A Purely End-to-end System for Multi-speaker Speech Recognition

Hiroshi Seki[1,2]*, Takaaki Hori[1], Shinji Watanabe[3], Jonathan Le Roux[1], John R. Hershey[1]
[1]Mitsubishi Electric Research Laboratories (MERL)
[2]Toyohashi University of Technology
[3]Johns Hopkins University

Abstract

Recently, there has been growing interest in multi-speaker speech recognition, where the utterances of multiple speakers are recognized from their mixture. Promising techniques have been proposed for this task, but earlier works have required additional training data such as isolated source signals or senone alignments for effective learning. In this paper, we propose a new sequence-to-sequence framework to directly decode multiple label sequences from a single speech sequence by unifying source separation and speech recognition functions in an end-to-end manner. We further propose a new objective function to improve the contrast between the hidden vectors to avoid generating similar hypotheses. Experimental results show that the model is directly able to learn a mapping from a speech mixture to multiple label sequences, achieving 83.1% relative improvement compared to a model trained without the proposed objective. Interestingly, the results are comparable to those produced by previous end-to-end works featuring explicit separation and recognition modules.

1 Introduction

Conventional automatic speech recognition (ASR) systems recognize a single utterance given a speech signal, in a one-to-one transformation. However, restricting the use of ASR systems to situations with only a single speaker limits their applicability. Recently, there has been growing interest in single-channel multi-speaker speech recognition, which aims at generating multiple transcriptions from a single-channel mixture of multiple speakers' speech (Cooke et al., 2009).

To achieve this goal, several previous works have considered a two-step procedure in which the mixed speech is first separated, and recognition is then performed on each separated speech signal (Hershey et al., 2016; Isik et al., 2016; Yu et al., 2017; Chen et al., 2017). Dramatic advances have recently been made in speech separation, via the *deep clustering* framework (Hershey et al., 2016; Isik et al., 2016), hereafter referred to as DPCL. DPCL trains a deep neural network to map each time-frequency (T-F) unit to a high-dimensional embedding vector such that the embeddings for the T-F unit pairs dominated by the same speaker are close to each other, while those for pairs dominated by different speakers are farther away. The speaker assignment of each T-F unit can thus be inferred from the embeddings by simple clustering algorithms, to produce masks that isolate each speaker. The original method using k-means clustering (Hershey et al., 2016) was extended to allow end-to-end training by unfolding the clustering steps using a permutation-free mask inference objective (Isik et al., 2016). An alternative approach is to perform *direct mask inference* using the permutation-free objective function with networks that directly estimate the labels for a fixed number of sources. Direct mask inference was first used in Hershey et al. (2016) as a baseline method, but without showing good performance. This approach was revisited in Yu et al. (2017) and Kolbaek et al. (2017) under the name permutation-invariant training (PIT). Combination of such single-channel speaker-independent multi-speaker speech separation systems with ASR was first considered in Isik et al. (2016) using a conventional Gaussian Mixture Model/Hidden Markov Model

*This work was done while H. Seki, Ph.D. candidate at Toyohashi University of Technology, Japan, was an intern at MERL.

(GMM/HMM) system. Combination with an end-to-end ASR system was recently proposed in (Settle et al., 2018). Both these approaches either trained or pre-trained the source separation and ASR networks separately, making use of mixtures and their corresponding isolated clean source references. While the latter approach could in principle be trained without references for the isolated speech signals, the authors found it difficult to train from scratch in that case. This ability can nonetheless be used when adapting a pre-trained network to new data without such references.

In contrast with this two-stage approach, Qian et al. (2017) considered direct optimization of a deep-learning-based ASR recognizer without an explicit separation module. The network is optimized based on a permutation-free objective defined using the cross-entropy between the system's hypotheses and reference labels. The best permutation between hypotheses and reference labels in terms of cross-entropy is selected and used for backpropagation. However, this method still requires reference labels in the form of senone alignments, which have to be obtained on the clean isolated sources using a single-speaker ASR system. As a result, this approach still requires the original separated sources. As a general caveat, generation of multiple hypotheses in such a system requires the number of speakers handled by the neural network architecture to be determined before training. However, Qian et al. (2017) reported that the recognition of two-speaker mixtures using a model trained for three-speaker mixtures showed almost identical performance with that of a model trained on two-speaker mixtures. Therefore, it may be possible in practice to determine an upper bound on the number of speakers.

Chen et al. (2018) proposed a progressive training procedure for a hybrid system with explicit separation motivated by curriculum learning. They also proposed self-transfer learning and multi-output sequence discriminative training methods for fully exploiting pairwise speech and preventing competing hypotheses, respectively.

In this paper, we propose to circumvent the need for the corresponding isolated speech sources when training on a set of mixtures, by using an end-to-end multi-speaker speech recognition without an explicit speech separation stage. In separation based systems, the spectrogram is segmented into complementary regions according to sources, which generally ensures that different utterances are recognized for each speaker. Without this complementarity constraint, our direct multi-speaker recognition system could be susceptible to redundant recognition of the same utterance. In order to prevent degenerate solutions in which the generated hypotheses are similar to each other, we introduce a new objective function that enhances contrast between the network's representations of each source. We also propose a training procedure to provide permutation invariance with low computational cost, by taking advantage of the joint CTC/attention-based encoder-decoder network architecture proposed in (Hori et al., 2017a). Experimental results show that the proposed model is able to directly convert an input speech mixture into multiple label sequences without requiring any explicit intermediate representations. In particular no frame-level training labels, such as phonetic alignments or corresponding unmixed speech, are required. We evaluate our model on spontaneous English and Japanese tasks and obtain comparable results to the DPCL based method with explicit separation (Settle et al., 2018).

