デジタルプラクティス　Vol.10 No.2(Apr. 2019)

特集号招待論文

Web検索等に利用される分散型音声認識システムへのディープラーニングの実装

藤田 悠哉[1]

[1]ヤフー（株）

音声認識技術は議事録の文字起こしやテレビ字幕作成，スマートフォン等からの検索クエリの入力やスマートスピーカの操作に利用されている技術であり，近年ディープラーニング技術の利用により性能が飛躍的に向上した．本稿では，ディープラーニング技術を検索クエリ入力等に利用される分散型音声認識に適用するにあたって生じる2つの課題を取り扱う．1つ目は，大量のデータを用いたモデル学習の高速化，2つ目は学習済みのモデルをリアルタイムに推論するための高速化である．これら2つの課題解決にあたっての工夫や得られた知見を紹介する．

1. はじめに

音声認識技術とは，音声信号を単語列に変換する技術である．これは，長さTの音声信号$X=(x_1,x_2,\cdots,x_T)$に対応する長さMの単語列$W=(w_1,w_2,\cdots,w_M)$を見つけ出すパターン認識問題と捉えることができる．しかし，音声信号の長さと単語列の長さが一致しないこと，つまりある単語列に対応する音声信号の時間方向の伸縮を考慮する必要があることから，音声認識技術特有の方法が研究され，発展してきた．

音声認識技術の歴史を大まかに振り返ると，**表1**の様に60年の時を経てさまざまな取り組みにより進歩して来た[1],[2]．まだ計算機が一般的ではなかった1950年代に「1，5，7，9」といった孤立数字や，「あ，い，う，え，お」といった母音の認識が試みられたのが音声認識技術の研究開発の発端と言えるだろう．その後，1970年代のアメリカ国防高等研究計画局（DARPA,Defense Advanced Research Projects Agency）による共通の学習・評価データの整備による各研究機関の競争の促進など，さまざまな要因により研究開発が進展し，1990年代の商用システムの発売に繋がったと考えられる．また，2009年にディープラーニング技術が登場し[3]，さまざまなタスクにおいて大幅な性能改善が達成された[4]．そして，スマートフォンからの検索クエリ入力の手段として音声認識技術が用いられ，数千から数十万時間の音声データを収集可能になった．加えて，GPU（Graphics Processing Unit）の性能向上およびディープラーニングフレームワークにより大量の音声データを用いたモデル学習が現実的な時間で行える様になった．そのため，近年の音声認識技術の研究開発はディープラーニングを利用したものが主流となっている．

© 2019 Information Processing Society of Japan

しかし，ディープラーニングを実際の製品で利用するためにはさまざまなノウハウや工夫が必要となる．本稿では特に，スマートフォンからの検索クエリ入力などに用いられる分散型音声認識システムへディープラーニングを導入した事例を紹介する．

表1　音声認識技術の歴史概観

1950~60 年代	孤立数字認識 母音認識
1970 年代	大語彙音声認識システムの開発が始まる DARPA 主導による共通の学習・評価データ整備
1980 年代	現在の主流となった隠れマルコフモデルと N-gram 言語モデルが登場
1990 年代	誤り率最小化学習による性能改善 商用システムが販売される
2009 年	ディープラーニングを音声認識に利用する研究が発表される
2012 年～	各研究機関がディープラーニングの有効性を報告 実システムでの利用が進む

2. 分散型音声認識システムの利用場面と特徴

分散型音声認識システム（DSR，Distributed Speech Recognition system）とは，サーバとクライアントから構成される音声認識システムのことである．ヤフーではYJVOICEという名称のシステムを開発・運用している[5]．YJVOICEでは**図1**の様に，スマートフォンの様なクライアント端末から音声信号または特徴量をネットワークを介してサーバに送信し，サーバ側で音声認識処理を行って，クライアントに認識結果を返す．

© 2019 Information Processing Society of Japan

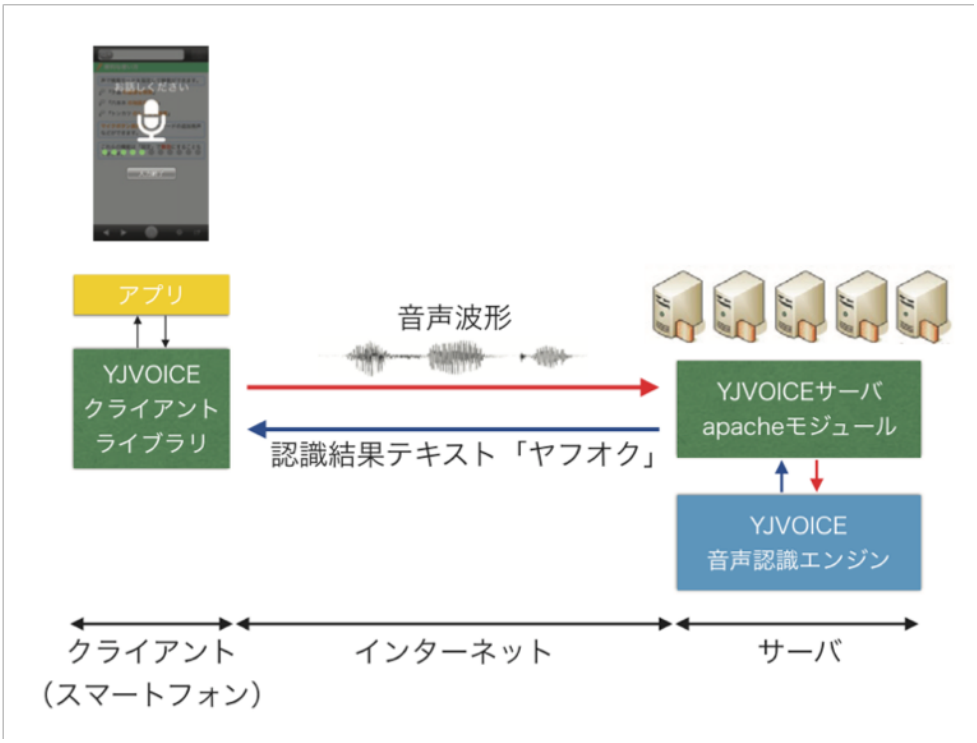

図1　分散型音声認識の構成

DSRの利点は主に（1）クライアント端末で行う演算を最小限に抑えられることから計算機リソース等が限られた端末でも音声認識を利用でき，（2）サーバ側はデータセンタ等に設置された計算機リソースが豊富な計算機を利用できることから計算量やメモリ使用量の制限が少ない，という点が挙げられる．

