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

Multi-level Language Modeling and Decoding for Open Vocabulary End-to-End Speech Recognition

Hori, T.; Watanabe, S.; Hershey, J.R.

TR2017-181 December 2017

Abstract

We propose a combination of character-based and word-based language models in an end-to-end automatic speech recognition (ASR) architecture. In our prior work, we combined a character-based LSTM RNN-LM with a hybrid attention/connectionist temporal classification (CTC) architecture. The character LMs improved recognition accuracy to rival state-of-the-art DNN/HMM systems in Japanese and Mandarin Chinese tasks. Although a characterbased architecture can provide for open vocabulary recognition, the character-based LMs generally under-perform relative to word LMs for languages such as English with a small alphabet, because of the difficulty of modeling linguistic constraints across long sequences of characters. This paper presents a novel method for end-to-end ASR decoding with LMs at both the character and word level. Hypotheses are first scored with the character-based LM until a word boundary is encountered. Known words are then re-scored using the word-based LM, while the character-based LM provides for out-of-vocabulary scores. In a standard Wall Street Journal (WSJ) task, we achieved 5.6 % WER for the Eval'92 test set using only the SI284 training set and WSJ text data, which is the best score reported for end-to-end ASR systems on this benchmark.

IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)

MULTI-LEVEL LANGUAGE MODELING AND DECODING FOR OPEN VOCABULARY END-TO-END SPEECH RECOGNITION

Takaaki Hori, Shinji Watanabe, John R. Hershey

Mitsubishi Electric Research Laboratories (MERL)
201 Broadway, Cambridge, MA 02139, USA
{thori, hershey}@merl.com, shinjiw@ieee.org

ABSTRACT

We propose a combination of character-based and word-based language models in an end-to-end automatic speech recognition (ASR) architecture. In our prior work, we combined a character-based LSTM RNN-LM with a hybrid attention/connectionist temporal classification (CTC) architecture. The character LMs improved recognition accuracy to rival state-of-the-art DNN/HMM systems in Japanese and Mandarin Chinese tasks. Although a character-based architecture can provide for open vocabulary recognition, the character-based LMs generally under-perform relative to word LMs for languages such as English with a small alphabet, because of the difficulty of modeling linguistic constraints across long sequences of characters. This paper presents a novel method for end-to-end ASR decoding with LMs at both the character and word level. Hypotheses are first scored with the character-based LM until a word boundary is encountered. Known words are then re-scored using the word-based LM, while the character-based LM provides for out-of-vocabulary scores. In a standard Wall Street Journal (WSJ) task, we achieved 5.6 % WER for the Eval'92 test set using only the SI284 training set and WSJ text data, which is the best score reported for end-to-end ASR systems on this benchmark.

Index Terms— End-to-end speech recognition, language modeling, decoding, connectionist temporal classification, attention decoder

1. INTRODUCTION

Automatic speech recognition (ASR) is currently a mature set of widely-deployed technologies that enable successful user interface applications such as voice search [1]. However, current systems lean heavily on the scaffolding of complicated legacy architectures that grew up around traditional techniques, including hidden Markov models (HMMs), Gaussian mixture models (GMMs), hybrid HMM/deep neural network (DNN) systems, and sequence discriminative training methods [2]. These systems also require hand-made pronunciation dictionaries based on linguistic assumptions, extra training steps to derive context-dependent phonetic models, and text preprocessing such as tokenization for languages without explicit word boundaries. Consequently, it is quite difficult for non-experts to develop ASR systems for new applications, especially for new languages.

End-to-end ASR has the goal of simplifying the above module-based architecture into a single-network architecture within a deep learning framework, in order to address these issues. End-to-end ASR methods typically rely only on paired acoustic and language data without linguistic knowledge, and train the model with a single algorithm. Therefore, the approach makes it feasible to build ASR systems without expert knowledge.

There are two major types of end-to-end architectures for ASR: connectionist temporal classification (CTC) and attention-based methods, both of which are based on recurrent neural networks. Connectionist temporal classification (CTC) uses a system in which the acoustic model network emits scores not only for each output symbol, but also for extra *blank*, or neutral, symbols that act as filler between output symbols. These neutral symbols allow the network to implicitly align the input sequence of acoustic features to the much shorter sequence of output symbols. The Markov conditional independence assumption allows dynamic programming to search for the best scoring sequential alignment by absorbing the neutral symbols, while incorporating language modeling scores. Attention-based methods, in contrast, use a neural attention mechanism to explicitly estimate an alignment between input acoustic frames and recognized output symbols within the network [3, 4, 5, 6, 7]. Attention-based methods do not rely on the frame-level Markov assumption, and hence they can model a more general class of statistical dependency between input and output sequences. However, without special constraints, the attention mechanism is too flexible in the sense that it allows arbitrarily non-sequential alignments. Although such alignments may be appropriate for applications such as machine translation where word order may differ arbitrarily between languages, they are inappropriate for speech recognition, where the alignment between spoken and written speech is essentially monotonic.

