手話コミュニケーション研究会

論文集　２０２０・２１合併号

2022 年 5 月 1 日

手話コミュニケーション研究会

発行所　NPO 手話技能検定協会

巻 頭 言

疫病禍というのは本当に恐ろしいものである。実際に罹患しなくても経済的、社会的な影響が大きい。2020 年度は学会開催もできず、2021 年度からようやくオンラインと会場でのハイブリッド開催が定着してきた感がある。オンライン参加は実際に行かなくてもよいメリットもあり、国際学会などでは何日も日程が必要になるので、日程が重なり地域が異なる場合は選択せざるをえなかったが、そうしたデメリットはある程度解消された。しかし会場に行き、その土地の雰囲気や学会の雰囲気を知るには現地に行かねばならない。また会議以外の場でのトークが重要な場合もある。非公式な情報はそうしたロビーでの会話からしか得られない。そのメリットが失われたことも否定できない。

本研究会も開催ができなかった。そういう事態であれば、リモートワークなどにより、論文提出や発行作業なども進むはずだが、実際そうはならなかった。研究論文というものは机上論のこともあるが、実際の実験や実地調査の結果による検証が必要である。いわゆるエビデンスは物理的な場において得られるものがほとんどである。そしてそのエビデンスに基づく議論によって法則性が見いだされ、論理構成が出来上がっていく。研究会とはそういう場であるから、研究会がなくなると研究論文もでにくくなるというのは必然的なことである。本論文集を史上初めて年度合併号としたのは、論文数が少なかったからではなく、検証の場がなくて、後になって編集したことによる。年度版を一回飛ばして、全部をまとめて号にするという考えもあろうが、論文執筆時点という実績を明確にするため、発行は合併であるが、年度別に分け、ページ配分も同じように年度に分けた。体裁上は発行だけが合併ということにしている。

米国では植民地時代を colonial age と呼び、独立した後の時代を post-colonial age と呼んでいる。それにちなめば corona 禍の時代が過ぎれば post-coronial age がやってくる。カタカナだと同じポスト・コロニアルなので日本英語としては語呂がいい。ただ欧米ではコロナと呼ばず Covid19 という政治的な名称が流布したため、国際化は無理であろう。日本独自のガラパゴス的進化ということになってしまうが、名称はともかく、コロナ禍が終息した後は顕著な変化が起こることが予想される。リモートワークの普及はその典型であろうが、一方で情報産業とそれ以外の生産現場との乖離も進むであろう。農業や建設などはある程度の情報化による合理化はありえても、情報だけでは生産できない。そもそも人間は情報だけを食べて生きてはいけない。「人はパンのみにて生くるものにあらず」とは精神の重要性を説いたものだが、「パンなしで生きる」ことはできない。「のみ」という点が重要なのだ。すなわちポスト・コロニアルに重要なのは実在であり物理的な存在となる。実際コロナ禍の対策は三密を避ける、ワクチン、治療薬という物理的な効果であり、大量の情報が錯綜した中で情報操作はあまり有効でないことが証明されたといえる。マスコミの報道が有効に作用したというエビデンスがない。情報は今後、量より質に変化することも予想される。いわゆるコスパがこれまで判断されてこなかったことがむしろ不思議である。

目次

姿勢推定ソフトの違いにおける手話認識の評価

豊田工業高等専門学校　情報工学科　細川　治度，木村　勉

あらまし　本研究は，豊田高専 木村研究室で開発中の手話辞書システムの改良の一環として，姿勢推定処理エンジンの比較・検討を目的としている．これは，現在使用中の姿勢推定エンジンである OpenPose は精度が高い一方で処理に時間がかかり，検索時間が長くなる問題を抱えているからである．より軽量な姿勢推定エンジンの PoseNet を利用して同様の処理を行い，学習をさせたところ，手話の認識率は低くなる一方で処理の高速化が達成された．これにより，二つの姿勢推定エンジンを利用者のニーズに合わせて選んでもらうなど，両者を使い分けていく方向性を示すことができた．一方で，PoseNet の最大の利点である「ウェブブラウザ上での姿勢推定処理」はまだ完成しておらず，今後の課題となった．

キーワード　手話，機械学習，ディープラーニング，OpenPose，PoseNet，姿勢推定

1. はじめに

豊田高専・木村研究室では，手話認識機能を備えた手話辞書システム[1]を開発している．このシステムでは図 1 のように，まず利用者が手話をする様子を動画に撮り，サーバに送る．サーバ内では，受け取った動画を姿勢推定エンジン（OpenPose[2]）で姿勢推定処理にかける．姿勢推定処理とは，人体の関節などのキーポイントの位置を画像から推定する処理のことである．姿勢推定処理によって生成された動作データ（CSV 形式ファイル）を基に，手話認識エンジンが手話を判別し，確率の高い上位 10 個の単語を利用者に提示している．

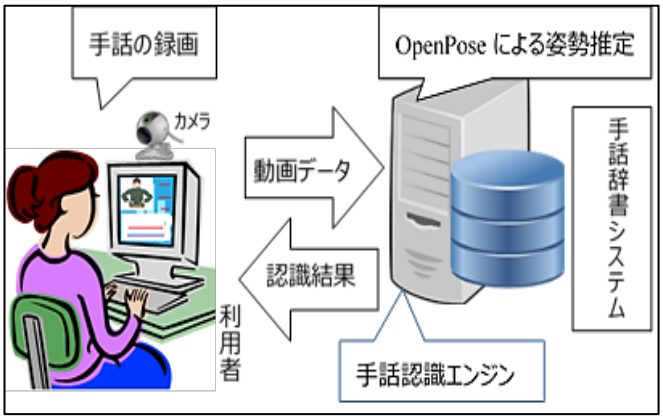


図 1．既存のシステム

ただし，現状ではこのシステムには問題点がある．それは，動画のアップロードや OpenPose による姿勢推定の負荷が高いことなどの理由により，処理に時間がかかっていることである． これにより，現在のシステムでは 7 秒程度の手話動画でも検索に 30 秒以上かかっており，利用者が不便に感じてしまうことが考えられる．また，処理速度の問題は OpenPose に依存しており，サーバ側のスペックの向上だけで処理速度が改善するかどうかは不透明である．

これを解決するための案として，本研究では異なる姿勢推定エンジンを用いたシステムを構築して姿勢推定処理を行い，機械学習を行う．その過程と結果を基に現行のシステムとの比較・検討を行い，両エンジンの利点・欠点を評価する．

2. 研究内容

本研究では姿勢推定エンジンの PoseNet[3]を利用した，図 2 のようなシステムを試作して既存のシステムと比較する．今年度は第 1 段階として PoseNet をサーバ上で動作させることで，主に認識率の違いを評価する．最終的には， PoseNet を用いて利用者のデバイス側で姿勢推定を行い，動作データのみを送るようにすることが目標である．

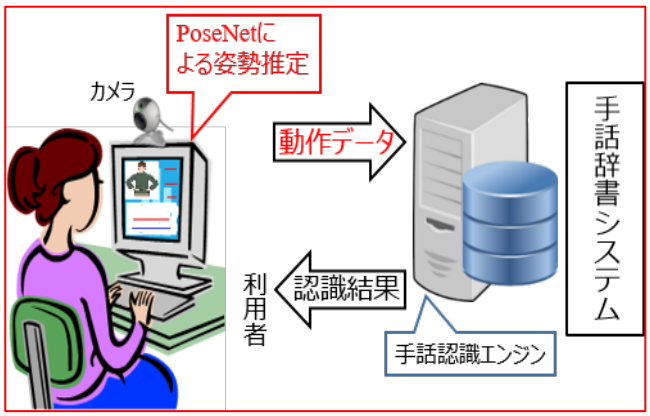


図 2．PoseNet による利用者側での姿勢推定

2.1. OpenPose

現在のシステムで使用している OpenPose は，カーネギーメロン大学が開発した姿勢推定の精度が高いエンジンである．この姿勢推定処理エンジンの最大の利点は，指のキーポイントが推定できることである．OpenPose に手話動画を実行ファイルに読み込ませて，人体のキーポイントの位置を画面上の座標値として取得している．この OpenPose の欠点としては，使用している学習済みモデルが Caffe[2]であり姿勢推定処理に時間がかかっているということが挙げられる．また，

CPU で姿勢推定処理をする場合はさらに時間がかかることになるため，現在のプログラムを CPU で動作させることは難しいと考えられる．

2.2. PoseNet

PoseNet は Google 社が開発した姿勢推定エンジンで，学習済みモデルに MobileNetV1[3]を使用している．この MobileNet はモバイル端末での動作を想定しているため，CPU でも動かせる程度に軽量化されていることが利点である．この他に使用できる学習済みモデルには ResNet[3]があるが，本研究では使用する Python プログラムが対応していないため利用しない．また，PoseNet の推定モードには一人用と複数人用の 2 種類が用意されている．しかし，一人用モードでは入力画像が正方形という制約があるほか，人物でないものを誤認識してしまいやすいという問題がある．そのため精度を要求されるような場面では，画面に映る人物が一人であっても複数人推定モードを使うことが推奨されている[3]．よって本研究においても，姿勢推定の際は基本的に複数人モードを利用することとする．

一方で PoseNet の姿勢推定の精度は，複数人推定モードを使用したとしても OpenPose より低下するほか，指のキーポイントを認識することができないという欠点もある．これらはあくまで研究前にデモプログラムや公式のホームページから得られた知見であり，研究を進める中で詳しく検証していく．

2.3. 姿勢推定処理

手話認識エンジンに手話のデータを読み込ませるためには，動画のフレームごとに画面内の利用者の姿勢を動作データとして数値化する必要がある．

そのために PoseNet では手話動画を読み込ませて，人体のキーポイントの位置を画面上の座標値として取得する．PoseNet では全身の最大 17 箇所のキーポイントを推定することができるが，手話認識のために取得しているキーポイントの位置は，両肩・両肘・両手首の各 XY 座標の合計 12 点である． なお，本年度は処理が高速で，既存のプログラムの流用が容易な GPU 版 PoseNet を主に利用する．理由は，PoseNet の特性を知るためにより多くのデータが必要とされるからである．また，PoseNet は GPU 版もブラウザ版も，姿勢推定において同じ学習済みモデルを使うことができるため，推定結果は同じになる．このことから，認識率の比較においては GPU 版でより多くのデータを集め，ブラウザ版 PoseNet で開発を行う際の参考にすることが望ましいと判断した．ブラウザ版 PoseNet についても可能な限り開発を進めて，比較のためのデータを収集する．PoseNet での姿勢推定の様子を図 3 に示す．

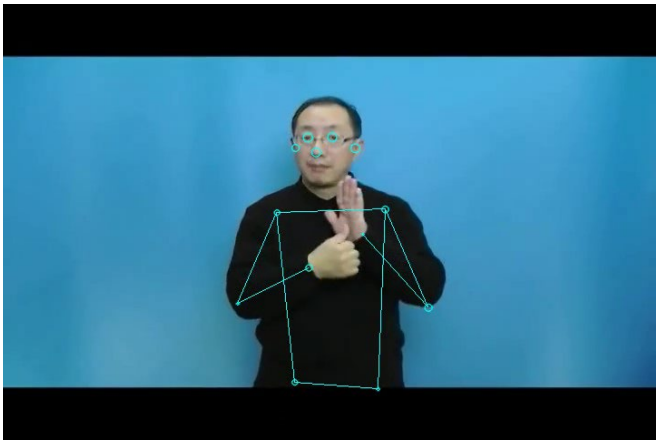


図 3． PoseNet での姿勢推定の様子

一方，現行システムで利用している OpenPose では，1 フレームごとに画面内に映った人物のキーポイントの位置を推定し，結果を JSON ファイルに出力している．OpenPose では全身の最大 136 箇所のキーポイントを推定することができるが，手話認識のために取得しているキーポイントは，両肩・両肘・両手首と両手の XY 座標の計 96 点である．その他にも，JSON ファイルには各座標の信頼度も格納している．OpenPose での姿勢推定の様子を図 4 に示す．

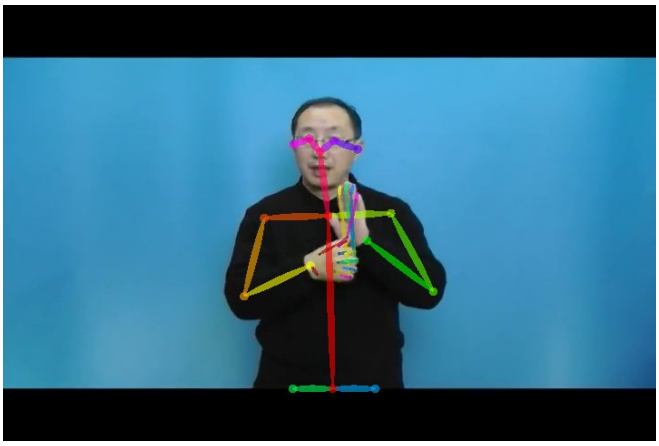


図 4． OpenPose での姿勢推定の様子

2.4. 動作データ

この節では動作データについて述べる．まず OpenPose においては，1 フレームごとに 1 つの JSON 形式のファイルが出力されるので，それを時系列順に並べる．そして，1 つの動画で 1 つのファイルになるようにまとめている．一方，PoseNet では 1 フレームごとに 1 つの Python の配列が出力されているため，それを時系列順に 2 次元配列にまとめ，1 つの動画で 1 つのファイルになるようにファイルに書き込んでいる．OpenPose で生成される JSON ファイルを図 5 に，PoseNet で生成される Python 配列を図 6 に示す．元となる動画データは，手話技能検定試験 6 級の出題範囲

の単語 101 単語を各 100 回撮影したもので，合計 10,100 個を利用する．このうち各単語 90 個ずつの 9,090 個を学習に，各単語 10 個ずつの 1,010 個をテストデータに分配する．

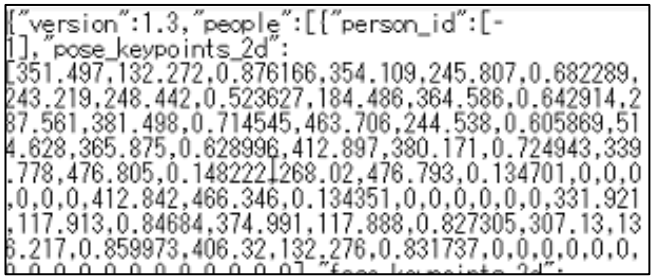


図 5．OpenPose から生成される JSON ファイルの例

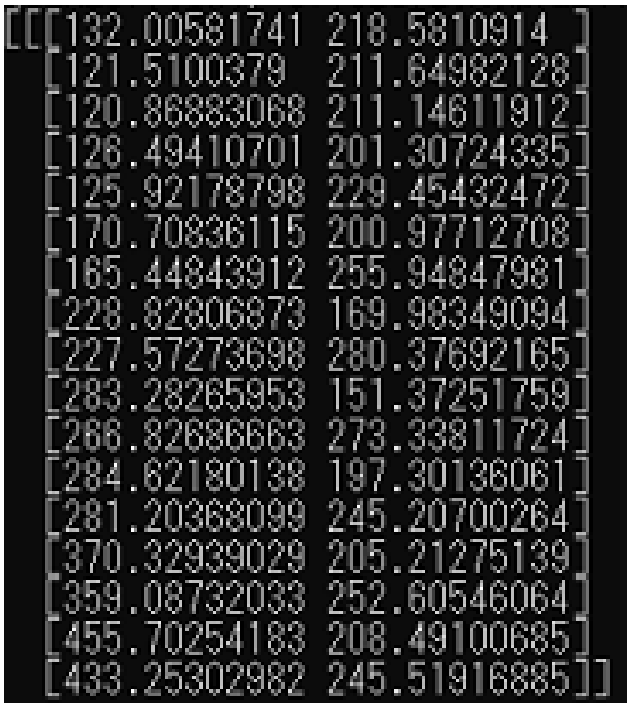


図 6．PoseNet で生成される Python 配列の例

2.5. Chainer と CTC を利用した機械学習

認識率を比較するためには，OpenPose や PoseNet で作成したデータをそれぞれ機械学習にかけて手話の学習をする必要がある．また，キーポイント数は両者で異なるため，学習用およびテストデータ用の動作データは分けられている．

学習に使用する機械学習のフレームワークは Chainer[4]で，ニューラルネットワークの構造は CTC（Connectionist Temporal Classification）[5]を使用する．これらは，機械学習をさせるための Python のプログラムを，Anaconda で作成した Jupyter 用の仮想環境上で動作させている．

2.6. 座標データの正規化

姿勢推定エンジンでは画像からキーポイントの位置（座標）を出力するが，この座標は画像のピクセルに対応した座標系である．例えば，VGA（640x480）の画像の場合，左上が（0,0），右下が（639,479）の座標となる．そのため画像の解像度が異なると，キーポイントの位置も大きく変化する．これに正規化の処理を加えてキーポイントの座標値を 0～1 の範囲に収め，縦横比と人物の位置を統一する．正規化後のキーポイントの配置のイメージが図 7 である．

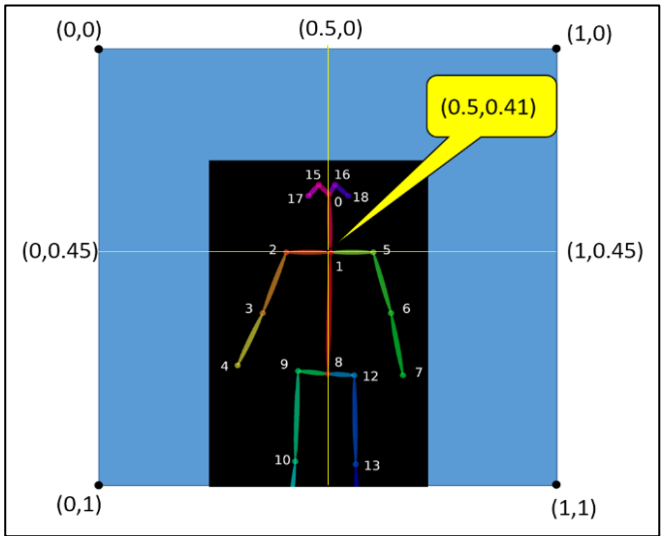


図 7.　正規化後のキーポイントの配置のイメージ

まず，肩幅は身長の約 1/4 であること[6]を利用して，両肩のキーポイントの座標値から利用者の肩幅，そして身長を推定する．これを基に利用者の腕の可動範囲を求め，その人の首の付け根を中心とした正方形（図 7 の水色の範囲）を定める．そして，この正方形の左上を正規化用の座標系の原点（0,0）とする．つまり，全座標値を正規化用の座標系の原点からの相対距離に変更して利用者の身長で割ることにより，キーポイントを適切な座標値にしている．これにより，解像度が異なったり，利用者が画面内で片寄った位置にいたりする場合でも，姿勢推定処理に必要な範囲のみを切り取る事ができるため，認識率の向上が予想される．正規化前の動作データが図 8，正規化後の動作データが図 9 である．1 行に 1 フレームごとの各キーポイントの座標が格納されている．

	A	B	C	D	E
1	451.6364	252.7146	253.8272	258.15	522.
2	451.2404	252.723	253.891	257.4578	522.
3	451.4877	252.5653	253.8923	258.6669	520.
4	454.1862	244.0348	255.7611	249.0874	521.

図 8.　正規化前の動作データ

	A	B	C	D	E
1	0.438343	0.417244	0.384384	0.488965	0.45
2	0.438382	0.41682	0.38297	0.481897	0.44
3	0.438383	0.417561	0.382908	0.483873	0.45
4	0.439527	0.411693	0.384144	0.488761	0.45
5	0.43933	0.411603	0.384808	0.488349	0.45

図 9．正規化後の動作データ

2.7. 姿勢推定エンジンの比較

先述したとおり，現在手話認識エンジンに採用している OpenPose は高精度・高負荷であるが，PoseNet は OpenPose とは対照的に，処理の軽量化と精度の低下が予想される．本研究で比較する項目は，手話辞書システムにおいて求められる性能と両者の特徴的な点を考慮して，認識率・処理時間と GPU 等のリソース使用率の 3 つとする．以下に，それぞれの項目を選択した理由を説明する．

最初の比較項目は，認識率である．ニューラルネットワークを使って学習させるためには，ある程度の精度を持った動作データが必要である．つまり，それぞれのキーポイントが高い精度で推定できているかによるということである．動画によっては，腕の重なりやシャッタースピードによるブレなどによって，推定が難しいキーポイントもあるが，これらは推定エンジンによって傾向が変わると思われる．また，手話認識に必要な学習済みモデルとそのために必要な学習用データは，推定エンジンごとに用意する必要がある．もし学習用データの推定精度が低ければ，学習の際に手話単語ごとの特徴を掴めないことも発生すると考えられる．このようにいくら処理が早くても，姿勢推定の精度が低ければ辞書システムとしての使用に耐えうるとは言えなくなる．このことから，最優先の比較項目として手話単語の認識率を比較することとした．また PoseNet は，GPU 版もブラウザ版も共通の学習済みモデルで姿勢推定処理を行うため，認識率の比較の際は両者を同一のものとして扱う．そのため，PoseNet で学習データを作成する場合は処理が高速で動作データを多く作成できる GPU 版を利用する．

2 つ目の比較項目は，処理時間である．これは，動画を姿勢推定エンジンに送った後，どれだけの時間で推定結果を出力するかを比較する．計測するのは，1 つの手話動作の姿勢推定にかかった時間である．一単語を調べるために何十秒も待たなければならないのは，利用者にとってはストレスとなる．本研究では，この処理時間を短縮することも目的としている．

最後の比較項目は，リソースの使用率である．リソースとは，GPU や CPU の姿勢推定処理中の平均使用率を示す．先行研究では姿勢推定処理中の GPU の負荷を計測した事例はなく，処理に時間がかかる理由が明らかになっていなかった．本研究ではこの理由を探る一環として，姿勢推定処理中のリソース使用率を計測する．また，先述の通り PoseNet は処理時間が短いとはいえ，GPU 版 PoseNet でも複数人推定を使用することになる．これによって OpenPose と比較して負荷が高くならないかどうかという点についても検証する．他にも，ブラウザ版 PoseNet は CPU で姿勢推定処理を行うため，GPU での処理に比べて負荷が高いことが予想されている．そのため，ブラウザ版 PoseNet においても準備ができ次第，CPU の使用率を計測する．これは，あまりに負荷が高いと利用者の端末では動かないことも考えられるためである．

3. 研究結果

3.1. 開発環境

今年度は，PoseNet を動かす Python のプログラムを，GPU 上で動作させることを想定している．そのために，Python 用の仮想環境構築ツールである Anaconda を用いる．開発環境を表 1 に示す．

表 1．開発環境

使用した PC のスペック	
CPU	AMD Ryzen 5 2600X 3.6GB@3.60Hz
RAM	16.0GB
GPU	NVIDIA Geforce RTX 2060
使用したエンジン・ライブラリ	
機械学習を行う際に使用	Chainer7.2.0
	CuPy7.2.0
	NumPy1.19.0
骨格情報を取得する際に使用	OpenPose1.5.1
	PoseNet2.0
	TensorFlow-gpu1.15
	Pillow7.2.0
	OpenCV3.4.5.20
	PyTorch1.5.1
	Matplotlib3.2.2
	CUDA10.2
開発言語	
Python3.7	

なお，ブラウザ上で動作させる場合プログラミング言語は JavaScript になり，なおかつ HTML ファイルからスクリプトを CPU へ読み込むため，GPU は使用していない．他にも，現行の OpenPose のシステムについては，必要としているライブラリは OpenCV のみであるため，GPU 版の PoseNet と同じ環境を用いて動作させることとした．

3.2. プログラムの実装と機械学習

現行システムの OpenPose，及び GPU 版の PoseNet を利用できる環境を構築し，手話の様子を撮影した 10,100 個の動画から動作データを生成するプログラムを作成した．その後，プログラムから生成された動作データを学習データとテストデータに振り分け，Chainer 上で CTC 手法を用いたネットワークを利用し，学習させた．比較するのは，現在使用している OpenPose を利用したプログラムで生成した動作データを，PoseNet と同じ Chainer 上で CTC 手法を用いて学習させたときの認識率である．なお，動作データの元となる手話動画は共通だが，学習とテストに用いた動作データは両エンジンで分けられている．これは，OpenPose では手型の推定も行っているため，データサイズが PoseNet での動作データと異なるからである．

ブラウザ版 PoseNet においては，姿勢推定処理を行うプログラムを，HTML ファイル内に JavaScript で記述した．また，この HTML ファイルを動作させることで，姿勢推定処理を CPU 上で済ませることにも成功した．ただし，現状で姿勢推定処理をするためには Python プログラムであらかじめ動画をフレームごとに分割する必要がある．そのため，ブラウザ上だけで実行することはできなかった．

3.3. 認識率の比較

プログラムを実装し，生成した動作データを学習させてテストした結果， 既存の OpenPose を利用した学習データでは認識率が 90.3%であったが，GPU 版の PoseNet で構築した学習データでの認識率は 73.3%となり，PoseNet の方が約 17%低下した．

ただし先行研究[3]において，CTC 手法を用いる前は OpenPose で生成した学習データであっても認識率が 75%程度であったので，現段階でも PoseNet を利用した学習データは初期の手話辞書システムと同等の認識率が確保されているといえる．

3.4. 処理時間とリソース使用率の比較

処理時間とリソース使用率は，姿勢推定処理をしながら測定用のプログラムを動かすことで測定した．測定は，各推定エンジンで手話動画から学習データとテストデータを合計 10,100 個生成する間に行った．なお，ブラウザ版 PoseNet での測定結果は，約 100 個の動画を姿勢推定処理にかけた際の結果である．