特に筆者らが開発・運用しているシステムにおいては，スマートフォンからの検索クエリの入力が主な利用用途であることから，多種多様な検索クエリに対応するため語彙数（認識可能な単語の数）が100万程と大きく，モデルサイズが大きくなる．たとえば，筆者らが運用しているシステムでは，モデルを展開するだけで数十GByteのメモリを消費する．また，認識結果をなるべく早く返すことがユーザ体験上望ましいため，リアルタイムでの処理が求められる．音声認識処理では数万に及ぶ仮説を展開することから，リアルタイムで認識処理を行うためにはCPUおよびメモリを多く消費する．大きなモデルを用いて展開する仮説の数を大きくする方が認識性能は高くなるため，計算機リソースの制約が小さいDSRを利用している．

さらに，DSRには次のような利点もある．クライアント端末で完結する音声認識と違い，必ず音声もしくは音響特徴量をサーバへ送信するため，大量の音声データの収集が容易に行える．音声認識のモデルは，多くの音声データを用いた方が性能が高くなる．また，学習したモデルの更新はサーバ側のアップデートだけで済み，クライアント端末側の対応は必要ない．

3. 音声認識技術とディープラーニング

2009年に音声認識にディープラーニングを適用する研究が報告された[3]．その後，2012年に各研究機関からディープラーニングによって種々のタスクの音声認識性能が改善することが報告された[4]．特に，DARPAが提供している学習・評価データにおいて従来法と比べて単語誤り

© 2019 Information Processing Society of Japan

率（ある音声の人間による書き起こしと，音声認識が出力した結果を比べて，どの程度誤りがあるかを測る指標）が33%改善したことが非常に大きなインパクトとなり，その後ディープラーニングを用いた研究が活性化するきっかけとなったと言えるだろう．

本章では音声認識技術の基本的な仕組みと，音声認識におけるディープラーニングの活用方法を述べる．

3.1 音声認識技術の仕組み

先述の通り，音声認識は長さTの音声信号$X=(x_1,x_2,\cdots,x_T)$に対応する長さMの単語列$W=(w_1,w_2,\cdots,w_M)$を見つけ出すパターン認識問題と捉えることができる．これらの間に何らかの確率モデルを仮定して定式化するとこの問題は，

$$\hat{W} = \arg\max_{W} p(W|X)$$

と表せる．音声認識においてはこの問題を直接は扱わず，ベイズの定理から，

$$\hat{W} = \arg\max_{W} p(X|W)p(W)$$

と変形して扱う[6],[7]．このうち$p(X|W)$は音響モデルと呼ばれ，単列語Wが与えられた時の音声信号Xが生成される確率を与える．また，$p(W)$は言語モデルと呼ばれる，音響特徴に依存しない単語列Wの確率モデルである．

3.1.1 音響モデル

音響モデルには隠れマルコフモデル（HMM, Hidden Markov Model）が用いられる．HMMはさまざまな構造を持ち得るが，音声認識では3状態のleft-to-right構造かつ出力確率が混合ガウス分布（GMM, Gaussian Mixture Model）

$$p(x|s) = \sum_{n=1}^{N} \pi_{s,n} \frac{1}{\sqrt{(2\pi)^D |\Sigma_{s,n}|}} \exp\left(-\frac{1}{2}(x-\mu_{s,n})^T \Sigma_{s,n}^{-1}(x-\mu_{s,n})\right)$$

で表されるものが用いられる[6],[7]．ここでNはガウス分布の混合数，$\pi_{s,n}$は状態sのn番目のガウス分布の混合係数を表す．$\mu_{s,n}$と$\Sigma_{s,n}$は状態sのn番目のガウス分布の平均ベクトルと共分散行列である．

音響モデルでは直接単語列に対してHMMを定義するのではなく，音声の物理的な特徴で分類された音素と呼ばれる単位を，さらに前後の音素に依存したトライフォンと呼ばれるものに対して1つのHMMを定義する．たとえば「あした」という単語は音素では「a sh i t a」と表され

© 2019 Information Processing Society of Japan

る．これを文の始まりと終わりを「*」，前に接続する音素を「-」，後ろに接続する音素を「+」という記号で表記するものがトライフォンであり，「a sh i t a」のトライフォン表記は「*-a+sh a-sh+i sh-i+t i-t+a t-a+*」となる．

理屈からは，各トライフォンに1つのHMMが割り当てられることになり，HMMの状態数は音素数の3乗に比例する個数となるので，GMMの数も大量となる．しかし，実際には学習データに滅多に出現しないトライフォンがあるため，状態をクラスタリングする．そのため，状態数つまりGMMの数は高々数千から数万となる．出力確率がGMMで与えられるHMMのこと一般的にGMM-HMMと呼ぶ．一方，音声認識の文脈においてはディープラーニング以前の音声認識システムと区別する呼称として用いられることが慣例となっている．

3.1.2　言語モデル

言語モデルは音響特徴量には依存しない，単語の出やすさを表現する確率モデルである．これは，単語履歴が与えられたときに 次に出現する単語の確率分布を与えるモデルであり，たとえば

$$p(W) = p(w_T|w_{t-1}, w_{t-2}, \cdots, w_1)$$

の様にすべての単語履歴に依存するモデルが考えられる．しかし，言語モデルの学習に用いるテキストデータには，滅多に出現しない単語履歴があることから，履歴を適当なNで打ち切ったモデルを用いる．これを*N-gram*言語モデルと呼ぶ[8]．Nの値は3や4が用いられることが多い．

3.1.3　サーチ

音声認識処理では，音響モデルおよび言語モデルを用いて入力される音声信号に最もマッチする単語列を探索する．これをサーチまたデコード処理と呼ぶ．各モデルは非常に多くのパラメータを持ち，すべての仮説を展開して最良のものを選ぶには計算コストが膨大にかかることから，近似や枝刈りを用いたアルゴリズムによって実現されている．近年は，重み付き有限状態トランスデューサ（WFST, Weighted Finite State Transducer）を用いて音響モデルと言語モデルをあらかじめ統合して探索空間を狭めた上でサーチを行う方法がよく用いられる[6],[7].