2 Single-speaker end-to-end ASR

2.1 Attention-based encoder-decoder network

An attention-based encoder-decoder network (Bahdanau et al., 2016) predicts a target label sequence $Y = (y_1, \dots, y_N)$ without requiring intermediate representation from a T-frame sequence of D-dimensional input feature vectors, $O = (o_t \in \mathbb{R}^D | t = 1, \dots, T)$, and the past label history. The probability of the n-th label y_n is computed by conditioning on the past history $y_{1:n-1}$:

$$p_{\text{att}}(Y|O) = \prod_{n=1}^{N} p_{\text{att}}(y_n|O, y_{1:n-1}). \quad (1)$$

The model is composed of two main sub-modules, an encoder network and a decoder network. The encoder network transforms the input feature vector sequence into a high-level representation $H = (h_l \in \mathbb{R}^C | l = 1, \dots, L)$. The decoder network emits labels based on the label history y and a context vector c calculated using an attention mechanism which weights and sums the C-dimensional sequence of representation H with attention weight a. A hidden state e of the decoder is

updated based on the previous state, the previous context vector, and the emitted label. This mechanism is summarized as follows:

$$H = \mathrm{Encoder}(O), \tag{2}$$
$$y_n \sim \mathrm{Decoder}(c_n, y_{n-1}), \tag{3}$$
$$c_n, a_n = \mathrm{Attention}(a_{n-1}, e_n, H), \tag{4}$$
$$e_n = \mathrm{Update}(e_{n-1}, c_{n-1}, y_{n-1}). \tag{5}$$

At inference time, the previously emitted labels are used. At training time, they are replaced by the reference label sequence $R = (r_1, \ldots, r_N)$ in a *teacher-forcing* fashion, leading to conditional probability $p_{\mathrm{att}}(Y_R|O)$, where Y_R denotes the output label sequence variable in this condition. The detailed definitions of $\mathrm{Attention}$ and Update are described in Section A of the supplementary material. The encoder and decoder networks are trained to maximize the conditional probability of the reference label sequence R using backpropagation:

$$\mathcal{L}_{\mathrm{att}} = \mathrm{Loss}_{\mathrm{att}}(Y_R, R) \triangleq -\log p_{\mathrm{att}}(Y_R = R|O), \tag{6}$$

where $\mathrm{Loss}_{\mathrm{att}}$ is the cross-entropy loss function.

2.2 Joint CTC/attention-based encoder-decoder network

The joint CTC/attention approach (Kim et al., 2017; Hori et al., 2017a), uses the connectionist temporal classification (CTC) objective function (Graves et al., 2006) as an auxiliary task to train the network. CTC formulates the conditional probability by introducing a framewise label sequence Z consisting of a label set $\mathcal{U}$ and an additional blank symbol defined as $Z = \{z_l \in \mathcal{U} \cup \{\texttt{'blank'}\} | l = 1, \cdots, L\}$:

$$p_{\mathrm{ctc}}(Y|O) = \sum_{Z} \prod_{l=1}^{L} p(z_l|z_{l-1}, Y) p(z_l|O), \tag{7}$$

where $p(z_l|z_{l-1}, Y)$ represents monotonic alignment constraints in CTC and $p(z_l|O)$ is the frame-level label probability computed by

$$p(z_l|O) = \mathrm{Softmax}(\mathrm{Linear}(h_l)), \tag{8}$$

where h_l is the hidden representation generated by an encoder network, here taken to be the encoder of the attention-based encoder-decoder network defined in Eq. (2), and $\mathrm{Linear}(\cdot)$ is the final linear layer of the CTC to match the number of labels. Unlike the attention model, the forward-backward algorithm of CTC enforces monotonic alignment between the input speech and the output label sequences during training and decoding. We adopt the joint CTC/attention-based encoder-decoder network as the monotonic alignment helps the separation and extraction of high-level representation. The CTC loss is calculated as:

$$\mathcal{L}_{\mathrm{ctc}} = \mathrm{Loss}_{\mathrm{ctc}}(Y, R) \triangleq -\log p_{\mathrm{ctc}}(Y = R|O). \tag{9}$$

The CTC loss and the attention-based encoder-decoder loss are combined with an interpolation weight $\lambda \in [0, 1]$:

$$\mathcal{L}_{\mathrm{mtl}} = \lambda \mathcal{L}_{\mathrm{ctc}} + (1 - \lambda)\mathcal{L}_{\mathrm{att}}. \tag{10}$$

Both CTC and encoder-decoder networks are also used in the inference step. The final hypothesis is a sequence that maximizes a weighted conditional probability of CTC in Eq. (7) and attention-based encoder decoder network in Eq. (1):

$$\hat{Y} = \arg\max_{Y} \big\{ \gamma \log p_{\mathrm{ctc}}(Y|O) + (1 - \gamma) \log p_{\mathrm{att}}(Y|O) \big\}, \tag{11}$$

where $\gamma \in [0, 1]$ is an interpolation weight.

3 Multi-speaker end-to-end ASR

3.1 Permutation-free training

In situations where the correspondence between the outputs of an algorithm and the references is an arbitrary permutation, neural network training faces a *permutation problem*. This problem was first addressed by deep clustering (Hershey et al., 2016), which circumvented it in the case of source separation by comparing the relationships between pairs of network outputs to those between pairs of labels. As a baseline for deep clustering, Hershey et al. (2016) also proposed another approach to address the permutation problem, based on an objective which considers all permutations of references when computing the error with the network estimates. This objective was later used in Isik et al. (2016) and Yu et al. (2017). In the latter, it was referred to as permutation-invariant training.

This permutation-free training scheme extends the usual one-to-one mapping of outputs and labels for backpropagation to one-to-many by selecting the proper permutation of hypotheses and

references, thus allowing the network to generate multiple independent hypotheses from a single-channel speech mixture. When a speech mixture contains speech uttered by S speakers simultaneously, the network generates S label sequence variables $Y^s = (y_1^s, \ldots, y_{N_s}^s)$ with N_s labels from the T-frame sequence of D-dimensional input feature vectors, $O = (o_t \in \mathbb{R}^D | t = 1, \ldots, T)$:

$$Y^s \sim g^s(O), \ s = 1, \ldots, S, \tag{12}$$

where the transformations g^s are implemented as neural networks which typically share some components with each other. In the training stage, all possible permutations of the S sequences $R^s = (r_1^s, \ldots, r_{N_s'}^s)$ of N_s' reference labels are considered (considering permutations on the hypotheses would be equivalent), and the one leading to minimum loss is adopted for backpropagation. Let $\mathcal{P}$ denote the set of permutations on $\{1, \ldots, S\}$. The final loss $\mathcal{L}$ is defined as

$$\mathcal{L} = \min_{\pi \in \mathcal{P}} \sum_{s=1}^{S} \mathrm{Loss}(Y^s, R^{\pi(s)}), \tag{13}$$

where $\pi(s)$ is the s-th element of a permutation π. For example, for two speakers, $\mathcal{P}$ includes two permutations $(1, 2)$ and $(2, 1)$, and the loss is defined as:

$$\mathcal{L} = \min(\mathrm{Loss}(Y^1, R^1) + \mathrm{Loss}(Y^2, R^2), \\ \mathrm{Loss}(Y^1, R^2) + \mathrm{Loss}(Y^2, R^1)). \tag{14}$$

Figure 1 shows an overview of the proposed end-to-end multi-speaker ASR system. In the following Section 3.2, we describe an extension of encoder network for the generation of multiple hidden representations. We further introduce a permutation assignment mechanism for reducing the computation cost in Section 3.3, and an additional loss function $\mathcal{L}_{KL}$ for promoting the difference between hidden representations in Section 3.4.