使用率の比較を表 2 に示す．この結果から，GPU 版の PoseNet では OpenPose と比較して，処理時間と GPU の使用率が改善されていることがわかる．特に処理時間に関しては，PoseNet の処理時間は OpenPose での処理時間の 1/5 以下となっている．なお，ブラウザ版の PoseNet では GPU 版の PoseNet よりも姿勢推定処理にかかる時間は大幅に長くなっているが，現在の OpenPose での処理時間よりは短くなっている．ただ，一単語の検索に 20 秒以上かかるのは辞書システムとしては実用性に乏しいため，今後の改善が求められる．また，リソース使用率については，ブラウザ上で動作するプログラムであることから CPU の使用率を測定している．そのため，数値が高くなっている．この CPU 使用率は，あくまでもデスクトップ型パソコンに搭載された CPU での結果であり，よりスペックの低いノートパソコンや携帯端末などの CPU で測定した場合は，数値がさらに高くなる可能性がある．これらのブラウザ版 PoseNet のデータは，検証回数が他の二種類のプログラムよりも少なくデータ収集に使用した環境も一つだけであるため，来年度以降さらに検証する必要がある．

表 2．処理時間とリソース使用率の比較

	OpenPose	PoseNet（GPU）	PoseNet（ブラウザ）
処理時間（秒）	31.30	5.93	22.37
平均リソース使用率（%）	6.50	3.48	34.92

3.5. 正規化

OpenPose と PoseNet の両者で生成した動作データに正規化の処理を行い，再度同じ機械学習の環境で学習させた．姿勢推定するために切り取る範囲については，数値が 0 から 1 に収まらないものがあったため，切り取る正方形の長さを動画に写っている人物の身長から身長の 2 倍へと変更した．これは，動画に写っている人物が前かがみであった場合に，肩幅を実際よりも狭く推定していたためであると考えられる．

正規化により，OpenPose での学習結果は認識率が 90.3%から 94.3%に，PoseNet での学習結果は 73.3%から 74.8%に上昇した．ただし，上昇の幅は限定的で，テストデータはそれぞれ 1,010 個であることを考慮す

ると，違いは動画の数にして数個分に満たない．

なお，これは使用している手話動画の性質によるものが大きいと考えられる．学習に使用した手話動画は，いずれも手話をする人物が画面中央にいるため，正規化の役割の一つである「動画内で片寄った位置にいる人物を，姿勢推定する範囲の中央に移動させる」という機能が必要とされないからである．また，解像度も720x480 と 800x600 の 2 種類しか学習データの動画にはなかったので，正規化がどの程度効果的であったかどうかは確認できなかった．今年度は解像度の異なる動画や人物が片寄った動画を用意することができなかったので，今後の研究課題として引き継ぐ予定である．

また，OpenPose と PoseNet で共通しているキーポイントについては，動作データ内でのキーポイントの並びを統一した．これにより，今後新しい姿勢推定エンジンを使う際でも，統一されたフォーマットで比較することができるようになった．

3.6. 手話単語ごとの認識率の比較

単語別の認識率を比較すると，OpenPose では 101 単語中 80%以上の認識率がある単語は 97 個，そのうち 100%正解したものは 62 個にのぼる．その他の結果は図 10 に示す．先行研究[1]では，「明日」と「明後日」など，使う指の違いによる誤認識が課題となっていたが，今年度は OpenPose においてはそれらの単語も最低 60%の精度を記録できた．これは，Chainer と CTC による学習に切り替えたことが要因として考えられる．また，奥行き方向の動きの差についても，OpenPose では 80%の精度で認識できた．

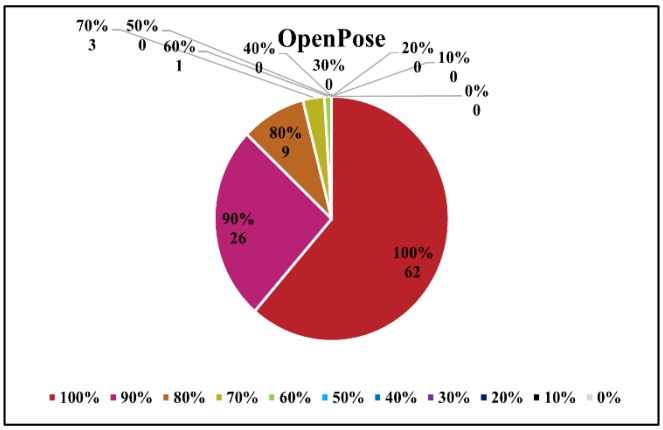


図 10．OpenPose の単語別の認識率

一方 PoseNet では，認識率が 80%以上の単語は 58 個，そのうち 100%正解したものは 25 個に過ぎない．その他の結果は図 11 に示す．さらに，認識率が 10%にまで低下したものもあった．ただし，認識率が 0%のものはない．このことから，PoseNet でも全く特徴をつかめなかった手話単語はないとみられる．

なお，PoseNet の認識率向上策として，指の動作データを取り除いて PoseNet と同じデータサイズにした OpenPose の学習データと，PoseNet のテストデータを使った学習も行った．しかし認識率は向上せず，逆に 50%にまで低下した．

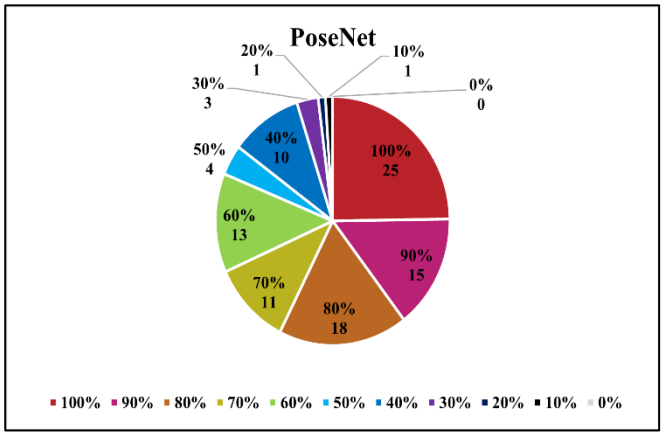


図 11．PoseNet の単語別認識率

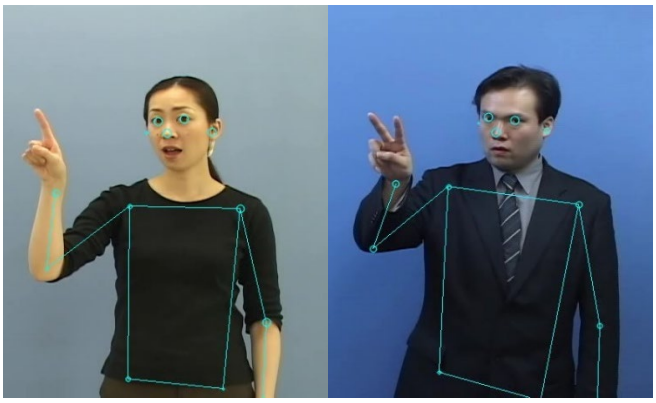
4. 考察

4.1. 認識率に関する考察

先行研究[1]においては，PoseNet と同等の 75%の認識率であっても，翻訳には適さないが辞書として利用する際は実用に耐えうるとする結論になっている．このことから，PoseNet は姿勢推定エンジンとして辞書システムに組み込めるだけの十分な推定精度を有していると言える．なおブラウザ版 PoseNet においても，同じ学習済みモデルを利用して姿勢推定処理を行っているため，動作データもほぼ同じものになり同程度の認識率になると考えられる．

4.2. 単語別の認識率に関する考察

PoseNet において認識率が下がった理由としては，OpenPose では手型を推定できるが，図 12 のように PoseNet では推定できないことが挙げられる．

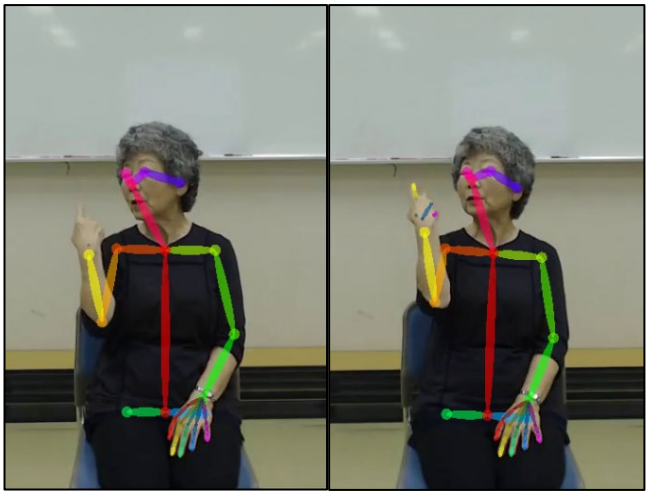


（a） 明日　　（b） 明後日

図 12． PoseNet による姿勢推定の様子

例えば，「明日」という単語は図 12（a）のような手型で，顔の横から前に出す動作をする．「明後日」は，図 12（b）のような手型である．「あなた」は，相手を人差し指だけで指し示す動作である．OpenPose では手型の推定もできるため，「明日」の単語の認識率が 90%であったが，PoseNet では手型の推定ができず，「明後日」や「あなた」と誤認識してしまい，認識率が 10%に低下した．今後 PoseNet での精度の向上を図るためには，追加の処理が必要であると考えられる．

一方で，両方の姿勢推定エンジン共通の課題も確認された．一つ目は，「昨日」と「一昨日」という単語などでみられた，特徴が見分けづらい単語の区別である．「昨日」に至っては，OpenPose でも認識率は全単語中唯一の 60%で最下位，PoseNet でも 40%であった．この単語は，「明日」「明後日」の関係と同じで，違いは手型，しかも指一本の違いである．それでも認識率が下がったのは，この手話単語の動きにある．「昨日」は図 13（a）のように，手を自分の肩越しに後ろへ動かす．「一昨日」も同様である．このことによって，指が隠れてしまうことが多いのである．「一昨日」の手話の手型はチョキであるが，実際に学習用の手話動画でも図 13（b）のように指が数フレームしか映らずに手の甲に隠れてしまっていたケースが発見された．



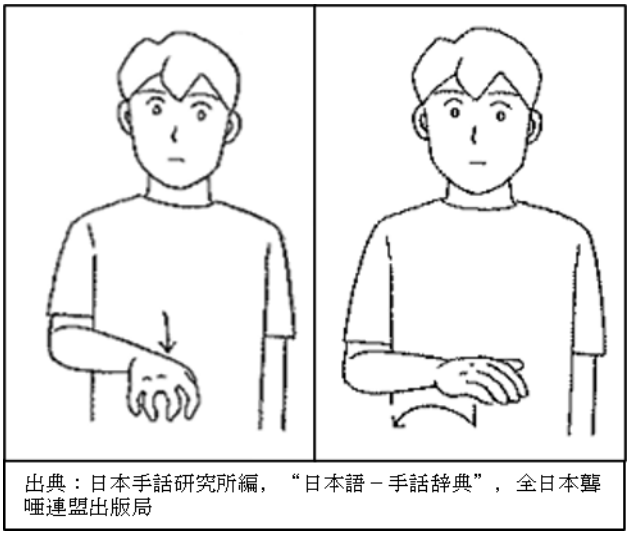
（a）昨日　　（b）一昨日

図 13．OpenPose による手型の推定の失敗例

OpenPose の推定精度は高いが，手型の推定になると，シャッタースピードの遅さによるブレなどで精度は落ちてしまう．また，OpenPose は写っていないキーポイントは 0 と出力するため，手型の推定は安定しない場合がある．手型の推定ができない PoseNet では，なおさら両者の区別は困難である．これらのことから，認識率の向上の余地はまだあることが分かった．では逆に，なぜ 60%や 40%の認識率が確保されたのか．それは，「昨日」「一昨日」の手話の動きだけは特徴として掴めていたからであると考えられる．指の状態が分からなくても，動作からは少なくとも「昨日」か「一昨日」であると判断したのではないかと考えられる．

なお，利用する先の手話辞書システムでは，確率が高い単語は確率順に候補として提示される．よって，利用者はそこから探している単語を選択すればよいので，手話辞書システム自体の変更は必要ないと考えられる．同様の理由により，同じ動作・手型で複数の意味を持つ単語も，検索する際はどちらの意味として認識されても問題ないと考えられる．

もう一つは，利用者ごとの動きの違いである．手話はその単語にかかわる事物を表現する言語のため，利用者による多少の違いがある．これが誤認識につながる場合がある．特に誤認識の原因となっていたのは，手首のみを上下・左右に揺らす動作である．例えば，図 14（a）の「場所」という単語は，全指を曲げた利き手を前に置く動作である．図 14（b）の「子ども」は手のひらを下にして，左から順番に置くように移動する動作である．手型の認識がうまくできず手首の動きが小さいと，「子ども」と表現したつもりが「場所」として認識されてしまう．逆に，正確に手首の座標を認識できず，座標値のぶれを揺らす動作と判断した例もあった．前者は精度が高い OpenPose に，後者は精度が低い PoseNet で多く発生していた．このことから，指のキーポイントの有無だけではなく，認識エンジン自体の特徴が手話の認識に影響していると思われる．



（a）場所　　（b）子ども

図 14．手話単語の動作の違い

他には，指を強調するために動作を止め，それが原因で別の単語として認識されてしまったものもある．手話の初学者がこのシステムを利用する際には個人差という問題は起こらないが，手話を知らない人が単語を見よう見まねで調べたいときは，この問題が発生してしまうのではないかと考えられる．

4.3. 認識率向上のための解決策についての考察

機械学習での認識率が高止まりになった理由として，もう一つの理由が考えられる．それは，学習においては各キーポイントの動作の大きさが重要とされるではないかということである．この推測は，4.2 において手首の動きの差の例があったように，動作の差が少ないと手話認識が難しくなることから推測したものである．動作量に注目すれば，キーポイントの動画内での座標値の変化に注目することになる．つまり機械学習では，その前のフレームの同じキーポイントの位置との相対的な差を重視しているのではないか，ということである．次年度以降は，機械学習では解像度の違いが学習に関係するかを調べるために，今ある動画の縦横比を引き延ばしたり，別の解像度で撮影した新しい動画を用意したりして検証する必要がある．

4.4.「ドラえもん手話」との関連

本年度の研究では，指をキーポイントとして推定しない PoseNet を利用したが，認識率は大幅には下がらなかった．結果として，PoseNet でも約 6 割の単語は 80%以上の認識率を確保している．これに関しては，手を握った状態での手話（ドラえもん手話）でも意思疎通はある程度できるという研究[7]があり，この研究の通りになったといえる．ただし，違いが手型のみの単語同士の場合は認識率が著しく低下している．

5. 今後の課題

本年度は，主に現行の OpenPose と新規の PoseNet で手話の動作データを作成し，機械学習にかけることで手話の認識率の比較や，現在の課題とされている処理時間やリソース使用率の検証を行った．このために多くのデータを必要としていたため，処理時間が短い GPU 版 PoseNet を利用して，サーバ側で PoseNet を動作させていた．また，より多くのデータを得られたことで，前処理としての正規化の有用性や，OpenPose よりも認識率が下がる理由を考察することができた．しかし，システムのコストや負荷を改善するためには，PoseNet を利用者のデバイス上で動作させる必要がある．現在，JavaScript を利用してブラウザ上で姿勢推定をすることはできたが，これにはあらかじめ動画をフレームごとに分割する必要がある．そのため，ブラウザのみで完結しているシステムとは言えない．これらのことから，今後は動画をそのまま姿勢推定処理できるような JavaScript のプログラムが必要であると考えられる．さらに，他の姿勢推定エンジンについても調査・比較し，手話辞書システムに最適なエンジンを探し出す．

6. おわりに

今回の研究の結果から，PoseNet を用いることで，認識率は劣るが処理時間を短縮させることができた．また，画像の前処理を行うことで，OpenPose は従来よりも高い認識率を発揮できることが確認できた．

しかし，PoseNet の最大の利点は，その認識率のままにブラウザ上で姿勢推定処理を動作させることができる点である．これを実現するため，今後は成果を基に PoseNet をブラウザ上で動作させるシステムの開発を進める．

また，OpenPose と比較した際の PoseNet の利点・欠点を調べることができたため，利用者のニーズに応じて両者を使い分ける方向性を示すことができた．具体的には，利用者が検索する前に早さを優先するか正確さを優先するかを選択してもらうことである．これにより選択に応じて，正確さを優先する人には OpenPose を，速さを優先する人には PoseNet を組み込んだシステムを利用してもらうことができる．

他にも，PoseNet での認識結果が不満だった利用者には，時間はかかるものの，追加でより正確な OpenPose を用いたシステムの方を利用してもらうように web ページ上で誘導するなどの使い分けが将来的には考えられる．

7. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）の助成を受けたものである．

文　献

[1] 髙橋佑汰，木村勉，神田和幸，"機械学習を用いた手話認識に関する研究",電子情報通信学会技術研究報告， 118(440)，59-64，2019

[2] GitHub-CMU-Perceptual-Computing-Lab_openpose_OpenPose_Real-time multi-person keypoint detection library, https://github.com/CMU-Perceptual-Computing-Lab/openpose 2020 年 11 月 4 日閲覧

[3] “tfjs-models_posenet at master ・ tensorflow _tfjs-models ・GitHub”, https://github.com/tensorflow/tfjs-models /tree/master/posenet, 2020 年 7 月 22 日閲覧

[4] ディープラーニング入門: Chainer チュートリアル，https://tutorials.chainer.org/ja 2021 年 1 月 12 日閲覧

[5] Connectionist Temporal Classification (CTC) を用いた音素認識，https://qiita.com/hirokisince1998/items/f59d3e8434d6326c48cc 2021 年 1 月 12 日閲覧

[6] インテリアデザインに必要な知識， https://www.jetc.co.jp/books/110076/110076-psample.pdf 2021 年 1 月 12 日閲覧

[7] 神田和幸,”ドラえもん手話の実例と NMS の情報伝達” 可視化情報学会誌. Suppl. 24(1), 277-278 (2004)

CTC を用いた手話動画からの単語認識に関する研究

豊田工業高等専門学校　情報工学科　秋田 大輔，木村　勉

あらまし　本研究では機械学習を用いた手話認識において，手話動画に含まれる手話とは関係のない動作による誤認識の問題を解決する．本研究では音声認識の手法である Connectionist Temporal Classification (CTC) を用いて，手話動作のみを認識させるようにする．また，画面内に複数の人が入ってしまった場合に認識精度が下がってしまうことが判明したため，利用する人だけを認識するようにプログラムの改良を行った．その結果，認識精度が昨年より向上した．動画を撮る際の余分な動作については，動作がある動画も機械学習させることで，認識させることが可能であるとわかった．

キーワード　手話，機械学習，CTC，OpenPose，Chainer

1. はじめに

これまで豊田高専木村研究室では，深層学習による手話の認識に関する研究を行ってきた．現在では，手話技能検定試験 6 級に指定されている単語に対して，約 93%の認識精度を持つ学習ネットワークの開発に成功している[1]．

だが，特定の環境の下で撮影された動画に対する認識は成功しているが，その他の撮影環境においては十分な評価が行われていない．深層学習に用いた学習用の手話動画は，ホームポジションと呼ばれる，手を降ろした姿勢から単語を表現する動作に移り，動作が終了したら再びホームポジションの姿勢に戻る（図 1）という動作である．一般的に流通している手話単語の動画データもこのような形式となっており，これまで行っていた研究でも，この形式のデータを認識して検証を行っている．

しかしながら，実際に利用者が手話辞書システムとして利用する場合には，動画撮影をする際，手話表現の前後に，図 2 に示すように録画操作などの不要な動作が混入してしまう．この不要な動作が手話単語認識において誤認識の原因となってしまうと考えられる．

本研究では，手話辞書システムが一般利用者の環境において使用される場合の問題点の解決を行う．

2. 目的

本研究は，手話表現中に無関係の動作を含む動画からでも正しく認識できる手話認識エンジンを開発することを目的とする．さらに，昨年度まで行われてきた研究で使用してきた手話動画に加え，様々な環境で撮影を行った動画に対しても手話が正しく認識できることを目指す．

先行研究[1]にて手話文章からの単語の抽出が行われたが，本研究では認識対象を文章ではなく単語とし，一般利用者自身が撮影を行う環境に適合した，認識エンジンを目指す．

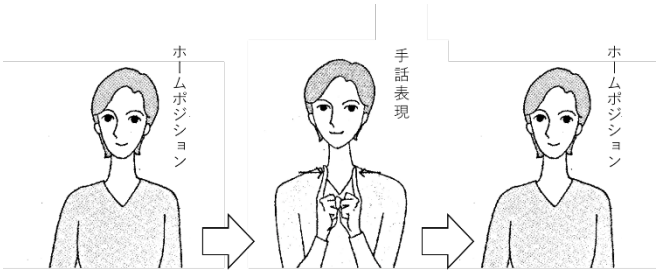


図 1　手話表現とホームポジション

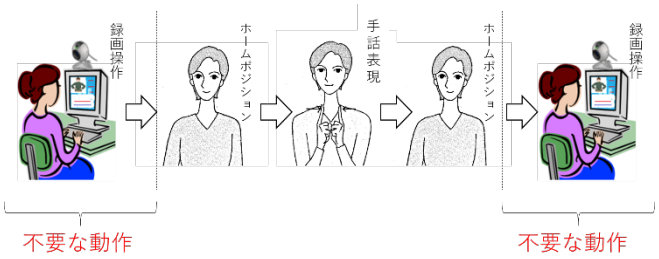


図 2　不要な動作を含んだ動作の流れ

3. 研究の方針と準備

本研究を進めるにあたって，重要な内容や採用した手法についてまとめる．

3.1. 手話単語認識の流れ

本研究では，手話単語認識を図 3 のように行っている．はじめに，手話を行っている動画を読み込み，姿勢推定ソフトの OpenPose[2]を用いて関節点などのキーポイントを取得する．これをフレームごとに JSON 形式のファイルに出力し，さらにこれを動画ごとに 1 つの CSV 形式の動作データにまとめる．これを用いて機械学習を行うことにより，手話単語の認識を行う．

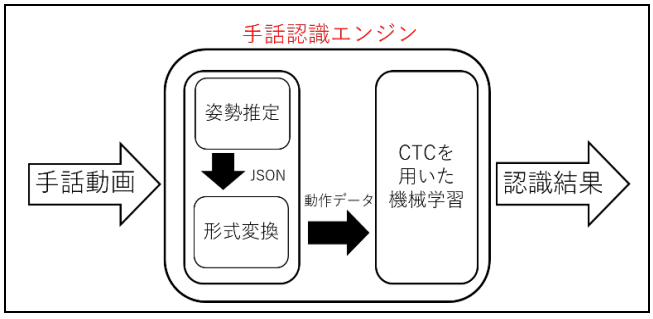


図 3　手話単語認識の流れ

また今回行う機械学習では，音声認識で用いられている Connectionist Temporal Classification 手法を採用する．

3.2. Connectionist Temporal Classification

Connectionist Temporal Classification[3]（以下 CTC)とは，End-to-End 音声認識に用いられるアプローチの 1 つである．損失関数として，解析後の出力データに対する損失値の計算に使用されている．また，音声認識の分野において,「音素」と「どの音でもないブランク表現」の認識をするために採用されている手法である[4].

