Vol.2020-SLP-131 No.9
2020/2/14

ストリーミング注意機構型 sequence-to-sequence モデルによる講演音声認識

稲熊 寛文[1]　三村 正人[1]　河原 達也[1]

概要：ラベル同期型モデルである注意機構型 sequence-to-sequence では，入力音声フレームごとの出力を獲得できないためストリーミング認識には不向きであった．これに対処するため，逐次トークンを出力できるモデルがこれまで提案されてきたが，アライメント境界は過去のアライメントに依存するため，出力系列長が長くなるにつれて認識誤りの影響が後方のトークン生成へ伝搬するという問題があった．そこで本稿では，CTC のアライメント情報を教師として学習中に与えることにより，アライメントの学習を頑健にする CTC 同期学習を提案する．また，一定区間の入力フレームごとに枝刈りを行うチャンク同期型ビームサーチを提案し，外部の音声区間検出モデルを用いない，講演単位の認識を行う．日本語話し言葉コーパスにおける実験において，CTC 同期学習による認識精度の改善を確認する．

キーワード：ストリーミング音声認識，monotonic chunkwise attention, CTC，CSJ

1. はじめに

近年，入力音声と書き起こしの写像を直接最適化する end-to-end 音声認識モデルに注目が集まっており，従来のハイブリッドモデルに迫る，または同等以上の認識精度が報告されている [1]．End-to-end 音声認識モデルは大きく分けて，connectionist temporal classification（CTC）[2] や RNN transducer（RNN-T）[3] に代表されるフレーム同期型モデルと，注意機構型 sequence-to-sequence モデル [4,5] に代表されるラベル同期型モデルに分類される．注意機構型 sequence-to-sequence モデルはこれらの end-to-end モデルの中で最も高い認識精度を達成しているものの [6,7]，入力音声全体をエンコードするまで最初のトークンを出力できないため，入力音声に対して逐次的にトークン生成を行うストリーミング認識には不向きである．一方，フレーム同期型モデルは各入力フレームに対してトークンを出力できるため，エンコーダを単方向，またはレイテンシ制御型双方向モデル [8] にすることでストリーミング認識が可能となる．そこで近年，音声認識における入出力のアライメントの単調性を利用し，入力音声全体をエンコードし終わる前にトークンを逐次出力することができるストリーミング注意機構型 sequence-to-sequenc モデルがいくつか提案されている．中でも monotonic chunkwise attention（MoChA）[9] はオフラインのモデルと同等の精度を示すことが報告されている．

MoChA は CTC や RNN-T と同様に前向きアルゴリズムを用いてアライメント確率の周辺化を行うが，デコーダの自己回帰性により後ろ向き確率を用いることができない．したがって，出力系列が長くなるにつれてアライメント誤りが後方のトークン生成へ伝搬する．実際，CTC とマルチタスク学習を行う場合，各トークンに対するアライメント境界が CTC スパイクより数フレーム遅延することを経験的に確認した．そこで，CTC の強制アライメント結果を利用して MoChA と CTC のアライメント境界の位置の差異を直接最小化する，CTC 同期学習を提案する．また，MoChA の推論はラベル同期型のビームサーチによって行われるため，有効仮説集合内の全ての仮説に対して次のトークンのアライメント境界が特定するまで仮説の枝刈りは行われない．これにより，デコーダ側で遅延が発生する．そこで本稿では，一定入力チャンクごとにトークンを逐次的に出力できるチャンク同期型ビームサーチを提案する．日本語話し言葉コーパスにおける実験により，学習中に CTC のアライメント情報を用いることで認識精度が大きく改善されることを確認した．さらに，MoChA と CTC のアライメントが一致するように学習されていることをアライメント境界の可視化により確認した．最後に，CTC の出力層から連続して出力される blank トークンの累積数を数え上げることでエンコーダとデコーダの状態を初期化するタイミングを決定し，外部の音声区間検出モデルを用い

[1] 京都大学 大学院情報学研究科

ない講演単位のストリーミング認識を行う．

2. ストリーミング注意機構型 sequence-to-sequence モデル

2.1 Monotonic chunkwise atteniton (MoChA)

本稿では，ストリーミング注意機構型 sequence-to-sequence モデルの一つである，monotonic chunkwise attention（MoChA）[9] に着目する．各出力トークン生成時において入力系列全体に対する相対的なアライメントスコアを計算する従来の注意機構型モデル [4] と異なり，MoChA は入力系列に対するアライメント境界の位置を学習する．音声認識における入出力アライメントの単調性を利用し，推論時のデコーディングの計算量は二次から線形へと削減される．二値系列で表されるアライメント境界を学習するのは困難であるため，学習時はアライメント境界の期待値を計算することで，通常の注意機構型モデルと同様に微分可能なクロスエントロピー損失 $\mathcal{L}_{\text{mocha}}$ を最小化する．ただし，学習時のデコーディングの計算量は二次となる．入力音声系列を $\boldsymbol{x} = (x_1, \cdots, x_T)$，出力トークン系列を $\boldsymbol{y} = (y_1, \cdots, y_L)$ とすると，モデル全体の損失関数は以下のようになる．

$$\mathcal{L}_{\text{total}} = (1 - \lambda_{\text{ctc}})\mathcal{L}_{\text{mocha}} + \lambda_{\text{ctc}}\mathcal{L}_{\text{ctc}} \tag{1}$$