To avoid the shortcomings of both models, while retaining their advantages, we have proposed a hybrid attention/CTC model for end-to-end ASR training [8]. The hybrid model attaches a CTC mechanism and objective to the attention model during training. The CTC objective acts as a regularization term that encourages the alignments in the attention model to be monotonic, and improves its performance. Moreover, we proposed using the combined CTC and attention-based objective for decoding, and demonstrated that a one-pass search strategy can incorporate both objectives and efficiently find better aligned hypotheses with an improved accuracy [9].

In our previous work [10], we have also shown that integrating a recurrent neural network language model (RNN-LM) with the decoder network significantly improves the recognition accuracy in Japanese and Mandarin Chinese tasks, reaching a comparable or higher accuracy to those of state-of-the-art DNN/HMM systems. For example, our result on the Chinese HKUST task is 28.0 % word error rate (WER), which is slightly better than 28.2% given by a state-of-the-art time-delay neural network (TDNN) system with lattice-free maximum muMMI training [11]. Since the Japanese and Chinese systems were designed to output character sequences, the RNN-LM

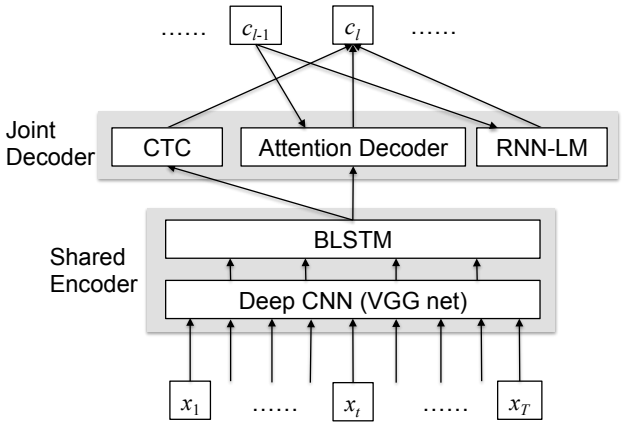

Fig. 1. Hybrid attention/CTC network with LM extension: the shared encoder contains a VGG net followed by BLSTM layers and trained by both CTC and attention model objectives simultaneously. The joint decoder predicts an output label sequence by the CTC, attention decoder and RNN-LM.

was also designed as a character-based LM, and effectively combined with the decoder network to jointly predict the next character.

A character-based architecture achieves high-accuracy ASR for languages with a large set of characters such as Japanese and Chinese. It also enables open vocabulary ASR, in contrast to word-based architectures, which suffer from the out-of-vocabulary (OOV) problem. However, the character-based LMs generally under-perform relative to word LMs for languages with a small alphabet, such as English, because of the difficulty of modeling linguistic constraints across long sequences of characters. To overcome this problem, we present a novel method for end-to-end ASR decoding with LMs at both the character and word level. Hypotheses are first scored with the character-based LM until a word boundary is encountered. Known words are then re-scored using the word-based LM, while the character-based LM provides for out-of-vocabulary scores. This approach exploits the benefits of both character and word level architectures, and enables high-accuracy open-vocabulary end-to-end ASR. We evaluate the proposed method with a standard Wall Street Journal (WSJ) task and show significant improvement by the proposed method.

2. HYBRID ATTENTION/CTC ARCHITECTURE

In this section, we explain the hybrid attention/CTC framework, which utilizes both benefits of CTC and attention during training and decoding [8, 9].

2.1. Network architecture

Figure 1 shows the latest architecture of our CTC-attention network [10]. The encoder has deep convolutional neural network (CNN) layers with the VGG net architecture [12], which are followed by stacked bidirectional long short-term memory (BLSTM) layers. The decoder network has a CTC network, an attention decoder network, and an RNN-LM, which jointly predict the next label. Given input sequence $X = x_1, \ldots, x_T$, the encoder network accepts X and outputs hidden vector sequence $\mathbf{H} = \mathbf{h}_1, \ldots, \mathbf{h}_{T'}$, where $T' = T/4$ by using two max-pooling steps in the deep CNN. The decoder network iteratively predicts a single label c_l based on the hidden vectors $\mathbf{H}$ and the label context $c_1, \ldots, c_{l-1}$, and generates L-length label sequence $C = \{c_l \in \mathcal{U} | l = 1, \cdots, L\}$, where $\mathcal{U}$ is a set of labels. In this work, we assume $\mathcal{U}$ is a set of distinct characters or alphabet of the target language.