同じ意味を示す音声データでも，音素の発音の長さやブランクによって差が生じてしまう．CTC はこのデータに対して，ブランク（”_”）の挿入と，音素の連続を表現させる．ここで CTC により「あいう」の音声認識がどのように行われるかを図 4 に示す．文字の書いてある丸は音素を示しており，白い丸はブランクを示している．矢印で示す音の並びはどちらも「あいう」と発音した音声データであるが，これを CTC ではそれぞれ「ああ_い_う」,「あいうううう」というように認識を行う．その後，連続した音素を 1 つにまとめる，ブランクを削除する,という 2 つの処理を行うことで，どちらも「あいう」というように認識を行うことができる．このように CTC 手法では，データごとに生じてしまう差を，認識してから取り除くというプロセスにより，認識したデータの情報を簡潔にできる特徴を持っている．

CTC では音が含まれていないことだけでなく，認識結果として曖昧な場合にもブランクとして判断される．この処理により，音と音のつながりで発生してしまう連音による誤認識の発生を防いでいる．

音声認識において，既に音素単位から文字単位や単語単位へラベルを拡張し機械学習が行われていることから，本研究の手話認識においても，単語単位で CTC 手法を用いた認識を行う．

CTC 手法を用いた手話認識への試みは昨年度の研究[1]から行われており，この手法が採用された理由として，単語から文章を認識する際に発生するラベル数の差を解決できるということが挙げられる．例えば，クロスエントロピーや平均二乗誤差などを損失関数とするニューラルネットワークの学習では，出力データへ割り振られるラベルと正解ラベルの系列長が等しくなければならないため，入出力ラベルの数の統一，またはデータに対するフレーム単位でのラベルの整理が必要であった[5]．CTC 手法ではフレームの 1 つ 1 つに対して解析を行い，ラベルの割り振りを行う．ラベル数が膨大になるが，これをブランクの挿入とフレーム処理によって簡潔な形にする．これにより，文章データを学習させる場合に，必ずしも同じ単語数ではなくても認識が行えるようになっている．

CTC 手法では，複数の単語が入っているデータに対して，各ラベルを独立して認識する．これはデメリットとして，単語と単語間の関係性や，文法を一切考慮しない認識方法となってしまう．

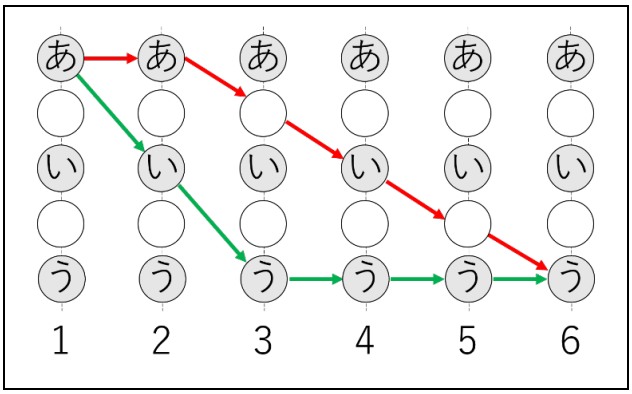


図 4　CTC による「あいう」の認識

3.3. OpenPose

本研究では，手話データから関節点などキーポイントの取得のため，OponPose を採用する．OpenPose とは，カーネギーメロン大学の Zhe Cao らが「Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields」[6]の論文で発表した，人物の姿勢を推定できる手法である．画像や動画から体の動きを検出することができ，目や鼻などの顔パーツから腰や足首などの関節点まで検出が行える．それらの関節を点で示し，線でつないで姿勢を推定する．図 5 は実際に OpenPose を利用して姿勢推定をしている画像である．入力データから取得した座標データは, x 座標と y 座標の 2D データとして出力される．また，特殊な設備がなくてもカメラ 1 台で複雑な姿勢の解析が可能であることが大きな利点である．

図 5　OpenPose による姿勢推定

出力された座標データは JSON 形式となっている，取得する関節は，肩，肘，手首，指を含む掌を取得しており，48 点となっている．なおそれぞれ x 座標と y 座標に分かれるため, 計 96 個のデータを入力データとする．

3.4. ニューラルネットワークの構築

本研究ではニューラルネットワークを LSTM（Long-Short Term Memory）で構成する．LSTM とは

時系列を持つデータの学習に優れた RNN（Recurrent Neural Network）の 1 つで，勾配消失問題の解決のために 1997 年に提唱されたものである．RNN では過去の出力値と現在入力値と合わせて扱うことができ，時系列データを持つため，前後の情報が重要となる音声認識や動画認識において有効性が活かされている．LSTM は RNN の改良モデルであり，過去のデータを「線形和」で保持することで，勾配消失問題の対策をしている．

今回のネットワークは，入力層から 1 番目の隠れ層が線形変換+ReLU，次が双方向 LSTM，最後に出力層まで線形変換という構造を持っている．

ReLU とは Rectified Linear Unit の略であり，活性化関数の 1 つである．この関数は入力値が 0 未満であれば 0 を，0 以上であれば入力値を出力値とする．

3.5. Chainer

本研究ではディープラーニング用のフレームワークとして Chainer[7]を採用する．Chainer とは，Preferred Networks が開発している深層学習を実装するためのフレームワークである．また Chainer では通常のライブラリに加えて，様々な分野に対応する拡張機能を取り揃えており，それらの拡張機能では，各分野のアルゴリズムや学習済みモデルが実装されている．

Chainer では CTC が関数として実装されているため，CTC を用いたネットワークの構築をより容易に行うことができる．

4. 研究内容

本研究では大きく分けて 4 つの研究内容に取り組む．これにより，引き継いだ研究成果の発展と，一般環境での実用化に近付くことを目指す．

4.1. CTC 手法による手話単語の認識

昨年度の研究[1]を引き継ぎ，CTC の仕様の確認と実装を行う．仕様の確認のために実際に音声認識を行い，CTC によるブランクの処理や，ラベル出力を確認する．その後認識対象を手話単語とする．OpenPose を用いて，手話動画からキーポイントの位置をフレームごとに収集する．出力された JSON 形式のデータを CSV 形式の動作データに変換するプロセスは引き継いだプログラムを利用する．この動作データを学習や検証に使用する．手話動画は手話技能検定試験 6 級の出題範囲の手話 101 種類，各 100 個ずつを用いる．各単語の動作データの内，90%を学習用に，残りの 10%を検証用として用いる．

手話動画は，ホームポジション（両手を膝に置いている姿勢）から手話を行い，ホームポジションに戻るという動作で，録画ボタンを押すなどの動作は含まれていないものを使用する．

4.2. 不要な動作を含む動画の認識

一般利用者が自宅などから手話辞書システムを利用することを想定し，図 2 のような録画の操作をする動作（録画開始・停止の操作など）を含む動画を用意する．また撮影環境は，PC や Web カメラでの撮影を想定し，自身もその環境で撮影を行う．さらに撮影したデータを，4.1 節で構築した学習済みモデルを用いて検証する．

4.3. 映り込みへの対応

一般利用者が手話辞書システムを利用した場合，利用者の背景に他の人が映り込む場合が想定される．また，OpenPose の誤認識により，人ではないオブジェクトを人として認識する場合も考えられる[8]．このような場合，架空の関節を取得してしまうことになる．これらが原因となり，利用者を正しく認識しない可能性があるため，改善する．

OpenPose には，画面内から 1 人分だけの関節点などのキーポイントを取得する機能がある．先行研究[7]にて開発されたプログラムにおいても，この機能を使っていたが，実際に画面内で手話を行っている人だけを認識しているのか定かではなかった．そのため，本研究では手話を行っている人を認識するための処理を新たに実装する．

4.4. データ拡張

学習用に使用している動画は，図 6 に示すような画角となっている．一般利用者の環境で撮影して送られてくる動画は，多様な状況が考えられるため，データ拡張を行って精度の変動を探る．今回は，1) 人物の位置の違い，2) 手話を行う速さの違い，3) 解像度の違い，の 3 つの違いを想定し，それぞれに対して 1) 画面のトリミング，2) スピードの調整，3) 解像度の変更といった処理を行う．

図 6　学習に使用している動画の画角

5. 研究結果

5.1. CTC 手法による手話単語の認識

CTC による手話認識を行うにあたり，はじめに CTC の仕様の確認を行う．本研究では，CTC を用いた音声認識機能の実装[9]を行うことで，仕様の確認とする．

この実験では音声データを用いて，音素単位でラベルを割り振り，実際に認識が行えるかを試す．

データは ATR 音素バランス 503 文と呼ばれる，ATR データベースの音韻バランス文を発話するための音声データを用いた．音声データは音素ごとにラベルがつけられ，そのラベルをもとに学習が行われる．テストデータには様々な文章があるが，実験の一例として，テストデータの 1 つである「小さな鰻屋に、熱気のようなものがみなぎる。」という文章の認識を行った際の説明を行い，実際にどのように出力されているのかを下図に示す．図 7 は，今回認識させたい文章の正解ラベル（音素）である．今回は，最終的な認識結果がこれに近くなっている場合に正解だと判断する．次にテストデータに対して認識を行った出力結果を図 8 に示す．この結果を見ると，ラベルとラベルの間にブランクを示す（”_”）が挿入されていることが見て取れる．さらに，検出された音素フレームも単一ではなく，同じ文字が複数並んでいることから，複数のフレームに渡って検出されていることが分かる．この出力結果から，連続した音素ラベルの省略と，ブランクの削除を行った出力を図 9 に示す．認識した結果は「小さなンなぎアエに、メっきのエオおなものがみなぎる。」といった音素による文章が出力されている．実際にこの図 9 と，正解ラベルである図 7 を見比べてみると，正解ラベルに近い認識結果が出力されていることが分かる．CTC では認識結果が曖昧である場合にはラベルではなくブランク記号を挿入することで，誤認識を減らしている．正解ラベルである図 7 と，ラベル処理を行った結果である図 9 を比較すると，いくつかのラベルを誤って判断していたが，大半のラベルを認識できていると言える．

次に，CTC による音声認識[9]を参考にし，先行研究により手話認識を行うプログラムが開発されているため，そのプログラムの実装，実行を行った．学習モデルを作り検証を行ったところ，単語ごとの認識率にばらつきがあるものの，概ね先行研究[1]の通りの認識率が得られた．また認識結果では図 10 に示すように，1 番左側に正解ラベル，中央にはテストデータが何として認識されたかのラベル，そして 1 番右側には実際に合っているかの正誤表記の，3 つの項目が出力されるようになっている．

```
'sil', 'ch', 'i', 'i', 's', 'a', 'n', 'a', 'u', 'n', 'a', 'g', 'i'
'y', 'a', 'n', 'i', 'pau', 'n', 'e', 'Q', 'k', 'i', 'n', 'o', 'y',
'o', 'o', 'n', 'a', 'm', 'o', 'n', 'o', 'g', 'a', 'm', 'i', 'n',
'a', 'g', 'i', 'r', 'u', 'sil'], dtype='<U3')
```

図 7　音声認識の正解ラベル

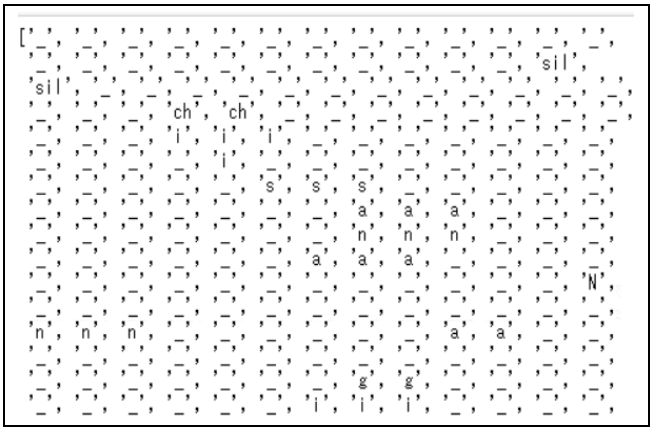


図 8　音声データの認識結果の一部

```
'sil', 'ch', 'i', 'i', 's', 'a', 'n', 'a', 'N', 'n', 'a', 'g', 'i'
'a', 'e', 'n', 'i', 'pau', 'm', 'e', 'Q', 'k', 'i', 'n', 'o', 'e',
'o', 'o', 'n', 'a', 'm', 'o', 'n', 'o', 'g', 'a', 'm', 'i', 'n',
'a', 'g', 'i', 'r', 'u', 'sil'], dtype='<U3')
```

図 9　ラベル処理を行った認識結果

```
 0 ['あう【会う】(6)'] ['あう【会う】(6)' 'さむい【寒い】(6)'] False
 1 ['あう【会う】(6)'] ['あう【会う】(6)'] True
 2 ['あう【会う】(6)'] ['あう【会う】(6)'] True
 3 ['あう【会う】(6)'] ['さむい【寒い】(6)'] False
 4 ['あう【会う】(6)'] [] False
 5 ['あう【会う】(6)'] ['あう【会う】(6)'] True
 6 ['あう【会う】(6)'] ['あう【会う】(6)'] True
 7 ['あう【会う】(6)'] ['あう【会う】(6)'] True
 8 ['あう【会う】(6)'] [] False
 9 ['あう【会う】(6)'] ['あう【会う】(6)'] True
10 ['あお【青】(6)'] ['あお【青】(6)'] True
11 ['あお【青】(6)'] ['あお【青】(6)'] True
12 ['あお【青】(6)'] ['あお【青】(6)'] True
13 ['あお【青】(6)'] ['あお【青】(6)'] True
14 ['あお【青】(6)'] ['あお【青】(6)'] True
```

図 10　手話の認識結果と正誤表記の一部

5.2. 不要な動作を含む動画の認識

利用時を想定して手話動画の撮影を行った．図 11 は実際に自身で撮影した動画の画面である．手話の種類は 4.1 節で記載した，手話技能検定試験 6 級の単語である 101 種類のうち，25 種類を撮影した．それぞれの種類につき 2 人が各 2 回ずつ手話を行い，計 100 個の動画データとなっている．

撮影した動画から，3.1 節の流れに従い動作データを抽出した．これを 5.1 節の学習モデルに対して検証を行ったところ，すべてのデータにおいて認識をすることができなかった．次に，利用時を想定して撮影した動画のうち半分を学習用，残りを検証用に分けて検証を行った．学習データの変更は，自身で撮影した 25 種類にのみ適応した．教師データの数は，5.1 節で学習に使用していた動作データと，自身で撮影した動作データを合わせて，各種類 100 個ずつとなっている．この学習モデルに対して，検証用データの認識を行った結果，54%の認識正解率であった．

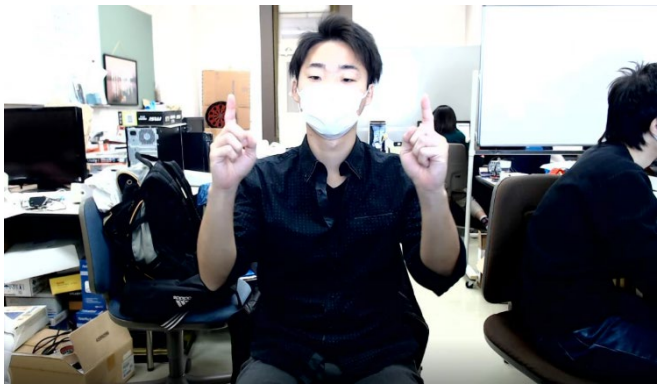


図 11　撮影した動画の画面の一例

5.3. 映り込みへの対応

他者の映り込みについて，先行研究のプログラムに機能を追加することで対応した．図 12 に示すように，フレーム単位の JSON 形式のデータから，1 動画の CSV 形式の動作データに変換するときに，手話をしている人のキーポイントのみを対象とするようにした．利用者が手話辞書システムを利用し撮影を行う際，ほとんどの場合ではカメラと手話表現を行う人の間に，遮蔽物や人が入り込むことは無いという前提を立てた．これより，図 13 のように画面内で 1 番大きく映っている人（1 番肩幅が大きく映っている人）が手話を行っていると考えた．まず，本節にて前述した OpenPose の機能を省き，画面内に映っている全員分のキーポイントを取得するように変更した．次に，それぞれの両肩を比較し，長さを取得する．この長さを他の人と比べていき，肩幅の最も大きい人を導出する仕組みとなっている．

この新たなプロセスにより，画面内に他の人が映り込んでしまっている場合にも，利用者の情報を取得することができる．さらにカバンなどを誤認識して関節点を推定したとしても，ほとんどの場合は 1 番手前の人物よりも小さく映る．よって，利用者の方が優先して認識されるため，この誤認識も防ぐことができる．また，3.1 節で使用した動画データにこの手法を適用した結果，認識率が約 2%上昇した．この 2%という数値は，実験を複数回行った平均値であるため誤差ではないと考えた．

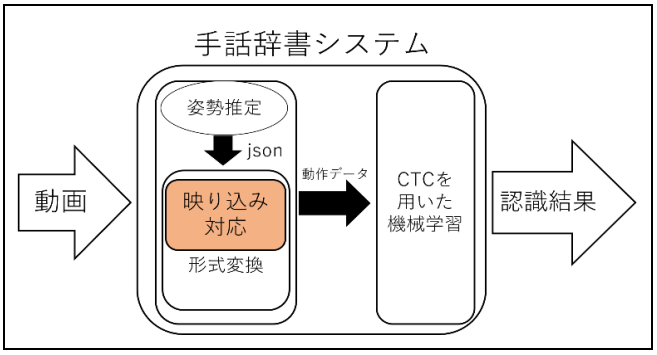


図 12　改良後のシステム概要図

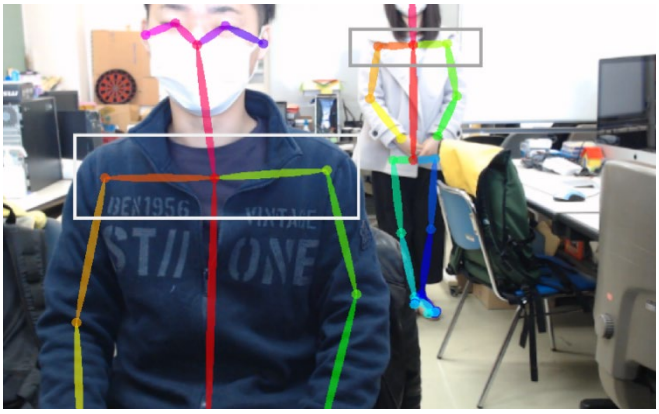


図 13　肩幅の比較

5.4. データ拡張

動画データを加工し，データ拡張用の動画データを自動的に生成するソフトウェアを開発した．動画データの加工には FFmpeg[10]という，動画と音声を記録・変換・再生するためのフリーソフトウェアを用いた．

次に作成したソフトウェアの仕様について説明する．加工したい動画の入ったディレクトリの指定と，加工後の動画データを出力したいディレクトリの指定を行うと，入力ディレクトリ内の動画がすべて加工され，指定されたディレクトリに出力するようになっている．画面のトリミングでは，左右上下から切り取る場所の選択と，指定した方向の何%を切り取るのかを選択するようになっている．スピードの変更では，元の動画に対して何%の速度変化を行うかを指定するようになっている．解像度の変更では，元の動画に対して何倍の解像度に変更したいかを指定するようになっている．

動画に対してトリミングの加工を行った際，はじめに約 30%を切り取る加工を行った．しかしこの加工では度々手話を行っている動作まで切れてしまっていたため，今回は 30%での切り取り加工を行わず，新たに 20%での切り取り加工を行うことにした．

今回動画に対して行った 3 種類の拡張方法を表 1 にまとめた．1 つ目のデータ拡張方法では，わずかに認識精度が向上した．2 つ目のデータ拡張方法では，わずかに認識率が減少した．

当初はデータ拡張を行った後，すべてのデータで学習や認識率の検証を行う予定であったが，現在，一部のデータでは検証まで至っていない．

表 1　データ拡張による数値の比較

	行った加工	向上した認識率
データ拡張 1	右側 20%切り取り	2.48 %
データ拡張 2	動画速度を 0.5 倍	-0.69 %
データ拡張 3	解像度を 3/4 倍	未検証

6. 考察

5 節の実験結果を経て得られた問題点について，発生してしまった原因などを考察する．

6.1. システム利用を想定時の認識率低下の原因

5.2 節の実験にて，自身で撮影を行った動画で検証を行った際，認識結果が著しく低くなってしまったことの理由を考察する．まず，学習モデルの教師データに採用した動画に比べ，自身で撮影した動画はカメラからの距離が近くなっている．これは図 6 と図 11 の画像を比較したように，自身で撮影した動画の方が手話を行っている人が大きく映り，キーポイントの座標の幅が大きくなっていることを示す．この座標のズレが誤認識を引き起こしていると考えた．また横向きの座標に関しても，座標にズレが生まれている．これは画角の違いが原因であると考える．教師データに採用した動画の画角に比べて，自身で撮影した動画の画角は横に広くなっている．

また，自身で撮影した動画をテストデータとしてのみ利用した際の認識率から，ラベルの特徴量とテストデータの情報が，一部一致した時点で認識結果として出力される可能性が薄いのではないかと考える．

現状では，遷移動作と関節の動きが類似した手話の識別が難しく，認識率の低下につながっている．例えば図 14，15 に示す「兄」や「姉」などの手話は，指を立てた状態で拳を上にあげる動作である．この動作が，撮影時に混入する動作と似ているため，誤認識が発生してしまったと考える．

本認識エンジンの最終的な完成目標である手話辞書システムでは，認識結果として 10 個の候補が羅列され，その中から自身で行った手話を選択する仕様になる予定である．また，これまでの認識エンジンではテストデータを配列にまとめ，ラベルの中から最もテストデータの情報に近いものを得るために，NumPy の np.argmax という関数を用いてきた．現段階ではこれを用いて確率が最も高いものだけを出力しているが，これを 10 番目までの候補を出すことで，類似したものを誤認識してしまったとしても，10 番目までの候補に正解のラベルが含まれていた場合には，システムとしては成立すると考える．

6.2. 映り込みへの対応による機能改善

5.3 節で行った映り込みへの対応について考察する．

研究結果にて実際に出力された値により，認識率が上がったことが確認できた．しかし，認識率の比較のために利用した動画には，手話を行っている人のみが映っており，今回開発した趣旨としては画面への映り込みへの対応であるため，本節の趣旨に対しては十分な検証が行えていなかったと考える．このことが原因で認識率があまり向上しなかったと予想する．

次に，わずかに認識率の向上が見られた要因について，2 人しか映っていない状況下でも，人ではないオブジェクトにキーポイントを投影してしまうなどの誤認識が起こっており，映り込みへの対応によって本来取得したい手話を行っている人のキーポイントが取得できたためだと推察する．

また，映り込みへの対応のプロセスにより，処理が重くなってしまった．想定している辞書システムでは，1 回の認識で処理する動画は単一であるため，その差はわずか数秒であるが，今回の研究中に行ったように，多くの数のデータを一括で行うとなると，膨大に処理時間が増加してしまった．

図 14　兄の手話表現

図 15　姉の手話表現

6.3. データ拡張による精度変動の予想

今回行ったデータ拡張による精度変動について考察する．

はじめに，画面の一部分を切り取る拡張方法について，行ったのは人の位置変動による認識率の変化を見るための実験であったため，今回行った加工では人の位置が変化したと捉えるにはあまり効果がなかったと考えた．具体的には，切り取り処理を行う対角面に対して，切り取った分と同じだけ，何もない部分を追加する対応が理想的であった．

次に今回行った加工に関して，これは切り取る方向によって精度が変動すると考える．今回行った右側を切り取る動作において，検証前までの予想では，精度の変動は無いと予想していた．この理由としては，手話を行っている人のキーポイントの座標を取得する際，x 軸と y 軸方向で座標を取得しているため，動画の右側を切り取る加工では座標にズレが生じないと考えたためである．実際に，加工を行っていない状態の動画から生成した動作データと，加工後の動作データの数値を大まかに比較しても，取得座標に大きな違いは見られなかった．実際に検証を行った際には認識率の向上が多少見られたが，検証回数の少なさからこれを向上だと断言できないため，誤差だと考える．

次に速度変動のデータ拡張について考察する．動画の速度を変動させることで，手話を行う人の表現速度の違いを再現した．検証を行った動画では，手話表現がゆっくりである場合を想定した加工を施している．表 1 の結果により，もし，手話を行うスピードが本来想定されるスピードに比べて遅かったとしても，認識精度をほとんど変えずに認識が行えると分かった．逆にスピードが速い場合の精度変動については，認識率が低下するのではないかと予想する．これは手話表現を行う際に,「ブレ」が生じることでキーポイントの取得精度が低下すると予測されるからである．

次に，解像度の変動による精度の変動を考察する．解像度を下げることにより，認識精度は減少すると予想する．この理由についても，キーポイントの取得精度が低下するためだと考える．特に指先のキーポイントの取得が難しくなってしまうと予想し，解像度が低下することで指を用いた表現が曖昧になってしまい，精度が低下するのではないかと考える．

7. 今後の課題

今回の研究では，不要な動作を含んだ単語データであっても，学習させることによってある程度までは認識が可能であることが分かった．しかしまだ十分な認識率とは言えず，多くの改善点が浮かび上がった．この改善点を反省し，本節では今後行うべき課題についてまとめる．

まずはじめに，今回撮影した動画は撮影環境や手話を行っている人が偏ってしまっている．今後，一般利用を想定した実験のために，今よりも多くの動画の用意が必要であると考える．

今回，学習させることにより手話動画からの認識を行うことができた．今後行う認識の進め方として，2 通りの方針が浮かび上がってきた．1 つ目は今回の様に，学習用データに不要な動作が入ったものを利用する方針である．これを行うためには，教師データにどのような割合で，不要な動作を含んだデータを採用するのかという検討の必要がある．もう 1 つの方針は，動画データから動作データを生成する段階で標準化を行い，不要な動作を含んでいたとしても認識ができるようにする方針である．標準化の理想形は，図 16 のような形が理想であると考える．この標準化は，1) 手話を行う人の大きさを揃える，2) 人の位置を揃える，3) x 座標と y 座標を 0 から 1 の間に収めるという 3 つの処理を施したものである．

次に，取得した動作データから，姿勢を推定できる手段が欲しいと考える．これは，新たな標準化方法が見つかった際に，実際に標準化された数値を目で見て分かるようにするためである．

6.2 節の「映り込みへの対応による機能改善にて」における考察を経て，趣旨に対応する正しい検証方法を検討する．これは，同じ人や環境にて手話を行うが，1 人で手話を行っているものと背後に人が映り込んだものを 2 種類用意し，認識率を比較することが理想だと考えた．

データ拡張に関する課題として，6.3 節より画面の一部を切り取るシステムの改善が必要であると考える．FFmpeg の機能には画面に特定色で帯を付けられるものがあり，この処理を，切り取り処理の後に追加することで実現が可能であると考える．また，今回の研究ではデータ拡張による精度の検証が一部できておらず，これを行う必要がある．この検証では，標準化方法を考案するにあたり，精度の向上をさせられる処理が何なのか，手掛かりをつかめると考える．

さらに本研究成果を研究室内で開発中の手話辞書システムに組み込むことを目指す．この際，6.1 節に記載した，確率が最も高いものだけを候補として出す方法から，10 個の候補を出す仕組みに変更しておく必要がある．また，本研究での学習モデルの開発において，取得できた「重み」を Web ページの開発者へ受け渡し，統合の実現を図る．

本研究のフィードバックとして，撮影の際，カウントダウン機能や，動画の撮影時間を制限することで，撮影をする際に含まれるであろう不要な動作を根本的に省いてしまう案が提案された．これを受けて，認識エンジン以外での改良となるが，撮影ツールの改良も視野に入れるべきかと考えた．

Chainer のメジャーアップデートが終了することを受け，今後は別のフレームワークへの移行を検討する必要がある．また，移行先としては PyTorch[11]への移行が有力である．Chainer と PyTorch はともに define by run 方式をとっており，2 つのフレームワークの性質が非常に似ていることから，PyTorch へコードの移植が容易に行える．この 2 つのフレームワーク間での移行に関する文献もインターネット上に多く見受けられ，

負担が比較的少なく移行を行えると考える．

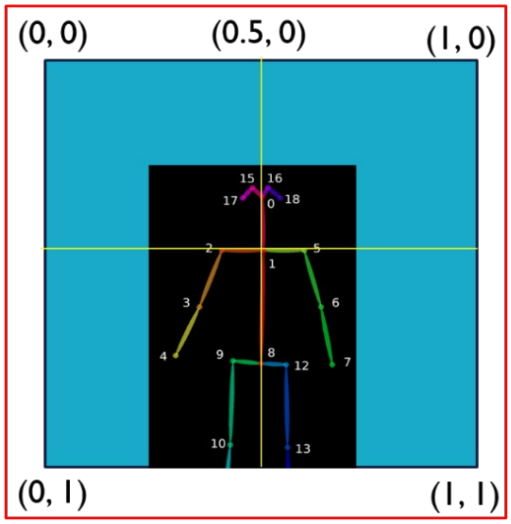


図 16　標準化後の理想形

8. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）の助成を受けたものである．

文　献

[1] 磯谷光，手話文章からの単語の抽出に関する研究，2019

[2] CMU-Perceptual-Computing-Lab/OpenPose，https://github.com/CMU-Perceptual-Computing-Lab/openpose，2021 年 02 月 19 日閲覧

[3] Connectionist Temporal Classification の理論と実装について，http://musyoku.github.io/2017/06/16/Connectionist-Temporal-Classification/，2020 年 11 月 06 日閲覧