λ_{ctc} は CTC 損失の重みであり，本稿では CTC とエンコーダを共有してマルチタスク学習を行い，MoChA と同時に最適化する．本稿では全ての実験で $\lambda_{\text{ctc}} = 0.3$ とした．CTC とのマルチタスク学習は注意機構の入出力のアライメントの単調性を促進して収束を速める効果があり，推論時にもそのスコアを使うことで認識精度が改善することが報告されている [10]．さらに，本稿では CTC から出力される blank トークンを用いることで音声区間検出を行う（3.3 節）．

学習時においては，可能な全てのアライメント $\{\alpha_{i,j}\}$ に対する事後確率を周辺化することで最適化を行う．$\{\alpha_{i,j}\}$ は出力ステップ i において j 番目の入力フレームでトークンを出力する期待値を表す（*monotonic attention*）．$\{\alpha_{i,j}\}$ は選択確率 $\{p_{i,j}\} \in [0, 1]$ によって以下のように再帰的に計算され，$\{p_{i,j}\}$ はエンコーダ状態およびデコーダ状態の関数としてパラメータ化される．

$$\begin{aligned}
e^{\text{mono}}_{i,j} &= g\frac{v^{\text{T}}}{||v||}\text{ReLU}(\mathbf{W}_{\text{h}}h_j + \mathbf{W}_{\text{s}}s_i + b) + r \\
p_{i,j} &= \sigma(e^{\text{mono}}_{i,j}) \\
\alpha_{i,j} &= \text{MonotonicAttn}(s_i, h_j) \\
&= p_{i,j}\sum_{k=1}^{j}\left(\alpha_{i-1,k}\prod_{l=k}^{j-1}(1 - p_{i,l})\right) \\
&= p_{i,j}\left((1 - p_{i,j-1})\frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j}\right)
\end{aligned}$$

ここで g, v, $\mathbf{W}_{\text{h}}$, $\mathbf{W}_{\text{s}}$, b, r は学習パラメータであり，s_i は i 番目のデコーダ状態，h_j は j 番目のエンコーダ状態，σ はシグモイド関数，そして $e^{\text{mono}}_{i,j}$ は monotonic attention のエネルギー関数を表す．$\{\alpha_{i,j}\}$ は累積和と累積積を組み合わせることで，各入力フレーム j に対して並列化して効率的に計算することができる [9]．

monotinic attention に加えて，各出力トークンに対応するアラメイント境界からの一定区間のチャンク（w フレーム）に対する相対的なアライメントスコアを計算する chunkwise attention $\beta_{i,j}$ を導入する．Chunkwise attention のエネルギー関数 $e^{\text{chunk}}_{i,j}$ は monotonic attention のエネルギー関数 $e^{\text{mono}}_{i,j}$ と同様に定式化されるが（ただしオフセット r は使用しない），異なるパラメータで実装される．

$$\begin{aligned}
\beta_{i,j} &= \text{ChunkwiseAttn}(s_i, h_j) \\
&= \sum_{k=j}^{j+w-1}\left(\alpha_{i,k}\exp(e^{\text{chunk}}_{i,j}) / \sum_{l=k-w+1}^{k}\exp(e^{\text{chunk}}_{i,l})\right)
\end{aligned}$$

$\{\beta_{i,j}\}$ は移動総和を用いることで効率的に計算される．コンテキストベクトル c_i の期待値は，通常の注意機構型モデルと同様にチャンクに対するスコア $\beta_{i,j}$ を用いて以下のように計算される．

$$c_i = \text{Score}(\beta_{i,j}, h_j) = \sum_{j=1}^{T}\beta_{i,j}h_j$$

以降のデコーダ状態の更新，およびトークン生成は通常の注意機構型モデルと同様に行われる．

テスト時においては，終端ラベルを除く各ラベルは $p_{i,j}$ が閾値 0.5 を超えた場合に出力され，このとき $\alpha_{i,j}$ は 1.0 にセットされる．次のトークン生成は，直前のトークンに対するアライメント境界 $\arg_j(\alpha_{i-1,j} = 1)$ より右側で行われる．より詳細な定式化は [9, 11] を参照されたい．

2.2 レイテンシ制御型双方向 LSTM エンコーダ（LC-BiLSTM）

双方向 LSTM エンコーダ（bidirectional LSTM，以降 BiLSTM）は入力発話の最後からの現在までの音声フレームを考慮するため，最初から現在までのフレームしか考慮しない単方向 LSTM エンコーダ（unidirectional LSTM，以降 UniLSTM）よりも高い精度を示すことが知られている [1]．しかし，エンコードの遅延が発生するため，ストリーミング認識には適していない．そこで本稿では，現在の音声フレームの位置から右側の一定フレームを未来のコンテキストとして用いる，レイテンシ制御型 BiLSTM をエンコーダ（latency-controlled BiLSTM，以降 LC-BiLSTM）を採用する．LC-BiLSTM は，通常の BiLSTM と同様に順方向 LSTM と逆方向 LSTM から構成される．現在のチャンク（N_{l} フレーム）における順方向 LSTM の最終状態は，次のチャンクにおける順方向 LSTM の初期状態として用いる．ただし，隣接するチャンクの音声フレームはオーバー