2.2. Multi-task learning

In [8], we used the CTC objective function as an auxiliary task to train the attention model encoder based on the multi-task learning (MTL) framework. This approach substantially reduced irregular alignments during training and inference, and provided improved performance in several end-to-end ASR tasks.

CTC [13] is a latent variable model that monotonically maps an input sequence to an output sequence of shorter length. We assume here that the model outputs letter sequence C. CTC introduces framewise letter sequence with an additional "blank" symbol $Z = \{z_t \in \mathcal{U} \cup \text{blank} | t = 1, \cdots, T\}$. By using conditional independence assumptions, the posterior distribution $p(C|X)$ is factorized as follows:

$$p(C|X) \approx \underbrace{\sum_{Z} \prod_{t} p(z_t|z_{t-1}, C) p(z_t|X)\, p(C)}_{\triangleq p_{\text{ctc}}(C|X)} \tag{1}$$

As shown in Eq. (1), CTC has three distribution components by the Bayes theorem similar to the conventional hybrid ASR case, i.e., framewise posterior distribution $p(z_t|X)$, transition probability $p(z_t|z_{t-1}, C)$, and letter-based language model $p(C)$. We also define the CTC objective function $p_{\text{ctc}}(C|X)$ used in the later formulation.

The framewise posterior distribution $p(z_t|X)$ is conditioned on all inputs X, and it is quite natural to be modeled by using BLSTMs:

$$p(z_t|X) = \text{Softmax}(\text{Lin}(\mathbf{h}_t)) \tag{2}$$
$$\mathbf{h}_t = \text{BLSTM}(\text{CNN}(X)). \tag{3}$$

Softmax$(\cdot)$ is a softmax activation function, and Lin$(\cdot)$ is a linear layer to convert hidden vector $\mathbf{h}_t$ to a $(|\mathcal{U}| + 1)$ dimensional vector ($+1$ means a blank symbol introduced in CTC).

Although Eq. (1) has to deal with a summation over all possible Z, we can efficiently compute this marginalization by using dynamic programming thanks to the Markov property. In summary, although CTC and hybrid systems are similar to each other due to conditional independence assumptions, CTC does not require pronunciation dictionaries and omits an HMM/GMM construction step.

Compared with CTC approaches, the attention-based approach does not make any conditional independence assumptions, and directly estimates the posterior $p(C|X)$ based on the chain rule:

$$p(C|X) = \underbrace{\prod_{l} p(c_l|c_1, \cdots, c_{l-1}, X)}_{\triangleq p_{\text{att}}(C|X)}, \tag{4}$$

where $p_{\text{att}}(C|X)$ is an attention-based objective function. $p(c_l|c_1, \cdots, c_{l-1}, X)$ is obtained by

$$p(c_l|c_1, \cdots, c_{l-1}, X) = \text{Decoder}(\mathbf{r}_l, \mathbf{q}_{l-1}, c_{l-1}) \tag{5}$$
$$\mathbf{h}_t = \text{Encoder}(X) \tag{6}$$
$$a_{lt} = \text{Attention}(\{a_{l-1}\}_t, \mathbf{q}_{l-1}, \mathbf{h}_t) \tag{7}$$
$$\mathbf{r}_l = \sum_t a_{lt} \mathbf{h}_t. \tag{8}$$

Eq. (6) converts input feature vectors X into a framewise hidden vector $\mathbf{h}_t$ in an encoder network based on BLSTM, i.e., Encoder$(X) \triangleq$ BLSTM(CNN(X)). Attention$(\cdot)$ in Eq. (7) is based on a content-based attention mechanism with convolutional features, as described in [14]. a_{lt} is an attention weight, and represents a soft alignment of hidden vector $\mathbf{h}_t$ for each output c_l based on the weighted summation of hidden vectors to form letter-wise hidden vector $\mathbf{r}_l$ in Eq. (8). A decoder network is another recurrent network conditioned on previous output c_{l-1} and hidden vector $\mathbf{q}_{l-1}$, similar to RNN-LM, in addition to letter-wise hidden vector $\mathbf{r}_l$. We use Decoder$(\cdot) \triangleq$ Softmax(Lin(LSTM$(\cdot)$)).

Attention-based ASR does not explicitly separate each module, but it implicitly combines acoustic models, lexicon, and language models as encoder, attention, and decoder networks, which can be jointly trained as a single deep neural network. Compared with CTC, attention-based models make predictions conditioned on all the previous predictions, and thus can learn language. However, the cost of using an explicit alignment without monotonic constraints means the alignment can become impaired.