3.2 End-to-end permutation-free training

To make the network output multiple hypotheses, we consider a stacked architecture that combines both shared and unshared (or specific) neural network modules. The particular architecture we consider in this paper splits the encoder network into three stages: the first stage, also referred to as mixture encoder, processes the input mixture and

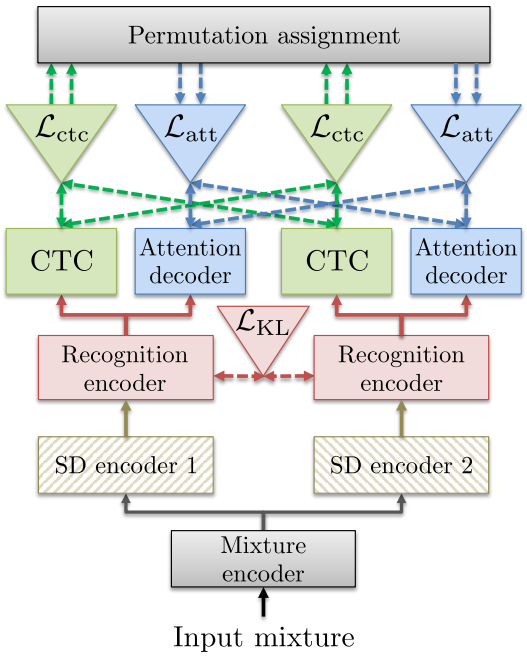

Figure 1: End-to-end multi-speaker speech recognition. We propose to use the permutation-free training for CTC and attention loss functions $\mathrm{Loss}_{\mathrm{ctc}}$ and $\mathrm{Loss}_{\mathrm{att}}$, respectively.

outputs an intermediate feature sequence H; that sequence is then processed by S independent encoder sub-networks which do not share parameters, also referred to as speaker-differentiating (SD) encoders, leading to S feature sequences H^s; at the last stage, each feature sequence H^s is independently processed by the same network, also referred to as recognition encoder, leading to S final high-level representations G^s.

Let $u \in \{1 \ldots, S\}$ denote an output index (corresponding to the transcription of the speech by one of the speakers), and $v \in \{1 \ldots, S\}$ denote a reference index. Denoting by $\mathrm{Encoder}_{\mathrm{Mix}}$ the mixture encoder, $\mathrm{Encoder}_{\mathrm{SD}}^{\mathrm{u}}$ the u-th speaker-differentiating encoder, and $\mathrm{Encoder}_{\mathrm{Rec}}$ the recognition encoder, an input sequence O corresponding to an input mixture can be processed by the encoder network as follows:

$$H = \mathrm{Encoder}_{\mathrm{Mix}}(O), \tag{15}$$
$$H^u = \mathrm{Encoder}_{\mathrm{SD}}^{\mathrm{u}}(\mathrm{H}), \tag{16}$$
$$G^u = \mathrm{Encoder}_{\mathrm{Rec}}(H^u). \tag{17}$$

The motivation for designing such an architecture can be explained as follows, following analogies with the architectures in (Isik et al., 2016) and (Settle et al., 2018) where separation and recog-

nition are performed explicitly in separate steps: the first stage in Eq. (15) corresponds to a speech separation module which creates embedding vectors that can be used to distinguish between the multiple sources; the speaker-differentiating second stage in Eq. (16) uses the first stage's output to disentangle each speaker's speech content from the mixture, and prepare it for recognition; the final stage in Eq. (17) corresponds to an acoustic model that encodes the single-speaker speech for final decoding.

The decoder network computes the conditional probabilities for each speaker from the S outputs of the encoder network. In general, the decoder network uses the reference label R as a history to generate the attention weights during training, in a teacher-forcing fashion. However, in the above permutation-free training scheme, the reference label to be attributed to a particular output is not determined until the loss function is computed, so we here need to run the attention decoder for all reference labels. We thus need to consider the conditional probability of the decoder output variable $Y^{u,v}$ for each output G^u of the encoder network under the assumption that the reference label for that output is R^v:

$$p_{\text{att}}(Y^{u,v}|O) = \prod_n p_{\text{att}}(y_n^{u,v}|O, y_{1:n-1}^{u,v}), \quad (18)$$

$$c_n^{u,v}, a_n^{u,v} = \text{Attention}(a_{n-1}^{u,v}, e_n^{u,v}, G^u), \quad (19)$$

$$e_n^{u,v} = \text{Update}(e_{n-1}^{u,v}, c_{n-1}^{u,v}, r_{n-1}^{v}), \quad (20)$$

$$y_n^{u,v} \sim \text{Decoder}(c_n^{u,v}, r_{n-1}^{v}). \quad (21)$$

The final loss is then calculated by considering all permutations of the reference labels as follows:

$$\mathcal{L}_{\text{att}} = \min_{\pi \in \mathcal{P}} \sum_s \text{Loss}_{\text{att}}(Y^{s,\pi(s)}, R^{\pi(s)}). \quad (22)$$

3.3 Reduction of permutation cost

In order to reduce the computational cost, we fixed the permutation of the reference labels based on the minimization of the CTC loss alone, and used the same permutation for the attention mechanism as well. This is an advantage of using a joint CTC/attention based end-to-end speech recognition. Permutation is performed only for the CTC loss by assuming synchronous output where the permutation is decided by the output of CTC:

$$\hat{\pi} = \arg\min_{\pi \in \mathcal{P}} \sum_s \text{Loss}_{\text{ctc}}(Y^s, R^{\pi(s)}), \quad (23)$$

where Y^u is the output sequence variable corresponding to encoder output G^u. Attention-based decoding is then performed on the same hidden representations G^u, using teacher forcing with the labels determined by the permutation $\hat{\pi}$ that minimizes the CTC loss:

$$p_{\text{att}}(Y^{u,\hat{\pi}(u)}|O) = \prod_n p_{\text{att}}(y_n^{u,\hat{\pi}(u)}|O, y_{1:n-1}^{u,\hat{\pi}(u)}),$$

$$c_n^{u,\hat{\pi}(u)}, a_n^{u,\hat{\pi}(u)} = \text{Attention}(a_{n-1}^{u,\hat{\pi}(u)}, e_n^{u,\hat{\pi}(u)}, G^u),$$

$$e_n^{u,\hat{\pi}(u)} = \text{Update}(e_{n-1}^{u,\hat{\pi}(u)}, c_{n-1}^{u,\hat{\pi}(u)}, r_{n-1}^{\hat{\pi}(u)}),$$

$$y_n^{u,\hat{\pi}(u)} \sim \text{Decoder}(c_n^{u,\hat{\pi}(u)}, r_{n-1}^{\hat{\pi}(u)}).$$