3.2　音声認識におけるディープラーニングの活用方法

3.2.1　音響モデル

音響モデルにおけるディープラーニングの活用方法は，GMM-HMMの出力確率を適当なディープラーニングのモデル，たとえば多層のニューラルネットワーク（DNN, Deep Nerual Network）の出力に置き換える．

図2は，「明日の東京の天気は」という発話からDNNを学習する過程を概念的に描いたものである．音声波形は一定の長さ（たとえば25msec）で切り出されたのち周波数スペクトルに変換される．この周波数スペクトルを得る一定の長さのことをフレームと呼ぶ．さらに，周波数スペクトルは人間の聴覚特性を模したメルフィルタバンクと呼ばれるフィルタ処理が施され，40次元の特徴量となる．この特徴量を過去5フレーム，未来5フレーム分連結してDNNに入力し，正解ラベルとしてトライフォンHMMの状態系列を与えてクロスエントロピー基準で学習する．

© 2019 Information Processing Society of Japan

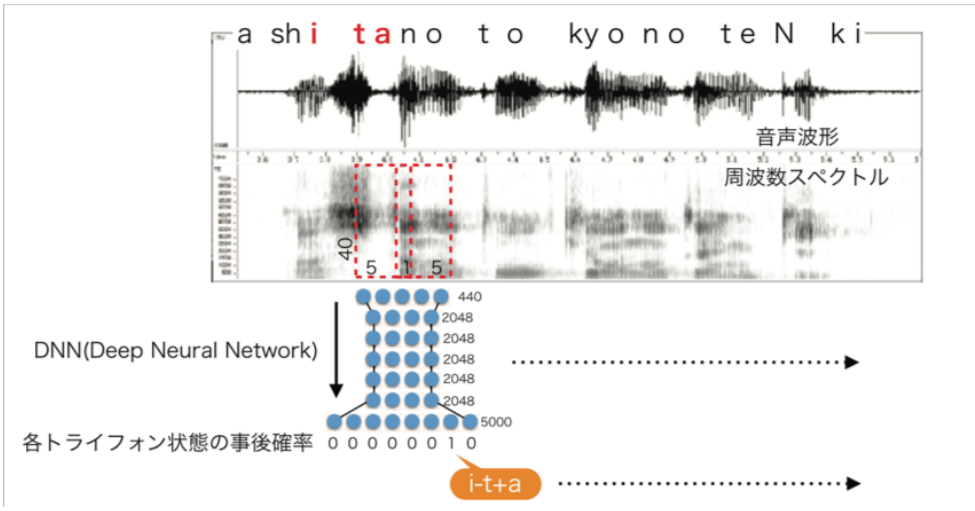

図2　音響モデルにおけるディープラーニングの活用方法

学習されたDNNをデコードに利用する際は，同様の特徴量をDNNに入力する．クロスエントロピー基準で学習しているので，DNNの出力はトライフォンHMMのある状態の事後確率となる．これをベイズの定理により尤度に変換し，GMMの出力確率を置き換える．この方式を特にDNN-HMMハイブリッド方式と呼ぶ．この手法により，従来のGMM-HMMに比べて大幅な性能改善が報告されている[4]．

3.2.2　音声区間検出

音声認識処理の前段として，音声が発話されているか否かを検出する音声区間検出（VAD, Voice Activity Detection）を用いることがある．VADもディープラーニングを用いることによる性能改善が報告されている[9]．VADのモデルは，図2の音響モデルにおける出力ラベルが，音声か非音声かの2状態になったものである．

3.2.3　その他の活用方法

本稿のスコープからは外れるが，他にもさまざまな活用方法がある．言語モデルにおいてもDNNやリカレントニューラルネットワーク（RNN, Reccurent Neural Network）を用いることによる性能改善が報告されている[10],[11]．

また，特定のトリガーワードを検出するキーワードスポッティング（KWS, Keyword Spotting）にDNNやLSTM（Long Short-term Memory）を利用する研究もある[12],[13]．

近年特に注目を集めているのは，End-to-End音声認識と呼ばれる手法で，本稿で紹介したDNN-HMMハイブリッド方式と異なり音声信号から直接単語列を出力する手法である[14]．

4．分散型音声認識へディープラーニングを実装するにあたっての工夫

本章では，DSRへのディープラーニング実装にあたり必要となるモデル学習の高速化およびサービス運用時の推論の高速化について述べる．

4.1　モデル学習時

ディープラーニングモデルの学習には，Theano[15]やTensorFlow[16]，Chainer[17]等のディープラーニングフレームワークを用いる．これらのフレームワークに応じた学習用のソースコードを書き，GPU搭載サーバで学習を実行する．

© 2019 Information Processing Society of Japan

大量のデータを用いる学習の場合，GPUの利用は必須といえる．**図3**にいくつかのGPUとCPUを用いた場合の誤差逆伝搬にかかる時間を示す．左から3列目までは3機種のGPUの結果であり，右に行くほど高性能なGPUである．4列目はあるCPUの結果である．なお，256サンプルあたりの処理時間を示している．また，以下2つのモデルの学習時間を示している．