[4] 上乃　聖　他，”CTC による文字単位のモデルを併用した Attention による単語単位の End-to-End 音声認識”，”情報処理学会研究報告”，vol.2018-SLP-120，no.16，pp.1-6，2018

[5] 稲熊　寛文　他，”End-to-End モデルによる Social Signals 検出および音声認識との結合”，”情報処理学会研究報告”，vol.2017-SLP-117，no.7，pp.1-6，2017

[6] Zhe Cao, Tomas Simon, Shin-En Wei, Yaser Sheikh, Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields，volume.1，pp.1-9，2016

[7] Chainer: A flexible framework for neural networks, https://chainer.org/，2021 年 01 月 21 日閲覧

[8] Windows で姿勢推定 (tf pose estimation)，https://qiita.com/kayu0516/items/754c6719fb55d2a6d563，2021 年 01 月 20 日閲覧

[9] Connectionist Temporal Classification（CTC）を用いた音素認識 - 2019 年 01 月 07 日更新，https://qiita.com/hirokisince1998/items/f59d3e8434d6326c48cc，2020 年 07 月 27 日閲覧

[10] 髙橋佑汰，木村勉，神田和幸，"機械学習を用いた手話認識に関する研究"，電子情報通信学会技術研究報告，査読無，118(440)，59-64，2019

[11] FFmpeg，https://ffmpeg.org/，2021 年 02 月 19 日閲覧

[12] PyTorch，https://pytorch.org/，2021 年 02 月 19 日閲覧

手話認識機能を搭載した辞書システムの開発

豊田工業高等専門学校　情報工学科　石浜 春，木村　勉

あらまし　豊田高専　木村研究室では「日本手話・日本語辞書システム」の開発をしてきた．前年度の研究ではカメラで撮影した手話動画を認識し，対応する日本語を掲示することができたが，このシステムは利用者にとって使いやすいものであるとは言えなかった．本研究では，前年度の研究成果を基に，より使いやすい「手話辞書システム」の開発を行う．検索することが可能な手話単語は，手話技能検定試験 6 級 101 単語を対象とする．結果として，前年度は複数回，画面遷移が行われていたが，これを無くしすべての動作を一画面で行える．手話辞書システムが完成した．性能向上のための改良点は，フロントエンドとバックエンドの完全な分離によってオンライン上で正常に動作が行えるようにすること，動画録画の際に録画のタイミングを遅らせること，研究室内で開発が進んでいる CTC 手法を用いた手話認識エンジンの採用を行うことが挙げられる．

キーワード　Windows，OpenPose，Flask，HTML，日本手話，辞書，GUI，手話認識

1. はじめに

豊田高専 木村研究室ではこれまで手話から日本語を検索することのできる「日本手話・日本語辞書システム[1]」を開発してきた．この理由として，日本語から手話を調べるという日本語―手話辞書が多く存在するのに対し，手話からそれに対応する日本語を引き当てる辞書（日本手話・日本語辞書）の数が少ないためである．図 1 に日本手話・日本語辞書システムの GUI を示す．

図 1　日本手話・日本語辞書システムの GUI

しかし従来の日本手話・日本語辞書システムは，手話表現における 6 種類の音素から調べたい手話に当てはまるものを選んでいき，目的の手話になるまで単語を減らす必要があるため，手話の初心者には使いにくいシステムであった．また先天性聴覚障害者の場合，多くは書記日本語に明るくない．そのため知らない手話を調べようとしても，GUI に書かれている選択肢の日本語の理解が難しいという問題がある．それらの問題点を踏まえ，前年度の研究[2]ではカメラで撮影した手話動画を認識し，対応する日本語を提示する辞書システムを開発した．この辞書システムは，ブラウザ上に GUI を実装した．使用するには処理ごとに画面が遷移する構造となっている．図 2 にその様子を示す．

図 2　前年度の辞書システムの GUI

従来の日本手話・日本語辞書システムよりは手話検索が容易になったが，使う際に複数の遷移が必要であるため使い勝手が悪いこと，また，オンラインに対応できていないなどの問題点が挙げられる．

2. 目的

前節で述べた問題点より，本研究では前年度の研究結果を利用して動画の録画，送信，表示された手話単語と対応する動画の確認まで一画面で行えるようにす

る（図 3）． GUI の改善を行うことで，初めての利用者でも視覚情報から操作方法を汲み取れるため，スムーズにシステムを利用できる．そうすることにより，手話学習が容易になり，手話学習者の意欲・増加につながると考えた．

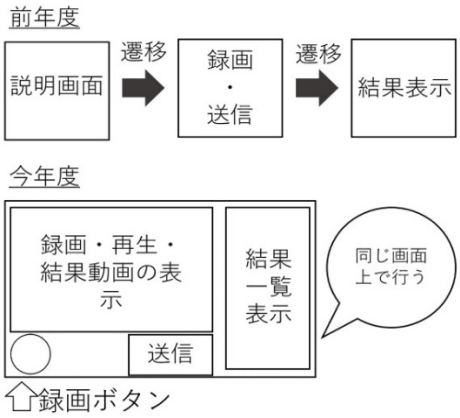


図 3　GUI の改善

3. 手話辞書システムの仕様

本システムでは次のようなシステムを想定している．

3.1. 外部仕様

本研究では，昨年度の研究成果を踏まえ，下記のような外部仕様を満たす手話辞書システムを開発する．

1.手話動画の録画

動画プラグインの videojs-record[3]を導入して録画と再生が 1 つの画面上で行えるようにする．利用者は USB カメラなどを用いて手話を録画，送信する．

2.手話認識

録画した動画をサーバに送信し，認識を行う．認識には前年度開発された手話認識エンジン[4]を用いる．この処理は時間がかかるため，利用者から見ると，システムが動作しているのか停止しているのかがわからない．そこで処理中であることを示すために，アニメーションを表示する．

3.認識結果の表示

手話動画の認識結果を GUI 上で一覧表示する．ここに表示された単語をクリックするとその単語の手話動画が再生される．

3.2. 内部仕様

ここではシステムの内部仕様について概略を述べる．

今回開発するシステムはオンライン辞書システムを想定している．そのため大きく分けて，クライアントとサーバの 2 つのサブシステムとそれらを繋ぐ通信から成り立つ．

クライアントでは，手話動画の撮影とサーバへの送信を行い，サーバからの認識結果を利用者に提示する．

サーバでは，クライアントから送信されてきた手話動画を，手話認識エンジンを用いて認識を行い，その結果をクライアントに返す．

下記に各サブシステムの仕様について述べる．

1）クライアントの仕様

クライアント側は，HTML と JavaScript を用いて実装を行されている．録画した手話との一致率が高い順に 10 個の単語を提示し，利用者はそれをクリックすると，単語に対応する手話動画の再生を行うことができるようにする．

2）サーバの仕様

サーバは Python API である Flask[5]を用いて以下の仕様を満たすように実装する．サーバが処理を行う部分は，動画送信ページから結果表示ページへ遷移する部分である．クライアント側から送信されたファイルは，手話認識エンジンで認識が行われる．認識に使えるのは動画ファイルのみであるため，送信されたファイルが一度動画かどうかを確認する．姿勢推定ソフトウェア OpenPose への入力は，利用者からどのようなフォーマットがアップされても対応できるように，動画形式をすべて mp4 に統一する処理を行う．

手話認識エンジンは，姿勢推定アプリケーションで行う．姿勢推定の結果と学習に使用された手話を比較し，一致率として高い順に 10 個選別し，見本として用意された手話動画とともにクライアント側に返す．

3.3. 手話認識エンジン

図 4 に前年度開発された手話認識エンジンのネットワーク構造を示す．

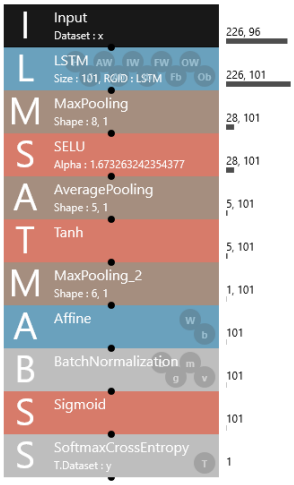


図 4　前年度開発されたネットワーク構造

本システムで使用する手話認識エンジンは，前年度で開発された再帰型ニューラルネットワークを用いた手話認識システムを用いる．手話認識システムは Long Short Term Memory（LSTM）を用いて構築されたニューラルネットワークである．ニューラルネットワーク

とは，人間の脳のしくみから着想を得た，脳機能の特性のいくつかをコンピュータ上で表現するために作られた数学モデルである．LSTM は再帰型ニューラルネットワークが長期の時系列を学習する際に用いられる手法であり，時系列を持つ手話動画のベクトルを学習させている．

手話認識エンジンは，手話動画を姿勢推定アプリケーションである OpenPose[6]を使って姿勢推定を行い，動画の縦と横の長さを用いて正規化された動作データを入力する．システムでは手話技能検定試験 6 級 101 単語が学習されており，入力される動作データが，学習済みである手話単語にどれだけ一致しているかを一致率として出力する．この一致率の高い順から，認識結果として提示する．

4. 研究内容

本研究で用いる開発環境を表 1 に示す．

表 1　開発環境

OS	Windows10
PC	CPU：AMD Ryzen 5 2600X 3.6GHz RAM：16GB GPU：NVIDIA GeForce RTX 2060 (6GB)
姿勢推定	OpenPose
ソフトウェア開発環境	Atom1.4.0
機械学習	Sony Neural Network Console
サーバ用フレームワーク	Flask v0.5.1
Python 実行環境	Anaconda

前年度のシステム環境を引き継いでいるため今回は必要なかったが，本システムは Python を用いたサーバ，HTML5，JavaScript を用いて実装される．そのため，Python のライブラリを使用するために Anaconda を用いてライブラリの導入を行う．また，Sony が開発したニューラルネットワークを構築する統合開発環境である Neural Network Console[7]（以下 NNC）から出力された学習済み機械学習プログラムを利用するには Neural Network library（以下 nnabla）というライブラリが必要になる．また，結果の出力には numpy ライブラリを，サーバの実装には Flask ライブラリを使用するのでこれらをインストールする必要がある．

本研究では，以下の機能の改善・実装を行う．

1. ページの融合

問題を解決する方法として，前年度ではトップページ，動画送信ページ，結果表示ページ，テストデータ登録完了ページと複数のページに分かれていた．本研究ではトップページとテストデータ送信ページを削除し，動画送信ページと結果表示ページを一画面にまとめる．前年度は，前述のように機能ごとにページの遷移が必要だったのに対し，本研究では一画面上で録画，送信，結果一覧表示のすべての処理を行えるように GUI の改善を行う．録画画面には外部プラグインである videojs-record を利用する．これを導入することでボタンの数が減り，利用者が視覚情報から操作方法を容易に汲み取ることができる．

2. 手話単語に対応する動画の再生

前年度の結果表示ページは，認識結果順に手話単語と対応する動画が並べられていた．しかし本研究では，認識結果は動画の録画と送信を行うブラウザと同ページに表示されるようにする．また，動画をはじめから表示しておくのではなく，利用者が確認したい単語をクリックしたら，手話動画が同じ画面上で再生されるようにする．

3. 認識中に進歩状況を追加

前年度のシステムでは，動画の認識時間が長く，その間正常に動作しているのか利用者がわからないといった問題点があった．そのため，動画の送信ボタンが押されたら認識が始まったことや，認識中であることを示すアニメーションなどを追加する．

5. 研究結果

研究結果についてここに述べる．本システムのオブジェクト図を図 5 に示す．

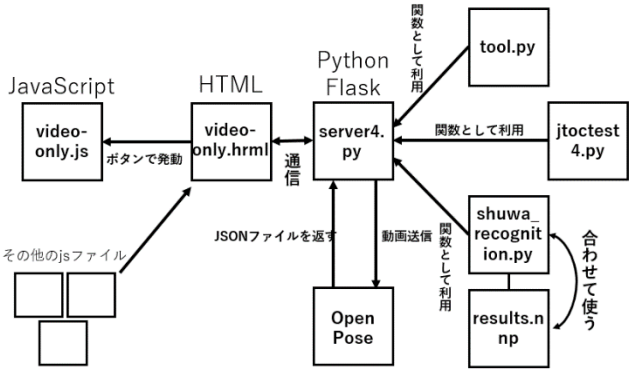


図 5　本システムのオブジェクト図

本システムは図 5 のように，主にサーバプログラムである server4.py とクライアントである video-only.html 間の通信によって成り立っている．各ライブラリ名とその機能を表 2 に示す．また，本システムで利用する関数を表 3 に示す．

5.1. GUI の実装

図 6 に実装したページを示す．まず，“会う”の手話動画を録画し，認識を行った．録画中の様子を図 7，認識結果を図 8 に示す．図 8 より，画面右側の認識結果の 7 番目に“会う”が提示されている．

利用者は画面中央より下にあるビデオアイコン（図 9）をクリックすることで USB カメラを利用した録画画面を表示することができる．

表 2　利用する主なライブラリの一覧

ライブラリ名	機能
video-only.js	録画・保存・結果提示とローディングアニメの再生
video-only.html	画面構成
server4.py	サーバプログラム
tool.py	手話動画の名前とアドレスの導出・並べ替え
jtoctest4.py	JSON ファイルの CSV 化
shuwa_recognition.py	機械学習の学習結果のプログラム
results.nnp	ニューラルネットワークの学習済みモデル
OpenPose	送信された動画から姿勢推定を行い，JSON ファイルを返す

表 3　利用する関数の一覧

関数名	ライブラリ名	機能
output_url	tool.py	動画のアドレスを一致率順に並べ替える
output_name	tool.py	動画名を一致率順に並べ替える
dfs	jtoctest.py	JSON ファイルの CSV 化を行う
recognition	shuwa_recognition.py	CSV 化した動画と学習データの比較を行う
click submit	video-only.js	ローディングアニメーションを表示する
click Btn	video-nly.js	手話動画を表示・再生する
start Record	video-only.js	録画ボタンを押されたら動作する
handle Click	record-toggle.js	録音ボタンが押されたら動作する

図 6　実装した GUI

図 7　録画中の様子

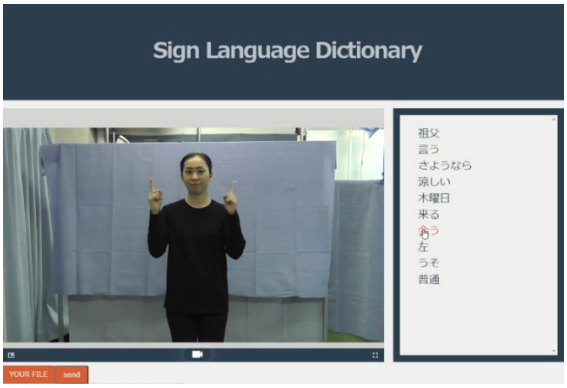


図 8　認識結果

図 9　ビデオアイコン

また，録画画面の左下にある録画ボタン（図 10）を押すことで 10 秒間動画を取ることができ，撮影後は自動的に自分の PC のローカルフォルダに保存される．この機能の実装には，外部プラグインである videojs-record を導入した．

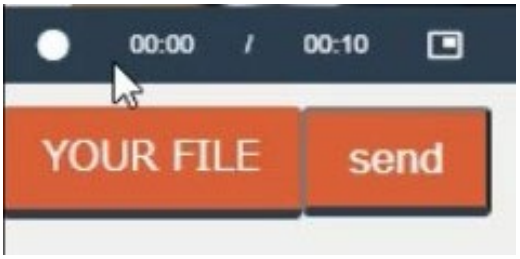


図 10　実装したボタン

【YOUR　FILE】ボタンからローカルフォルダに保存された動画を選択し，send ボタンで動画を送信する．send ボタンは POST メソッド[8]を用いてサーバに動画を送信する．POST メソッドは，サーバにデータを送信する HTTP 通信の一つで，HTML 上におけるリクエストの本文のタイプは Content-Type で示される．Content-Type とは，そのファイルの種類を表す情報が書いてある項目のことである．

認識には 40 秒ほどかかるため，認識している間，「探す」を意味する手話単語と検索中の文字が入ったアニメーションを表示した（図 11）.

Searching...

図 11　認識中に表示されるアニメーション

アニメーションの表示方法について示す．まず，HTML 上でアニメーションの GIF と全画面オーバーレイを指定した（図 12）.

GIF と全画面オーバーレイを JavaScript を用いてブラウザで非表示になるようにした（図 13）．”none”が非表示という意味である.

```
<!--全画面オーバーレイ-->
<div class="over" id="over">
</div>
<!--ローディングアニメ-->
<div class ="loading "id="loading">
   <img src="../static/images/loading.GIF"/>
</div>
```

図 12　検索中表示の HTML 記述

```
//ローディング GIF を非表示にする
document.getElementById("loading").style.display="
none";
//全画面オーバーレイを非表示にする
document.getElementById("over").style.display="no
ne";
```

図 13　アニメーションの非表示設定

そして，send ボタンが押されたら web 上にオーバーレイとアニメーションが表示される関数を記述した（図 14）.

```
//submit が押されたらローディング GIF，全画面
オーバーレイを表示
function clicksubmit(){
   const loading =
document.getElementById("loading");
   const over = document.getElementById("over");
   loading.style.display="block";
   over.style.display="block";
   loading.animate([{opacity: '0'}, {opacity: '1'}],
1500);
   over.animate([{opacity: '0'}, {opacity: '1'}], 1500);
   }
```

図 14　アニメーションを表示するプログラム

animate()メソッド[9]は，新たなアニメーションを作成して，対象要素へ適用するメソッドである．また，opacity は対象要素の不透明度を表している．ボタンを押してすぐアニメーションが現れると不自然なため，ここでは各要素の不透明度を 1500 ミリ秒かけて 0 から 1 に変更することで，フェードインで表示させる.

認識結果で提示された単語を図 15 に表す.

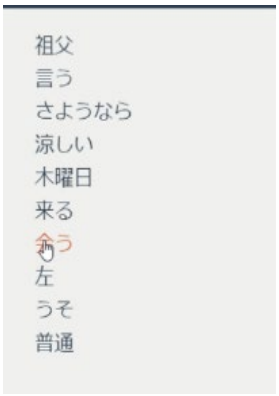


図 15　認識結果で提示された単語

単語の提示方法は次のようになっている．まず，サーバプログラムの server4.py で tool ライブラリを呼び出す．tool ライブラリには，手話動画の名前とアドレスを利用者が送信した動画と一致率の高い順に並べ替えを行う output_name 関数と output_url 関数が用意してあるので（表 3），それを用いて並べ替えを行い，配列に格納する．これは単語名と見本手話動画のファイル名が格納される．この配列を Flask のテンプレートエンジンである Jinja2[10]に渡して HTML の記述を生成する（図 16）.

```
<li>
<a href = "javascript:clickBtn("http://localhost:8090/
static/video/sign_video/おととい.mp4")">一昨日
</a>
<li>
```

図 16　見本手話動画の提示

手話認識を行っていないときはプログラムを非表示にしないと，ブラウザ上で名前の入っていない状態の配列が文字として出てしまい見栄えが悪い．そのため，認識後に提示したいプログラムを Jinja2 の if 文の中に記述する（図 17）．これを行うことで，図 6 のように認識前のページ上では何も表示されない.

また，単語名をクリックすると見本手話動画を再生するプログラムは図 18 のように実装した.

まず，単語名を HTML の<a>タグでリンク化した（図 18）．これはクリックすると JavaScript で記述した clickBtn()関数が実行される仕組みになっている．clickBtn()関数は手話動画のアドレスを指定した HTML の要素に代入する関数である.

```
<ul>
{%if video_url %}
<li>
<a href="javascript:clickBtn({{video_url[0]}})">
{{name[0]}} </a>
</li>
<li>
<a  href="javascript:clickBtn2({{video_url[1]}})">
{{name[1]}}</a>
</li>
<li>
<a href="javascript:clickBtn3({{video_url[2]}})">
{{name[2]}}</a>
</li>
<li>
<a href="javascript:clickBtn4({{video_url[3]}})">
{{name[3]}}</a>
</li>
<li>
<a href="javascript:clickBtn5({{video_url[4]}})">
{{name[4]}}</a>
</li>
<li>
<a href="javascript:clickBtn6({{video_url[5]}})">
{{name[5]}}</a>
</li>
<li>
<a  href="javascript:clickBtn7({{video_url[6]}})">
{{name[6]}}</a></li>
<li>
<a href="javascript:clickBtn8({{video_url[7]}})">
{{name[7]}}</a></li>
<li>
<a href="javascript:clickBtn9({{video_url[8]}})">
  {{name[8]}}</a></li>
<li>
<a href="javascript:clickBtn10({{video_url[9]}})">
{{name[9]}}</a>
</li>
{%endif %}
</ul>
```

図 17　認識結果の一覧提示用 HTML

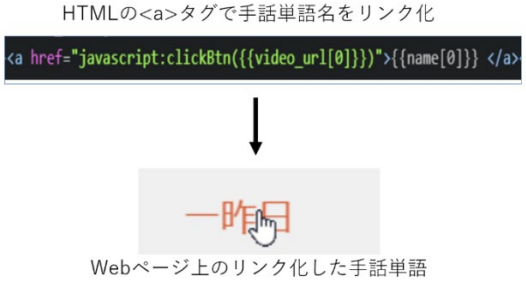


図 18　リンク化した単語名

```
function clickBtn(video_url){
```

図 19　video_url の引数指定

これを次のように実装した．まず，video_url（動画ファイルのアドレス）を引数とする（図 19）．

getElementById()[11]は指定した id を持つ HTML の文字列の要素を取得するメソッドである．getElementById()を利用して，myvideo_htm5_api という id を持つ文字列の要素を，myvideo に代入する（図 20）．

図 20　文字列の要素代入

setAttribute()は指定した要素に新たな属性とそれに対応する値を格納できるメソッドである．myvideo.getAttribute()[12]で myvideo_htm5_api という id が指定された HTML 文字列の src 属性の要素を取得し，myvideo_src に代入する（図 21）．

```
var myvideo_src　=
myVideo_html5_api.getAttribute("src");
```

図 21　src 属性の取得

src 属性は，画像や動画を指定するときに使う HTML タグである．src 属性の中身が null だった場合と，null 以外が格納されていた場合（録画を行った後の動画が格納されている場合）に src 属性に video_url を格納する．図 22 に src 要素が null だった場合の if 文を示す．条件が変わっても if 文の中身は同じである（図 22）．同時に，自動再生を行う autoplay 属性と，再生・一時停止・再生位置の移動・ボリュームなど，動画を利用しやすくするインターフェースを，ブラウザが自動で表示する controls 属性を追加する．図 23 に単語がクリックされた後の myvideo_htm5_api という id が指定された文字列を示す．

src に動画のアドレス，autolay 属性，controls 属性が加されていることがわかる．

```
if(myvideo_src==null){
    myvideo.setAttribute("src",video_url);
    myvideo.setAttribute("autoplay","autoplay");
    myvideo.setAttribute("controls","controls");
}
```

図 22　src 属性の判定

```
<video id="myVideo_html5_api"
playsinline="playsinline" class="vjs-tech"
tabibox="-1"
src="http://localhost:8090/static/video/sign_video/おとといmp4" autoplay="autoplay"
controls="controls"></video>
```

図 23　見本手話動画の属性の追加

6.1. サーバ側の実装

サーバは前年度を引き継ぎ，Python フレームワークである Flask で構築されている．Flask では，サーバプログラムである server4.py で HTML ファイルを呼び出している（図 5）．前年度は処理ごとに呼び出す HTML ファイルが異なっていたため，複数の HTML ファイルを呼び出していた．それに対し，本研究では全処理を同画面上で行うため，処理が行われた後に指定された HTML ファイルを呼び出す return render_template 関数に，同じ HTML ファイルを指定した．よって，どの処理後も同じページを表示することができた．

6.2. ポート開放

本システムはオンライン上で利用することを目標としているため，外部からアクセスした際に問題なく動作するか確認を行った．今回は web サーバ上にアップロードするのではなく，ローカルアドレスのポート開放を行うことで，同じネットワークにつながっている同研究室内の PC でアクセスした．まず，以下のサイトを参考にして，解放したい特定のローカルポートに“8090“を入力し，ポート開放を行った．Flask はデフォルトでは外部公開を許可していないため，図 24 のようにサーバプログラムへ明示的に “0.0.0.0”を指定する必要がある．

```
if __name__ == "__main__":
    app.run(host = “0.0.0.0”, port = 8090,
debug=True)
```

図 24 ホスト指定

結果として，同研究室内の PC からアクセスすることは成功した．しかし，動画の送信の際にエラーが出てしまい，正常に動作しなかった．

6.3. 録画開始の遅延機能の追加

手話動画を録画する際，録画を開始・終了ボタンを押す動作が動画内に混入してしまい，正しい結果が出ないといった問題がある．今回のシステムで導入した外部プラグインである videois-record に JavaScript にデフォルトでスリープ関数は実装されていないため，スリープ機能を持った関数やカウントダウンを行う関数を追加する必要がある．録画開始ボタンに該当する HTML の id を取得し，録画ボタンが押されたら 5 秒スリープしてから録画を開始するスリープ関数を記述したが，動作しなかったため，録画開始ボタンを押したときに動作する startRecord 関数を書き換える必要があると考える．record-toggle.js ファイルは録音のみを行う際に利用されるプログラムで，録音ボタンが押されたら動作する関数が記述されている．この中の handleClick 関数を書き換えて録音を遅らす方法は確認できた．改良した関数は図 25 のとおりである．

```
handleClick(event){
  let recoder = this.player_.record();
  if(!recorder.isRecording()){
        recordr.start();
  }else{
        recorder.stop();
  }
 }
```

図 25　録音遅延関数

しかし，改良した関数では録画に対応させることはできなかった．

6. 考察

作成したシステムでは，録画した動画から想定していた手話単語を摘出することに成功した．しかし，図 15 より，上から 7 番目に正しい手話単語名がある．単語は一致率の高い順に提示されるため，本来なら先頭に望んだ単語が提示されるのが理想的である．今後一般公開を目指すためには，録画方法や認識率の改善が必要だと考えられる．

