiplication. Finally, the recursion process for the numerator forward and backward variables can be simply denoted by

$$\alpha_{t+1}^{(N)} = A\alpha_t^{(N)} \circ Bp_t, \beta_t^{(N)} = A^{\mathrm{T}}\beta_{t+1}^{(N)} \circ B^{(N)}p_t \tag{14}$$

where $\alpha_t^{(D)}$, $\beta_t^{(D)}$ and p_t are column vectors at time step t and $\circ$ denotes the element-wise multiplication. As for the denominator, we generate the transition matrix according to the topology and the transform matrix here is just an identical matrix. The recursion process is similar to eq. (14).

4. EXPERIMENTS

4.1. Settings

We evaluate our models trained with CTC, and MMI-CTC on WSJ dataset. We apply ESPnet [17] and Kaldi [18] for feature extraction and PyTorch [19] for model training and evaluation. We extract 80-dim fbank features and 3-dim pitch features, in total 83-dim input features, as the input of neural networks. The model is 4-layer bi-directional LSTM [20] with one projection layer between each pair of LSTMs, which is named as RNNP in ESPnet. The number of the units in LSTM for one direction is 512 and that of the projection layers is 512 as well. There is also a dropout rate of 0.2 between each pair of LSTMs. We train our models with Adam optimiser and a learning rate scheduler that reduces the learning rate automatically when there is no loss reduction on development set. We stop training when there is no loss reduction on development set, which is regarded as convergence. When decoding, we apply a beam search decoding algorithm as in [21] with a beam size of 50. The decoding algorithms of CTC and MMI-CTC are similar. However, one essential difference from CTC is that we only allow the paths compatible with MMI-CTC topology. Besides, we also decode with an 3-gram language model in ARPA format from WSJ dataset, where the language model coefficient during decoding is 0.5, which is tuned on development set.

4.2. Results and Analysis

The results with and without language model are shown in tables 1 and 2, where MMI-CTC-oN denotes the model trained by MMI-CTC but without normalisation. To evaluate the effect of normalisation in eq. (5), for MMI-CTC-oN, we manually assign the first item of eq. (12) to zero, but we still use eq. (6) to calculate the loss to make them comparable.

Table 1. The performance of models w/o language model.

WER%	CTC	MMI-CTC	MMI-CTC-oN	ESPnet
dev93	29.9	27.3	37.0	30.5
eval92	24.0	22.8	29.8	24.3

Table 2. The performance of models with language model.

WER%	CTC	MMI-CTC	MMI-CTC-oN
dev93	17.4	16.4	20.7
eval92	13.3	12.7	14.6

In order to verify our implementation, we also train a model with the same neural network structure and hyperparameters based on pure CTC by ESPnet. We can see that in both two cases, with and without the language model, MMI-CTC is consistently better than CTC.

We also notice that normalisation is significantly important for MMI-CTC to achieve a better performance. In addition to the overall performance, we also find that the convergence speed of MMI-CTC is faster than CTC as shown in fig. 2, as it stops earlier than CTC. One may say the numbers of the tokens in MMI-CTC and CTC are different, and this could be a reason for the different convergence speed. Thus, we also show the curve for MMI-CTC-oN. As we know the only difference between MMI-CTC-oN and MMI-CTC is that there is no denominator minimisation in the former one, we can prove that the faster convergence speed arises from the denominator minimisation. Note that the loss in fig. 2 has a physical meaning of the average $-\log p(Y|X)$ and at last $\mathcal{L}_{MMI-CTC}$ is smaller than $\mathcal{L}_{CTC}$ on the development set, which implies the former has a better generalisation ability.

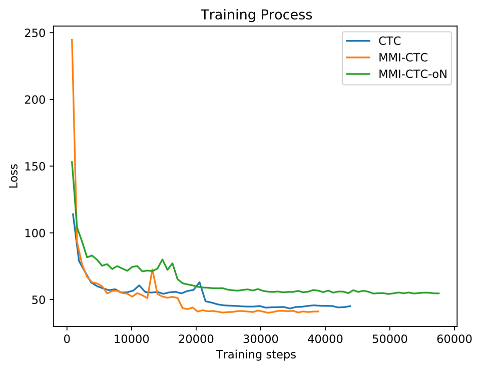

Fig. 2. The training loss on development set.

5. CONCLUSION

To deal with the undesired aspects of CTC, we propose to change the topology the introduce individual blank tokens. The new topology drives us to introduce a denominator for normalisation, which is the main difference between CTC and MMI-CTC in terms of equations. As we can see from the experiment results, MMI-CTC has a better performance and converges faster than CTC. This is because the normalisation enables the model to directly minimise the probability of other possible word sequences. In future work, we are planning to apply MMI-CTC in more powerful DNN architecture and expecting to achieve state-of-the-art performance.

6. REFERENCES

[1] Mark Gales and Steve Young, "The application of hidden Markov Models in speech recognition," *Foundations and Trends in Signal Processing*, vol. 1, no. 3, pp. 195–304, 2007.

[2] L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[3] L.R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[4] Geoffrey Hinton, Li Deng, et al., "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] Karel Veselý, Arnab Ghoshal, et al., "Sequence-discriminative training of deep neural networks," in *Proc. of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2013, pp. 2345–2349.

[6] Alex Graves, Santiago Fernández, et al., "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *ACM International Conference Proceeding Series*, 2006, vol. 148, pp. 369–376.

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

[8] William Chan, Navdeep Jaitly, et al., "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016, pp. 4960–4964.

[9] Hossein Hadian, Hossein Sameti, et al., "End-to-end speech recognition using lattice-free MMI," in *Proc. of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018, pp. 12–16.

[10] Daniel Povey, Vijayaditya Peddinti, et al., "Purely sequence-trained neural networks for ASR based on lattice-free MMI," in *Proc. of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2016, pp. 2751–2755.

[11] Andrew Rosenberg, Kartik Audhkhasi, et al., "End-to-end speech recognition and keyword search on low-resource languages," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 5280–5284.

[12] Taku Kudo and John Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[13] Jaesong Lee and Shinji Watanabe, "Intermediate loss regularization for CTC-based speech recognition," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2021, pp. 6224–6228.

[14] Yosuke Higuchi, Shinji Watanabe, et al., "Mask CTC: Non-autoregressive end-to-end ASR with CTC and mask predict," in *Proc. of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2020, pp. 3655–3659.

[15] Takafumi Moriya, Hiroshi Sato, et al., "Distilling Attention Weights for CTC-Based ASR Systems," in *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020, pp. 6894–6898.

[16] Khe Chai Sim and Arun Narayanan, "An efficient phone n-gram forward-backward computation using dense matrix multiplication," in *Proc. of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2017, pp. 1646–1650.

[17] Shinji Watanabe, Takaaki Hori, et al., "ESPnet: End-to-end speech processing toolkit," in *Proc. of Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2018, pp. 2207–2211.

[18] Daniel Povey, Arnab Ghoshal, et al., "The kaldi speech recognition toolkit," in *IEEE 2011 workshop on automatic speech recognition and understanding*. IEEE Signal Processing Society, 2011.

[19] Adam Paszke, Sam Gross, et al., "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, pp. 8024–8035. Curran Associates, Inc., 2019.

[20] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] Awni Y Hannun, Andrew L Maas, Daniel Jurafsky, and Andrew Y Ng, "First-pass large vocabulary continuous speech recognition using bi-directional recurrent dnns," *arXiv preprint arXiv:1408.2873*, 2014.