Segmental Recurrent Neural Networks for End-to-end Speech Recognition

Citation for published version:

Digital Object Identifier (DOI):
10.21437/Interspeech.2016-40

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings of Interspeech 2016

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 07. Jan. 2020
Segmental Recurrent Neural Networks for End-to-end Speech Recognition

Liang Lu\textsuperscript{1*}, Lingpeng Kong\textsuperscript{2}, Chris Dyer\textsuperscript{2}, Noah A. Smith\textsuperscript{3}, and Steve Renals\textsuperscript{1}

\textsuperscript{1}Centre for Speech Technology Research, The University of Edinburgh, Edinburgh, UK
\textsuperscript{2}School of Computer Science, Carnegie Mellon University, Pittsburgh, USA
\textsuperscript{3}Computer Science \\& Engineering, The University of Washington, Seattle, USA

\{liang.lu, s.renals\}@ed.ac.uk, \{lingpenk, cdyer\}@cs.cmu.edu, nasmith@cs.washington.edu

Abstract

We study the segmental recurrent neural network for end-to-end acoustic modelling. This model connects the segmental conditional random field (CRF) with a recurrent neural network (RNN) used for feature extraction. Compared to most previous CRF-based acoustic models, it does not rely on an external system to provide features or segmentation boundaries. Instead, this model marginalises out all the possible segmentations, and features are extracted from the RNN trained together with the segmental CRF. In essence, this model is self-contained and can be trained end-to-end. In this paper, we discuss practical training and decoding issues as well as the method to speed up the training in the context of speech recognition. We performed experiments on the TIMIT dataset. We achieved 17.3\% phone error rate (PER) from the first-pass decoding — the best reported result using CRFs, despite the fact that we only used a zeroth-order CRF and without using any language model.

Index Terms: end-to-end speech recognition, segmental CRF, recurrent neural networks.

1. Introduction

Speech recognition is a typical sequence to sequence transduction problem, i.e., given a sequence of acoustic observations, the speech recognition engine decodes the corresponding sequence of words (or phonemes). A key component in a speech recognition system is the acoustic model, which computes the conditional probability of the output sequence given the input sequence. However, directly computing this conditional probability is challenging due to many factors including the variable lengths of the input and output sequences. The hidden Markov model (HMM) converts this sequence level classification task into a frame-level classification problem, where each acoustic frame is classified into one of the hidden states, and each output sequence corresponds to a sequence of hidden states. To make it computationally tractable, HMMs usually rely on the conditional independence assumption and the first-order Markov rule — the well-known weaknesses of HMMs. Furthermore, the HMM-based pipeline is composed of a few relatively independent modules, which makes the joint optimisation nontrivial.

There has been a consistent research effort to seek architectures to replace HMMs and overcome their limitation for acoustic modelling, e.g., \cite{1,2,3,4}. However, these approaches have not yet improved speech recognition accuracy over HMMs. In the past few years, several neural network-based approaches have been proposed and demonstrated promising results. In particular, the connectionist temporal classification (CTC) \cite{6,7,8,9} approach defines the loss function directly to maximise the conditional probability of the output sequence given the input sequence, and it usually uses a recurrent neural network to extract features. However, CTC simplifies the sequence-level error function by a product of the frame-level error functions (i.e., independence assumption), which means it essentially still does frame-level classification. It also requires the lengths of the input and output sequence to be the same, which is inappropriate for speech recognition. CTC deals with this problem by replicating the output labels so that a consecutive frames may correspond to the same output label or a blank token.

Attention-based RNNs have been demonstrated to be a powerful alternative sequence-to-sequence transducer, e.g., in machine translation \cite{10}, and speech recognition \cite{11,12,13}. A key difference of this model from HMMs and CTCS is that the attention-based approach does not apply the conditional independence assumption to the input sequence. Instead, it maps the variable-length input sequence into a fixed-size vector representation at each decoding step by an attention-based scheme (see \cite{11} for further explanation). It then generates the output sequence using an RNN conditioned on the vector representation from the source sequence. The attentive scheme suits the machine translation task well, because there may be no clear alignment between the source and target sequence for many language pairs. However, this approach does not naturally apply to the speech recognition task, as each output token only corresponds to a small size window of acoustic spectrum.

In this paper, we study segmental RNNs \cite{14} for acoustic modelling. This model is similar to CTC and attention-based RNN in the sense that an RNN encoder is also used for feature extraction, but it differs in the sense that the sequence-level conditional probability is defined using an segmental (semi-Markov) CRF \cite{15}, which is an extension on the standard CRF \cite{16}. There have been numerous works on CRFs and their variants for speech recognition, e.g., \cite{4,5,17} (see \cite{18} for an overview). In particular, feed-forward neural networks have been used with segmental CRFs for speech recognition \cite{19,20}.