録画方法の改善案は 2 つ挙げられる．一つ目は，図 26 のように，録画画面に利用者が録画を行う位置を示すことである．なぜなら，利用者と USB カメラの位置が近いと，関節などのキーポイントがうまく取れないことがあり，これが誤認識の原因となるからである．

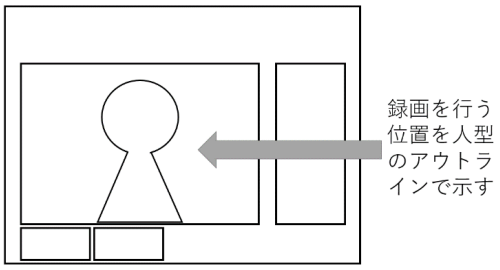


図 26　誤認識を防ぐ改善案

二つ目は，導入予定であった録画遅延機能の実装である．開始・停止ボタンを押す動作が混入しないようにすることで，誤認識を減らすことが可能になる．

次に，作成したシステムの利点は以下のとおりである．

1．GUI 改善の成功

従来のシステムは処理ごとにページの遷移が必要であった．また，ページ上に文字やボタンが多く，白地の背景に黄色のボタンと明度の近い二色をメインとしたカラーデザインであった．これは色を認識しにくい人にとっては見づらい可能性がある．今回作成したシステムでは，GUI を改善することによって遷移を検索時のみに変更し，すべての処理を一画面で行うことに成功した．また，ボタンを減らし簡単な操作でシステムを利用できること，明度に差がある色を使うこと

で様々な事情の利用者に対応できる．これによって，前年度より利用者にとって使いやすいシステムになったといえる．

2．検索中アニメーションの導入

前年度問題点としてあげられていた，認識中に正常にシステムが動作しているかわからないといった問題を解決できた． OpenPose を用いて姿勢推定を行うとき，動画をフレームに区切って最初のフレームから姿勢推定を行う．ひとつのフレームの姿勢推定を行うのにかかる時間はとても長いというわけではないが，1 動画におけるフレームの数だけ認識するには時間がかかる. 解決策として, 動画送信ボタンが押されたら「探す」を表す手話単語と，検索中の文字のアニメーションを表示した．

7. 今後の課題

今回作成したシステムの不十分な点である．

1. オンライン上で正常に動作しない

前年度の問題点で，複数台のコンピュータから同時にアクセスされた時の挙動の確認が行われていないことが挙げられていた．そのため，ポート開放を行うことで同研究室内の PC からシステムにアクセスする実験を行ったが，正常に動作しなかった．この原因として，システムがサーバとクライアントに完全に分離していないといった点が挙げられる．サーバプログラムで指定されている動画のアップロード先にローカルファイルが指定されているため，オンラインでは動作しない．また，手話認識エンジンの比較データや，認識後掲示される動画の保存場所が同じくローカル PC 上であることも原因の一つである．

これらの問題を解決策として，手話認識エンジンの比較データや認識後掲示する動画をデータベースとして管理したり, Flask で立ち上げたサーバを書き換えたり等，バックエンドの大幅な変更が必要になる．

2. 録画開始遅延機能の未追加

手話動画を録画するときに，開始・停止ボタンを押す動作が混入しないようにするために，録画が始まるまでにカウントダウンを行ったり，一定の時間だけ録画するような機能を実装する予定であった．余分な動作が混入することで，解析結果に誤差が生じ，想定していた単語と合致しない結果となるのを防ぐためである．しかし， 今回導入した外部プラグインの videojs-record の使用方法を習得するのに時間がかかり機能の追加に至らなかった．

3. CTC 手法を用いた手話認識エンジンの導入

同研究室内で Chainer をフレームワークとして開発が進んでいる CTC 手法を用いた新たな手話認識エンジンを採用することによって，手話動画に認識率の向上を目指す．shuwa_recognition.py は学習済みモデルを用いて手話を認識するプログラムである．学習済みモデルは NNC から得られる result.nnp というファイルである. サーバプログラムである server4.py では NNC のライブラリをインポートして，この学習済みモデル利用している. これを CTC 手法で行った学習済みモデルが利用できるように変更する．しかし，今回問題点となった学習済みモデルの格納場所をどう変更するか考える必要がある．

8. おわりに

今回の研究では，利用者が使いやすい手話辞書システムの GUI 設計の見直しを目標とし, その結果として遷移を減らすこと，認識中であることをわかりやすくするなどの改善点を解決できた．

7. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）の助成を受けたものである．

文　献

[1] T. Kimura and K. Kanda, "Sign Language Recognition through Machine Learning by a New Linguistic Framework", Association for the Advancement of Assistive Technology in Europe 2019, Proceedings S144-S145 (2019)

[2] 鈴木洸輔，手話認識を使用した辞書システムの試作，豊田高専卒業論文，2020

[3] Videojs-record - https://collab-project.github.io/videojs-record/#/(2021 年 2 月 19 日閲覧)

[4] 吉田あすか，機械学習を用いた手話単語認識の拡張，豊田高専卒業論文，2020.

[5] Flask へ ようこそ - Flask v0.5.1 documentation https://a2c.bitbucket.io/flask/（2021 年 2 月 19 日閲覧）

[6] CMU-Perceptual-Computing-Lab/openpose , https://github.com/CMU-Perceptual-Computing-Lab/openpose （2021 年 2 月 19 日閲覧）

[7] Neural Network Console – SONY - https://dl.sony.com/ja/（2021 年 2 月 19 日閲覧）

[8] POST - MDN – Mozilla https://developer.mozilla.org/ja/docs/Web/HTTP/Methods/POST（2021 年 2 月 19 日閲覧）

[9] element.animate() - MDN - Mozzilla https://developer.mozilla.org/ja/docs/Web/API/Element/animate（2021 年 2 月 19 日閲覧）

[10] Jinja – https://jijja.palletsprojects.com/em/2.11.x/（2021 年 2 月 19 日閲覧）

[11] Document.getElementById() – MDN – Mozilla https://developer.mozilla.org/ja/docs/Web/API/Document/getElementById（2021 年 2 月 19 日閲覧）

[12] element.setAttribute - Web API | MDN (mozilla.org) https://developer.mozilla.org/ja/docs/Web/API/Document/ setAttribute（2021 年 2 月 19 日閲覧）

研究ノート：新手話学の分類辞(CL)の研究

神田和幸　民博

１．はじめに

CL は英語の Classifiers を語源とする手話用語だが、日本での使われ方は非常に曖昧である。アメリカでは CL は手型に限定されている。日本でも神田(2010)は日本手話の動詞の語幹となる手型として、手型に限定している。本論では、巷間に使われているジェスチャ的な表現という広義ではなく、手話学上の定義として手型に限定して議論する。CL の文法機能はアメリカ手話と日本手話では異なることがありうるので、アメリカの CL 議論はここでは深く立ち入らないが、どのような手型を CL としているのかだけを紹介しておく。紙幅の都合で図は省略する。

CL-1

CL-3

CL-4

CL-5

CL-A

CL-B- flat things[roof, flat, wall]

CL-C-[thick things, round pole-like things]

CL-C-(modified)(index and thumb)　pepperoni, cookies, campaign buttons

CL-F - small round things: buttons, quarters, tokens, eyeballs, instrumental classifier for holding on to small things, (also for showing movement of small flying insects)

CL-G- thin things (or degree of thinness), also "eyelids"

CL-L (bent)-[large, check, card, square.　Related concept: EGO bigheaded/egoistic/conceited]

CL-L-[check, card, square]

CL-R Rope-like, braided, rolled,and/or twisted things.

CL-V- legs, a person walking-(upside-down V), two people walking, [stand, walk-to, lay down, toss-and-turn, dive, jump, skate board, scooter, get up]

CL-V (bent fingers) = a small animal, or a larger animal sitting.

CL-Y　Very wide things.　A fat person walking.　A hippopotamus's mouth.

CL:O - cylindrical pipe, hose, ...

CL:S - head

CL:U - ribbon, tongue, strip of paper,

CL:Y - fat animal or person, large, thick tires of a fancy sports car, big yawn, ...

(https://www.lifeprint.com/asl101/pages-signs/classifiers/classifiers-frame.htm)

Schick の分類

シックは初めて CL を研究したと考えてよい。その分類を以下に示す。

Schick 1990 の分類

	Movement Roots（動作原形）		
	MOV（動作）	IMIT（模倣）	DOT（提示）
CLASS（具象分類辞）	S-V(+LOC)	S-V	S-V:be(+LOC)
SASS（抽象分類辞）	V:adj	S-V	V:adj+LOC
HANDLE（取扱分類辞）	(S+)V-O(+IO)	(S+)V-O(+LOC)	(S+)V-O+LOC

(p.20, Schick 1990)

この図だけでは分かりにくいので少し解説をする。シックは CL を動作の原形（意味）により、動作、模倣、提示の３つに分類し、CL に具象、抽象、取扱の種類があるとした。上記の S は主語、V は動詞、LOC は位置、be は be 動詞、 adj は形容詞、O は目的語、IO は間接目的語の意味である。いろいろな文法概念が入り乱れていて分かりにくい。

シックの分類法により、日本手話の例を考えてみると以下のようになるが、検証が不十分なので、ここでは理解の助けになればという程度である。意味や文法とのリンクはよくわからないので、省略する。

ASL の分類辞の適用

CLASS に該当　１（人）：＜行く＞　２下（動作）：＜歩く＞　コ（車）：＜自動車＞
　I-L-Y（飛行機）：＜飛行機＞

SASS に該当　　１＜線を引く＞　２＜線を引く＞　テ：＜ある＞　　メ：＜管＞　　C２：＜太い管＞　　C５：＜太い管＞

HANDLE に該当　＜つまみ＞　＜ドライブ＞　　＜仕事＞　　＜場所＞

２．日本手話の CL

日本手話の CL（拘束形態素）は動きと結合し（語形変化し）動詞を作る

CL と結合する動きには以下のような種類があると想定している。

①　押印

　押印は繋辞（copula）の文法機能を持つ。押印の変化形には「痕跡をなぞる」があり、ここでは痕跡と命名する。一般に、どの言語でも繋辞により 自動詞または形容詞になる。

手話辞典で「男」と掲載される手話語彙は、実際には親指を立てる手型（CL）が押印動作で示されることで＜男です＞という手話になる。押印動作を伴う指文字も同様に繋辞と結合されている。手話では、CL も押印動作も拘束形態素で、結合により動詞となる。

痕跡は下記に例を挙げるが、唇をなぞる（＜赤＞）、顔をなぞる（＜顔＞）など、該当の

物をなぞるしぐさをする。文法機能としては繋辞と同じと考えられる。

② 動詞類

動く空間と動きの種類による意味の分化がある。たとえば空間的に目上という意味は肩より上の空間 において、上への動きで示される。上昇という動きを伴う語彙＜兄＞などは上昇動作が押印動作とは相補分布の関係にあり、同じく繋辞の機能を持つ。一方、＜行く＞では、ひとさし指(CL)が動くことで一般動詞になる。このように CL は結合する動きにより動詞としての意味が変わるので、動きの意味についても考察の必要がある。

動きの分類はひとまず脇において、形だけの分類をすると、CL の種類は以下のようになる。

CL1（ひとさし指）
CL2 (ひとさし指と中指)
CL5 （全指を開く）
CL 男
CL 女
CL ヤ
CL 刀（４指を閉じる）
CL 甲
CL 小丸
CL シ
CL C（親指とひとさし指）
CL オ
CL 拳
CL コ
CL 場
CL 岩
CL セ
(CL 薬) 消滅的傾向にある
(CL ヌ) 特殊例

指さし（PT）の構造

CL と深い関係にあるのが指さし（PT）である

指さしの手型には１、テ片手、テ両手があり、動きとして押印、跡、 接触がある

これらは意味が同じことが多いので異形態とも考えられる。

PT 1 手型１ 指先向き R 動き押印 例：あなた
PT 甲 手型テ 掌向き下 動き押印 例：です
PT 掌 手型テ 掌向き上 指先 R 動き押印 例：そちら様

PT 跡 上記の型が定められた図形をトレース　例：みなさん、顔、赤
PT 両 掌の両手タイプ 例：今 派生形
PT 触 PT 甲（PT 掌？）が接触 例：技、ベテラン
PT1 変形 接触の結果の変化（同化？）例：白
結合の例として
　赤の構造 PT1 痕跡 赤いもの（下唇）赤いもの
　白の構造 PT1 痕跡（同化変形）白いもの（歯）
　黒の構造 PT 甲 痕跡 黒いもの（髪）

指差しの機能は基本として指示であり、指示されるものは空間位置または身体である。空間位置は方向比喩、身体は内包的意味を顕在化したものである。

手話語彙の多く、とくに身体部位は PT と指さしまたは痕跡との組み合わせがほとんどで、世界共通であることも多く、最もジェスチャに近い。そのため、スワディシュの基本語彙を用いて言語の親縁関係や年代を測定しようとすると失敗する。音声言語においては、記号の恣意性が厳密であるから、こうした推論が成立するが、手話言語においては、ジェスチャとの関係が深いため、単純に語形だけから推定する手法は誤りである。実際、こうした試みを見ると、あらゆる手話間に親縁性があることになってしまう。それをもって言語普遍論の根拠とするのは屋上屋を重ねることになる。

そもそも指差しが手話として特異であることは早くから知られてきた。形態素としてみれば「存在を示す機能」であり、位置はそのままでは提示できないので、何かで指示することにより初めて認識される。この指示機能は 1 本指の指差しだけでなく、掌でもよく、時には顎が使われることもある。共通していえるのは、動きに＜静止＞があることで、この静止動作が＃提示の意味をもつ。すなわち＃PT／＃位置／＃静止という一般形式が成立している。意味の中心は位置である。この一般形の応用として＃PT に手の形を代入することで＜男＞のようなジェスチャ的な手話語彙が成立している。この場合、位置は中立で意味を失い、手の形の意味が中心となる。PT は CL であっても、ほぼ意味をもたないゼロ形態素と考えるのが妥当であろう。

謝辞
本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）助成を受けた研究成果の一部である。

Conformer を用いた手話単語認識に関する研究

豊田工業高等専門学校　情報工学科 稲田　渉，木村　勉

あらまし　筆者らは，これまでに深層学習を用いた手話単語認識において，約 93%の認識率を挙げている．本研究では，認識率の向上と手話文認識システムとの統合を見据えて，新たな手法を取り入れる．従来研究では音声認識に用いられる Connectionist Temporal Classification (CTC)と姿勢推定アプリの OpenPose を用いているが，これを音声認識で使われている Conformer という学習モデルと姿勢推定アプリの MediaPipe に変更する．その結果，認識率が向上した．この手法を応用することにより，手話文，読唇，指文字の認識も可能となる．

キーワード　手話，機械学習，深層学習，MediaPipe，Conformer，PyTorch

1. はじめに

これまで豊田高専木村研究室では深層学習による手話の認識に関する研究を行ってきた．現時点では，手話技能検定試験 6 級に指定されている単語に対して，約 93%の認識精度を持つ学習ネットワークの開発に成功している[1].

現在の語彙数での認識率は約 93%であるが，語彙数が増えた場合，低下する恐れがある．また，今後手話文認識エンジンとの統合も考え，本研究では，音声認識に用いられる Connectionist Temporal Classification (CTC)[2]を用いる代わりに自然言語処理で飛躍的な性能をもっている Transformer[3]と画像認識などに用いられる CNN を組み合わせた Conformer[4]を適用させ，その性能を検証する．

また，先行研究[1]で用いられていた姿勢推定アプリである OpenPose[5]は姿勢推定に時間がかかるため，手話辞書システムの検索時間が長くなってしまう．そこで，本研究では OpenPose から MediaPipe[6]に変更し，検索時間の短縮を図る．

本研究では，従来研究から姿勢推定の手法と学習モデルを変え，有効性を検証する．

2. 目的

本研究は，手話単語の認識率を向上させることを目的とする．また本研究とは別に木村研究室では，手話文中に含まれる単語の認識に関する研究を行っている[7]．現在のところ認識率は約 29%となっている．Conformer は音声認識でも活用されており，これによる学習の有効性を確認できれば，手話文の認識にも応用が可能であると考えられる．また，本研究室内で進められている読唇や指文字の認識にも Conformer が応用できるのか，その有効性を検証する．

3. 研究の方針と準備

木村研究室で行っている手話認識は，姿勢推定を行い，関節などのキーポイントを特徴量として学習させている．これにより，背景の変化や衣服による影響を少なくすることができるという利点がある．

この節では，従来研究と本研究で採用している手話単語認識フローの変更点と新しく採用した手法について述べる．

3.1. 従来研究での認識フロー

従来研究では，姿勢推定ソフトの OpenPose を用いて関節点などの座標を取得している．これをフレームごとに JSON 形式のファイルに出力し，さらにこれを動画ごとに 1 つの CSV 形式の動作データにまとめる．これらを学習データとして機械学習を行う．また，学習モデルには RNN (Recurrent Neural Network)の一種であり，長期的な依存関係を学習することのできる LSTM (Long Short-Term Memory)を採用している．損失関数には，End-to-End の音声認識に使用されており，どの音素でもない「ブランク」という概念が認識できる CTC を採用している．さらに，機械学習用フレームワークとして Chainer[8]を使用し，手話単語の学習を行っている．

3.2. 本研究での認識フロー

本研究では，姿勢推定に MediaPipe を用いている．MediaPipe によって関節点などの座標取得し，肩幅を基準として座標を正規化する．その後，CSV 形式のファイルとして出力する．それを学習データとする．学習モデルには Conformer を採用し，損失関数には交差エントロピーを使用している．機械学習用フレームワークには PyTorch[9]用いて学習をし，手話単語の認識を行う．

3.3. MediaPipe

本研究では姿勢推定に MediaPipe を用いる．MediaPipe とは，Google 社が提供しているストリーミングメディアに対して推論を実行するパイプラインを構築するためのフレームワークである．しかし，今回は関節点の座標取得にのみ使用する．図 1 に MediaPipe によって姿勢推定している様子を示す．

従来研究で使用している OpenPose はいくつかの欠点がある．まず，GPU が必須であるため，使用環境が制限されてしまう．また，GPU の性能にも左右される

が，推定に時間がかかるため，手話辞書システムを実行したときに検索時間が長くなる．さらに取得できる座標は 2 次元であるため，奥行きの変化を伴う手話の認識が低下するという問題がある．

MediaPipe は OpenPose と比べて，GPU が必要ではなく，動作が軽い．また，3 次元で座標が取得でき，奥行きの認識も可能である．さらに，OpenPose と比べて取得できる関節点の数も多いという特徴がある．

図 1：MediaPipe による姿勢推定

3.4. Conformer

本研究では，学習モデルとして Conformer を採用する．これは，自然言語処理で飛躍的な性能をもっている Transformer と CNN と組み合わせた学習モデルである．Transformer とは，2017 年に発表されたエンコーダデコーダ型の深層学習モデルである．その特徴として，RNN の再帰や CNN の畳み込みなど採用しておらず，Attention のみを用いている．そうすることで，並列計算が可能になり，訓練の時間が短縮される．その上，翻訳では精度が高いモデルの多くが Transformer をベースとしている．さらに，Transformer は自然言語処理だけでなく画像処理や音声認識などにも応用されている．その音声認識に応用するために考案されたモデルが Conformer である．

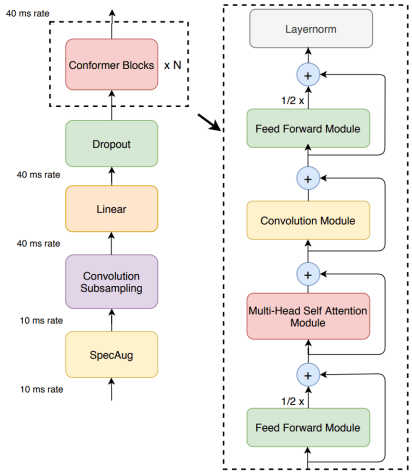


図 2：ConformerEncoder の構造（左）と ConformerBlock の構造（右）

Conformer の構造を図 2 に示す．文献[4]では，Conformer はエンコーダとして使われており，デコーダには LSTM を使用している．図 2 の左側の部分が文献[4]の論文内で使われている ConformerEncoder である．ConformerEncoder の最初の部分である SpecAugment は音声データを画像データとして扱うためのスペクトログラムに変換し，マスクをかける前処理であるので，本研究では使用しない．最後の部分である ConformerBlock がこの Conformer の特徴である．右側の図は ConformerBlock を表す図である．Transformer の特徴である Multi-Head Self Attention と CNN の特徴である Convolution 層を組み合わせている．これによって長い時間の依存性を捉えることができ，さらに局所的な特徴も学習することができる．この深層学習モデルは，音声認識において大きな成果を出している．

3.5. PyTorch

PyTorch とは，2016 年に Facebook により開発された Python のオープンソース機械学習ライブラリである．先行研究[1]で利用していた Chainer のメジャーアップデートの終了を受け，本研究では PyTorch に移行した．Chainer と PyTorch はどちらも Preferred Networks が開発しており，計算グラフの構築と順伝播処理の実行を同時に行う Define-by-run 方式である．機能や性質も似ており，Chainer と同じようにデータの取得やデータセットの作成，ネットワークの構築，学習などを容易に行うことができる．

4. 研究内容

本研究では大きく分けて 5 つの研究内容に取り組む．これにより，認識精度の向上を目指し，Conformer の有効性を確認する．

4.1. 開発環境

開発環境は，Python 3.9.6，PyTorch 1.10.0+cuda 11.3，姿勢推定に MediaPipe[6]を用いて，統合開発環境には Jupyter Notebook[10]を使用している．

4.2. ConformerBlock を用いた多クラス分類手話単語の認識

まず，ConformerBlock を使い，手話技能検定試験 6 級の 101 単語を多クラス分類として認識させる．ネットワークの構造は，ConformerBlock に全結合層をつなげたものである．

使用する学習データは，単語の動画を 1 単語につき 100 個データを用意し，MediaPipe を用いて手の 21 か所，肘と手首 2 か所，および左右分として計 46 か所の 3 次元座標を取得し，それを CSV 形式でまとめたファイルである．つまり，特徴量を 138 個とり学習する．各単語の学習データのうち 80%を訓練データ，20%をテストデータとする．先行研究[1]では行っていないが，

本研究では取得した座標を，肩幅の倍を画面全体の大きさとして正規化する．これによって手話辞書システムにおいて，人の位置やカメラとの距離によって認識率が下がることを防ぐ．パラメータは，バッチサイズを 16，エポック数を 120，損失関数に交差エントロピーを用いて，最適化関数は Adam を採用する．学習率を 0.00001 として学習する．101 単語に ID を割り振り，多クラス分類をする．

4.3. 唇の特徴点を含めた手話単語認識

手話には，同じ動作で複数の意味をもつ同形異義語が存在する．しかし，手話をするときは動作で表している単語を口でも表現する．つまり，同形異義語でも口の動きは異なるので，4.1 では取得していない唇の座標を含めた学習データで学習させる．これによって同形異義語を識別できるようにする．MediaPipe では唇で取得できる座標が 17 か所ある．口は奥行きの方向の変化がほぼないため，3 次元座標ではなく 2 次元座標を追加する．つまり，特徴量を 172 個として学習する．パラメータは，バッチサイズを 16，エポック数 100，損失関数と最適化関数，学習率は，4.1 と同様にする．

4.4. ConformerEncoder を用いた手話単語認識

次に，文献[4]の論文と同じエンコーダデコーダ型の学習モデルを使って学習を行う．文献[4]の論文では，英文の音声データを学習させているが，本研究では手話の 1 単語を 1 文として学習させる．ネットワークの構造は，文献[4]同じように ConformerEncoder を用いて，デコーダには single-LSTM-layer を採用する．学習データは 4.1 と同じものを使用する．バッチサイズを 16，エポック数を 120，損失関数と最適化関数は 4.1 と同様にした．学習率を 0.001 として学習する．学習率が 4.1 と異なるのは，モデル自体が変わり，適切な学習率が変わったためである． 101 単語にと文の最初を示す記号である sos と終わりを示す記号である eos を加え，合計 103 個に ID を割り振り，文として認識させる．

4.5. 語彙数の拡張

4.1 と同様の方法で手話技能検定試験 5 級の単語のうち 78 単語を新たに追加して学習を行った．これによって新たな単語を認識できるネットワークを開発し，さらに語彙数の増加によって認識率が大きく変化するかを検証する．バッチサイズを 32，エポック数は 120，損失関数と最適化関数，学習率は 4.1 と同様にした．

4.6. 手話辞書システムへの適用

本研究で作成する認識エンジンを，本研究室の手話辞書システムに適用する．また，同様に姿勢推定に MediaPipe を用いる．そして，その実用性や検索時間の変化を検証する．

5. 研究結果

5.1. ConformerBlock を用いた多クラス分類手話単語の認識

ConformerBlock を用いて学習させた結果，単語ごとの認識率にばらつきがあるものの，約 97.4%の認識率が得られた．先行研究[1]の認識率である約 93%と比べて，認識率が約 4.4%向上した．今回の学習での損失の推移を図 3 に，認識率の推移を図 4 に示す．また，テストデータの認識において 20 個中 3 つ以上誤認識した単語を表 1 に示す．表 1 より，「暑い」と「南」の 2 つの認識がうまくいっていないことがわかる．これは，同じ手話動作で異なる意味を表す同形異義語であるからである．図 5 に示した手話は，利き手を顔の近くで仰ぐ動作であるが，この手話は「暑い」，「南」，「夏」，「うちわ」の 4 つの意味を持ち，それを区別するためには口の動きでそれぞれの言葉を表現する必要がある．「昨日」と「一昨日」の手話を図 6 に示す．この 2 つの手話はどちらも手を前から後ろに動かすものだが，指を一本立てるか，二本立てるかという違いがある．「金曜日」と「OK」の手話を図 7 に示す．これはどちらも人差し指と親指で輪をつくるという同じ手型であるが，「金曜日」は手を振る動作が入り，「OK」は手型を提示するだけである．