The hybrid attention/CTC network shares the same CNN/BLSTM encoder with CTC and attention decoder networks. Unlike the solitary attention model, the forward-backward algorithm of CTC can enforce monotonic alignment between speech and label sequences during training. That is, rather than solely depending on the data-driven attention mechanism to estimate the desired alignments in long sequences, the forward-backward algorithm in CTC helps to speed up the process of estimating the desired alignment. The objective to be maximized is a logarithmic linear combination of the CTC and attention objectives, i.e., $p_{\text{ctc}}(C|X)$ in Eq. (1) and $p_{\text{att}}(C|X)$ in Eq. (4):

$$\mathcal{L}_{\text{MTL}} = \lambda \log p_{\text{ctc}}(C|X) + (1-\lambda) \log p_{\text{att}}(C|X), \tag{9}$$

with a tunable parameter $\lambda : 0 \leq \lambda \leq 1$.

2.3. Joint decoding

It has already been shown that the CTC objective helps guide the attention model during training to be more robust and effective, and produce a better model for speech recognition [8]. Furthermore, we can use CTC predictions also in the decoding process [9, 10].

The inference step of attention-based speech recognition is performed by output-label synchronous decoding with a beam search. However, we take the CTC probabilities into account to find a better aligned hypothesis to the input speech, i.e., the decoder finds the most probable character sequence $\hat{C}$ given speech input X, according to

$$\hat{C} = \arg\max_{C \in \mathcal{U}^*} \left\{ \lambda \log p_{\text{ctc}}(C|X) + (1-\lambda) \log p_{\text{att}}(C|X) \right\}. \tag{10}$$

In the beam search process, the decoder computes a score of each partial hypothesis, which is defined as the log probability of the hypothesized character sequence. With the attention model, the score for hypothesis h can be computed recursively as

$$\alpha_{\text{att}}(h) = \alpha_{\text{att}}(g) + \log p(c|g, X), \tag{11}$$

where g is an existing partial hypothesis, and c is a character label appended to g to generate h, i.e., $h = g \cdot c$. The score for h is obtained as the addition of the original score $\alpha(g)$ and the conditional log probability given by the attention decoder in Eq. (5). During the beam search, the number of partial hypotheses for each length is limited to a predefined number, called a *beam width*, to exclude hypotheses with relatively low scores, which dramatically improves the search efficiency.

However, it is non-trivial to combine CTC and attention-based scores in the beam search, because the attention decoder operates character-by-character while CTC decodes at the frame rate. To incorporate CTC probabilities in the score, we compute the probability of each partial hypothesis based on the CTC prefix probability [15] defined as the cumulative probability of all label sequences that have h as their prefix:

$$p(h, \dots | X) = \sum_{\nu \in (\mathcal{U} \cup \{\texttt{<eos>}\})^+} P(h \cdot \nu | X), \tag{12}$$

and we use the CTC score as

$$\alpha_{\text{ctc}}(h) \triangleq \log p(h, \dots | X), \tag{13}$$

where ν represents all possible label sequences except the empty string, and <eos> indicates the end of sentence. The CTC score cannot be obtained recursively as in Eq. (11), but it can be computed efficiently by keeping the forward probabilities over input frames for each partial hypothesis. Then the hypothesis score is computed as

$$\alpha(h) = \lambda \alpha_{\text{ctc}}(h) + (1-\lambda) \alpha_{\text{att}}(h). \tag{14}$$

2.4. RNN-LM integration

We combine an RNN-LM network in parallel with the attention decoder, which can be trained separately or jointly, where the RNN-LM is trained with character sequences without word-level knowledge. Although the attention decoder implicitly includes a language model as in Eq. (5), we aim at introducing language model states purely dependent on the output label sequence in the decoder, which potentially brings a complementary effect.

As shown in Fig. 1, the RNN-LM probabilities are used to predict the output label jointly with the decoder network. The RNN-LM information is combined at the logits level or log probability level. If we use a pre-trained RNN-LM without any joint training, we need a scaling factor on the log probabilities. If we train the model jointly with the other networks, we may combine their pre-activations before the softmax without a scaling factor as this is learnt.

In this work, we use only pre-trained RNN-LMs without joint training, because we assume a large text corpus is available to train the RNN-LMs and joint training rarely improves recognition accuracy in such conditions.

We compute the decoding objective including an RNN-LM as

$$\begin{aligned} \hat{C} = \arg\max_{C \in \mathcal{U}^*} \{ & \lambda \log p_{\text{ctc}}(C|X) + (1-\lambda) \log p_{\text{att}}(C|X) \\ & + \gamma \log p_{\text{lm}}(C) \}, \end{aligned} \tag{15}$$

where the LM probability $p_{\text{lm}}(C)$ is added with scaling factor γ in the log probability domain, and the probability is computed as

$$p_{\text{lm}}(C) = \prod_{i=1}^{L} p_{\text{lm}}(c_i | c_1, \dots, c_{i-1}) \tag{16}$$

using the RNN-LM.