However, segmental CRF is different in that it is an end-to-end model — it does not depend on an external system to provide segmentation boundaries and features, instead, this model is trained by marginalising out all possible segmentations, while the features are derived from the encoder RNN, which is trained jointly with the segmental CRF. Our experiments were performed on the TIMIT dataset, and we achieved 17.3\% PER from first-pass decoding with zeroth-order CRF and without using any language model — the best reported result using CRFs.
2. Segmental Recurrent Neural Networks

2.1. Segmental Conditional Random Fields

Given a sequence of acoustic frames \( X = \{x_1, \ldots, x_T\} \) and its corresponding sequence of output labels \( y = \{y_1, \ldots, y_T\} \), where \( T \geq J \), segmental (or semi-Markov) conditional random field defines the sequence-level conditional probability with the auxiliary segment labels \( E = \{e_1, \ldots, e_J\} \) as

\[
P(y, E \mid X) = \frac{1}{Z(X)} \prod_{j=1}^{J} \exp f(y_j, e_j, X),
\]

where \( e_j = (s_j, n_j) \) is a tuple of the beginning \( (s_j) \) and the end \( (n_j) \) time tag for the segment of \( y_j \), and \( n_j > s_j \) while \( n_j, s_j \in [1, T] \); \( y_j \in Y \) and \( Y \) denotes the vocabulary set; \( Z(X) \) is the normaliser that sums over all the possible \((y, E)\) pairs, i.e.,

\[
Z(X) = \sum_{y, E} \prod_{j=1}^{J} \exp f(y_j, e_j, X).
\]

Here, we only consider the zeroth-order CRF, while the extension to higher order models is straightforward. Similar to other CRF-based models, the function \( f(\cdot, \cdot, \cdot) \) is defined as

\[
f(y_j, e_j, X) = w^\top \Phi(y_j, e_j, X),
\]

where \( \Phi(\cdot) \) denotes the feature function, and \( w \) is the weight vector. Previous works on CRF-based acoustic models mainly use heuristically handcrafted feature function \( \Phi(\cdot) \). They also usually rely on an external system to provide the segment labels. In this paper, we define \( \Phi(\cdot) \) using neural networks, and the segmentation \( E \) is marginalised out during training, which makes our model self-contained.

2.2. Feature Representations

We use neural networks to define the feature function \( \Phi(\cdot) \), which maps the acoustic segment and its corresponding label into a joint feature space. More specifically, \( y_j \) is first represented as a one-hot vector \( v_{y_j} \), and it is then mapped into a continuous space by a linear embedding matrix \( M \) as

\[
u_j = M v_{y_j}.
\]

Given the segment label \( e_j \), we use an RNN to map the acoustic segment to a fixed-dimensional vector representation, i.e.,

\[
h^0_j = r(h^0, x_{s_j})
\]
\[
h^1_j = r(h^1, x_{s_j+1})
\]
\[
\vdots
\]
\[
h^d_j = r(h^d, x_{n_j})
\]

where \( h^0 \) denotes the initial hidden state, \( d_j = n_j - s_j \) denotes the duration of the segment and \( r(\cdot) \) is a non-linear function. We take the final hidden state \( h^d_j \) as the segment embedding vector, then \( \Phi(\cdot) \) can be represented as

\[
\Phi(y_j, e_j, X) = g(u_j, h^d_j),
\]

where \( g(\cdot) \) corresponds to one layer or multiple layers of linear or non-linear transformation. In fact, it is flexible to include other relevant features as additional inputs to the function \( g(\cdot) \), e.g., the duration feature which can be obtained by converting \( d_j \) into another embedding vector. In practice, multiple RNN layers can be used transform the acoustic signal \( X \) before extracting the segment embedding vector \( h^d_j \) as Figure 1.

Figure 1: Segmental RNN using first-order CRF. The coloured circles denote the segment embedding vector \( h^d_j \) in Eq. (7). Using bi-directional RNNs is straightforward.