This corresponds to the "permutation assignment" in Fig. 1. In contrast with Eq. (18), we only need to run the attention-based decoding once for each output G^u of the encoder network. The final loss is defined as the sum of two objective functions with interpolation λ:

$$\mathcal{L}_{\text{mtl}} = \lambda \mathcal{L}_{\text{ctc}} + (1-\lambda)\mathcal{L}_{\text{att}}, \quad (24)$$

$$\mathcal{L}_{\text{ctc}} = \sum_s \text{Loss}_{\text{ctc}}(Y^s, R^{\hat{\pi}(s)}), \quad (25)$$

$$\mathcal{L}_{\text{att}} = \sum_s \text{Loss}_{\text{att}}(Y^{s,\hat{\pi}(s)}, R^{\hat{\pi}(s)}). \quad (26)$$

At inference time, because both CTC and attention-based decoding are performed on the same encoder output G^u and should thus pertain to the same speaker, their scores can be incorporated as follows:

$$\hat{Y}^u = \arg\max_{Y^u} \big\{ \gamma \log p_{\text{ctc}}(Y^u|G^u) + (1-\gamma) \log p_{\text{att}}(Y^u|G^u) \big\}, \quad (27)$$

where $p_{\text{ctc}}(Y^u|G^u)$ and $p_{\text{att}}(Y^u|G^u)$ are obtained with the same encoder output G^u.

3.4 Promoting separation of hidden vectors

A single decoder network is used to output multiple label sequences by independently decoding the multiple hidden vectors generated by the encoder network. In order for the decoder to generate multiple different label sequences the encoder needs to generate sufficiently differentiated hidden vector sequences for each speaker. We propose to encourage this contrast among hidden vectors by introducing in the objective function a new term based on the negative symmetric Kullback-Leibler

(KL) divergence. In the particular case of two-speaker mixtures, we consider the following additional loss function:

$$\mathcal{L}_{\mathrm{KL}} = -\eta \sum_{l} \big\{ \mathrm{KL}(\bar{G}^1(l) \,||\, \bar{G}^2(l)) + \mathrm{KL}(\bar{G}^2(l) \,||\, \bar{G}^1(l)) \big\}, \quad (28)$$

where η is a small constant value, and $\bar{G}^u = (\mathrm{softmax}(G^u(l)) \,|\, l = 1, \ldots, L)$ is obtained from the hidden vector sequence G^u at the output of the recognition encoder $\mathrm{Encoder}_{\mathrm{Rec}}$ as in Fig. 1 by applying an additional frame-wise softmax operation in order to obtain a quantity amenable to a probability distribution.

3.5 Split of hidden vector for multiple hypotheses

Since the network maps acoustic features to label sequences directly, we consider various architectures to perform implicit separation and recognition effectively. As a baseline system, we use the concatenation of a VGG-motivated CNN network (Simonyan and Zisserman, 2014) (referred to as VGG) and a bi-directional long short-term memory (BLSTM) network as the encoder network. For the splitting point in the hidden vector computation, we consider two architectural variations as follows:

- Split by BLSTM: The hidden vector is split at the level of the BLSTM network. 1) the VGG network generates a single hidden vector H; 2) H is fed into S independent BLSTMs whose parameters are not shared with each other; 3) the output of each independent BLSTM $H^u, u=1, \ldots, S$, is further separately fed into a unique BLSTM, the same for all outputs. Each step corresponds to Eqs. (15), (16), and (17).
- Split by VGG: The hidden vector is split at the level of the VGG network. The number of filters at the last convolution layer is multiplied by the number of mixtures S in order to split the output into S hidden vectors (as in Eq. (16)). The layers prior to the last VGG layer correspond to the network in Eq. (15), while the subsequent BLSTM layers implement the network in (17).

4 Experiments

4.1 Experimental setup

We used English and Japanese speech corpora, WSJ (Wall street journal) (Consortium, 1994; Garofalo et al., 2007) and CSJ (Corpus of spontaneous Japanese) (Maekawa, 2003). To show the effectiveness of the proposed models, we generated mixed speech signals from these corpora to simulate single-channel overlapped multi-speaker recording, and evaluated the recognition performance using the mixed speech data. For WSJ, we used WSJ1 SI284 for training, Dev93 for development, and Eval92 for evaluation. For CSJ, we followed the Kaldi recipe (Moriya et al., 2015) and used the full set of academic and simulated presentations for training, and the standard test sets 1, 2, and 3 for evaluation.

Table 1: Duration (hours) of unmixed and mixed corpora. The mixed corpora are generated by Algorithm 1 in Section B of the supplementary material, using the training, development, and evaluation set respectively.

	TRAIN	DEV.	EVAL
WSJ (UNMIXED)	81.5	1.1	0.7
WSJ (MIXED)	98.5	1.3	0.8
CSJ (UNMIXED)	583.8	6.6	5.2
CSJ (MIXED)	826.9	9.1	7.5

We created new corpora by mixing two utterances with different speakers sampled from existing corpora. The detailed algorithm is presented in Section B of the supplementary material. The sampled pairs of two utterances are mixed at various signal-to-noise ratios (SNR) between 0 dB and 5 dB with a random starting point for the overlap. Duration of original unmixed and generated mixed corpora are summarized in Table 1.