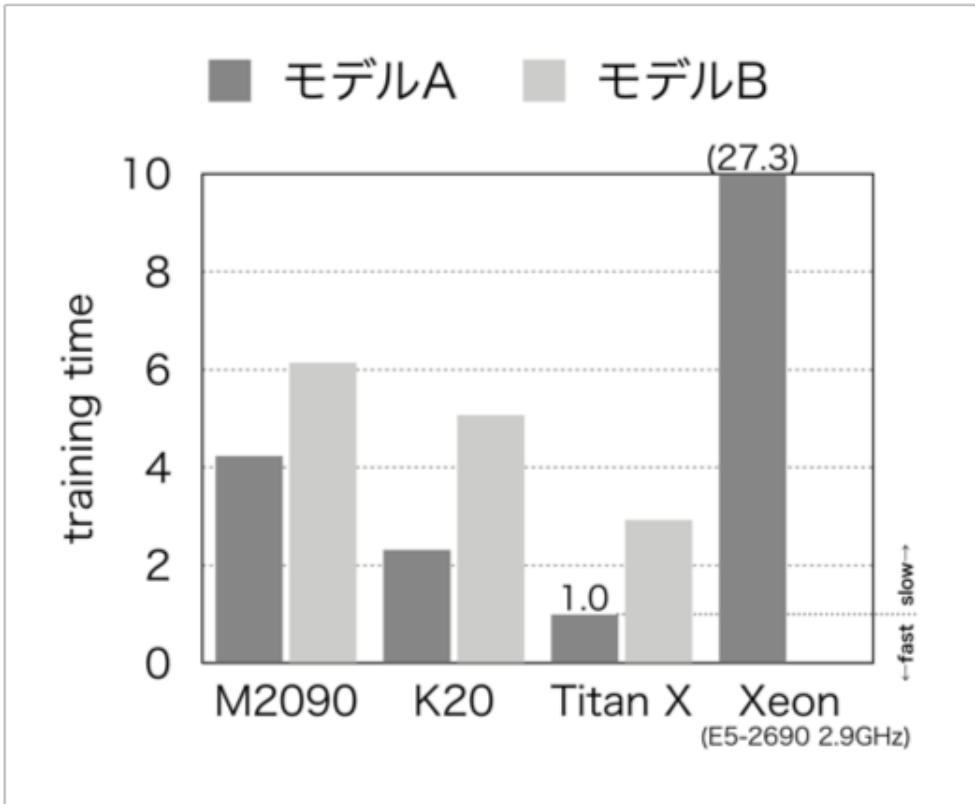

図3　DNNモデル学習時間の比較

- モデルA
  隠れユニット1,024個，レイヤ数5層
- モデルB
  隠れユニット2,048個，レイヤ数5層

モデルAをTitan XというGPUで学習した時の時間を1として表示している．筆者らが運用している分散型音声認識では，学習に用いる音声データの総量は1200時間ほどである．Titan Xを利用してもモデルAの学習には1週間ほどかかる．CPUを用いると数カ月かかってしまうため，GPUを使わなければ現実的な時間で学習を終えることができない．

また，主記憶装置からGPUメモリへ効率的にデータを転送することも学習の高速化の鍵となる．**図4**に示す様に，主記憶装置からGPUメモリへデータを転送する個所がボトルネックになる．したがって，できる限りデータ転送のI/Oを減らすことが望ましい．ベストな方法は，学習に利用するすべてのデータをGPUメモリに読み込んだ後，学習の反復を行うことである．しかし，筆者らが学習に用いた学習用音声データは数TBに及ぶため，高々数十GBの容量のGPUメモリには展開できない．そこで，あらかじめ学習データをGPUメモリにて保持可能なできるだけ大きなサイズに分割しておき，主記憶装置からGPUメモリへのデータ転送回数を減らすことでI/Oを減らす工夫を行った．

© 2019 Information Processing Society of Japan

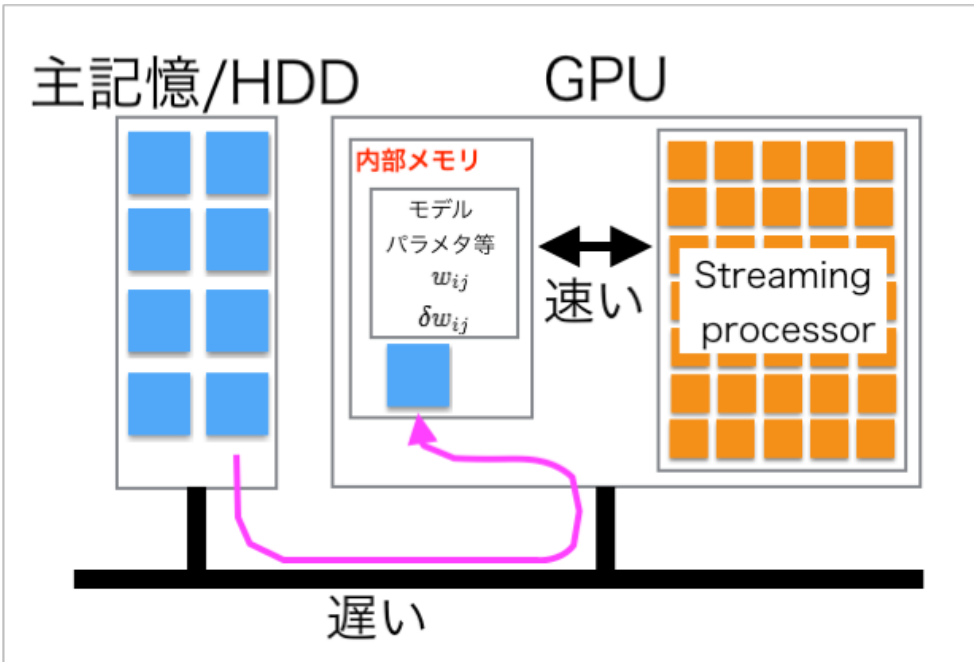

図4　GPUメモリへのデータ転送

また，複数のGPUを用いることで学習を高速化することもできる．異なるGPUで異なるデータを処理するデータパラレルと，モデルを分割して異なるGPUで処理するモデルパラレルの2つの手法がある[18]．本稿では実装が容易なデータパラレルについて述べる．これは，複数のGPUに異なるデータをアサインして学習を行う手法である．誤差逆伝搬をそれぞれのGPUで行ったのち，各GPUで得られた微分値の和を用いてモデルを更新する．すべてのGPUのモデルを同様に更新する必要があるため，学習の反復ごとにすべてのGPU間で微分値を共有するための通信が発生する．したがって，単純に複数GPUに異なるデータをアサインすると，1回の学習の反復ごとに微分値の共有のための通信が発生してしまい，それがボトルネックとなり 1つのGPUで学習した時よりも学習が遅くなってしまう．その対策として，1回の学習の反復に用いるデータ数（ミニバッチサイズ）を大きくして微分値の共有のための通信の頻度を減らすことで高速化が実現できる．**図5**に，ミニバッチサイズを変えた時の学習時間を示す．ミニバッチサイズが512の場合，複数のGPUを用いることで計算時間は小さくなるが，通信時間が加わることでトータルの学習時間は大きくなってしまう．ミニバッチサイズを大きくすることで1つのGPUよりも学習時間が短くなることが分かる．