Algorithm 1 Chunk-synchronous vectorized beam search decoding

function ChunkSyncDecode($\boldsymbol{h}^{\text{chunk}}, \Omega, N_{\text{beam}}, M_{\text{len}}$)
  // Initialization
  $i \leftarrow 0$, $\Omega_{\text{end}} \leftarrow \{\}$
  $L_{\max} \leftarrow T_{\text{chunk}} \times M_{\text{len}}$

  for $i = 0, \cdots, L_{\max} - 1$ **do**
    $B \leftarrow |\Omega|$
    if $i > 0$ and $\sum_b \boldsymbol{\alpha}_{i-1}^{(b)}$=0 **then**
      break ▷ No additional boundary
    end if
    $\boldsymbol{s}_{i-1}^{(1:B)}, \boldsymbol{c}_{i-1}^{(1:B)}, \boldsymbol{y}_{i-1}^{(1:B)} \leftarrow$ Vectorize(Ω)
    $\boldsymbol{s}_{i}^{(1:B)} =$ DecoderRNN($\boldsymbol{s}_{i-1}^{(1:B)}, \boldsymbol{c}_{i-1}^{(1:B)}, \boldsymbol{y}_{i-1}^{(1:B)}$)
    $\boldsymbol{\alpha}_{i}^{(1:B)} =$ MonotonicAttn($\boldsymbol{s}_{i}^{(1:B)}, \boldsymbol{h}^{\text{chunk}}$)
    $\boldsymbol{\beta}_{i}^{(1:B)} =$ ChunkwiseAttn($\boldsymbol{s}_{i}^{(1:B)}, \boldsymbol{h}^{\text{chunk}}$)
    $\boldsymbol{c}_{i}^{(1:B)} =$ Score($\boldsymbol{\beta}_{i}^{(1:B)}, \boldsymbol{h}^{\text{chunk}}$)
    $\boldsymbol{p}_{i}^{(1:B)} =$ Generate($\boldsymbol{s}_{i}^{(1:B)}, \boldsymbol{c}_{i}^{(1:B)}$)
    // Optionally integrate CTC and LM scores here
    $b \leftarrow 0$, $\Omega_{\text{next}} \leftarrow \{\}$
    for l in Ω **do**
      for k in TopkIndex($\boldsymbol{p}_{i}^{(b)}$) **do**
        if $p_{i,k}^{(b)}$ =<eos> **then**
          add l to Ω_{end}
        else
          add to $l + [k]$ to Ω_{next}
        end if
      end for
      $b \leftarrow b + 1$
    end for
    select top N_{beam} hypotheses in Ω_{next}
  end for
  return (Ω_{end}, Ω_{next})
end function

ラップしない．一方，逆方向 LSTM は現在のチャンクの N_{l} フレームに加えて，さらに右側の N_{r} フレームもコンテキストして使用する（合計 $N_{\text{l}} + N_{\text{r}}$ フレーム）．逆方向 LSTM の初期状態は，チャンクに関わらず常にゼロベクトルとする．$N_{\text{l}}, N_{\text{r}}$ を適切に設定することで，推論時のレイテンシを制御することが可能となる．

3. 提案手法

3.1 CTC 同期学習

CTC が前向き・後ろ向きアルゴリズムを用いてアライメントの事後確率の周辺化を行う一方で，MoChA は前向きアルゴリズムのみを用いてアライメントの事後確率の周辺化を行う．これはデコーダが自己回帰モデルであるためである．通常の注意機構が各トークン生成時に入力系列全体に対する相対的なアライメントスコアを計算するのに対して，MoChA のアライメントスコアは過去のアライメントに依存する．これにより，出力系列が長くなるにつれてアライメント誤りが後方のトークン生成へ伝搬し，認識に悪影響を及ぼす．予備実験において，MoChA によって出力されたアライメント境界が同じトークンに対応する CTC の事後確率のスパイクに比べて数フレーム遅延することを経験的に観測した（図 1，2）．

そこで本稿では，MoChA のアライメント境界を CTC の事後確率のスパイクに近づけるような CTC 同期学習を提案する．CTC のアライメントは後ろ向き確率を考慮して計算されるため，MoChA によって学習されるアライメントよりも信頼度が高いと考えられる．CTC のアライメント情報を学習中に使用することで，MoChA によるアライメントの学習を容易にすることが期待される．MoChA の学習時，エンコーダを共有することで CTC とマルチタスク学習を行い，さらに CTC の強制アライメント結果を用いる．可能なすべての CTC パスのうち，最も事後確率の高いものを選択し，アライメント境界の正解位置 $\mathbf{b}^{\text{ctc}} = (\text{b}_1^{\text{ctc}}, \cdots, \text{b}_L^{\text{ctc}})$ とする．ただし，blank 以外のトークンが連続する箇所では，最も左側のインデックスを境界として使用する．$\mathbf{b}^{\text{ctc}}$ は学習の各ステップにおけるパラメータを用いてオンラインで決定される．