3. DECODING WITH MULTI-LEVEL LMS

In this section, we incorporate a word-level LM into our hybrid attention/CTC based ASR. In most end-to-end ASR systems, a finite lexicon and an N-gram language model are compiled into a Weighted

Finite-State Transducer (WFST), and used for decoding [16, 17]. The WFST framework efficiently handles frame-synchronous or label-synchronous decoding with the optimized search network and reduces the word error rate [18, 19]. However, this approach is not suitable for RNN-LMs because an RNN-LM cannot be represented as a static state network. In addition, we lose the benefit of the open-vocabulary property given by the character-based architecture.

In this paper, we extend the character-based decoding to enable open-vocabulary end-to-end ASR with a word-level RNN-LM. We consider that the character-based systems can predict space characters between words as well as letters within the word. Note that the space character has an actual character code, which is different from the CTC's blank symbol. With the space characters, it is possible to deterministically map any character sequence to a word sequence, e.g., character sequence

```
a <space> c a t <space> e a t s
```

is mapped to a unique word sequence

```
a cat eats
```

where <space> formally represents the space character. Accordingly, only when the decoder hypothesizes a space character, it computes the probability of the last word using the word-level RNN-LM and simply accumulates it to the hypothesis score. No special treatment is necessary for different types of homonyms: words with the same spelling but different pronunciation are handled in a context-dependent way by the word language model, whereas words with the same pronunciation but different spellings are automatically handled as different word hypotheses in the beam search. Similarly, ambiguous word segmentations are automatically handled as different decoding hypotheses.

The proposed mechanism can be implemented by modifying the character-level LM probabilities as follows. Let $\mathcal{V}$ be the vocabulary of the word-level RNN-LM and be including an abstract symbol of OOV word such as <UNK>. We compute the conditional character probabilities in Eq. (16) as

$$p_{\text{lm}}(c|g) = \begin{cases} \frac{p_{\text{wlm}}(w_g|\psi_g)}{p_{\text{clm}}(w_g|\psi_g)} & \text{if } c \in S, w_g \in \mathcal{V} \\ p_{\text{wlm}}(\texttt{<UNK>}|\psi_g)\tilde{\beta} & \text{if } c \in S, w_g \notin \mathcal{V} \\ p_{\text{clm}}(c|g) & \text{otherwise} \end{cases} \tag{17}$$

where S denotes a set of labels that indicate the end of word, i.e., $S = \{\texttt{<space>}, \texttt{<eos>}\}$, w_g is the last word of the character sequence g, and ψ_g is the word-level history, which is the word sequence corresponding to g excluding w_g. For the above example, g, w_g, and ψ_g are set as

$$\begin{aligned} g &= \texttt{a}, \texttt{<space>}, \texttt{c}, \texttt{a}, \texttt{t}, \texttt{<space>}, \texttt{e}, \texttt{a}, \texttt{t}, \texttt{s} \\ w_g &= \texttt{eats} \\ \psi_g &= \texttt{a}, \texttt{cat}. \end{aligned}$$

$\tilde{\beta}$ is a scaling factor used to adjust the probabilities for OOV words.

The first condition on the right-hand side of Eq. (17) is used when the character c indicates the end of the previous word. In this case, the word-level probability $p_{\text{wlm}}(w_g|\psi_g)$ is computed using the word-level RNN-LM. The denominator $p_{\text{clm}}(w_g|\psi_g)$ is the probability of w_g obtained by the character-level RNN-LM and used to cancel the character-based LM probabilities accumulated for w_g. The probability can be computed as

$$p_{\text{clm}}(w_g|\psi_g) = \prod_{i=1}^{|w_g|} p_{\text{clm}}(w_{g,i}|\psi_g w_{g,1} \cdots w_{g,i-1}), \tag{18}$$

where $|w_g|$ is the length of word w_g in characters and $w_{g,i}$ indicates the i-th character of w_g. The second term, $p_{\text{wlm}}(\texttt{<UNK>}|\psi_g)$ acts as a weight on the character-level LM and ensures that the combined language model is normalized over character sequences both at word boundaries and in-between.