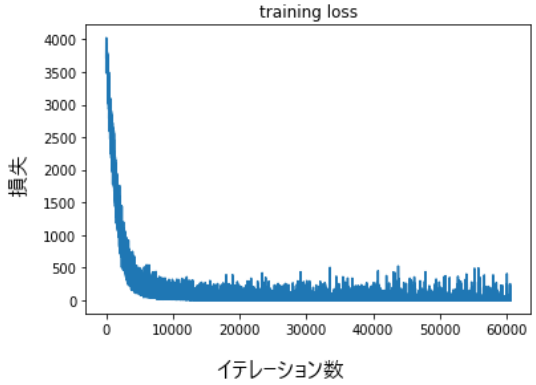


図 3：ConformerBlock を用いた多クラス分類による損失の推移

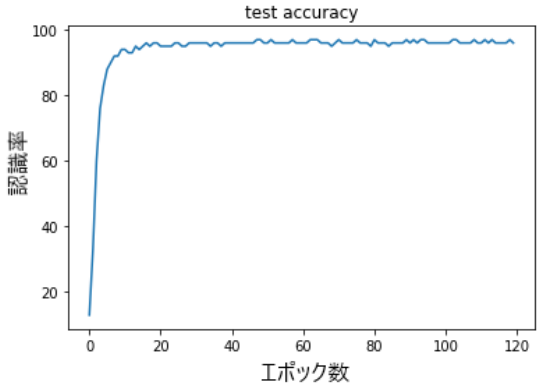


図 4：ConformerBlock を用いた多クラス分類による認識率の推移

表 1：ConformerBlock を用いたモデルでの誤認識した主な単語

単語	誤認識した数（20 個中）	主に誤認識された単語
暑い	7	南
南	5	暑い
一昨日	3	昨日
金曜日	3	OK

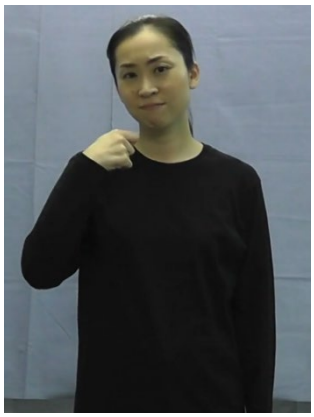


図 5：「暑い」と「南」の手話

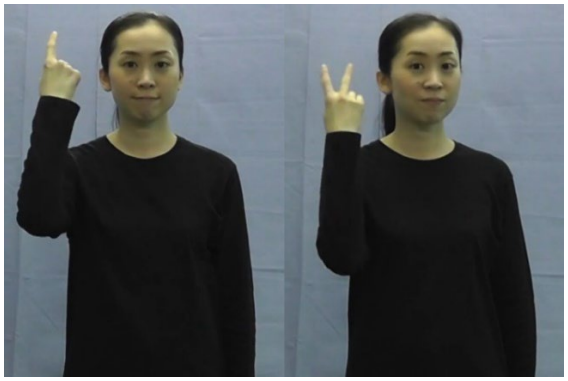


図 6：「昨日」（左）の手話と「一昨日」（右）の手話

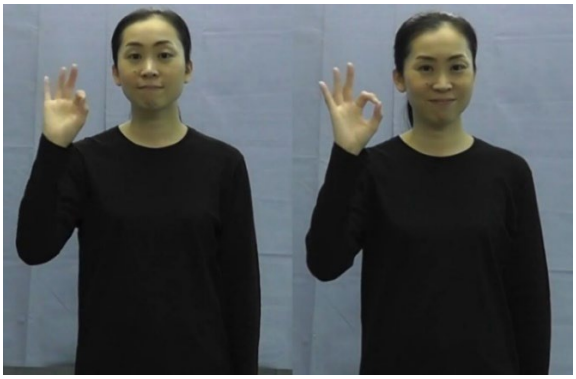


図 7：「金曜日」（左）の手話と「OK」（右）の手話

5.2. 唇の特徴点を含めた手話単語認識

唇の座標を含めた学習データを用いて学習した結果，約 93.7%の認識率となり，若干低下した．認識率が向上しなかった原因としては，肩の大きさで正規化した唇の座標は，値の変化が小さく，口の形の変化をうまく学習できなかったことが考えられる．さらに，動画によっては口を動かしていないものもあったため，同時に学習するのは困難であると考える．また，データに単語によって差の生じない特徴量が増えてしまったため，認識率が低下したと考えられる．よって同形異義語を認識するための読唇については，専用のシステムを用意したほうがよいと考えられる．

5.3. ConformerEncoder を用いた手話単語認識

ConformerEncoder を用いて学習させた結果，4.1 と同様に単語ごとの認識率にばらつきがあるものの，約 96.1%の認識率が得られた．学習での損失の推移と認識率の推移を図 8 と図 9 に示す．認識率は 4.1 に比べて少しだけ低下した．よって，単語単位の認識では多クラス分類を用いた方が良いと考えられる．また，手話文の認識においても先行研究を上回る認識率が期待できる．

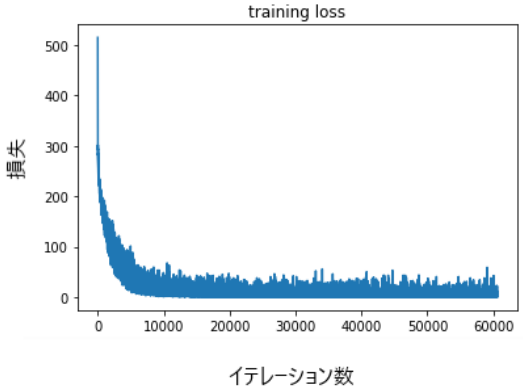


図 8：ConformerEncoder を用いた手話単語認識による損失の推移

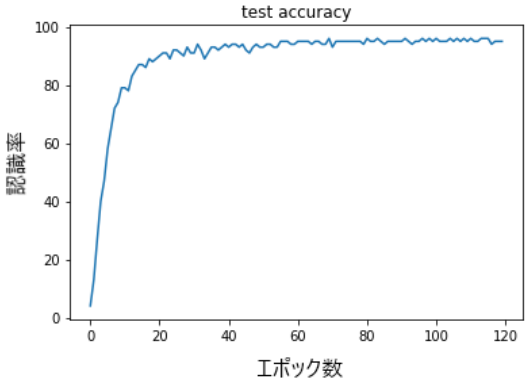


図 9：ConformerEncoder を用いた手話単語認識による認識率の推移

5.4. 語彙数の拡張

語彙数を増やして学習させたところ，約 96.9%の認識率が得られた．また，テストデータの認識において 4 つ以上誤認識した単語を表 2 に示す．4.1 と同様に同形異義語の「夏」と「南」が認識できていない．さらに，「数」と「いくつ」，「辛い」と「甘い」，「売る」と「買う」，「コーヒー1」と「コーヒー2」の認識率が低

い．

まず，「数」の手話と「いくつ（数)」の手話を図 10 に示す．この 2 つの手話はどちらも親指から順に 5 本の指を曲げるというものであるが,「数」はこれを 1 回，「いくつ（数)」はこれを 2 回行うという違いがある．

次に「辛い」と「甘い」の手話を図 11 に示す．この 2 つの手話はどちらも手で口の前に円を描くように動かすが，手型が「辛い」は指を曲げ，「甘い」は指を伸ばすという違いがある．

そして，「売る」と「買う」という手話を図 12 に示す．この 2 つの手話は，利き手はお金を表す「OK」の手型，非利き手は，商品を持っていることを表す「手のひらを上」にした手型である．「売る」はお金をもらって商品を渡す動作なので，利き手を自分側に，非利き手を相手側に移動させる動作となる．「買う」はその逆となる．

最後に「コーヒー1」と「コーヒー2」は，意味は同じであるが，手話表現が異なるものである．2 つの「コーヒー」の手話を図 13 に示す．この手話は，利き手の動作（スプーンでかき混ぜる動作）が同じで，非利き手の手型が異なる(カップの形状とカップを持つ仕草)だけである．

このように，誤認識してしまう単語には，手型や動作の回数が異なるものや動きが逆の手話が多い．

しかし，複数個誤認識してしまう単語があるものの，結果としては語彙数の増加によって認識率が大きく低下することはなかった．よって本手法の有効性が確かめられた．

表 2：ConformerEncoder を用いたモデルでの誤認識した主な単語

単語	誤認識した数	主に誤認識された単語
数	9	いくつ（数）
南	8	暑い
辛い	5	甘い
売る	5	買う
コーヒー1	4	コーヒー2

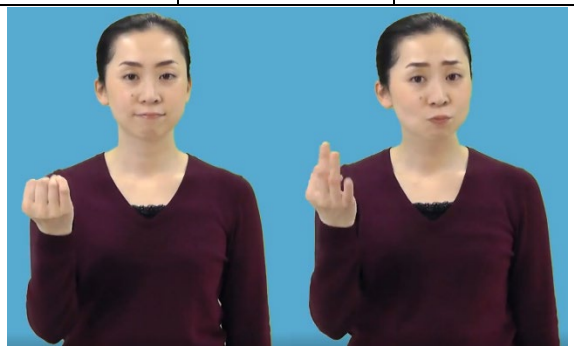


図 10：「数」（左）の手話と「いくつ（数)」（右）の手話

図 11：「辛い」（左）の手話と「甘い」（右）の手話

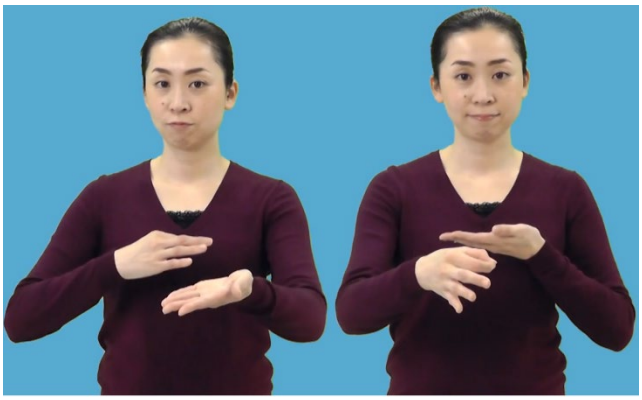


図 12： 「売る」（左）の手話と「買う」（右）の手話

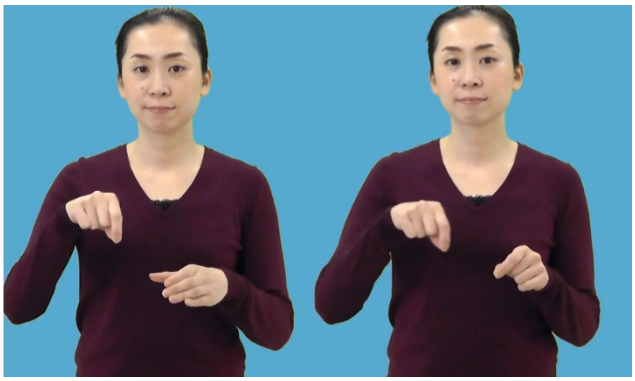


図 13： 2 つの「コーヒー」の 手話

5.5. 手話辞書システムへの適用

MediaPipe と 4.1 で開発した手話技能検定試験 6 級の 101 単語をクラス分類するネットワークを本研究室の手話辞書システムに適用した．従来システムと本研究のシステムで認識結果が出るまでの時間を計測した．3 秒，4 秒，5 秒の動画をそれぞれ 5 回送信し，その時間の平均値を調べる． それぞれの実行環境を表 3 に，計測した結果を表 4 に示す．録画する手話動画が 3 秒の場合，従来システムでは動画の送信から姿勢推定とモデルの推論の時間に約 13.84 秒かかった．本研究のシステムでは， 3 秒の動画を送信してから推論結果が出るまでに約 7.44 秒かかっていた．他の秒数の動画でも認識結果が出るまでの時間が短くなっており，この

実行環境では，従来システムに比べて検索時間が平均約 44.7%短縮された．従来システムはローカル PC 上に実装し，本研究ではクラウド上（AWS : Amazon Web Server）に構築しているため，単純な比較は出来ないが，検索時間がおよそ半分になっている．

表 3 に示した従来システムでの実行環境で OpenPose と MediaPipe で姿勢推定をし，CSV 形式のファイルを出力するまでにかかった時間を計測し，比較する．OpenPose に対する MediaPipe の姿勢推定時間の比率を表 5 に示す．表 5 より，動画の秒数が 3 秒である場合，OpenPose を用いた場合は 9.94 秒であるが，MediaPipe を用いた場合は 4.76 秒と短くなっている．どの秒数でも MediaPipe での姿勢推定の方がかかった時間が短い．さらに，比率の平均が約 47.3%となっており，本研究のシステムと同様におよそ半分の時間になっている．以上のことから，本研究のシステムで時間が短縮されたのは，OpenPose から MediaPipe に変更したことで，姿勢推定から CSV 形式のファイルを作成する時間が短くなったことが要因であることがわかる．

また，認識の結果はうまく認識できる単語とうまく認識できず他のものと誤認識しやすい単語があり，ばらつきがあった．

表 3：実行環境

	以前のシステム	現在のシステム
OS	Ubuntu20.04.3 LTS	Windows11 home
CPU	Intel(R)Xeon(R)Platinum 8259CL 2.5GHz	AMD Ryzen 5 2600X Six-Core Processor 3.60 GHz
GPU	なし	NVIDIA GeForce RTX 2060
RAM	7.7GB	32GB

表 4：動画の秒数と認識結果が出るまでの時間と短縮された割合

秒数	以前のシステム	現在のシステム	割合
3 秒	13.84 秒	7.44 秒	46%
4 秒	17.28 秒	9.74 秒	44%
5 秒	18.42 秒	10.32 秒	44%

表 5:動画の秒数と OpenPose と MediaPipe での姿勢推定時間とその比率

秒数	OpenPose	MediaPipe	比率
3 秒	9.94 秒	4.76 秒	48%
4 秒	12.2 秒	5.60 秒	46%
5 秒	13.6 秒	6.51 秒	48%

6. 考察

5 節の研究結果を経て得られた新たに用いた手法の有効性と，手話認識における問題点について考察する．

6.1. 新たに用いた手法の評価

手話の認識率向上に向けて新たに用いた手法について考察する．

本研究では，姿勢推定の手法と学習モデル，損失関数を変更した．姿勢推定は OpenPose から MediaPipe に変更した．また，学習モデルと損失関数は，LSTM と CTC を用いていたが，本研究では Conformer と交差エントロピーを使用した．手法の変更と認識率の向上について考察する．先行研究[1]での認識率である約 93%から約 4.4%向上した．これは，姿勢推定と学習の 2 つの手法の変化によるものであると考えられる．まず，OpenPose から MedaiPipe の変更によって認識率が向上したことについて考察する．先行研究では，本研究でも認識できていない同形異義語の「夏」と「南」に加えて，「思う」を「昨日」と誤認識してしまう．また，「何」の認識率が低い．まず，「昨日」と「思う」の手話を図 14 に示す．図 14 のように，どちらの手話も手型は同じである．しかし，「昨日」は手を後ろへ動かし，「思う」は額に近づける．この違いでは，2 次元での手の変化を考えると，手の位置の変化が少ないため，「思う」を「昨日」と誤認識してしまうと考えられる．これは，奥行きの座標の変化が学習できないと，うまく認識できない．よって，OpenPose を使った姿勢推定でこの 2 つの手話が認識できなかったが，MediaPipe を用いて認識できたといえる．

図 14：「昨日」の手話（左）と「思う」の手話（右）

次に，「何」の手話を図 15 に示す．この手話は，図 15 のように人差し指を立てて左右に動かすというものである．この手話に認識率が低いのは，指の座標の変化がうまく取得できなかったことにあると考えられる．座標の変化が小さくなったのは，座標の正規化を行っていなかったことが理由である．よって，今回座標の正規化をしたことにより，「何」のように他の手話に比べて指の変化が手話動画に依存してしまう，学習しづらい動作の単語を学習できたといえる．したがって，MediaPipe に変更し，座標の正規化を行ったことで，認識率が向上したと考えられる．

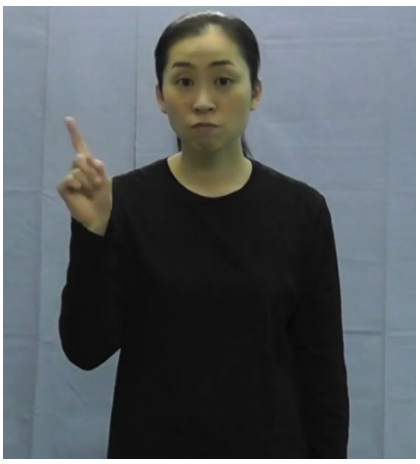


図 15:「何」の手話

また，手話認識システムに適用した結果，以前のシステムよりも認識結果が出るまでの時間が短くなったのは，OpenPose に比べて動作が軽い MediaPipe の影響であると考える．

最後に，Conformer による認識率の向上について考察する．本研究室内で，MediaPipe を用いて関節点の座標を取った本研究と同じデータを使い，学習モデルに GRU (Gated Recurrent Unit)を採用して学習した結果，約 95%の認識率となっている．GRU とは，RNN の一種であり，LSTM よりも計算量が少なく，性能はほぼ同等といわれている学習モデルである．GRU を用いた学習に比べて，本研究で使用した Conformer を使用した学習結果の方が，認識率が高くなっていることから，Conformer は手話認識において有効な学習モデルであると考えられる．

以上のことから，本研究で新たに採用した姿勢推定と学習モデルの 2 つの手法は，どちらも手話単語の認識率の向上に役立っており，有効な手法であるといえる．

6.2. 認識率が低かった単語

手話技能検定試験 6 級と 5 級の単語を学習した結果，表 1 と表 2 のように，認識精度が高くない単語がいくつかあった．その原因について考察する．

まず，「夏」と「南」だが，先も述べたように，全く同じ動きである同形異義語なので，認識できなかった．これを解決するためには，口の動きの学習を改善する必要がある．この問題は，本研究室内で進められている，読唇の研究と組み合わせることで改善できると考えられる．その手法としては，本研究で開発したネットワークによって「夏」または「南」と認識された場合に，読唇専用のモデルを使うことで，2 つのうちどちらを意味しているのかを識別するものが有効であると考える．読唇では，口の座標を口の中心点を基準にして正規化している．これによって学習をしているが，これを本研究にも適用できれば同形異義語の認識も可能になるといえる．つまり，手や腕の動きを本研究で行ったように肩幅を基準にして正規化し，唇の座標のみを口の中心を基準に正規化したものを 1 つにした学習データを用いることで，同形異義語の認識ができると考えられる．

次に，表 1 で示した「一昨日」と「昨日」，「OK」と「金曜日」であるが，「一昨日」と「昨日」の手話は，どちらも手を後ろに動かすものであるが，図 6 からわかるように，立てる指が変わる．「一昨日」が人差し指と中指で，「昨日」が人差し指のみである．また，「金曜日」と「OK」であるが，図 7 からわかるように，どちらも手型は同じであるが，「金曜日」は顔の横で手首を何回かひねり，「OK」は顔の横から少し前に出す．このように，2 つの手話は似ているため，認識率が高くならなかったと考える．この 2 組の手話を誤認識した数はそれぞれ 20 個中 3 つであり，個別の認識率としては 85%となる．この認識率は，手話辞書システムとしての実用性はあるが，改善点でもあると考える．また，5 級の単語である「辛い」と「甘い」，「数」と「いくつ（数)」，「買う」と「売る」といった単語も，この 2 組の単語と同じように手話動作が似ているものである．これらの単語では，手話動作が小さい場合や，手型がはっきりしていないときに誤認識してしまうと考えられる．

このように，似ている単語を識別できるようにする方法としては，アンサンブル学習が有効であると考える．アンサンブル学習とは，複数の学習済みモデルの推論結果を用いて，単体のモデルの性能を上回るようにするための手法である．つまり，似ている単語のみを識別できるようにするネットワークを開発し，実際に手話を認識する際には複数のモデルの推論結果を考慮して認識結果を出力することで，より認識率の安定したシステムになるのではないかと考える．しかし，似ている単語は手話技能検定試験 6 級と 5 級の範囲において，先に示したようにいくつもある．今後，語彙数が増えるたびに手話動作や表現が似ている単語も増えていくと考えられる．これら全てに対応するように学習させたいくつかのモデルを手話辞書システムに実装するのは，システムの動作を遅らせる原因になる可能性が高い．もしこのような状況になれば認識率を高くするために動作を遅らせるか，高い精度を求めないが動作が軽いものにするかというトレードオフの関係になってしまうと考えられ，現実的ではないといえる．

6.3. Conformer の応用

Conformer の本研究室内で進められている研究への応用について考察する．

本研究では，手話単語の認識に Conformer を利用した．特に，4.4 で使用したエンコーダデコーダ型のモデ

ルは，手話文，指文字，読唇などの認識に応用できる．本研究の 4.4 では，1 単語を 1 文として学習した．その際に，101 単語と文の最初と最後を表す記号の合計 103 個に ID を割り振って学習した．この単語に割り振ったものを，指文字では五十音に置き換え，読唇では母音に置き換えることで同じように学習することが可能である．さらに，将来的には同じモデルを用いてそれぞれの学習をすることで，違う認識エンジンを用いることなく，1 つのネットワークで手話を認識できるようになると考える．

7. 今後の課題

今回の研究では，手話単語の認識率が向上し，新たな手法の有効性も確認できたが，これから Conformer を活用していくにあたっていくつかの問題点と改善点がある．

Conformer は，End-to-End の音声認識に用いられている．End-to-End の学習には，大量の学習データを必要とする．しかし，手話文，指文字，読唇の研究に応用するためには，十分なデータ数がない．今あるデータを使った End-to-End の学習では，学習データにある手話文や指文字の単語を認識することは可能であるが，学習データにない手話文や指文字の単語を認識することは困難である．しかし，手話辞書システムにおいて手話文を認識する場合には，学習データにない文を認識できるようにしなければならない．そのために，本研究室で行われていた手話文から手話単語ごとに分割する研究[11]を利用するという手段が考えられる．これによって，認識を単語単位で行い，その認識結果を組み合わせることで，学習データにない手話文の認識が可能になる．同じように，指文字も五十音ごとに分割すると，うまく学習できるといえる．

他には，本研究のように単語のみを学習したうえで，手話文の学習を行う転移学習という手法も考えられる．これによって，少ないデータ数でも手話文の認識ができる可能性がある．

さらに，文献[4]の音声認識の際に前処理として使われていた SpecAugment のように，学習データを時間軸においてシフトやマスクをかけることによって学習データの水増しをし，学習するという Data Augmentation という手法もある．しかし，この手法は認識結果が学習データに偏りすぎてしまい，うまく認識できなくなるという過学習を引き起こす原因にもなるため，検証が必要である．このように，本研究で行った姿勢推定や学習モデルの変更ではなく，学習時の手話動画に対する前処理を工夫することで，解決できる可能性がある．

今回の研究で約 97.4%と高い認識率のネットワークの開発に成功したが，手話辞書システムに適用するにあたって認識率が下がってしまう可能性が高い．これは，テストデータが学習データとほぼ同じ環境で撮影された手話動画であることが原因になると考えられる．これにネットワークの学習段階において対応する方法として，先ほど挙げたようにマスクをかけるというものと，学習させる手話動画の長さを考慮するというものがある．Conformer は，元は音声認識に使われており，音声データは話す文によって長さが変わる．そのため，モデルに入力するときにはパディングで長さをそろえるが，同時にモデルにそれぞれのデータの長さも入力している．本研究では，クラス分類をする場合において動画の長さを全て同じとして学習させたが，この点を変更し，それぞれの手話動画の長さを考慮した学習を行うことで，手話辞書システムにおいて高い認識率を保つことができると考える．

8. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）および電気通信普及財団の助成を受けたものである．

文　献

[1] 秋田大輔， CTC 手法を用いた手話単語認識エンジンの作成，豊田高専卒業論文，pp1-8，2021

[2] “Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks”， ICML '06: Proceedings of the 23rd international conference on Machine learning ，pp. 369-376，2006

[3] Ashish Vaswani， Noam Shazeer， Niki Parmar，Jakob Uszkoreit， Llion Jones， Aidan N. Gomez，Lukasz Kaiser， Illia Polosukhin，” Attention Is All You Need”， 31st Conference on Neural Information Processing Systems (NIPS 2017)， Long Beach，pp5998-6008， CA， USA，2017

[4] Anmol Gulati， James Qin， Chung-Cheng Chiu，Niki Parmar， Yu Zhang， Jiahui Yu， Wei Han，Shibo Wang， Zhengdong Zhang， Yonghui Wu，Ruoming Pang，”Conformer:Convolution-augmented Transformer for Speech Recognition ，” ，INTERSPEECH 2020，no. 10. 21437 ， pp. 5036-5040， Shanghai， China， 2020.

[5] Zhe Cao，Tomas Simon，Shin-En Wei，Yaser Sheikh，”Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields”，volume. 1，pp. 1-9，2016

[6] Mediapipe， https://google. github. io/mediapipe/，2021 年 12 月 22 日閲覧

[7] 磯谷光，木村勉，神田和幸，” ディープ・ラーニングを用いた手話認識に関する研究”，信学技報，vol1.120，no.419，WIT2020-38，pp.47-52，2021.

[8] Chainer: A flexible framework for neural networks，https://chainer. org/，2021 年 01 月 26 日閲覧

[9] PyTorch， https://pytorch. org/， 2021 年 12 月 22 日閲覧

[10] JupyterNotebook，https://jupyter.org/，2022 年 2 月 14 日閲覧

[11] 磯谷光， 手話文章からの単語の抽出に関する研究，豊田高専卒業論文，pp1-8，2020

手話認識システムの小型デバイスへの実装と開発ツール群の作成

豊田工業高等専門学校　情報工学科　深津 昂希，木村　勉

あらまし　豊田高専 木村研究室で開発中の手話認識システムは，手話技能検定試験 6 級レベルの 101 単語の手話表現を学習させており，93%の精度を出しているが，特定の語句に対しての認識率が低いことや，処理が遅いため，実用に向かなかった．本研究では，姿勢推定エンジンの変更を行い，新たにシステムを作成し，これらの問題の解決を図った．さらに，開発したシステムを小型デバイスに実装し，評価を行った．

キーワード　手話認識，深層学習，ディープラーニング，Raspberry Pi

1. 研究背景

豊田高専 木村研究室では機械学習を用いて手話単語を学習させ，人の動きから特定の手話を認識するシステム[1]（以下従来研究）を開発している．これは処理の負荷が高く GPU が必須のため搭載できる端末に制限があった．これは，このシステムに搭載されている姿勢推定エンジンの OpenPose[2]の影響によるものである. OpenPose の手話単語動画の処理時間を表 1 に，動画の仕様を表 2 に示す．また，実行環境を表 3 に示す．このように，手話辞書システムとして利用するには OpenPose では動作が重いと考えられる．

また従来研究で，このシステムに手話技能検定試験 6 級の 101 単語を学習させたところ，奥行きによって差のある手話単語の識別ができなかった．識別できなかった手話単語の例を図 1，図 2 に示す．これらの単語は，手型は同じだが，奥行きのある動きに差があり（図 1 は後方，図 2 は前方に移動），3 次元の動きが認識できないため, これを識別することができなかった. さらに，この他にも動作の似ている単語や同じ動作であるが，口の動きや文脈で意味が変わる同形異義語の認識がうまくできていなかった．同形異義語には対応できないが，より軽量で，奥行きの検知ができる姿勢推定エンジンを用意することで, システムの動作速度, 認識精度を向上させることができると考えられる.