MoChA の推論時のハードアライメント境界 $\{j|\alpha_{i,j} = 1\}_{i=1,\cdots,L}$ は微分可能でないため，$\{\alpha_{i,j}\}$ を用いた境界の期待値（ソフトアライメント境界）を用いる．i 番目のトークンのソフトアライメント境界は以下のように表される．

$$\text{b}_i^{\text{mocha}} = \sum_{j=1}^{T} j\alpha_{i,j}$$

その上で，MoChA と CTC のアライメント境界の差異 $\mathcal{L}_{\text{sync}}$ を以下のように定義する．

$$\mathcal{L}_{\text{sync}} = \sum_{i=1}^{L} |\text{b}_i^{\text{ctc}} - \text{b}_i^{\text{mocha}}|$$

このとき，式（1）の損失関数は以下のように修正され，同時最適化を行う．

$$\mathcal{L}_{\text{total}} = (1 - \lambda_{\text{ctc}})\mathcal{L}_{\text{mocha}} + \lambda_{\text{ctc}}\mathcal{L}_{\text{ctc}} + \lambda_{\text{sync}}\mathcal{L}_{\text{sync}}$$

ここで λ_{sync} は $\mathcal{L}_{\text{sync}}$ の重みであり，本稿では $\mathcal{L}_{\text{sync}} = 1.0$ とした．

3.2 チャンク同期型ビームサーチ

注意機構型 sequence-to-sequence モデルはラベル同期型モデルであり，推論時にビームサーチを行う場合，有効な（終端記号が出力されていない）仮説集合内に含まれる仮説長は同じである．MoChA は推論時のデコーディングの計算量を二次から線形に落とすものの，ビームサーチは通常の注意機構型モデルと同様に行われる．しかし，MoChA によって特定されるアライメント境界は仮説ごとに異なる可能性があるため，有効仮説集合 Ω 内の全ての仮説に対して次に出力されるトークンに対するアライメント境界が決定するまで仮説集合を更新することができない（追加の入力音声を待たなければならない）．これにより，デコーダ側でトークン生成の遅延が発生する．このように，ラベル同期型のビームサーチはストリーミング認識には不向きである．

そこで本稿では，一定フレーム数で区切られたの入力音声チャンクごとに仮説集合を更新する，チャンク同期型ビームサーチを提案する（Algorithm 1）．各チャンク内では，対応するエンコーダ出力 $\boldsymbol{h}^{\text{chunk}}$ に対してラベル同期型のビームサーチが行われる．次のトークンのアライメント境界が現在のチャンクに存在しない仮説に対しては，次のトークンの展開は行わずに Ω 内に残す．その他の仮説に対しては，トークン生成を行う度にビーム幅 N_{beam} で枝刈りを行う（30 行目）．終端集合が出力された場合，終了仮説集合 Ω_{end} に追加する．Ω 内の全ての仮説に対して追加のアライメント境界が特定されなくなった場合（8-10 行目），またはトークン生成のステップ数が閾値 $L_{\max}$ を超えた場合，現在のチャンクにおけるビームサーチを終了する．仮説集合内の各仮説の系列長が異なる場合があるため，仮説長でスコアを正規化する．

本稿ではレイテンシ制御型エンコーダを使用するため，N_{chunk} をエンコーダの入力音声チャンク幅 N_{l} に設定する．実装に関しては，Ω 内の各仮説をバッチとしてまとめてビームサーチを行い，デコーディングを効率化した．

3.3 CTC による音声区間検出（CTC-VAD）

CTC の学習では，語彙内のトークンを出力しないことに対応する blank トークンを導入することにより，入出力の系列長を揃えて前向き・後ろ向きアルゴリズムによって可能な全てのアライメントの事後確率を周辺化して最適化を行う．CTC によって学習されたアライメントは出力トークンのアライメント境界に必ずしも対応しておらず，学習後の CTC の出力層から得られる事後確率は時間方向に対してスパイク状になることが知られている [2]．しかし，一定区間連続する blank トークンは無音区間に対応していると捉えることができる．そこで本稿では，連続して出力された blank トークンの数が閾値 N_{b} を超えた場合，次に初めて blank 以外のトークンが出力される入力フレームまでの区間を無音区間としてみなす．無音区間が検出された場合，その時点で最も確率の高い仮説を出力する．また，MoChA のエンコーダおよびデコーダの LSTM の状態とメモリセルをゼロベクトルで初期化（状態リセット）し，有効仮説集合 Ω を空集合で初期化する．ただし，頻繁に状態リセットが行われるのを防ぐため，前回の状態リセット位置からの入力音声フレームの累積数が N_{acc} を超え，かつ無音区間が検出された場合にのみ次の状態リセットを行う（Algorithm 2）．