If w_g is an OOV word as in the second condition, we assume that a word-level probability for the OOV word can be computed with the word and character-level RNN-LMs as

$$p_{\text{oov}}(w_g|\psi_g) = p_{\text{wlm}}(\texttt{<UNK>}|\psi_g) p_{\text{clm}}(w_g|\texttt{<UNK>}, \psi_g). \tag{19}$$

Since the character-level probability satisfies

$$p_{\text{clm}}(w_g|\texttt{<UNK>}, \psi_g) \propto p_{\text{clm}}(w_g|\psi_g), \tag{20}$$

we approximate it as

$$\begin{aligned} p_{\text{clm}}(w_g|\texttt{<UNK>}, \psi_g) &= \frac{p_{\text{clm}}(w_g|\psi_g)}{1 - \sum_{w \in \mathcal{V}} p_{\text{clm}}(w|\psi_g)} && (21) \\ &= \beta(\psi_g)\, p_{\text{clm}}(w_g|\psi_g) && (22) \\ &\approx \tilde{\beta}\, p_{\text{clm}}(w_g|\psi_g), && (23) \end{aligned}$$

and obtain

$$p_{\text{oov}}(w_g|\psi_g) = p_{\text{wlm}}(\texttt{<UNK>}|\psi_g)\, \tilde{\beta}\, p_{\text{clm}}(w_g|\psi_g), \tag{24}$$

where we assume the scaling factor $\beta(\psi_g) = \tilde{\beta}$, and set it as a tunable parameter. In the second condition of Eq. (17), character-based probability $p_{\text{clm}}(w_g|\psi_g)$ is eliminated since it is already accumulated for the hypothesis. This term allows predicting OOV words as well as in-vocabulary words and enables open-vocabulary ASR.

The third case gives the character-level LM probabilities to the hypotheses within a word. Although the character-level LM probabilities are canceled at the end of every known word hypothesis and so are only used to score OOV words, they serve another important role in keeping the correct word hypotheses active in the beam search until the end of the word where the word-level LM probability is applied.

Finally, the log probability of sentence-end label <eos> is added to the log probability of each complete hypothesis g' as

$$\alpha(g') = \alpha(g) + \gamma \log p_{\text{wlm}}(\texttt{<eos>}|\psi_g w_g) \tag{25}$$

in the beam search process.

4. RELATED WORK

There are some prior works related to our multi-level LM approach to oven-vocabulary ASR [20][21][22], which were designed for conventional hybrid ASR systems, i.e., not for end-to-end systems. In [20], subword units such as syllables were introduced to model OOV words, and a subword-level LM was used together with a word-level LM based on the WFST framework. In [21] and [22], character and word-level N-gram LMs were combined with a grapheme-to-phoneme model to handle multiple pronunciations of OOV words.

Our proposed method, in contrast, is specially designed for end-to-end ASR. The proposed method is simpler than the prior methods since we do not have to handle pronunciation variants of the character sequences. In our case, the multi-level character and word-level RNN-LMs are incorporated in our hybrid CTC/attention end-to-end ASR. However, we have not normalized the character LM to exclude character sequence probabilities of in-vocabulary words as done in [22], which would make for interesting future work.

5. EXPERIMENTS

We evaluate our proposed method with the Wall Street Journal (WSJ) corpus, which is a well-known English clean speech database [23, 24]. We used the si284 data set for training, the dev93 data set for validation, and the eval92 data set for evaluation. The data sets are summarized in Table 1.

Table 1. WSJ data sets used for evaluation

	# utterances	Length (h)
Training (WSJ1 si284)	37,416	80
Validation (dev93)	503	1.1
Evaluation (eval92)	333	0.7

As input features, we used 80 mel-scale filterbank coefficients with pitch features and their delta and delta delta features for the CNN/BLSTM encoder [25]. For the attention model, we used only 32 distinct labels: 26 English letters, apostrophe, period, dash, space, noise, and sos/eos tokens. The CTC model used the blank instead of sos/eos, and our MTL model used both sos/eos and the blank.

Our encoder network is boosted by using deep CNN, which is motivated by prior studies [26, 25]. We used a 6-layer CNN architecture based on the initial layers of the VGG net architecture [12] followed by eight BLSTM layers in the encoder network. In the CNN architecture, the initial three input channels are composed of the spectral features, delta, and delta delta features. Input speech feature images are downsampled to $(1/4 \times 1/4)$ images along with the time-frequency axes through the two max-pooling layers. The BLSTM layers had 320 cells in each layer and direction, and the linear projection layer with 320 units is followed by each BLSTM layer. We used the location-based attention mechanism [14], where the 10 centered convolution filters of width 100 were used to extract the convolutional features. The decoder was a one-layer unidirectional LSTM with 300 cells.

The AdaDelta algorithm [27] with gradient clipping [28] was used for the optimization. We also applied a unigram label smoothing technique [29] to avoid over-confidence predictions. In the hybrid attention/CTC architecture, we used the $\lambda = 0.1$ for training and the $\lambda = 0.2$ for decoding. The beam width was set to 30 in decoding under all conditions. The joint CTC-attention ASR was implemented by using the Chainer deep learning toolkit [30].