2.3. Conditional Maximum Likelihood Training

For speech recognition, the segmentation labels \( E \) are usually unknown, training the model by maximising the conditional probability as Eq. (1) is therefore not practical. The problem can be addressed by defining the loss function as the negative marginal log-likelihood as

\[
\mathcal{L}(\theta) = - \log P(y \mid X)
\]
\[
= - \log \sum_{E} P(y, E \mid X)
\]
\[
= - \log \sum_{E} \prod_{j=1}^{J} \exp f(y_j, e_j, X) + \log Z(X),
\]

where \( \theta \) denotes the set of model parameters, and \( Z(X, y) \) denotes the summation over all the possible segmentations when only \( y \) is observed. To simplify notations, the objective function \( \mathcal{L}(\theta) \) is defined with only one training utterance.

However, the number of possible segmentations is exponential with the length of \( X \), which makes the naive computation of both \( Z(X, y) \) and \( Z(X) \) impractical. Fortunately, this can be addressed by using the following dynamic programming algorithm as proposed in [15]:

\[
\alpha_0 = 1
\]
\[
\alpha_t = \sum_{0<k<t} \alpha_k \times \sum_{y \in Y} f(y, (k, t), X)
\]
\[
Z(X) = \alpha_T
\]

In Eq. (11), the first summation is over all the possible segmentation up to timestep \( t \), and the second summation is over all the possible labels from the vocabulary. The computation cost of this algorithm is \( O(T^2 \cdot |Y|) \), where \( |Y| \) is the size of the vocabulary. The cost can be further reduced by introducing an upper bound of the segment length, in which case Eq. (11) can be rewritten as

\[
\alpha_t = \sum_{l<k<t} \alpha_k \times \sum_{y \in Y} f(y, (k, l), X)
\]
\[
l = \begin{cases} 0 & \text{if } t - L < 0 \\ t - L & \text{otherwise} \end{cases}
\]

where \( L \) denotes the maximum value of the segment length. The cost is then reduced to \( O(L \cdot T \cdot |Y|) \), and for long sequences
This involves minor modification of the recursive algorithm in Eq. (11) that instead of summing over all the possible labels, marginalising out all the possible segmentations:

$$
\beta_0,0 = 1 \quad (15)
\beta_{k, j} = \sum_{0 < k < t} \beta_{k, j-1} \times f(y_j, (k, t), X) \quad (16)
Z(X, y) = \beta_{T, T} \quad (17)
$$

Again, we can limit the length of the possible segments as Eq. (13). Given $Z(X)$ and $Z(X, y)$, the loss function $L(\theta)$ can be minimised using the stochastic gradient decent (SGD) algorithm similar to training other neural network models. Other losses, for example, hinge, can be considered in future work.

### 2.4. Viterbi Decoding

During decoding, we need to search the target label sequence $y$ that yields the highest posterior probability given $X$ by marginalising out all the possible segmentations:

$$
y^* = \arg \max_y \log \sum_E P(y, E | X) \quad (18)
$$

This involves minor modification of the recursive algorithm in Eq. (11) that instead of summing over all the possible labels, the Viterbi path up to the timestep $t$ is

$$
\alpha^*_t = \sum_{0 < k < t} \alpha^*_k \times \max_{y \in Y} f(y, (k, t), X) \quad (19)
$$

However, marginalising out all the possible segmentations is still expensive. The computational cost can be further reduced by greedy searching the most likely segmentation, i.e.,

$$
\alpha^*_t = \max_{0 < k < t} \alpha^*_k \times \max_{y \in Y} f(y, (k, t), X), \quad (20)
$$

which corresponds to the decoding objective as

$$
y^*, E^* = \arg \max_{y, E} P(y, E | X) \quad (21)
$$

### 2.5. Further Speedup

It is computationally expensive for RNNs to model long sequences, and the number of possible segmentations is exponential with the length of the input sequence as mentioned before. The computational cost can be significantly reduced by using the hierarchical subsampling RNN [21] to shorten the input sequences, where the subsampling layer takes a window of hidden states from the lower layer as input as shown in Figure 2. In this work, we consider three variants: a) **concatenate** – the hidden states in the subsampling window are concatenated before being fed into the next layer; b) **add** – the hidden states are added into one vector for the next layer; c) **skip** – only the last hidden state in the window is kept and all the others are skipped. The last two schemes are computationally cheaper as they do not introduce extra model parameters.