CTC の blank トークンを用いた音声区間検出は ESPnet[*1] や wav2letter++[*2] などでも実装されている．ただし，本稿では無音区間で分割された有声区間をオフライン認識するのではなく，状態リセットの位置を特定するために使用しており，無音区間に対しても推論を行っている．また，無音区間が検出されるかどうかに関わらず，逐次トークンの出力を行うことができる．MoChA によるアライメント境界の検出は CTC によらないため，CTC によって誤認識された入力音声フレームに対して正しい出力が得られる可能性がある．

Algorithm 2 Streaming decoding with CTC-based VAD

1: **function** DECODE($\boldsymbol{x}, N_{\text{chunk}}, N_{\text{acc}}, N_{\text{b}}, M_{\text{spike}}, M_{\text{len}}$)
2: // Initialization
3: $t \leftarrow 0$, $n_{\text{acc}} \leftarrow 0$, $n_{\text{b}} \leftarrow 0$
4: $IsReset \leftarrow True$
5: $\Omega \leftarrow$ {[<sos>,]}, $hyps \leftarrow []$
6:
7: **while** True **do**
8: // Reset encoder state if $IsReset = True$
9: $\boldsymbol{h}^{\text{chunk}} \leftarrow \text{Encode}(\boldsymbol{x}_{t:t+N_{\text{chunk}}}, IsReset)$
10: $\boldsymbol{p}^{\text{ctc}} \leftarrow \text{CTC}(\boldsymbol{h}^{\text{chunk}})$
11: $j_{\text{reset}} \leftarrow -1$
12: $IsReset \leftarrow False$
13: $n_{\text{acc}} \leftarrow n_{\text{acc}} + N_{\text{chunk}}$
14:
15: // CTC-based VAD if $n_{\text{acc}} \geq N_{\text{chunk}}$
16: **if** $n_{\text{acc}} \geq N_{\text{acc}}$ **then**
17: **for** $j = 0, \cdots, N_{\text{chunk}} - 1$ **do**
18: **if** $\underset{1 \leq k \leq K}{\text{argmax}}\, p^{\text{ctc}}_{j,k} =$<blank> **then**
19: $n_{\text{b}} \leftarrow n_{\text{b}} + 1$
20: **else if** $\max_{1 \leq k \leq K} p^{\text{ctc}}_{j,k} < M_{\text{spike}}$ **then**
21: $n_{\text{b}} \leftarrow n_{\text{b}} + 1$
22: **else**
23: $n_{\text{b}} \leftarrow 0$
24: **end if**
25: // Detect the first boundary
26: **if** not $IsReset$ **then**
27: $IsReset \leftarrow True$
28: $j_{\text{reset}} \leftarrow j$
29: **end if**
30: **end for**
31: **end if**
32:
33: // Chunk-synchronous beam search
34: $(\Omega_{\text{end}}, \Omega) \leftarrow \text{ChunkSyncDecode}(\boldsymbol{h}^{\text{chunk}}, \Omega, N_{\text{beam}}, M_{\text{len}})$
35: $\Omega_{\text{merge}} \leftarrow \Omega_{\text{end}} \cup \Omega$
36:
37: // Finish if the best candidate is in Ω_{end}
38: **if** $\text{argmax}(\Omega_{\text{merge}})[-1] =$<eos> **then**
39: **if** not $IsReset$ **then**
40: $j_{\text{reset}} \leftarrow |\boldsymbol{h}_{\text{chunk}}| - 1$
41: $IsReset \leftarrow True$
42: **end if**
43: **end if**
44:
45: // Detect the next state reset point
46: **if** $IsReset$ **then**
47: $hyps \leftarrow hyps + [\text{argmax}(\Omega_{\text{merge}})]$
48: $n_{\text{acc}} \leftarrow 0$, $n_{\text{b}} \leftarrow 0$
49: $\Omega \leftarrow$ {[<sos>,]} ▷ Reset beam
50:
51: // Next chunk will start from $j_{\text{reset}} + 1$
52: $t \leftarrow t - (N_{\text{chunk}} - 1 - j_{\text{reset}})$
53: **end if**
54: $t \leftarrow t + N_{\text{chunk}}$
55:
56: **if** $t \geq T - 1$ **then**
57: **break**
58: **end if**
59: **end while**
60:
61: // Hypothesis for the last segment
62: **if** not $IsReset$ **then**
63: $hyps \leftarrow hyps + [\text{argmax}(\Omega_{\text{end}} \cup \Omega)]$
64: **end if**
65:
66: **return** $hyps$ ▷ Session-level hypotheses
67: **end function**

[*1] https://github.com/espnet/espnet
[*2] https://github.com/facebookresearch/wav2letter