Character and word-based RNN-LMs were trained with the WSJ text corpus, which consisted of 37M words from 1.6M sentences. The character-level LM had a single LSTM layer with 800 cells and a 32-dimensional softmax layer while the word-level LM had a single LSTM layer with 1000 cells and a 20K-dimensional softmax layer, which equals the vocabulary size of the LM. We used the stochastic gradient descent (SGD) to optimize the RNN-LMs.

The first experiment evaluates the contributions of language models. Table 2 shows word error rate (WER) with different language models. The character-level LM reduces the WER from 13.4% to 11.5% even when using the transcriptions of si284 speech data for LM training, whose size is only 1.8% of the WSJ text corpus. This means that the separate language model has some complementary effect on the prediction performance. The WER is reduced to 7.7% by using more data from the WSJ text corpus. Next, we incorporate a word-level RNN-LM without any character-level RNN-LMs, where only word-based probabilities were applied at every position of space or end-of-sentence character. In this case, the WER increased up to 12.6%. Finally, when using both character and word-level RNN-LMs according to the proposed method, we have obtained a big improvement reaching 5.6% WER.

Table 2. Word Error Rate (WER) with different language models on WSJ.

Language models	LM train data	dev93	eval92
No LM	-	17.3	13.4
Character RNN-LM	si284	15.5	11.5
Character RNN-LM	WSJ	12.3	7.7
Word RNN-LM	WSJ	17.1	12.6
Word+charachter RNN-LM	WSJ	**9.6**	**5.6**

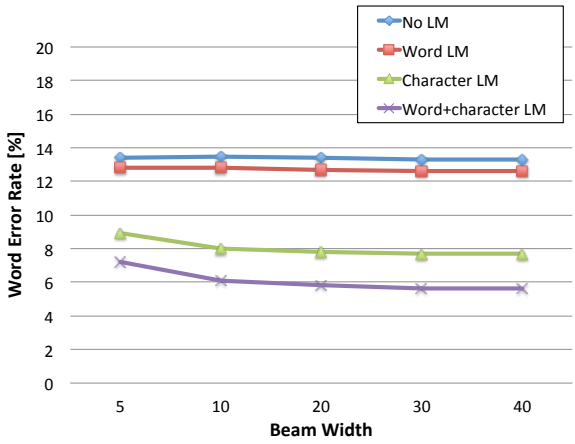

Fig. 2. Beam width vs. word error rate when using different language models

To investigate the reason for the high WER when using only the word-level RNN-LM, we conducted additional experiments with different beam widths ranging from 5 to 40. Figure 2 shows WERs for each beam width. Without character LM, i.e., No LM or Word LM, the WER is almost independent of the beam width. This means that the decoder network predicts labels with high confidence and does not change the result even with a wider beam. Consequently, applying the word-level LM probability at each word end is too late to recover better hypotheses using the word-level information. Our proposed method (word+character LM) achieves the lowest WER by combining the character-level LM as a guide for finding better word sequence hypotheses in beam search. Although the label smoothing technique we used for training the network mitigates this over-confidence problem to some extent, it seems difficult to recover the less-confident labels without any help of character-level LMs.

The second experiment investigates the benefit of the open vocabulary provided by the proposed method. Table 3 compares WERs when using different vocabulary sizes from 20K to 65K and the open vocabulary condition. The vocabularies were just used to constrain all hypotheses to be consisted of only in-vocabulary words. This constraint can be forced by allowing only character sequences appearing in the vocabulary during decoding. As shown in the table, when using a closed vocabulary, the WER does not reach the best WER in the open vocabulary condition. Even with a small OOV rate by the 65K vocabulary, there is still a 1.8% gap to the best WER. We checked the recognition results, and found that they had more deletion errors. This seems to be because when the decoder cannot hypothesize label sequences with high probabilities due to the

Table 3. Comparison of WERs with restricted vocabulary during decoding. We used a 20K-word RNN-LM with a character RNN-LM in all conditions.

Vocabulary size	OOV rate	WER
20K	1.98	8.1
40K	0.57	8.1
65K	0.18	7.4
open	-	**5.6**

Table 4. Comparison with other end-to-end ASR systems reported on WSJ.

End-to-end ASR systems	dev93	eval92
seq2seq [17]	-	9.3
CTC [31]	-	8.2
CTC [16]	-	7.3
seq2seq [29]	9.7	6.7
CTC-Attention+LM (this work)	**9.6**	**5.6**

vocabulary constraint, the scores for the active hypotheses become smaller, and therefore shorter hypotheses tend to be selected as the result.