© 2019 Information Processing Society of Japan

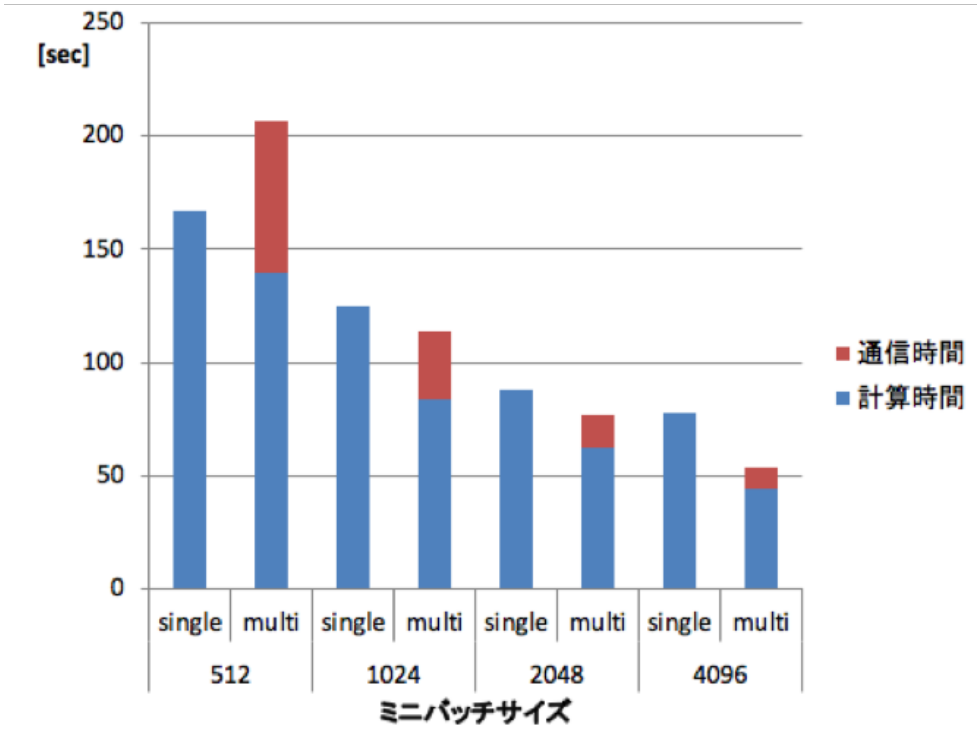

図5　複数のGPUを用いたときのミニバッチサイズと学習時間

しかしながら，ミニバッチサイズを大きくすると音声認識性能が低下してしまう．**図6**に，勾配法の学習率の初期値を0.1に設定してミニバッチサイズを変えた時の音声認識性能を示す．横軸は学習の反復回数（Epochs）を，縦軸は文正解率（S.Acc, Sentence Accuracy）である．なお，凡例のうち「256-fa」というのはフレーム正解精度（frame accuracy）を用いて反復ごとの学習率を調整している．その他の凡例では文正解率を用いて学習率を調整している．ミニバッチサイズが大きくなると性能が低下していることが分かる．これは，学習パラメータを調整することである程度の改善が可能である．**図7**にミニバッチサイズに応じて勾配法の学習率の初期値を変更した場合の音声認識性能を示す．たとえば，「512-0.3」の凡例では，ミニバッチサイズ512，学習率0.3を用いた場合の結果を示す．大きなミニバッチサイズに対して大きな学習率を用いることで，小さなミニバッチサイズと同程度の性能が得られる場合があることが分かる．