表 1: ラベル同期型ビームサーチによる発話単位での認識精度

モデル	Streaming	WER（%）			CER（%）		
		eval1	eval2	eval3	eval1	eval2	eval3
Baseline							
BiLSTM global attention	–	7.6	5.7	6.2	5.0	4.0	4.0
LC-BiLSTM global attention	–	7.9	5.8	6.7	5.3	4.0	4.4
BiLSTM MoChA	–	7.7	5.8	6.7	5.1	4.1	4.4
Proposed method							
UniLSTM MoChA	✓	11.1	8.3	9.1	7.8	6.2	6.3
+ Pre-training (seeded by UniLSTM MoChA)	✓	10.2	7.7	8.1	7.0	5.7	5.6
+ CTC-sync	✓	**9.8**	**7.2**	**7.6**	**6.7**	**5.2**	**5.1**
LC-BiLSTM MoChA (seeded by BiLSTM MoChA)	✓	8.4	6.2	**6.5**	5.6	4.5	**4.3**
+ CTC-sync	✓	**7.9**	**5.9**	6.7	**5.1**	**4.2**	**4.3**
References							
ESPnet BiLSTM global attention + speed perturb. [12]	–	N/A	N/A	N/A	6.6	4.8	5.0
ESPnet Transformer + speed perturb. [12]	–	N/A	N/A	N/A	5.7	4.1	4.5
Transformer	–	**7.1**	**5.4**	**5.7**	**4.5**	**3.8**	**3.7**

4. 評価実験

「日本語話し言葉コーパス」（CSJ）[13] を使用し，ストリーミング注意機構型 sequence-to-sequence モデルの発話単位および講演単位での音声認識精度の評価実験を行った．

4.1 コーパス

「日本語話し言葉コーパス」（CSJ）は学会講演，模擬講演から構成される（計 586 時間）．評価データは eval1, eval2，eval3 の 3 種類あり，それぞれ 10 講演から構成されている．発話単位の評価として，CSJ でアノテーションされている音声区間のタイムスタンプを用いて発話ごとに音声を分割する．講演単位の評価では，各講演の分割前の音声を使用し，最初の音声フレームから最後の音声フレームまで全てを用いて認識を行う．Kaldi のレシピに従い，開発データとして学習データの先頭 4000 発話を選択し，学習データから除いた．

4.2 実験条件

入力特徴量は，80 次元対数メルフィルタバンク出力を Kaldi を用いて抽出し，学習データ全体の統計量を用いてゼロ正規化を行った．ネットワークに関して，オフラインモデルのエンコーダには 4 層の CNN（各チャネル 32）と 5 層の BiLSTM を用いた．ストリーミングモデルには同様に 4 層の CNN と 5 層の LC-BiLSTM または UniLSTM を用いた．BiLSTM の各方向のメモリセルのユニット数は 512，UniLSTM のユニット数は 1024 とした．入力系列長を削減するため CNN 内で 2 層ごとに最大プーリングを行い，系列長が合計で 4 分の 1 になるようにサブサンプリングを行った．デコーダにはメモリセルのユニット数 1024 の 1 層の LSTM を用いた．MoChA のチャンクサイズは 4 とした．出力トークンとして byte pair encoding（BPE）アルゴリズムで学習した 10,000 クラスの語彙を採用した．BPE の学習には sentencepiece[*3] を使用した．

パラメータの更新は 30 発話をミニバッチとして，Adam によって行った．ドロップアウト率は 0.4 とし，label smoothing の確率は 0.1 とした．デコーディングはビームサーチによって行い，ビーム幅は 5 とした．言語モデルには，ユニット数 1024 の 2 層の LSTM から構成されるリカレントニューラルネットワーク言語モデルを使用し．shallow fusion [14] によって言語モデルのスコアを統合した．ネットワークの実装は PyTorch[*4]によって行った．

LC-BiLSTM エンコーダを用いて学習する場合，BiLSTM エンコーダを用いた MoChA で事前学習し，デコーダを含む全てのパラメータを初期値として使用した．LC-BiLSTM のチャンクサイズは $N_{\mathrm{l}} = 40, N_{\mathrm{r}} = 40$ とした．チャンク同期型ビームサーチに関して，$N_{\mathrm{acc}} = 800, N_{\mathrm{b}} = 40, M_{\mathrm{spike}} = 0.1, M_{\mathrm{len}} = 0.4$ とした．

評価尺度として，word error rate（WER）および character error rate（CER）を使用した．日本語では唯一の単語境界が存在しないが，CSJ でアノテーションされている単語境界を使用した．学習時は単語境界を考慮して出力系列をサブワードに分割し，CER は単語境界を削除して計算した．

4.3 実験結果：発話単位での評価

CSJ における発話単位での認識精度の評価結果を表 1 に示す．BiLSTM エンコーダを用いたオフライン設定では，MoChA は通常の注意機構型モデル（global attention）に近い精度を達成した．LC-BiLSTM エンコーダを用いた場合，未来のコンテキストが制限されるため精度が悪化した．また，比較として ESPnet によって実装された BiLSTM global attention，および Transformer の結果も示す．Transformer モデルの認識精度は BiLSTM global attention を上回り，これまでの CSJ における認識精度の中で最も高い精度を示した．ただし，本稿では speed perturbation や SpecAugment などのデータの水増しは行っていないことに注意されたい．