4.1.1 Network architecture

As input feature, we used 80-dimensional log Mel filterbank coefficients with pitch features and their delta and delta delta features ($83 \times 3 = 249$-dimension) extracted using Kaldi tools (Povey et al., 2011). The input feature is normalized to zero mean and unit variance. As a baseline system, we used a stack of a 6-layer VGG network and a 7-layer BLSTM as the encoder network. Each BLSTM layer has 320 cells in each direction, and is followed by a linear projection layer with 320 units to combine the forward and backward LSTM outputs. The decoder network has an 1-layer LSTM with 320 cells. As described in Section 3.5, we adopted two types of encoder architectures for multi-speaker speech recognition. The network architectures are summarized in Table 2. The split-by-VGG network had speaker differentiating encoders with a convolution layer

Table 2: Network architectures for the encoder network. The number of layers is indicated in parentheses. $\text{Encoder}_{\text{Mix}}$, $\text{Encoder}^{\text{u}}_{\text{SD}}$, and $\text{Encoder}_{\text{Rec}}$ correspond to Eqs. (15), (16), and (17).

SPLIT BY	$\text{Encoder}_{\text{Mix}}$	$\text{Encoder}^{\text{u}}_{\text{SD}}$	$\text{Encoder}_{\text{Rec}}$
NO	VGG (6)	—	BLSTM (7)
VGG	VGG (4)	VGG (2)	BLSTM (7)
BLSTM	VGG (6)	BLSTM (2)	BLSTM (5)

(and the following maxpooling layer). The split-by-BLSTM network had speaker differentiating encoders with two BLSTM layers. The architectures were adjusted to have the same number of layers. We used characters as output labels. The number of characters for WSJ was set to 49 including alphabets and special tokens (e.g., characters for space and unknown). The number of characters for CSJ was set to 3,315 including Japanese Kanji/Hiragana/Katakana characters and special tokens.

4.1.2 Optimization

The network was initialized randomly from uniform distribution in the range -0.1 to 0.1. We used the AdaDelta algorithm (Zeiler, 2012) with gradient clipping (Pascanu et al., 2013) for optimization. We initialized the AdaDelta hyperparameters as $\rho = 0.95$ and $\epsilon = 1^{-8}$. ϵ is decayed by half when the loss on the development set degrades. The networks were implemented with Chainer (Tokui et al., 2015) and ChainerMN (Akiba et al., 2017). The optimization of the networks was done by synchronous data parallelism with 4 GPUs for WSJ and 8 GPUs for CSJ.

The networks were first trained on single-speaker speech, and then retrained with mixed speech. When training on unmixed speech, only one side of the network only (with a single speaker differentiating encoder) is optimized to output the label sequence of the single speaker. Note that only character labels are used, and there is no need for clean source reference corresponding to the mixed speech. When moving to mixed speech, the other speaker-differentiating encoders are initialized using the already trained one by copying the parameters with random perturbation, $w' = w \times (1 + \text{Uniform}(-0.1, 0.1))$ for each parameter w. The interpolation value λ for the multiple objectives in Eqs. (10) and (24) was set to 0.1 for WSJ and to 0.5 for CSJ. Lastly, the model is retrained with the additional negative KL divergence loss in Eq. (28) with $\eta = 0.1$.

Table 3: Evaluation of unmixed speech without multi-speaker training.

TASK	AVG.
WSJ	2.6
CSJ	7.8

4.1.3 Decoding

In the inference stage, we combined a pre-trained RNNLM (recurrent neural network language model) in parallel with the CTC and decoder network. Their label probabilities were linearly combined in the log domain during beam search to find the most likely hypothesis. For the WSJ task, we used both character and word level RNNLMs (Hori et al., 2017b), where the character model had a 1-layer LSTM with 800 cells and an output layer for 49 characters. The word model had a 1-layer LSTM with 1000 cells and an output layer for 20,000 words, i.e., the vocabulary size was 20,000. Both models were trained with the WSJ text corpus. For the CSJ task, we used a character level RNNLM (Hori et al., 2017c), which had a 1-layer LSTM with 1000 cells and an output layer for 3,315 characters. The model parameters were trained with the transcript of the training set in CSJ. We added language model probabilities with an interpolation factor of 0.6 for character-level RNNLM and 1.2 for word-level RNNLM.

The beam width for decoding was set to 20 in all the experiments. Interpolation γ in Eqs. (11) and (27) was set to 0.4 for WSJ and 0.5 for CSJ.

4.2 Results

4.2.1 Evaluation of unmixed speech

First, we examined the performance of the baseline joint CTC/attention-based encoder-decoder network with the original unmixed speech data. Table 3 shows the character error rates (CERs), where the baseline model showed 2.6% on WSJ and 7.8% on CSJ. Since the model was trained and evaluated with unmixed speech data, these CERs are considered lower bounds for the CERs in the succeeding experiments with mixed speech data.

4.2.2 Evaluation of mixed speech

Table 4 shows the CERs of the generated mixed speech from the WSJ corpus. The first column indicates the position of split as mentioned in Section 3.5. The second, third and forth columns indicate CERs of the high energy speaker (HIGH E. SPK.), the low energy speaker (LOW E. SPK.), and the average (AVG.), respectively. The baseline model has very high CERs because

Table 4: CER (%) of mixed speech for WSJ.

SPLIT	HIGH E. SPK.	LOW E. SPK.	AVG.
NO (BASELINE)	86.4	79.5	83.0
VGG	17.4	15.6	16.5
BLSTM	14.6	**13.3**	14.0
+ KL LOSS	**14.0**	**13.3**	**13.7**

Table 5: CER (%) of mixed speech for CSJ.

SPLIT	HIGH E. SPK.	LOW E. SPK.	AVG.
NO (BASELINE)	93.3	92.1	92.7
BLSTM	**11.0**	**18.8**	**14.9**

it was trained as a single-speaker speech recognizer without permutation-free training, and it can only output one hypothesis for each mixed speech. In this case, the CERs were calculated by duplicating the generated hypothesis and comparing the duplicated hypotheses with the corresponding references. The proposed models, i.e., split-by-VGG and split-by-BLSTM networks, obtained significantly lower CERs than the baseline CERs, the split-by-BLSTM model in particular achieving 14.0% CER. This is an 83.1% relative reduction from the baseline model. The CER was further reduced to 13.7% by retraining the split-by-BLSTM model with the negative KL loss, a 2.1% relative reduction from the network without retraining. This result implies that the proposed negative KL loss provides better separation by actively improving the contrast between the hidden vectors of each speaker. Examples of recognition results are shown in Section C of the supplementary material. Finally, we profiled the computation time for the permutations based on the decoder network and on CTC. Permutation based on CTC was 16.3 times faster than that based on the decoder network, in terms of the time required to determine the best match permutation given the encoder network's output in Eq. (17).