### 3. Experiments

#### 3.1. System Setup

We used the TIMIT dataset to evaluate the segmental RNN acoustic models. This dataset was preferred for the rapid evaluation of different system settings, and for the comparison to other CRF and end-to-end systems. We followed the standard protocol of the TIMIT dataset, and our experiments were based on the Kaldi recipe [22]. We used the core test set as our evaluation set, which has 192 utterances. We used 24 dimensional log filterbanks (FBANKs) with delta and double-delta coefficients, yielding 72 dimensional feature vectors. Our models were trained with 48 phonemes, and their predictions were converted to 39 phonemes before scoring. The dimension of the LSTM cells was fixed to be 32. For all our experiments, we used the long short-term memory (LSTM) networks [23] as the implementation of RNNs, and the networks were always bi-directional. We set the initial SGD learning rate to be 0.1, and we exponentially decay the learning rate by a factor of 2 when the validation error stopped decreasing. Our models were trained with dropout.

### Table 1: Speedup by hierarchical subsampling networks.

<table>
<thead>
<tr>
<th>System</th>
<th>d(w)</th>
<th>d(hj)</th>
<th>layers</th>
<th>hidden</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>skip</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>128</td>
<td>∼3x</td>
</tr>
<tr>
<td>conc</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>128</td>
<td>23.2</td>
</tr>
<tr>
<td>add</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>128</td>
<td>21.3</td>
</tr>
<tr>
<td>skip</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>250</td>
<td>20.1</td>
</tr>
<tr>
<td>conc</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>250</td>
<td>20.5</td>
</tr>
<tr>
<td>add</td>
<td>64</td>
<td>64</td>
<td>3</td>
<td>250</td>
<td>21.5</td>
</tr>
</tbody>
</table>

This joint maximization algorithm may yield high search error, because it only considers one segmentation. In the future, we shall investigate the beam search algorithm which may yield a lower search error.
regularisation with an specific implementation for recurrent networks. The dropout rate was 0.2 unless specified otherwise. Our models were randomly initialised with the same random seed.

### 3.2. Results of Hierarchical Subsampling

We first demonstrate the results of the hierarchical subsampling recurrent network, which is the key to speed up our experiments. We set the size of the subsampling window to be 2, therefore each subsampling layer reduced the time resolution by a factor of 2. We set the maximum segment length $L$ in Eq. \[(d)\] to be 300 milliseconds, which corresponded to 30 frames of FBANKs (sampled at the rate of 10 milliseconds). With two layers of subsampling recurrent networks, the time resolution was reduced by a factor of 4, and the value of $L$ was reduced to be 8, yielding around 10 times speedup as shown in Table 4.

Table 4 compares the three implementations of the recurrent subsampling network detailed in section 3.3. We observed that concatenating all the hidden states in the subsampling window did not yield lower phone error rate (PER) than using the simple skipping approach, which may be due to the fact that the TIMIT dataset is small and it prefers a smaller model. On the other hand, adding the hidden states in the subsampling window together worked even worse, possibly due to that the sequential information in the subsampling window was flattened. In the following experiments, we stuck to the skipping method, and using two subsampling layers.

### 3.3. Hyperparameters and Different Features

We then evaluated the model by tuning the hyperparameters, and the results are given in Table 3. We tuned the number of LSTM layers, and the dimension of LSTM cells, as well as the dimensions of $w$ and the segment vector $h_j$. In general, larger models with dropout regularisation yielded higher recognition accuracy. Our best result was obtained using 6 layers of 250-dimensional LSTMs. However, without the dropout regularisation, the model can be easily overfit due to the small size of training set. In the future, we shall evaluate this model with a large dataset.

### 3.4. Comparison to Related Works

In Table 5, we compare our result to other reported results using segmental CRFs as well as recent end-to-end systems. Previous state-of-the-art result using segmental CRFs on the TIMIT dataset is reported in [28], where the first-pass decoding was used to prune the search space, and the second-pass was used to re-score the hypothesis using various features including neural network features. Besides, the ground-truth segmentation was used in [28]. We achieved considerably lower PER with first-pass decoding, despite the fact that our CRF was zeroth-order, and we did not use any language model. Furthermore, our results are also comparable to that from the CTC and attention-based RNN end-to-end systems. The accuracy of segmental RNNs may be further improved by using higher-order CRFs or incorporating a language model into the decode step, and using beam search to reduce the search error.

### 4. Conclusions

In this paper, we present the segmental RNN — a novel acoustic model that combines the segmental CRF with an encoder RNN for end-to-end speech recognition. We discuss the practical training and decoding algorithms of this model for speech recognition, and the subsampling network to reduce the computational cost. Our experiments were performed on the TIMIT dataset, and we achieved strong recognition accuracy using zeroth-order CRF, and without using any language model. In the future, we shall investigate discriminative training criteria, and incorporating a language model into the decoding step. Future works also include implementing a weighted finite state transducer (WFST) based decoder and scaling this model to large vocabulary datasets.
5. References