次に，ストリーミング認識モデルに関して，LC-BiLSTM MoChA が UniLSTM MoChA を大きく上回った．これよ

[*3] https://github.com/google/sentencepiece

[*4] https://github.com/pytorch/pytorch

り，未来のコンテキストが認識精度の改善に重要であることがわかる．CTC 同期学習によって UniLSTM MoChA の精度を大きく改善し，LC-BiLSTM MoChA では eval3 以外で改善がみられた．これは，UniLSTM MoChA のアライメント境界が LC-BiLSTM MoChA よりも右側にずれ込むことにより，発話後半のトークンのアライメント境界の特定がより困難になることに起因すると考えられる（図 1，2）．一方，CTC 同期学習によってアライメント情報を補助的に与えることで，MoChA のアライメント境界が左側に移動することを確認した．これにより，後半のトークン生成に対応するアライメント境界の特定がより容易になり，精度改善につながったと考えらえる．さらに，CTC 同期学習によって CTC のスパイクもやや左側に移動していることも確認できる．これはエンコーダを共有しているためであると考えられる．また，LC-BiLSTM MoChA に対して CTC 同期学習を行うことで，同じ LC-BiLSTM global attention と同等の認識精度を達成した．

次に，ラベル同期型およびチャンク同期型のビームサーチによる認識結果の比較結果を表 2 に示す．モデルは CTC 同期学習によって最適化を行った LC-BiLSTM MoChA を用いた．チャンク同期型ビームサーチにより認識精度が低下したが，ラベル同期型ビームサーチでは次のチャンクのエンコードによる遅延が発生する場合がある．

表 2: チャンク同期型ビームサーチによる発話単位でのストリーミング認識精度

モデル	WER(%)		
	eval1	eval2	eval3
ラベル同期（no LM）	8.8	6.5	7.3
＋ shallow fusion	**7.9**	**5.9**	**6.7**
チャンク同期（no LM）	9.3	6.9	7.8
＋ shallow fusion	8.5	6.6	7.3

表 3: チャンク同期型ビームサーチによる講演単位でのストリーミング認識精度

認識	eval1	eval2	eval3
	WER（%）	WER（%）	WER（%）
	Sub/Del/Ins	Sub/Del/Ins	Sub/Del/Ins
発話	8.5 5.5/1.8/1.2	6.6 4.6/1.1/0.8	7.3 5.0/1.1/1.2
講演	11.0 5.9/1.8/3.3	8.0 4.8/1.1/2.2	9.8 5.4/1.5/2.9

4.4 実験結果：講演単位での評価

最後に，講演単位でのストリーミング音声認識結果を表 3 に示す．モデルは CTC 同期学習によって最適化を行った LC-BiLSTM MoChA を用いた．CSJ のアノテーションは全ての発話に対してはなされておらず，講演単位の認識結果では発話単位のテストセットには含まれていなかった発話も余分に認識している．これにより，挿入誤りが増加していることがわかる．しかし，削除誤りに関してはほぼ同等であり，置換誤りも 0.2-0.4 ポイントほどの差に留まっている．外部の音声区間検出モデルを使用し