© 2019 Information Processing Society of Japan

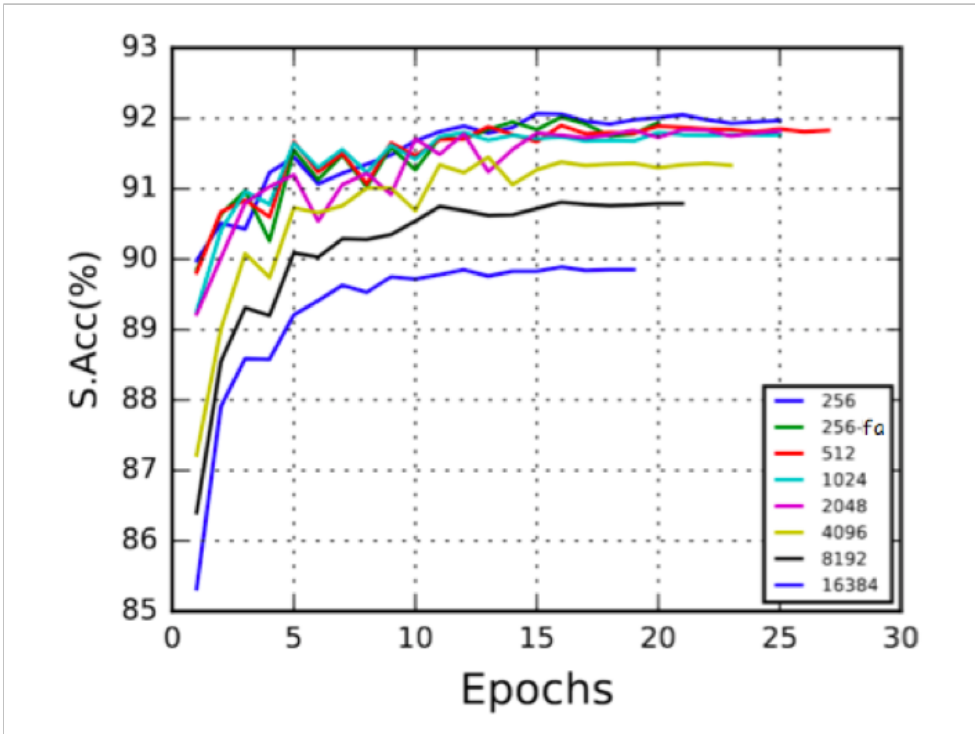

図6　ミニバッチサイズと音声認識性能の関係

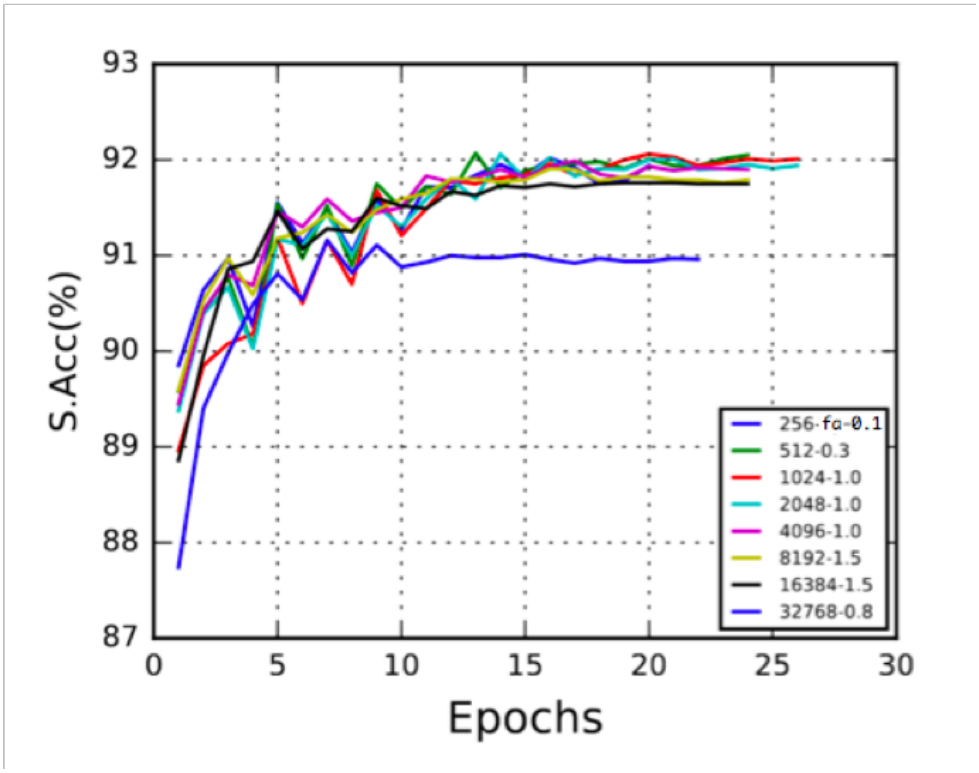

図7　学習率を調整した後の音声認識性能

また，モデルサイズ（隠れユニットの数とレイヤ数）の選択もDSR運用においては重要な要素である．一般的にはモデルサイズが大きい方が性能は良くなるが，学習が現実的な時間で終わるかどうかのみならず，後に述べる推論時にリアルタイムで動作可能なサイズでなければならない．筆者らが運用しているシステムでは，後述する工夫によりリアルタイムでの処理が可能となった隠れユニット1,024個，レイヤ数5層のモデルを利用している．

4.2　モデル推論時

© 2019 Information Processing Society of Japan

筆者らが運用しているDSRシステムでは，できるだけ早く認識結果を返すことが望ましい．そのため，音声信号の長さよりも短い時間で音声認識処理を行いたい．しかし，ディープラーニングモデルの推論は行列とベクトルの積が多く発生する．たとえば，隠れユニットn個のDNNの場合，10msecごとに$O(n^2)$オーダの計算が生じる．この計算の高速化には高性能なGPUやCPUの採用や，CPUに付随するSIMD（Single Instruction Multiple Data）命令拡張を用いたり，モデルを固有値分解して圧縮するなどいくつかのアプローチがある．筆者らは，高性能なGPU/CPUの取得コストとSIMDやモデル圧縮の実装コストのバランスを鑑みて，SIMDを採用した （Intel社のCPUではSSEやAVXといった呼称が用いられている）．

SIMDの利用はコンパイラオプションで指定する．コンパイラが自動的にSIMD拡張命令セットが利用可能な個所を見つけ，SSEやAVXに最適化されたバイナリを生成する．最適化アルゴリズムがコンパイラによって異なるため，**図8**の様に，コンパイラによって同じ演算でも演算速度に差が生じる．検証の結果，インテル社のコンパイラが最速であることが分かった．