Table 5 shows the CERs for the mixed speech from the CSJ corpus. Similarly to the WSJ experiments, our proposed model significantly reduced the CER from the baseline, where the average CER was 14.9% and the reduction ratio from the baseline was 83.9%.

4.2.3 Visualization of hidden vectors

We show a visualization of the encoder networks outputs in Fig. 2 to illustrate the effect of the negative KL loss function. Principal component analysis (PCA) was applied to the hidden vectors on the vertical axis. Figures 2(a) and 2(b) show the hidden vectors generated by the split-by-BLSTM model without the negative KL divergence loss for an example mixture of two speakers. We can observe different activation patterns showing that the hidden vectors were successfully separated to the individual utterances in the mixed speech, although some activity from one speaker can be seen as leaking into the other. Figures 2(c) and 2(d) show the hidden vectors generated after retraining with the negative KL divergence loss. We can more clearly observe the different patterns and boundaries of activation and deactivation of hidden vectors. The negative KL loss appears to regularize the separation process, and even seems to help in finding the end-points of the speech.

4.2.4 Comparison with earlier work

We first compared the recognition performance with a hybrid (non end-to-end) system including DPCL-based speech separation and a Kaldi-based ASR system. It was evaluated under the same evaluation data and metric as in (Isik et al., 2016) based on the WSJ corpus. However, there are differences in the size of training data and the options in decoding step. Therefore, it is not a fully matched condition. Results are shown in Table 6. The word error rate (WER) reported in (Isik et al., 2016) is 30.8%, which was obtained with jointly trained DPCL and second-stage speech enhancement networks. The proposed end-to-end ASR gives an 8.4% relative reduction in WER even though our model does not require any explicit frame-level labels such as phonetic alignment, or clean signal reference, and does not use a phonetic lexicon for training. Although this is an unfair comparison, our purely end-to-end system outperformed a hybrid system for multi-speaker speech recognition.

Next, we compared our method with an end-to-end explicit separation and recognition network (Settle et al., 2018). We retrained our model previously trained on our WSJ-based corpus using the training data generated by Settle et al. (2018), because the direct optimization from scratch on their data caused poor recognition performance due to data size. Other experimental conditions are shared with the earlier work. Interestingly, our method showed comparable performance to the end-to-end explicit separation and recognition network, without having to pre-train using clean signal training references. It remains to be seen if this parity of performance holds in other tasks and conditions.

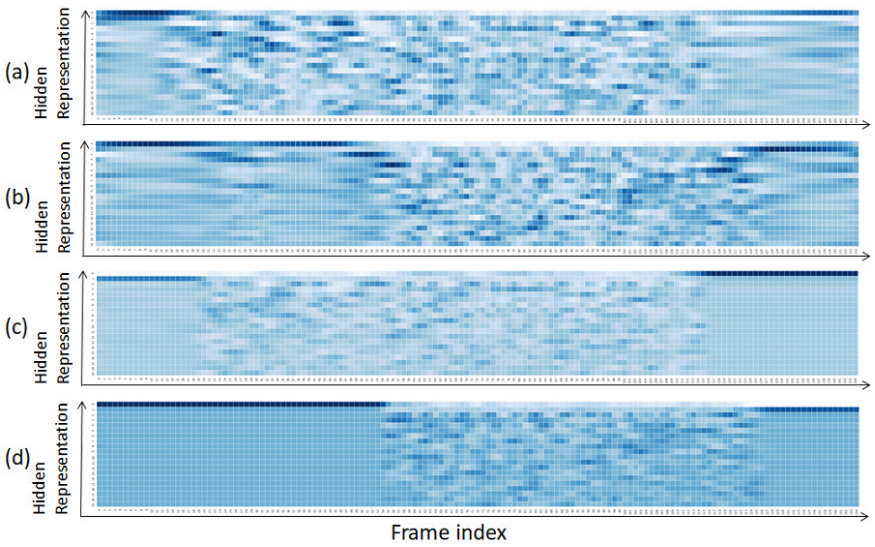

Figure 2: Visualization of the two hidden vector sequences at the output of the split-by-BLSTM encoder on a two-speaker mixture. (a,b): Generated by the model without the negative KL loss. (c,d): Generated by the model with the negative KL loss.

Table 6: Comparison with conventional approaches

METHOD	WER (%)
DPCL + ASR (ISIK ET AL., 2016)	30.8
Proposed end-to-end ASR	**28.2**
METHOD	CER (%)
END-TO-END DPCL + ASR (CHAR LM) (SETTLE ET AL., 2018)	**13.2**
Proposed end-to-end ASR (char LM)	14.0

5 Related work

Several previous works have considered an explicit two-step procedure (Hershey et al., 2016; Isik et al., 2016; Yu et al., 2017; Chen et al., 2017, 2018). In contrast with our work which uses a single objective function for ASR, they introduced an objective function to guide the separation of mixed speech.

Qian et al. (2017) trained a multi-speaker speech recognizer using permutation-free training without explicit objective function for separation. In contrast with our work which uses an end-to-end architecture, their objective function relies on a senone posterior probability obtained by aligning unmixed speech and text using a model trained as a recognizer for single-speaker speech. Compared with (Qian et al., 2017), our method directly maps a speech mixture to multiple character sequences and eliminates the need for the corresponding isolated speech sources for training.

6 Conclusions

In this paper, we proposed an end-to-end multi-speaker speech recognizer based on permutation-free training and a new objective function promoting the separation of hidden vectors in order to generate multiple hypotheses. In an encoder-decoder network framework, teacher forcing at the decoder network under multiple references increases computational cost if implemented naively. We avoided this problem by employing a joint CTC/attention-based encoder-decoder network.

Experimental results showed that the model is able to directly convert an input speech mixture into multiple label sequences under the end-to-end framework without the need for any explicit intermediate representation including phonetic alignment information or pairwise unmixed speech. We also compared our model with a method based on explicit separation using deep clustering, and showed comparable result. Future work includes data collection and evaluation in a real world scenario since the data used in our experiments are simulated mixed speech, which is already extremely challenging but still leaves some acoustic aspects, such as Lombard effects and real room impulse responses, that need to be alleviated for further performance improvement. In addition, further study is required in terms of increasing the number of speakers that can be simultaneously recognized, and further comparison with the separation-based approach.