表 1 OpenPose による動画処理時間

ライブラリロード	約 10 秒
動画処理	動画時間に対して約 2.2 倍

表 2 検証に用いた動画の仕様

FPS	60
動画時間	10.51 秒
コーデック	H.264/MPEG-4 AVC

表 3 実行環境

CPU	AMD Ryzen5 5600X 6-Core Processor@3.70Hz
RAM	16.0GB
GPU	NVIDIA GeForce RTX 3060 Ti
OS	Windows 10 Home

図 1 「先週」の手話単語の手の動き

図 2 「来週」の手話単語の手の動き

2. 目的

本研究では，従来研究で使用してきた姿勢推定アプリと学習フレームワークの変更により，処理速度の向上，前後移動を伴う単語，および動作の似ている単語の認識率を向上させるとともに，小型デバイスの Raspberry Pi へ実装することを目的とする. この際に，手話認識システムを開発する上で必要なツール群を作成する.

3. 開発環境

本研究の開発環境を表 4 に，システムを実装する小型デバイスの Raspberry Pi の仕様を表 5 に示す．また使用したアプリケーションについて述べる.

表 4 開発環境

CPU	AMD Ryzen5 5600X 6-Core Processor@3.70Hz
RAM	16.0GB
GPU	NVIDIA GeForce RTX 3060 Ti
OS	Windows 10 Home
開発言語	Python 3.9.6
開発ソフトウェア	Visual Studio Code 1.64.0
機械学習ライブラリ	PyTorch 1.10-cuda
姿勢推定ライブラリ	MediaPipe (Holistic) 0.8.9.1
動画編集ライブラリ	FFmpeg 4.4.1

表 5 使用した小型デバイスの詳細

デバイス名	Raspberry Pi 4 Model B
RAM	8GB
OS	Ubuntu Server 20.04 LTS

3.1. MediaPipe (Holistic)

本研究では姿勢推定アプリを Google 社が開発した MediaPipe[3]に置き換える．従来研究で利用されていた OpenPose と比較して，とても軽量であり，GPU を必要としない．また，3 次元情報の取得が可能である．

MediaPipe には，様々な解析モデルが内包されており，本研究で利用するのは，Holistic モジュールである．これは，手，体全体，顔の詳細な解析をするモデルを同時に実行するモジュールである．入力データを読み込む際の処理を共通して行えるため，別々に動作させる場合より速く処理を行うことができる．解析後は関節などのキーポイントの 3 次元座標を返す．図 3，図 4，図 5 にそれぞれのキーポイントを表した図を示す．各図の数値がキーポイントを表し（図 5 は交点がキーポイントであるが，数が多いため省略している），各キーポイントの 3 次元座標を返す．

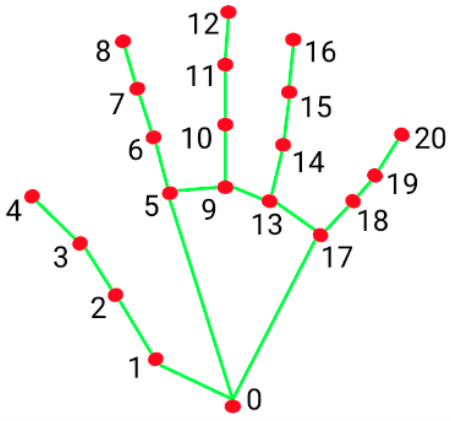


図 3 左手のキーポイント

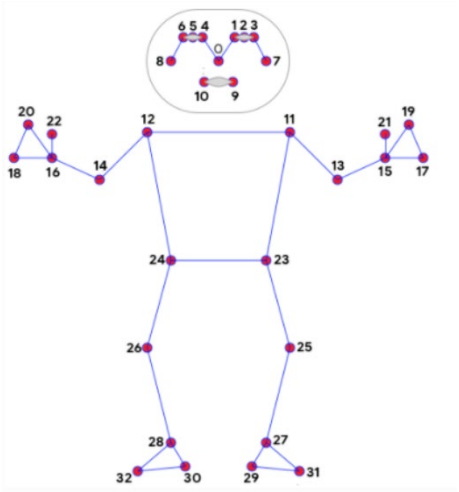


図 4 体全体のキーポイント

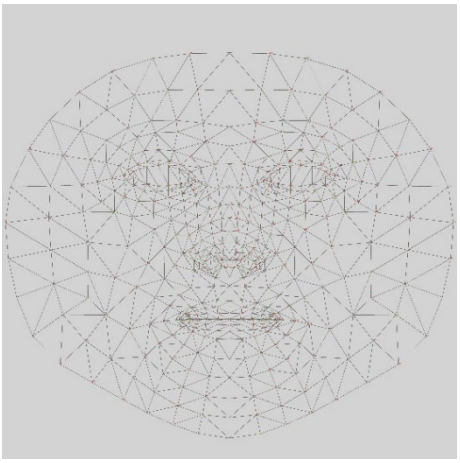


図 5 顔のキーポイント

3.2. PyTorch

学習フレームワークとしてオープンソースである PyTorch[4]を使用する．これは，様々な分野で活用できるように，最先端の手法の実装，多岐にわたる拡張ライブラリを備えている．PyTorch の特徴として，内部のデータ処理を詳細に変更することが可能であることが挙げられる．従来研究で用いられていた Sony の Neural Network Console [5] （以下 NNC）は，ウェブページ上の GUI で学習モデルを構成，学習を行うことができるウェブサービスである．これに学習データを入力する際，データ長をすべてそろえる必要があったため，一括でパディング処理を行う必要があったが，PyTorch では動的にデータの長さに対応して学習を行うことが可能になるため，より効率的に学習を行うことが可能になる．また，近年の機械学習の研究で多く用いられており，新たに開発された学習手法等を取り入れやすいことも利点として挙げられる．

3.3. GRU

ニューラルネットワーク構造として，ゲート付き回帰型ユニット（GRU : Gated recurrent unit）[6]を採用する．これは，従来研究で用いられていた，時系列デー

タの学習に向いているLSTMのゲート機構を簡略させることで，計算の効率化を狙った改善版である．LSTMと比較して，学習成果を落とすことなく，少ないパラメータ数でも学習を進めることが可能であり．言語解析の分野で多く利用されている．これは，PyTorch に移行することで新たに利用することができる．

3.4. RAdam 関数

学習の際に用いる最適化手法として，RAdam[7]を採用し，PyTorch 内に実装されている RAdam 関数[8]を利用する．従来研究で利用されていた Adam[9]は学習初期に起こる学習率の分散が大きく，学習が進みづらくなる場合があり，Warmup と呼ばれる別の学習工程を学習初期に加えることでこれを回避していた．Warmup を行うにはパラメータの適切な設定が必要で，何度も学習の推移を観察して，最適なパラメータを探す必要があった．RAdam は，学習初期に起こる分散を抑える機構を Adam に組み込んだものであり，Warmup の工程が不要となり，パラメータを設定することなく，学習をより進みやすくすることができる．

3.5. クロスエントロピー誤差

学習の際に正解との差分を計算する方法として，従来研究と同じくクロスエントロピー誤差を利用する．クロスエントロピー誤差の式を式(1)に示す．変数 p，q は確率分布であり，この 2 つの確率分布が似ているほどクロスエントロピー誤差は小さい値となる．作成する学習モデルの出力は確率分布であるため，誤差を計算する方法として適している．これについては，PyTorch 内に実装されている CrossEntropyLoss クラス[10]を利用する．

$$\mathbf{H}(\mathbf{p},\mathbf{q}) = -\sum_{x} \boldsymbol{p}(\boldsymbol{x}) \log(\boldsymbol{q}(\boldsymbol{x})) \qquad (1)$$

3.6. FFmpeg

動画編集には，オープンソースの動画編集ソフトウェアの FFmpeg[11]を利用する．これはコマンドラインで動作し，Python で操作することが可能である．

3.7. Raspberry Pi 4 Model B

小型デバイスとして Raspberry Pi 4 Model B[12]を利用する．Raspberry Pi とは，小型のシングルボードコンピュータである．国内外で広く普及しており，情報が手に入りやすい．また，MediaPipe を動作させることができるように高性能なタイプを利用する．

3.8. Ubuntu Server 20.04 LTS

小型デバイスの OS としては Ubuntu Server 20.04 LTS を採用する．広く普及しており，Raspberry Pi 用のインストーラが用意されている．また，動作速度を向上させるため，GUI アプリケーションをインストールしない，コンソールのみの Server タイプにする．

4. 研究内容

4.1. MediaPipe による姿勢推定

MediaPipe を利用し，手話認識で必要となる身体のキーポイントの取得を行うプログラムを作成する．3.1 に示した取得されるキーポイントには，手話認識に不要なキーポイントも含まれる．これらにより，学習の進みが悪くなることが考えられるため，この情報を取り除く処理を追加する．

また，これに動画撮影時の環境による影響を小さくするため，取得したキーポイントの正規化（人の位置と大きさ）と，異なるフレームレートに対応するためのフレーム数の調整を行う処理を追加する．

更に，動画の読み込み処理と MediaPipe による解析を並列に処理をすることで高速化を図る．動画読み込み処理は，すべての動画を読み込むまで次の処理に移行できない．そこで，プロセスを二つ利用してこの問題を解決する．一つが動画を読み込み，プロセス共有のバッファに格納する処理を行い，もう一つのプロセスが共有バッファから得たデータを MediaPipe により処理を行うようにする．

最後に，作成した姿勢推定プログラムの処理時間を検証する．

4.2. 手話認識システム向けのツール群の作成

手話認識システムを効率よく開発するために，幾つかのツールを作成する．

1) 姿勢推定プログラムの並列化ツール

現在木村研究室には，学習用のデータセットとして，手話技能検定試験 6 級の 101 単語について，各 100 個のデータ（計 10,100 個）を用意している．これを 4.1 で作成するプログラムで解析するには時間がかかってしまう．そこで，4.1 で作成するプログラムを並列化させるツールを作成する．処理したい動画が一つのディレクトリにあることを想定する．再帰的にディレクトリ下の動画を探し出し，マルチプロセスで同時にプログラムを実行するようにする．この際，4.1 で作成した動画の読み込みを並列化させる処理は利用できるプロセス数が減ってしまうため行わない．

2) 手話動画自動整理ツール

新しい単語のデータセットを用意する場合，撮影した動画から手話単語のみを切り出す必要がある．これは，撮影時にモデルの方の負担を軽減させるため，カメラは止めずに手話単語を連続して表現して貰っているからである．そのため 1 つの動画ファイルに複数の単語（約 300 個）が含まれている．これを 4.1 で作成する姿勢推定プログラムを利用し，その出力から自動的に単語ごとに分割させる．手話は手を顔近くに上げて行うため，手が腰より下に置かれている状態を単語の分割点とみなして抽出するツールを作成する．また

分割後，アノテーションを行うときに利用するため，動画撮影時に表現中の単語名を吹き込んでいる．この音声も含んだ動画とするため，姿勢推定プログラムの出力から分割点を算出したのち，FFmpeg を用いて動画を分割する．

3) 学習用データ確認ツール

MediaPipe を利用して得られたデータが適切かどうかを確認するため，各キーポイントを描画し，動画で出力するツールを作成する．

4.3. 学習フレームワークの PyTorch への移植

PyTorch を用いた学習済みモデルの作成を行う．使用するニューラルネットワーク構造は，新たに利用することができる GRU である．従来研究で利用されていた LSTM と認識精度，処理速度を比較し，検証を行う．

次に，クロスバリデーションで学習を行い，認識精度を求める．この際，一度に多くのデータを学習させると効率よく学習が進むため，パディング処理を行う．データの長さで，三つに分類しそれぞれ最大のデータ長を基準にパディング処理を行う．

また，PyTorch に用意されている AMP(AUTOMATIC MIXED PRECISION)[13]を学習時に利用する．これは変数のメモリ利用を最適化する手法であり，メモリ使用量が減ることでバッチサイズをより大きくすることが可能になり，計算量が少なることで処理が速くなることが期待できる．

4.4. 新システムの開発

4.1 で作成した姿勢解析プログラムと，4.2 で作成した各ツールにより生成した学習用手話データセットを 4.3 で構築したネットワークに学習させて，学習済モデルを作成する．そしてそれを利用するシステムを開発する．その後評価を行い，実用化に向けての検証を行う．

4.5. 小型デバイスへの実装による評価

3.7 で示した小型デバイスに MediaPipe，PyTorch の環境構築を行い，4.4 で作成した新システムを実装する．その際，小型デバイスの GPU は画像解析に用いるには，メモリと処理性能が低く，かえって遅くなるため，MediaPipe，PyTorch ともに GPU を利用しないタイプのものを選択する．実装後，実用に耐えうるかどうか，動作速度を基に検証する．

5. 研究結果

5.1. MediaPipe による姿勢推定

4.1 で述べた姿勢推定プログラムを完成させた．手話を認識するには手型と腕の動きが必要である．手のキーポイントは，図 5 で示した場所を取得することができる．体のキーポイントは図 6 の〇で囲んだ部分だけでよい．そこで，この部分だけを抽出するように処理を追加した．

正規化に関しては，中心点を，肩と肩の中心として行うようにした．手話においては，上半身のみで表現が可能なためである．図 7 に原点と基準点を示す．左上が原点であり，(0,0)から(1,1)の値で表現される．抽出する範囲は，肩幅の 4 倍を範囲としている．肩幅とは左右の腕の間隔である．腰より下は動画に映っていないことも考えられ．一番安定して人の大きさを測れる方法として，肩幅をとることとした．

図 8 の画像をここで述べた正規処理を行い，図 9 に図 8 の画像を正規化して出力したデータを学習用データ確認ツールで描画した図を示す．図 8 の画像ではアスペクト比が縦と横で差があり，右に人が寄っているが，正規化後の図 9 の画像では，アスペクト比が 1:1 となり，人物も中央に配置されている．

また，動画読み込みの並列処理を実装したところ，処理時間が約 3 分の 2 に縮小した．

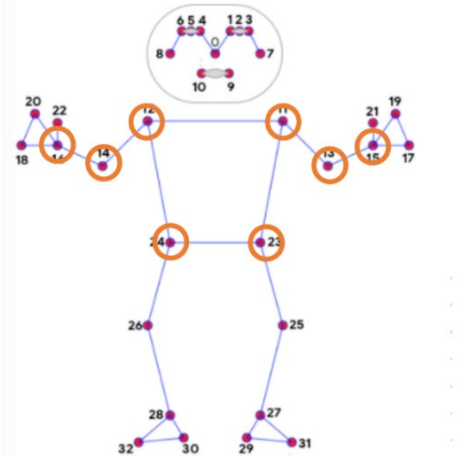


図 6 体全体の出力から取得するキーポイント

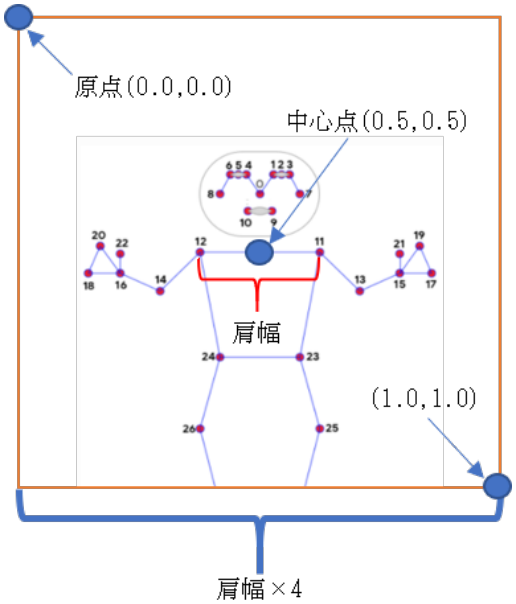


図 7 正規化の基準

図 8 正規化前の画像

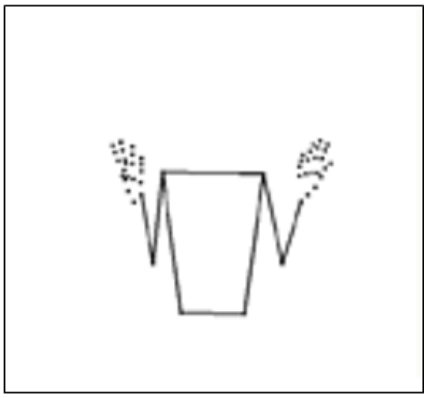


図 9 正規化後の画像

5.2. 手話認識システム向けのツール群の作成

4.2 で述べたツール群を作成し，手話認識システムの開発用ツールとして他の研究にも応用している．

姿勢推定プログラムの並列化ツールに関しては，利用しているコンピュータで利用できるプロセスを最大限利用して並列処理を行うことができるようになった．表 3 の開発環境では，12 プロセスを同時に利用することができ，並列処理なしでは 20 時間ほどかかる処理を約 2 時間で終えることができた．

手話動画自動整理ツールに関しては 5.1 で作成した姿勢推定プログラムにより取得した座標から，分割点を設定することで，適切に分割することができた．手話の動作の中で，手が一瞬，腰の高さまで下がることがある．これは分割点とする対象ではないため，一定時間以上手が腰の高さまで下がっている状態にならないと分割しないようにした．また，学習させた際，手話の始点，終点を認識することが可能になると考えられるため，分割点の手が腰より下にある状態の映像を一定時間含めるようにした．

学習用データ確認ツールに関しては，正面からと，側面から見た際の両方を描画できるものを作成した．図 10 に元動画，図 11 にツールによる正面から見た場合の出力，図 12 に側面からの出力を示す．図 11 から，手型が推定できていることが確認できる．また図 12 から右手（図中の上側の長い線）が前に，左手が膝の上に置かれている様子が確認できる．

図 10「来週」の手話

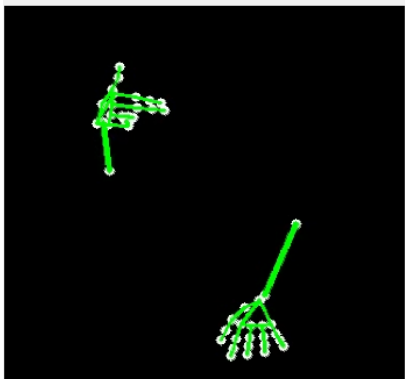


図 11 確認ツールの出力（正面）

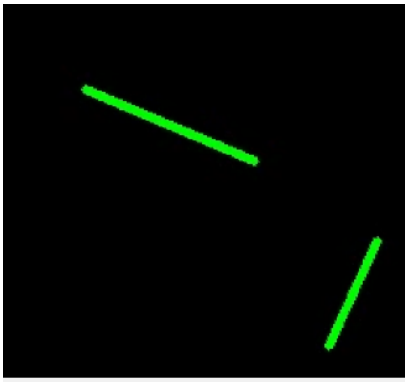


図 12 確認ツールの出力（側面）

最後に，処理時間に関して動画を入力して検証を行った．結果を表 6 に示す．ここで使用した動画は表 2 で使用した動画と同じものである．

表 6 姿勢推定プログラムによる動画処理時間

ライブラリロード	約 2 秒
動画処理	動画時間に対して約 0.8 倍

5.3. PyTorch への対応

これまで，NNC を利用して開発してきたシステムを PyTorch に対応させた．そして，LSTM，GRU それぞれで学習モデルを作成し，5.1，5.2 で開発したプログラムおよびツールを用いて作成したデータを学習させた．学習の推移を図 13 に示す．また，学習させる際のパラメータを表 7 に示す．LSTM で行った場合は，ロ

スの減少が進まなくなる，学習の停滞が頻繁に起き，ロスが約 24 から減少しなくなり，最後まで学習を進ませることができなかった．一方，GRU で行った場合，停滞することが少なく，ロスが 0 に近い値になるまで学習を進めることができた．

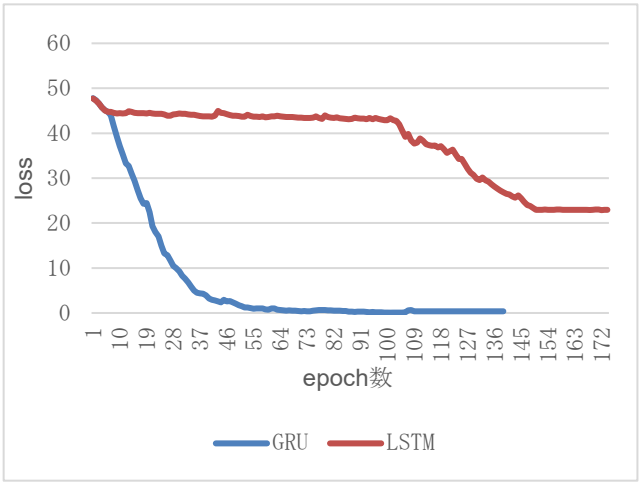


図 13 GRU と LSTM の学習の推移の比較

表 7 学習パラメータの概要

Epoch 数	1000
学習率	0.01
隠れ層数	100
損失関数	クロスエントロピー
最適化関数	RAdam 関数

次に，作成したプログラムの出力を 9:1 の割合で学習データとテストデータに分け，クロスバリデーションで学習を行った．ニューラルネットワーク構造は GRU であり，パラメータは先程示した表 5 のパラメータである．正答率と学習進度を示したグラフを図 14 に示す．ロスが 0 に近い値まで下がり，正答率はロスが下がるにつれて上昇し，最終的に 95%となった．また，テストデータによる検証により，主に間違えた単語，および，間違えた割合を表 8 に示す．

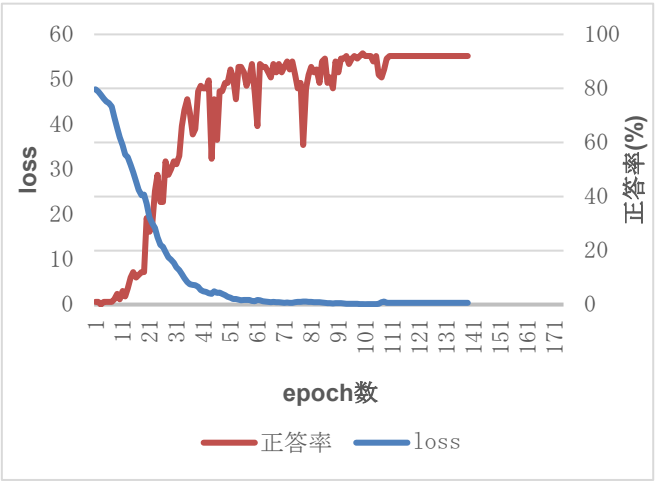


図 14　学習の推移

表 8 テストデータによる検証で間違えた箇所

単語	主に間違えた語	誤り率
南	暑い	40%
弟	妹	30%
昨日	一昨日	60%
両親	母	20%
すき	きらい	20%
祖父	父	20%

5.4. 新システムの開発

4.4 に述べた新システムを開発し，手話動画を新たに用意し，利用できるかどうか，検証した．その結果，正答率が著しく低く，約 10%だった．4.3 で述べたパディング処理を入力データに行うと，正答率が上がり，おおよそ 5.3 で検証した際の結果と同じような結果になった．

5.5. 小型デバイスへの実装による評価

4.5 に述べた通りに，小型デバイスへ 5.4 で作成した新システムを実装し，実際に動画を入力して処理速度を検証した．結果を表 9 に，使用した動画の仕様を表 10 に示す．

表 9 小型デバイス上のシステムによる処理時間

ライブラリロード	約 3 秒
動画処理	動画時間に対して 約 1.2 倍
学習済みモデルによる 解析	約 0.3 秒

表 10 検証に用いた動画の仕様

FPS	60
動画時間	12.21 秒
コーデック	H.264/MPEG-4 AVC

6. 考察

5 節の研究結果から，新たな手法の有効性と手話認識における問題点について考察する．

6.1. 正答率が低かった単語について

研究結果の表に示した，誤認識をした単語について考察する．まず，奥行きに差のある単語に関しては，誤認識がなかった．よって，新たに作成した姿勢推定プログラムによって奥行きの検知が可能になったと考えられる．

また，「暑い」と「南」においては，同形異義語であり，同じ表現の手話であるため，本研究で認識率を向上させる対象としていない．

次に，一番誤認識の多かった「昨日」と「一昨日」や「すき」と「きらい」，「祖父」と「父」，「両親」と「母」においては動作が似ており，指の細かい動作に違いのあるものである．「祖父」と「父」の場合，「祖父」は「父」の動作をしたのち，親指を曲げることで

区別がされる．親指の動作が細かく短いため，区別がしにくくなっていると考えられる．また，「昨日」の手話を図 16 に，「一昨日」の手話を図 17 に示す．この二つは，腕や手の動きは同じで，立てている指が人差し指だけか，中指も立てているかのみが異なる．この手話を読み込み，5.2 で作成した描画ツールにより，描画した図 18 と図 19 に示す．「昨日」では人差し指のみ，「一昨日」では人差し指に加え中指が立てられ，二本立てられていることがわかる．よって，作成した姿勢推定プログラムによる解析では指の検知ができていると考えられる．これらのことから，指の細かい動作は，学習が適切にできていないことによって認識ができていないと考えられる．