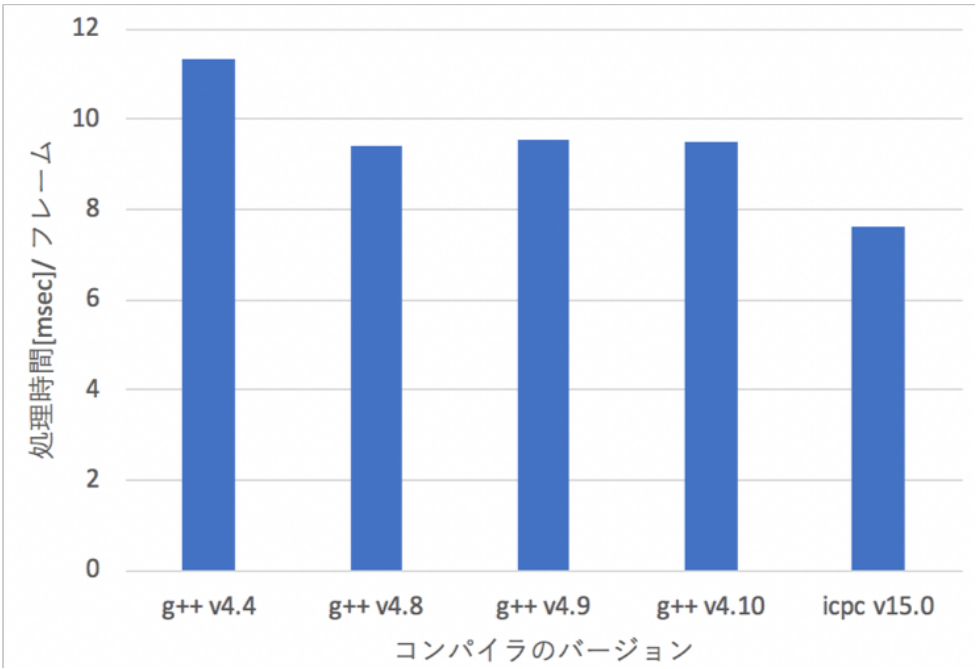

図8　コンパイラによる処理速度の違い

また，さらなる高速化のために，Intel社のMKL（Math Kernel Library）というライブラリを用いることにした．MKLを利用することで行列積の演算をマルチスレッドで実行できるため，さらなる高速化が可能である．**図9**にスレッド数による処理時間の違いを示す．スレッド数の増加により処理時間が短くなっていることが確認できる．

© 2019 Information Processing Society of Japan

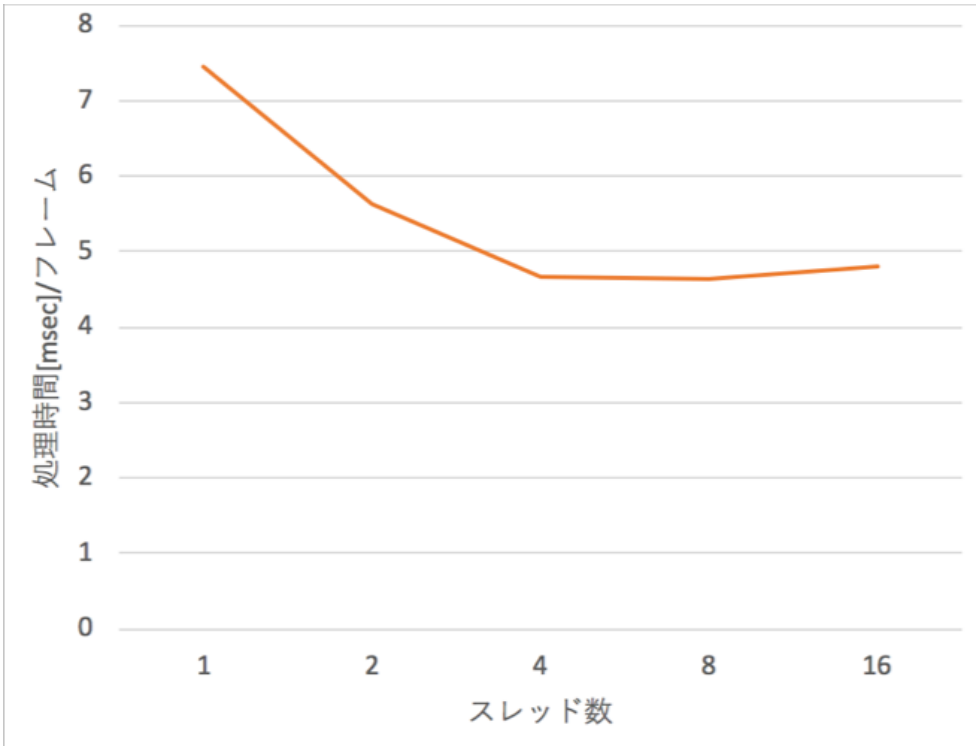

図9　スレッド数と処理時間の関係

また，行列演算は行列サイズがある程度大きい方が効率が良い．これまでは1フレームごとの処理を前提とした行列とベクトルの積だったが，複数のフレームをまとめてブロック化して行列と行列の積にすることで，単位フレームあたりの演算を高速化することができる．**図10**はブロック化するフレーム数と演算時間の曲線である．ブロック化するフレーム数が大きいほど，処理時間が短くなっていることが分かる．

© 2019 Information Processing Society of Japan

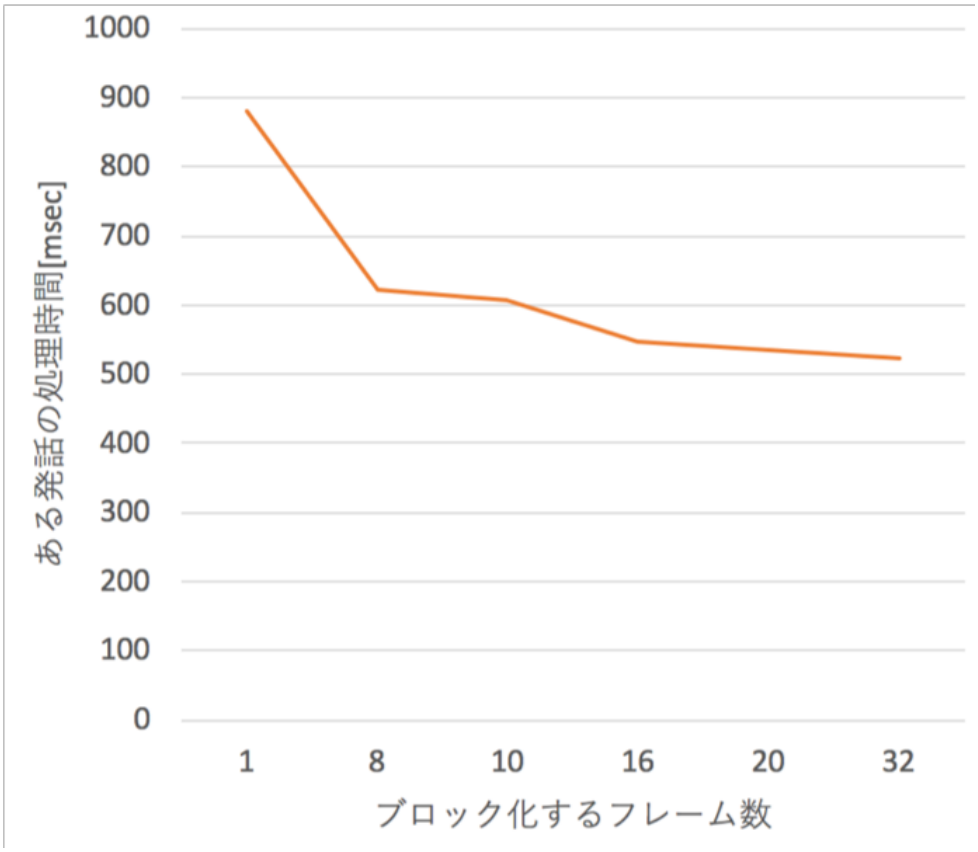

図10　ブロック化するフレーム数と処理時間の関係

4.3　その他の高速化手法

本稿のスコープからは離れるが，ほかにもさまざまな学習・推論の高速化手法が存在する．たとえば，非同期分散学習[19]では，複数の計算ノードを用いた学習時に，微分値の交換を非同期で行うことで高速化する手法が提案されている．