References

Takuya Akiba, Keisuke Fukuda, and Shuji Suzuki. 2017. ChainerMN: Scalable Distributed Deep Learning Framework. In *Proceedings of Workshop on ML Systems in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*.

Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. 2016. End-to-end attention-based large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949.

Zhehuai Chen, Jasha Droppo, Jinyu Li, and Wayne Xiong. 2018. Progressive joint modeling in unsupervised single-channel overlapped speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(1):184–196.

Zhuo Chen, Yi Luo, and Nima Mesgarani. 2017. Deep attractor network for single-microphone speaker separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 246–250.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–585.

Linguistic Data Consortium. 1994. CSR-II (wsj1) complete. *Linguistic Data Consortium, Philadelphia*, LDC94S13A.

Martin Cooke, John R Hershey, and Steven J Rennie. 2009. Monaural speech separation and recognition challenge. *Computer Speech and Language*, 24(1):1–15.

John Garofalo, David Graff, Doug Paul, and David Pallett. 2007. CSR-I (wsj0) complete. *Linguistic Data Consortium, Philadelphia*, LDC93S6A.

Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. 2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine learning (ICML)*, pages 369–376.

John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. 2016. Deep clustering: Discriminative embeddings for segmentation and separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 31–35.

Takaaki Hori, Shinji Watanabe, and John R Hershey. 2017a. Joint CTC/attention decoding for end-to-end speech recognition. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL): Human Language Technologies: long papers*.

Takaaki Hori, Shinji Watanabe, and John R Hershey. 2017b. Multi-level language modeling and decoding for open vocabulary end-to-end speech recognition. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Takaaki Hori, Shinji Watanabe, Yu Zhang, and Chan William. 2017c. Advances in joint CTC-Attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM. In *Interspeech*, pages 949–953.

Yusuf Isik, Jonathan Le Roux, Zhuo Chen, Shinji Watanabe, and John R. Hershey. 2016. Single-channel multi-speaker separation using deep clustering. In *Proc. Interspeech*, pages 545–549.

Suyoun Kim, Takaaki Hori, and Shinji Watanabe. 2017. Joint CTC-attention based end-to-end speech recognition using multi-task learning. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4835–4839.

Morten Kolbæk, Dong Yu, Zheng-Hua Tan, and Jesper Jensen. 2017. Multitalker speech separation with utterance-level permutation invariant training of deep recurrent neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(10):1901–1913.

Kikuo Maekawa. 2003. Corpus of Spontaneous Japanese: Its design and evaluation. In *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition*.

Takafumi Moriya, Takahiro Shinozaki, and Shinji Watanabe. 2015. Kaldi recipe for Japanese spontaneous speech recognition and its evaluation. In *Autumn Meeting of ASJ*, 3-Q-7.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *International Conference on Machine Learning (ICML)*, pages 1310–1318.

Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, Jan Silovsky, Georg Stemmer, and Karel Vesely. 2011. The kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*.

Yanmin Qian, Xuankai Chang, and Dong Yu. 2017. Single-channel multi-talker speech recognition with permutation invariant training. *arXiv preprint arXiv:1707.06527*.

Shane Settle, Jonathan Le Roux, Takaaki Hori, Shinji Watanabe, and John R. Hershey. 2018. End-to-end multi-speaker speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4819–4823.

Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Seiya Tokui, Kenta Oono, Shohei Hido, and Justin Clayton. 2015. Chainer: a next-generation open source framework for deep learning. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in NIPS*.

Dong Yu, Morten Kolbk, Zheng-Hua Tan, and Jesper Jensen. 2017. Permutation invariant training of deep models for speaker-independent multi-talker speech separation. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 241–245.

Matthew D Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Supplementary material for "A Purely End-to-end System for Multi-speaker Speech Recognition"

Hiroshi Seki[1,2]*, Takaaki Hori[1], Shinji Watanabe[3], Jonathan Le Roux[1], John R. Hershey[1]
[1]Mitsubishi Electric Research Laboratories (MERL)
[2]Toyohashi University of Technology
[3]Johns Hopkins University

1 Architecture of the encoder-decoder network

In this section, we describe the details of the baseline encoder-decoder network which is further extended for permutation-free training. The encoder network consists of a VGG network and bi-directional long short-term memory (BLSTM) layers. The VGG network has the following 6-layer CNN architecture at the bottom of the encoder network:

Convolution (# in = 3, # out = 64, filter = 3×3)
Convolution (# in = 64, # out = 64, filter = 3×3)
MaxPooling (patch = 2×2, stride = 2×2)
Convolution (# in = 64, # out = 128, filter = 3×3)
Convolution (# in=128, # out=128, filter=3×3)
MaxPooling (patch = 2×2, stride = 2×2)

The first 3 channels are static, delta, and delta delta features. Multiple BLSTM layers with projection layer $\mathrm{Lin}(\cdot)$ are stacked after the VGG network. We defined one BLSTM layer as the concatenation of a forward LSTM $\overrightarrow{\mathrm{LSTM}}(\cdot)$ and a backward LSTM $\overleftarrow{\mathrm{LSTM}}(\cdot)$:

$$\overrightarrow{H} = \overrightarrow{\mathrm{LSTM}}(\cdot) \tag{29}$$
$$\overleftarrow{H} = \overleftarrow{\mathrm{LSTM}}(\cdot) \tag{30}$$
$$H = [\mathrm{Lin}(\overrightarrow{H}); \mathrm{Lin}(\overleftarrow{H})], \tag{31}$$

When the VGG network and the multiple BLSTM layers are represented as $\mathrm{VGG}(\cdot)$ and $\mathrm{BLSTM}(\cdot)$, the encoder network in Eq. (2) maps the input feature vector O to internal representation H as follows:

$$H = \mathrm{Encoder}(O) = \mathrm{BLSTM}(\mathrm{VGG}(O)) \tag{32}$$

The decoder network sequentially generates the n-th label y_n by taking the context vector c_n and the label history $y_{1:n-1}$:

$$y_n \sim \mathrm{Decoder}(c_n, y_{n-1}). \tag{33}$$

The context vector is calculated in an location based attention mechanism (Chorowski et al., 2015) which weights and sums the C-dimensional sequence of representation $H = (h_l \in \mathbb{R}^C | l = 1, \ldots, L)$ with attention weight $a_{n,l}$:

$$c_n = \mathrm{Attention}(a_{n-1}, e_n, H), \tag{34}$$
$$\triangleq \sum_{l=1}^{L} a_{n,l} h_l. \tag{35}$$