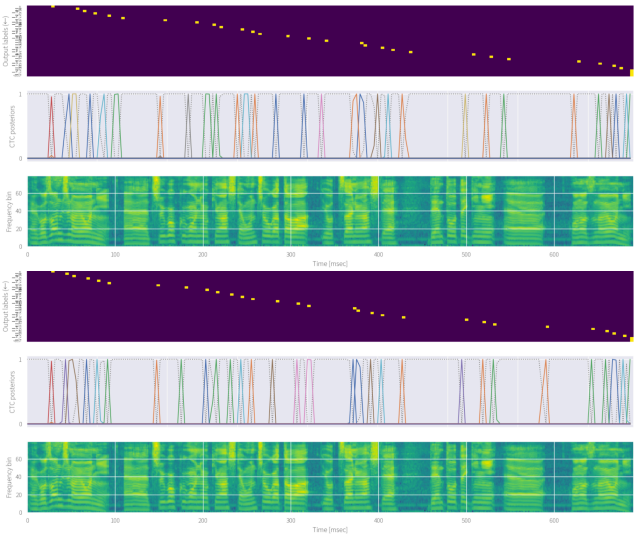

図 1: UniLSTM MoChA のアライメント境界の可視化（上：ベースライン，下：CTC 同期学習）

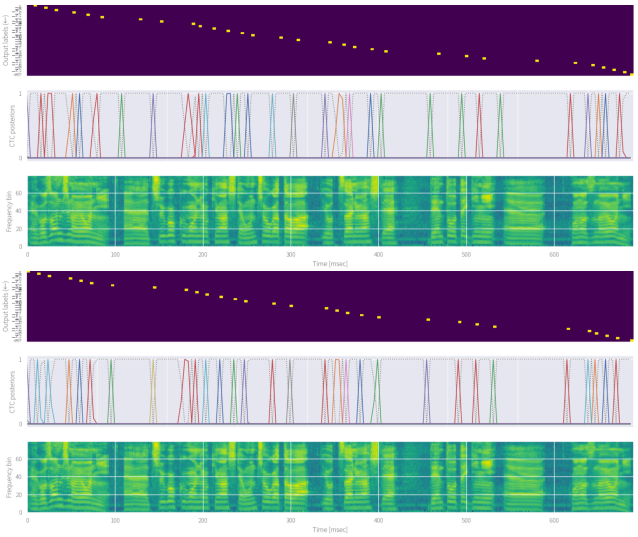

図 2: LCBiLSTM MoChA のアライメント境界の可視化（上：ベースライン，下：CTC 同期学習）

ていないことを考慮すると，ストリーミング注意機構型 sequence-to-sequence モデルでも長い講演を十分に高い精度で認識できることがわかる．

5. おわりに

本稿では，monotonic chunkwise attention (MoChA) を用いたストリーミング注意機構型 sequencet-to-sequence モデルによる CSJ における音声認識の評価を行った．頑健なアライメントを学習するため，CTC のアライメント情報を教師として与える CTC 同期学習を提案し，単方向およびレイテンシ制御型双方向 LSTM エンコーダを用いた場合の両方で認識精度の改善を確認した．また，チャンクごとに仮説の枝刈りを行うチャンク同期型ビームサーチを提案した．最後に，一定区間連続する CTC の blank トークンを検出することにより，外部の音声区間検出モデルを用いない講演単位での認識を行い，十分高い精度で認識で

きることを確認した．今後の展望として，TEDLIUM2 などの英語のコーパスにおいても評価を行い，また triggered attention [15] や RNN-T などのモデルとの比較も行う予定である．さらに，ラベル同期型ビームサーチと同等の認識を実現できるようにチャンク同期型ビームサーチのアルゴリズムを改良する．

参考文献

[1] Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, K. et al.: State-of-the-art speech recognition with sequence-to-sequence models, *Proceedings of ICASSP*, IEEE, pp. 4774–4778 (2018).

[2] Graves, A., Fernández, S., Gomez, F. and Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks, *Proceedings of ICML*, pp. 369–376 (2006).

[3] Graves, A.: Sequence transduction with recurrent neural networks, *arXiv preprint arXiv:1211.3711* (2012).

[4] Chorowski, J. K., Bahdanau, D., Serdyuk, D., Cho, K. and Bengio, Y.: Attention-based models for speech recognition, *Proceedings of NIPS*, pp. 577–585 (2015).

[5] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I.: Attention is all you need, *Advances in neural information processing systems*, pp. 5998–6008 (2017).

[6] Prabhavalkar, R., Rao, K., Sainath, T. N., Li, B., Johnson, L. and Jaitly, N.: A Comparison of Sequence-to-Sequence Models for Speech Recognition., *Proceedings of Interspeech*, pp. 939–943 (2017).

[7] Battenberg, E., Chen, J., Child, R., Coates, A., Li, Y. G. Y., Liu, H., Satheesh, S., Sriram, A. and Zhu, Z.: Exploring neural transducers for end-to-end speech recognition, *Proceedings of ASRU*, IEEE, pp. 206–213 (2017).

[8] Zhang, Y., Chen, G., Yu, D., Yaco, K., Khudanpur, S. and Glass, J.: Highway long short-term memory rnns for distant speech recognition, *Proceedings of ICASSP*, IEEE, pp. 5755–5759 (2016).

[9] Chiu, C.-C. and Raffel, C.: Monotonic chunkwise attention, *Proceedings of ICLR* (2018).

[10] Watanabe, S., Hori, T., Kim, S., Hershey, J. R. and Hayashi, T.: Hybrid CTC/attention architecture for end-to-end speech recognition, *IEEE Journal of Selected Topics in Signal Processing*, Vol. 11, No. 8, pp. 1240–1253 (2017).

[11] Raffel, C., Luong, M.-T., Liu, P. J., Weiss, R. J. and Eck, D.: Online and linear-time attention by enforcing monotonic alignments, *Proceedings of ICML*, pp. 2837–2846 (2017).

[12] Karita, S., Chen, N., Hayashi, T., Hori, T., Inaguma, H., Jiang, Z., Someki, M., Soplin, N. E. Y., Yamamoto, R., Wang, X. et al.: A comparative study on transformer vs rnn in speech applications, *arXiv preprint arXiv:1909.06317* (2019).

[13] Maekawa, K.: Corpus of Spontaneous Japanese: Its design and evaluation, *ISCA & IEEE Workshop on Spontaneous Speech Processing and Recognition* (2003).

[14] Kannan, A., Wu, Y., Nguyen, P., Sainath, T. N., Chen, Z. and Prabhavalkar, R.: An analysis of incorporating an external language model into a sequence-to-sequence model, *Proceedings of ICASSP*, IEEE, pp. 5824–5828 (2017).

[15] Moritz, N., Hori, T. and Le Roux, J.: Triggered attention for end-to-end speech recognition, *Proceedings of ICASSP*, IEEE, pp. 5666–5670 (2019).