推論の高速化のためには，パラメータ数の大きなモデルを最初に学習し，そのモデルの出力を用いて小さなモデルを学習する蒸留法が知られている．しかし，これはモデルの学習が2回必要になること，十分な性能が得られる小さなモデルのパラメータ数を探索する必要があり，学習時の計算コストが高い[20]．また，モデルパラメータを特異値分解して演算量を減らす手法も提案されているが，特異値数のパラメータの調整および特異値分解したモデルを再学習しなければ十分な性能が得られないため，やはり学習時のコストが高い [20]．ほかにも，モデルパラメータを量子化する手法などが提案されているが，同様の課題が存在する[21]．

5．まとめ

本稿では，分散型音声認識システムへディープラーニングを実装するにあたっての工夫や知見を紹介した．ディープラーニングのモデルは学習時に大量のデータを用いることから，GPUによる高速化が必須となり，GPUメモリに全データが読み込めないことから，GPUメモリに効率良くデータを転送する必要がある．また，運用時はできるだけ早く音声認識処理を行うために，SIMDやMKLを用いてモデルの推論を高速化する方法について紹介した．

ディープラーニングフレームワークの登場により，ディープラーニングを利用するハードルは下がったが，実システムでの運用にあたっては種々のノウハウが必要となる．モデルの学習だけでなく，モデル推論までを含めた高速化がディープラーニングを実システムで利用するにあたっ

© 2019 Information Processing Society of Japan

て重要な点だと言えるだろう.

謝辞　本稿を執筆するにあたり，各種資料を提供して頂いた弊社音声認識開発チームメンバ，特に磯 健一，石井 敬章，磯部 洋平氏に感謝いたします.

参考文献

1）Furui, S. : 50 Years of Progress in Speech and Speaker Recognition Research, ECTI Transactions on Computer and Information Technology (ECTI-CIT), Vol.1, No.2, pp.64-74, 1 (2005).
2）Furui, S. : Selected Topics from 40 Years of Research on Speech and Speaker Recognition, INTERSPEECH 2009.
3）Mohamed, A. R., Dahl, G. and Hinton, G. : Deep Belief Networks for Phone Recognition, NIPS Workshop on Deep Learnig for Speech Recognition and Related Applications, Vol.1, No.9, pp.39 (2009).
4）Hinton, G. et al. : Deep Neural Networks for Acoustic Modeling in Speech Recognition : The Shared Views of Four Research Groups, IEEE Signal Processing Magazine, Vol. 29, No.6, pp.82-97 (Nov. 2012).
5）Iso, K. I., Whittaker, E., Emori, T. and Miyake, J. : Improvements in Japanese Voice Search, INTERSPEECH 2012, pp.2109-2112.
6）河原達也編著：音声認識システム, オーム社（2016）.
7）篠田浩一：音声認識，講談社（2017）.
8）北　研二：確率的言語モデル，東京大学出版会（1999）.
9）Zhang, X.-L. and Wu, J. : Deep Belief Networks Based Voice Activity Detection, IEEE Transactions on Audio, Speech, and Language Processing, Vol.21, No.4, pp.697–710 (April 2013).
10）Bengio, y., Ducharme, R., Vincent, p. and Jauvin, C. : A Neural Probabilistic Language Model. Journal of Machine Learning Research, 3 : 1137-1155 (2003).
11）Mikolov, T., Karafiat, M., Burget, L., Cernocky, J. and Khudanpur, S. : Recurrent Neural Network Based Language Model, INTERSPEECH 2010, pp.1045–1048.
12）Chen, G., Parada, C. and Heigold, G. : Small-footprint Keyword Spotting Using Deep Neural Networks, ICASSP 2014, Vol.14, pp.4087-4091.
13）Chen, G., Parada, C. and Sainath, T. N. : Query-by-example Keyword Spotting Using Long Short-term Memory Networks, ICASSP 2015, pp.5236-5240.
14）Kim, S., Hori, T. and Watanabe, S. : Joint CTC-attention Based End-to-end Speech Recognition Using Multi-task Learning, ICASSP 2017, pp.4835-4839.
15）Theano : http://deeplearning.net/software/theano/
16）TensorFlow : https://www.tensorflow.org/
17）Chainer : https://chainer.org/
18）Krizhevsky, A. : One Weird Trick for Parallelizing Convolutional Neural Networks, ArXiv Preprint ArXiv, 1404.5997 (2014).
19）Dean, J. et al. : Large Scale Distributed Deep Networks, Advances in Neural Information Processing Systems, pp.1223-1231 (2012).
20）Tucker, G., Wu, M., Sun, M., Panchapagesan, S., Fu, G. and Vitaladevuni, S. : Model Compression Applied to Small-Footprint Keyword Spotting, IINTERSPEECH 2016, pp.1878-1882.
21）Quantization-aware Training : https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/quantize/README.md

藤田 悠哉（正会員）yuyfujit@yahoo-corp.jp

© 2019 Information Processing Society of Japan

2007年東京大学大学院情報理工学系研究科修士課程修了．同年日本放送協会に入局．2011年からNHK放送技術研究所にて音声認識の研究開発に従事．2014年にヤフー（株）に入社，引き続き音声認識の研究開発に従事．

採録決定：2018年12月18日
編集担当：新田 清（ヤフー（株））

© 2019 Information Processing Society of Japan