The location based attention mechanism defines the weights $a_{n,l}$ as follows:

$$a_{n,l} = \frac{\exp(\alpha k_{n,l})}{\sum_{l=1}^{L} \exp(\alpha k_{n,l})}, \tag{36}$$
$$k_{n,l} = w^{\mathrm{T}} \tanh(V^{\mathrm{E}} e_{n-1} + V^{\mathrm{H}} h_l + V^{\mathrm{F}} f_{n,l} + b), \tag{37}$$
$$f_n = F * a_{n-1}, \tag{38}$$

where w, V^E, V^H, V^F, b, F are tunable parameters, α is a constant value called inverse temperature, and $*$ is the convolution operation. We used 10 convolution filters of width 200, and set α to 2. The introduction of f_n makes the attention mechanism take into account the previous alignment information. The hidden state e is updated recursively by an updating LSTM function:

$$e_n = \mathrm{Update}(e_{n-1}, c_{n-1}, y_{n-1}), \tag{39}$$
$$\triangleq \mathrm{LSTM}(\mathrm{Lin}(e_{n-1}) + \mathrm{Lin}(c_{n-1}) + \mathrm{Emb}(y_{n-1})), \tag{40}$$

where $\mathrm{Emb}(\cdot)$ is an embedding function.

*This work was done while H. Seki, Ph.D. candidate at Toyohashi University of Technology, Japan, was an intern at MERL.

Table 1: Examples of recognition results. Errors are emphasized as capital letter. "_" is a space character, and a special token "*" is inserted to pad deletion errors.

(1) Model w/ permutation-free training (CER of HYP1: 12.8%, HYP2: 0.9%)
HYP1: t h e _ s h u t t l e _ * * * I S _ I N _ t h e _ f i r s t _ t H E _ l i f E _ o * f _ s i n c e _ t h e _ n i n e t e e n _ e i g h t y _ s i x _ c h a l l e n g e r _ e x p l o s i o n
REF1: t h e _ s h u t t l e _ W O U L D _ B E _ t h e _ f i r s t _ t * O _ l i f T _ o F f _ s i n c e _ t h e _ n i n e t e e n _ e i g h t y _ s i x _ c h a l l e n g e r _ e x p l o s i o n
HYP2: t h e _ e x p a n d e d _ r e c a l l _ w a s _ d i s c l o s e d _ a t _ a _ m e e t i n g _ w i t h _ n . _ r . _ c . _ o f f i c i a l s _ a t _ a n _ a g e n c y _ o f f i c e _ o u t s i d e _ c h i c a g o
REF2: t h e _ e x p a n d e d _ r e c a l l _ w a s _ d i s c l o s e d _ a t _ a _ m e e t i n g _ w i t h _ n . _ r . _ c . _ o f f i c i a l s _ a t _ a n _ a g e n c y _ o f f i c e _ o u t s i d e _ c h i c a g o
(2) Model w/ permutation-free training (CER of HYP1: 91.7%, HYP2: 38.9%)
HYP1: I T _ W A S _ L a s t _ r * A I S e * D _ * I N _ J U N E _ N I N E t E e N _ e * I G h T Y _ f I V e _ T O _ * T H I R T Y
REF1: * * * * * * * * a s t * r O N O M e R S _ S A Y _ T H A T _ * * * * t H e * _ e A R T h ' S _ f A T e _ I S _ S E A L E D
HYP2: * * * * a N D _ * * s t * r O N G e R S _ S A Y _ T H A T _ * * * * t H e * _ e * A R t H _ f A T e _ I S _ t o _ f o r t y _ f i v e _ d o l l a r s _ f r o m _ t h i r t y _ f i v e _ d o l l a r s
REF2: I T _ W a * S _ L A s t _ r A I S e * D _ * I N _ J U N E _ N I N E t E e N _ e I G H t Y _ f I V e * * * _ t o _ f o r t y _ f i v e _ d o l l a r s _ f r o m _ t h i r t y _ f i v e _ d o l l a r s

Algorithm 1 Generation of multi speaker speech dataset

$n_{\text{reuse}} \Leftarrow$ maximum number of times same utterance can be used.
$U \Leftarrow$ utterance set of the corpora.
$C_k \Leftarrow n_{\text{reuse}}$ for each utterance $U_k \in U$
for $U_k \in U$ **do**
  $P(U_k) = C_k \,/\, \sum_l C_l$
end for
for U_i in U **do**
  Sample utterance U_j from $P(U)$ while ensuring speakers of U_i and U_j are different.
  Mix utterances U_i and U_j
  if $C_j > 0$ **then**
    $C_j = C_j - 1$
    for $U_k \Leftarrow U$ **do**
      $P(U_k) = C_k \,/\, \sum_l C_l$
    end for
  end if
end for

2 Generation of mixed speech

Each utterance of the corpus is mixed with a randomly selected utterance with the probability, $P(U_k)$, that moderates over-selection of specific utterances. $P(U_k)$ is calculated in the first for-loop as a uniform probability. All utterances are used as one side of the mixture, and another side is sampled from the distribution $P(U_k)$ in the second for-loop. The selected pairs of utterances are mixed at various signal-to-noise ratios (SNR) between 0 dB and 5 dB. We randomized the starting point of the overlap by padding the shorter utterance with silence whose duration is sampled from the uniform distribution within the length difference between the two utterances. Therefore, the duration of the mixed utterance is equal to that of the longer utterance among the unmixed speech. After the generation of the mixed speech, the count of selected utterances C_j is decremented to prevent of over-selection. All counts C are set to n_{reuse}, and we used $n_{\text{reuse}} = 3$.

3 Examples of recognition results and error analysis

Table 1 shows examples of recognition result. The first example (1) is one which accounts for a large portion of the evaluation set. The SNR of the HYP1 is -1.55 db and that of HYP2 is 1.55 dB. The network generates multiple hypotheses with a few substitution and deletion errors, but without any overlapped and swapped words. The second example (2) is one which leads to performance reduction. We can see that the network makes errors when there is a large difference in length between the two sequences. The word "thirty" of HYP2 is injected in HYP1, and there are deletion errors in

HYP2. We added a negative KL divergence loss to ease such kind of errors. However, there is further room to reduce error by making unshared modules more cooperative.

References

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in Neural Information Processing Systems (NIPS)*, pages 577–585.