Finally, we compare our result with other end-to-end systems reported on the WSJ task. Table 4 summarizes the WER numbers obtained from other articles and this work. Since the systems in the table have different network architectures from each other, it is difficult to compare these numbers directly. However, we confirmed that our system has achieved the best WER in the state-of-the-art systems on the WSJ benchmark.

6. CONCLUSION

In this paper, we proposed a novel method that combines character-based and word-based language models in an end-to-end ASR architecture. Previously character-based end-to-end systems suffered in languages with small alphabets, due to the difficulty of modeling language constraints over long sequences of characters. The proposed end-to-end ASR decoding model combines the benefit of character-based open vocabulary recognition, while using word-based language modeling to overcome the weaker language modeling power of character-based language models. This paper demonstrates that a multilevel LM with both character-level and word-level dependencies outperforms models with only word-level or only character-level RNN-LMs, in the open vocabulary ASR condition. In a standard Wall Street Journal (WSJ) task, we achieved 5.6 % WER for the Eval'92 test set using only the SI284 training set and WSJ text data, which is the best score reported for end-to-end ASR systems on this benchmark. Future work will include evaluation of the proposed framework with different tasks in different languages, and extension to joint training for the end-to-end architecture including the multi-level language models.

7. REFERENCES

[1] Tara N. Sainath, Oriol Vinyals, Andrew Senior, and Hasim Sak, "Convolutional, Long Short-Term Memory, Fully Connected Deep Neural Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015.

[2] 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, "The Kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec. 2011.

[3] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "End-to-end continuous speech recognition using attention-based recurrent NN: First results," *arXiv preprint arXiv:1412.1602*, 2014.

[4] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015.

[5] Liang Lu, Xingxing Zhang, and Steve Renals, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 5060–5064.

[6] William Chan and Ian Lane, "On Online Attention-based Speech Recognition and Joint Mandarin Character-Pinyin Training," in *INTERSPEECH*, 2016.

[7] William Chan, Yu Zhang, Quoc Le, and Navdeep Jaitly, "Latent sequence decompositions," in *International Conference on Learning Representations*, 2017.

[8] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint CTC-attention based end-to-end speech recognition using multitask learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4835–4839.

[9] Takaaki Hori, Shinji Watanabe, and John R. Hershey, "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*, 2017.

[10] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *INTERSPEECH*, 2017.

[11] Daniel Povey, Vijayaditya Peddinti, Daniel Galvez, Pegah Ghahrmani, Vimal Manohar, Xingyu Na, Yiming Wang, and Sanjeev Khudanpur, "Purely sequence-trained neural networks for asr based on lattice-free MMI," in *Interspeech*, 2016, pp. 2751–2755.

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

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

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

[15] Alex Graves, "Supervised sequence labelling with recurrent neural networks," *PhD thesis, Technische Universität München*, 2008.

[16] Yajie Miao, Mohammad Gowayyed, and Florian Metze, "EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding," in *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2015, pp. 167–174.

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

[18] Mehryar Mohri, Fernando Pereira, and Michael Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002.

[19] Takaaki Hori and Atsushi Nakamura, "Speech recognition algorithms using weighted finite-state transducers," *Synthesis Lectures on Speech and Audio Processing*, vol. 9, no. 1, pp. 1–162, 2013.

[20] Issam Bazzi, *Modelling out-of-vocabulary words for robust speech recognition*, Ph.D. thesis, Massachusetts Institute of Technology, 2002.

[21] M Ali Basha Shaik, David Rybach, Stefan Hahn, Ralf Schlüter, and Hermann Ney, "Hierarchical hybrid language models for open vocabulary continuous speech recognition using wfst.," in *SAPA@ INTERSPEECH*, 2012, pp. 46–51.

[22] M Ali Basha Shaik, Amr El-Desoky Mousa, Stefan Hahn, Ralf Schlüter, and Hermann Ney, "Improved strategies for a zero oov rate lvcsr system," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 5048–5052.

[23] Linguistic Data Consortium, "CSR-II (wsj1) complete," *Linguistic Data Consortium, Philadelphia*, vol. LDC94S13A, 1994.

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

[25] Yu Zhang, William Chan, and Navdeep Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017.

[26] Ying Zhang, Mohammad Pezeshki, Philémon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville, "Towards end-to-end speech recognition with deep convolutional neural networks," *arXiv preprint arXiv:1701.02720*, 2017.

[27] Matthew D Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[28] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio, "On the difficulty of training recurrent neural networks," *arXiv preprint arXiv:1211.5063*, 2012.

[29] Jan Chorowski and Navdeep Jaitly, "Towards better decoding and language model integration in sequence to sequence models," *arXiv preprint arXiv:1612.02695*, 2016.

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

[31] Alex Graves and Navdeep Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *International Conference on Machine Learning (ICML)*, 2014, pp. 1764–1772.