図 16 「昨日」の手話

図 17 「一昨日」の手話

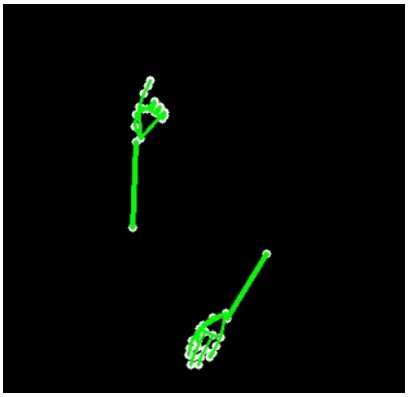


図 18 「昨日」の姿勢推定結果

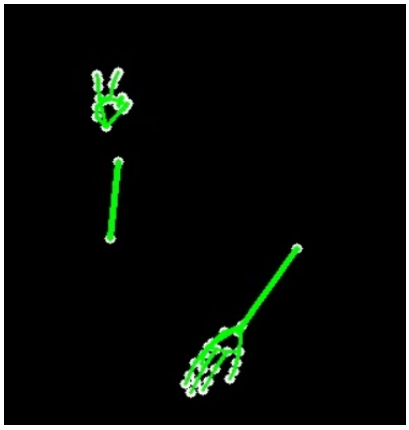


図 19 「一昨日」の姿勢推定結果

6.2. 新たに採用した手法の評価

新たに MediaPipe で作成した姿勢推定プログラムによる出力から学習させたモデルは 6.1 より，奥行きの検知ができるようになり認識率が向上した．さらに，5.1 で行った動作速度の検証により，OpenPose の場合より，ライブラリのロード時間が 8 秒削減され，処理速度が約 2.75 倍になったため，手話認識システムの処理速度，認識率を向上させたといえる．

また，現在のデータ数では，PyTorch により GRU を利用するほうが学習に適していると考えられる．来研究の手法で学習が進みづらくなったのは，MediaPipe を利用することにより取得できる座標点が増えたことによる影響と考えられる．OpenPose で得られる座標の数は，78 個に対して，本研究で作成した姿勢推定プログラムでは 138 個と約 1.8 倍に増えている．STM と GRU では大きな内部構造の違いはないため，GRU は LSTM に比べ簡潔に学習が進むことによる差と考えられる．よって，データ数を増やせば，LSTM でも学習が可能と考えられる．

6.3. 認識率の低下について

研究結果から，4.3 の学習する際のパディング処理がシステムに組み込んだときの学習済みモデルの正答率を下げている要因であるといえる．学習済みモデルは，パディング処理によって値埋めした箇所も含め学習しており，そこを単語の終点として学習していると考えられる．そのため，パディング処理をしていない動画を入力すると，単語が終わっていないとされ，異なる結果となってしまったと考えられる．

また，利用しているニューラルネットワーク構造である GRU は，最後に入力したデータに比較的大きく影響受ける特徴がある．動画は最初から入力しており，パディング処理はデータの末尾に行っているため，より影響をうけてしまったと考えられる．

6.4. 小型デバイスへの実装の評価

手話単語の表現時間は最大で約 10 秒であるので，

研究結果から，Raspberry Pi で実行しても実用には問題ないと考えられる．よって，小型デバイスでも利用できる軽量なシステムが構築できたといえる．

7. 今後の課題

まず，細かい動作の検知をより正確に行う必要がある．そのために，学習用データを増やす必要があると考えられる．学習用データを増やすには，さらに動画を撮影するか，既存の学習データを加工して増やす方法が考えられる．既存の学習データを加工する場合，例えば動作の速度を変更することが考えられる．

次に，本研究では対象としていなかったが，同形異義語の検知のために，口の動きの検知をする学習モデルを作成する必要がある．

更に，学習方法の再検討を行い，パディング処理の影響を受けにくい学習済みモデルを作成する必要がある．対応策として，パディング処理をなるべく少なくすることが考えられる．4.3 で述べたように，本研究では，データの長さで三つに分類してそれぞれにパディング処理を施した．これをさらに細分化してパディング処理を行うことで，影響を抑えることができると考えられる．また，学習方向を変更することも対応策として考えられる．6.3 で述べたように，GRU は最後に入力したデータに比較的大きく影響受ける特徴がある．そこで学習方向を動画の最後から読み込む方法もしくは双方向に読み込み合算する方法をとることで，影響を抑えることが可能と考えられる．

8. まとめ

研究の結果，MediaPipe を利用することにより，3 次元座標データを取得することが可能であることが確認でき，処理速度も大幅に上昇することが分かった．本研究において新たに見つかった課題を解決することで，手話認識システムの精度向上，処理速度の向上を図ることができ，小型デバイスへの実装も可能であると確認できた．

9. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）および電気通信普及財団の助成を受けたものである．

文　献

[1] 木村 勉他，"手話認識機能を備えた手話辞書システムの開発"， 信学技報， vol. 120， no. 419， WIT2020-39， pp. 53-58 (2021).

[2] “CMU-Perceptual-Computing-Lab/openpose” ， https://github.com/CMU-Perceptual-Computing-Lab/openpose， 2021 年 10 月 6 日閲覧

[3] "Holistic-mediapipe" ， https://google.github.io/mediapipe/solutions/holistic ， 2021 年 6 月 10 日閲覧

[4] 赤石 雅典， 最短コースでわかる PyTorch &深層学習プログラミング゛， 日経 BP，2021

[5] "Neural Network Console"， https://dl.sony.com/ja/， 2021 年 1 月 28 日 閲覧

[6] Junyoung Chung， Caglar Gulcehre， KyungHyun Cho， Yoshua Bengio，"Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling",arXiv:1412.3555，(2014)

[7] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, Jiawei Han,”On the Variance of the Adaptive Learning Rate and Beyond” ,arXiv:1908.03265(2019)

[8] “RAdam — PyTorch 1.10 documentation”, https://pytorch.org/docs/stable/generated/torch.optim.RAdam.html?highlight=radam#torch.optim.RAdam, 2021 年 12 月 12 日閲覧

[9] Diederik P. Kingma, Jimmy Ba,”Adam: A Method for Stochastic Optimization”, arXiv:1412.6980(2014)

[10] “CrossEntropyLoss — PyTorch 1.10 documentation”,https://pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html, 2021 年 12 月 11 日閲覧

[11] “FFmpeg”，https://www.ffmpeg.org，2021 年 6 月 10 日閲覧

[12] “Raspberry Pi 4 Model B specifications – Raspberry Pi” ， https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/，2021 年 12 月 11 日閲覧

[13] “Automatic Mixed Precision package - torch.cuda.amp — PyTorch 1.10 documentation”, https://pytorch.org/docs/stable/amp.html, 2021 年 12 月 11 日閲覧

手型推定による指文字認識の研究

豊田工業高等専門学校　情報工学科　坂口 孝志，木村　勉

あらまし　本研究では，これまでの研究において成果の出ている手話認識から，指文字の認識への拡張を目指す．音声認識の手法である Connectionist Temporal Classification を用いて音から音への遷移を除去し，指文字における音素部分のみを認識・学習させる．まず，指文字の認識が困難であると考えられていた要因を整理し，使用するツールやデータの選定を行った．例年利用されてきた姿勢推定ソフトウェア OpenPose を MediaPipe に変更することで，取得可能な手型の特徴点数を増やし，姿勢推定の精度向上を図る．深層学習フレームワークに関しても昨年度まで使用されていた Chainer から TensorFlow へ移行する．

キーワード　指文字，手話認識，深層学習，ディープ・ラーニング，CTC，MediaPipe，TensorFlow

1. 目的

これまでの豊田高専 木村研究室の研究により，手話単語の認識については成果が出ている[1].

しかし，指文字認識に関しては，研究を進めてこなかった．この主な原因として手型を正確に認識することができていなかったということが挙げられる．例えば従来研究でも図 1 の「明日」と図 2 の「明後日」のように動きが同じで手型のみが違うものについては認識精度が低かった.

これらの認識精度が低かった原因の一つとして，従来研究で使用されていた姿勢推定ソフト OpenPose[2]では，手型の推定精度が低かったことが挙げられる．そこで本研究では，より多くの特徴点に対しデータを取得できる MediaPipe[3]を使用することにより，手型の検出/推定精度を向上させて従来の手話認識から派生して指文字が認識できるようにすることを目標とする．

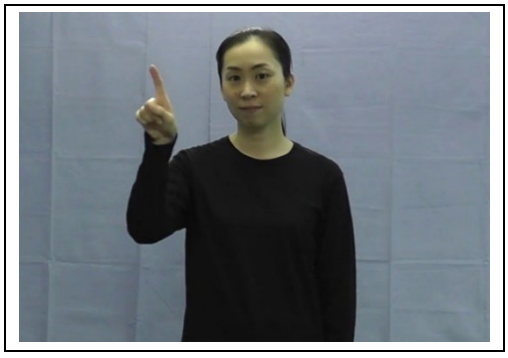


図 1:明日の手型

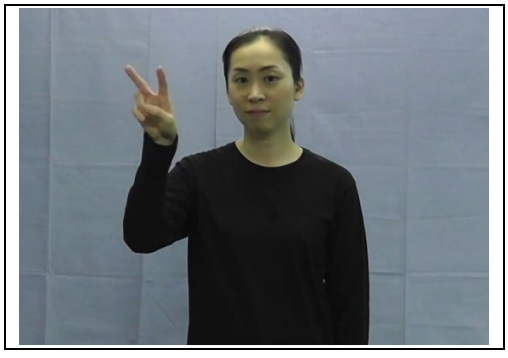


図 2:明後日の手型

2. 研究背景

これまでの研究において，CTC (Connectionist Temporal Classification)を用いて手話の文章から単語を抽出することができた[4]．この研究は，例えば「私の名前は佐藤です」という文章から「私」,「名前」「佐藤」,「です」の 4 つの手話単語を認識するものである．指文字認識を行うにあたり，基本的にはこの研究成果を応用する．例えば「さとう」と指文字で表現したときに「さ」,「と」,「う」の 3 つの音の指文字を認識させる．手話における単語認識との違い，指文字認識が困難である理由は以下の通りである.

1) 動画のブレ

手型の推定や識別が困難である．図 3 のようにカメラのシャッタースピードの影響により手がぶれてしまい，手型がはっきりしない動画が多いことに加えて，指のみで表現するため，動きが小さく識別が難しい.

図 3：手型がぶれている様子

2）奥行き情報の不足

指文字には手を手前に引いて表現するものがある．（拗音（ゃゅょ）や促音（っ）など）これにより，縦と横の動きだけでなく手の奥行き情報も必要である.

3）遷移の認識

指文字は 50 音に対応しているが，それらの連続した動作で単語を表現するため，1 つの指文字から別の指文字に遷移する際の区切りの判別が困難である.

4) 指文字の表現の違い

指文字表現には図 4 に示した「あ」や「か」のように「手型」のみで表現する静的なものと，図 5 に示した「の」や「り」のように「動き」を伴う動的なものがある．特に動的なものに関しては，指文字から指文字への遷移やほかの動きを伴う表現との区別が困難である．

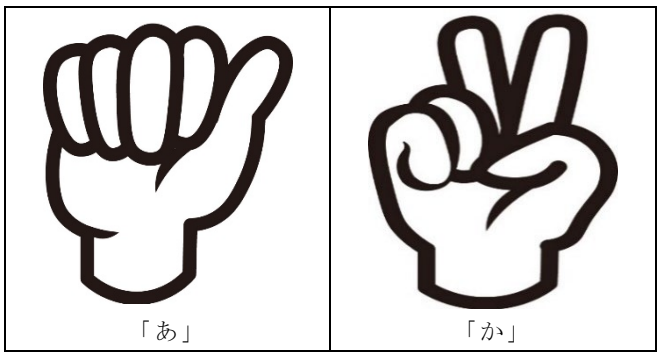


図 4：静的な指文字の例

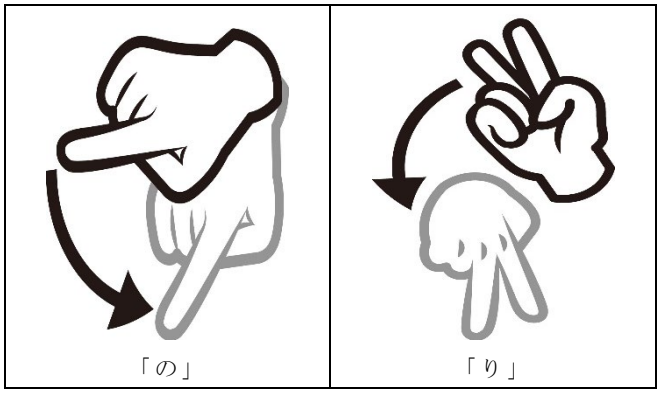


図 5：動的な指文字の例

これらの解決方法として，以下のように対応する．1) および 2) については，MediaPipe を利用することで改善を図る．図 6 の比較からわかるように，MediaPipe では，OpenPose よりも取得できる関節点などの特徴点の数が多く，より詳細に手型の情報を得ることができる．

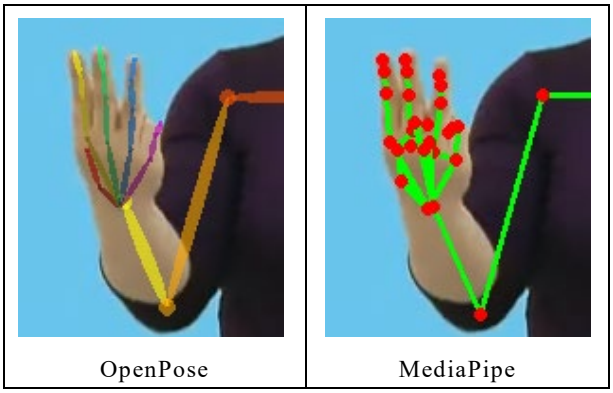


図 6：推定ソフトによる手型認識の違い

また，奥行き情報も含めた手型の推定が可能であるため従来では不可能だった Z 軸の動きにも対応できると期待される．3) および 4) については，これまでの研究成果を応用して，CTC 手法を用いて遷移の除去を行い対応する．

3. 関連研究

指文字認識の分野では，様々な手法で研究が行われてきた．その手法は 2 つに大別され，センサなどを取り付けたグローブを着用して認識を行う手法[5]と，カメラによって撮影した動画から座標点を取得して取り扱う手法[6]である．グローブによって認識を行う手法は比較的古くより行われてきたが，ハードウェアにかかるコストが高い点や装着時に指の動きに違和感が生じる点，外出先での利用が困難な点などがデメリットとして挙げられており，カメラによって撮影した動画を扱う手法にシフトしている傾向がある．また，2 つの手法に共通して深層学習を用いて認識を試みる事例が増えている．

これら 2 種類の方法では，表 1 に示すようも類似の手型を持たず，動きも伴わない静的な指文字表現に関しては 90%を超える精度での認識に成功している．

表 1：どの手法でも安定して精度が出やすい清音

わ	ら	や			な	た	さ	か	あ
					に		し	き	い
	る		む	ふ	ぬ		す	く	う
	れ		め	へ	ね	て	せ	け	え
	ろ	よ		ほ		と		こ	お

しかし，動的な指文字表現に関しては高くとも 80%程度，低い場合には 50%を下回ることもあり 60%から 70%程度の認識率に収まることが多い．横山 和明，森本 正志らの行った研究[7]においても遷移を伴うもの，静的，動的な指文字表現の出現順序などの条件を整えているにもかかわらず，表 2 のように比較的低い認識結果となっている．これは 2 節でも触れた通り，遷移や他の動的な指文字との区別が困難であることが原因と考えられる．本研究では，この問題を解決するために CTC 手法を取り入れ，遷移の切り分けを行うことで動的な指文字に関しても高い認識率を目指す．

表 2：指文字の認識結果

正解率	0.72		
	遷移	静的	動的
再現率	0.81	0.67	0.59
適合率	0.78	0.75	0.54
F 値	0.80	0.71	0.57

動的な指文字以外に認識率が低いとされているのは，図 7 に示した手型が同じで角度のみが異なる「ま」「み」「ゆ」などである[6]．これらは回転に関して不変な特徴量を使用している場合や，指先のみの情報を取得し手首の特徴点を抽出できない場合に低い認識率となってしまう．今回使用する MediaPipe は手首から指先までの特徴点を取得できることから，手型が同じで角度が異なる指文字にも対応できると予想される．

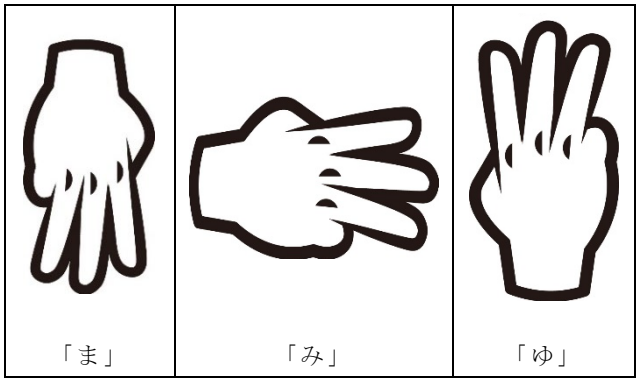


図 7：手型が同じで角度のみが異なる指文字の例

また，カメラによって撮影をした動画データやモーションセンサによって取得した座標点を扱う手法においては，限定条件下での認識に限られていることも問題点の一つであると考える．

これまでに提案されてきた手法では，認識精度を担保するために，一定以上の明るさや距離を保った状態の動画でなければならないなど，縛りを持ったシステムも多い．これらは汎用性の面では劣っており，実用化は厳しいという現状がある．撮影環境を統一しなければならない要因として掌と手の甲の区別がつかないことや従来の姿勢推定ソフトでは関節点が角度情報を持たず，システム側で複雑な計算処理が必要だったことが挙げられる．本研究で利用する MediaPipe は掌と手の甲の区別がつきやすく，また各関節点が隣り合う関節との角度情報を持つため，従来よりも汎用性を持った認識システムになることが期待される．

4. 研究の方針と準備

本研究を進めるにあたって採用した手法や内容についてまとめる．

4.1. 指文字認識の流れ

はじめに，指文字動画を読み込み，MediaPipe を用いて関節点など指の特徴点を取得する．それらを CSV 形式のファイルにし，単語リストの作成とアノテーションを行う．今回は，テストデータとして 105 単語，3,532 個のデータを用意する．

4.2. CTC

CTC は損失関数として，解析後の出力データに対する損失値の計算に使用されている．音声認識の深層学習では，音素の間に入るブランク表現に対して，CTC と呼ばれる手法でアプローチしている[8]．

例えば，CTC による「さとう」の音声認識は図 8 のようになる．同じ意味を持つ音声データでも，実際には音素の長さやブランクによる差が生じる．CTC はデータに対して，ブランク(”_”)の挿入と，音素の連続を表現させる．そのため，図 8 のように「ささ_と_う」「さとうううう」と表現される．その後，1) 同じ音素が連続した場合には，一つの音素にする．2) ブランクを削除する．というフレーム処理を当てはめることで「さとう」が認識できる．

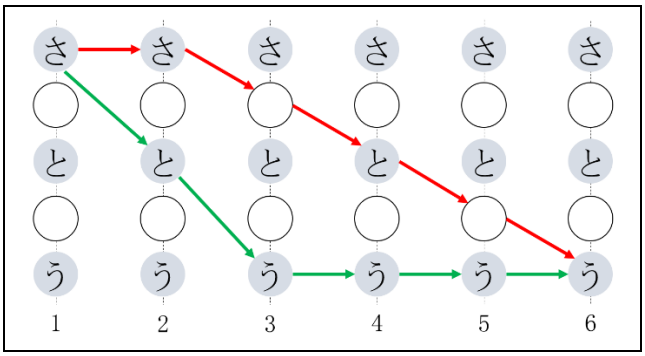


図 8：CTC による「さとう」の認識例

このように CTC は，個人差による音素の長さやブランクといった不安定要素を浮き彫りにしてから省略することで，認識したデータの情報を簡潔にする特性を持つ．ブランクの判定の際には，音の入力がない場合以外にも認識結果として曖昧だった場合にもブランクとして判断される．この処理によって，音と音の繋がりで発生する連音による誤認識の発生を防ぐ．音声認識でも音素から文字へラベル単位を拡張した機械学習が行われているため，同様にラベルが文字単位の手話認識でも CTC 手法を適応できると考えた．

また，CTC を利用することで，音素から単語を認識する際に発生するラベル数の差という問題も解決することができる．例えば，クロスエントロピーや平均二乗誤差などを損失関数とするニューラルネットワークを用いた学習では，出力データへ割り振られるラベルと正解ラベルの系列長が等しくなければならず，入出力ラベル数の統一，もしくはデータに対するフレーム単位でのラベルの整理が必要である．

これに対し CTC では，データのフレームごとに認識をすることで，この問題を解決している．具体的には，フレーム一つ一つに対して解析を行い，ラベルの割り振りを行う．フレームごとにラベルを割り振るとラベル数が膨大になるので，先ほどの図 8 のようにブランクの挿入とフレーム処理を用いて最終的に簡潔な形にする．欠点として，各ラベルの出力確率はフレームごとに独立であるため音素と音素前後関係は一切考慮されず，意味のある単語であるか単なる文字の羅列であるかの区別は行われない点が挙げられる．

4.3. MediaPipe

MediaPipe は Google 社が開発した機械学習向けのフ

レームワークである.TensorFlow[9]で実装されており，本研究では主に MediaPipe-Hands[10]ライブラリを使用する．Hands ライブラリでは，図 9 に示したように片手につき 21 個の特徴点を抽出することが可能であり，それぞれ XYZ の座標を取得できる．

また，指が交差しているような複雑な手型であっても認識が可能であるほか，カメラからの距離が 20cm から 2m 程度まで広く対応している点も大きなメリットである．

従来研究で利用していた OpenPose では Z 座標が取得できないため，奥行き情報を持つ動きに対応できない点や取得可能な特徴点自体も少ないことから，より高性能な MediaPipe への移行を決定した．本研究と直接の関連はないが，GPU が不要で CPU 単体でも動作可能な点も将来的なスマートフォンでのシステム利用などを考慮すると移行にあたってのメリットである．

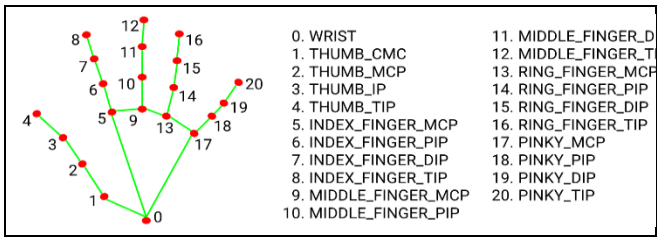


図 9：MediaPipe で取得可能な手の特徴点

4.4. ニューラルネットワークの構築

ニューラルネットワークには，時系列を持つデータの学習に優れた RNN（Recursive Neural Network）の一つである LSTM（Long Short-Term Memory）[11]を使用する．

RNN とは他の層から出力された出力データを入力データとして扱うことができるネットワークモデルであり，前の層で得た情報を次の層で入力データの一つとして扱うことができる．このため，時系列データを持ち，前後の情報が重要になる音声認識や動画認識において用いられることが多い．図 10 に示すように LSTM は RNN の中間層のユニットを LSTM Block と呼ばれるブロックに置き換えることで実現されている．

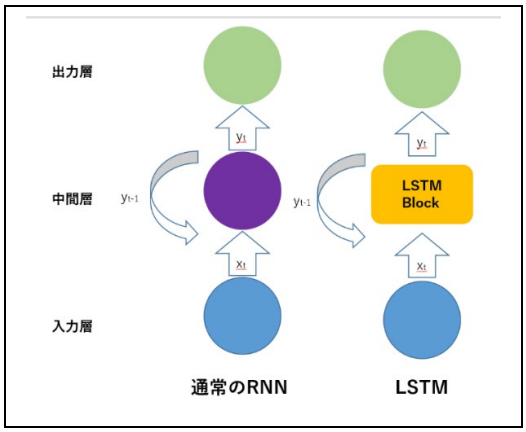


図 10：LSTM モデルと RNN モデルの概要図

LSTM Block は，忘却，入力，出力の 3 つのゲートとメモリを持つ．LSTM は RNN の改良モデルであり，次ステップへの情報の受け渡しに重みをつけることで，短い間の情報しか保持できなかった RNN よりも長いステップの情報の伝達が可能になった．LSTM の層による訓練の後に損失関数として CTC を使用している．

4.5. TensorFlow

従来研究で使用していた Chainer[12]のメジャーアップデートが終了したため，フレームワークの変更が必須となった．研究分野において直近数年では PyTorch[13]のほうが大きなシェアを占めており新しい情報が多い点や Chainer と PyTorch は共通して Define-by-run 方式であることを受け，PyTorch の利用を考えていた．しかし，PyTorch では CTC 利用のための関数の挙動が独特であり移行コストが高いため，今回は TensorFlow を使用する．

TensorFlow は Google 社が開発したオープンソースの機械学習ライブラリであり，Linux，macOS，Windows，Android など多くのプラットフォームで利用できるメリットがある．また，データの読み込みや前処理，計算，出力といった処理に対してテンソルを扱う特徴がある．デメリットとして，基本的に GPU の使用を前提とした設計になっているため，ハードウェア面での要求値が高くなってしまう点が挙げられる．

5. 研究内容

本研究における開発環境や研究の流れについて述べる．

5.1. 開発環境

本研究における開発・実行環境は表 3 の通りである．

表 3：開発環境

使用した PC のスペック	
CPU	IntelCore i9-10900X CPU@3.70Hz
GPU	NVIDIA GeForce RTX 3090
RAM	96 GB
OS	Ubuntu 20.04.3
使用したアプリケーション	
フレームワーク	TensorFlow 2.7.0
IDE	Jupyter notebook
使用言語	Python 3.9.7

5.2. 動画データの作成

従来研究では手話単語のみを取り扱っていたため，指文字の動画データは作成されていなかった．そこでまず，手話技能検定試験の 5 級と 6 級の学習 DVD に含まれる指文字動画を単語ごとに切り分け，1 つの動

画に 1 つの単語が対応した形の動画データを作成する．

指文字表現において，一連の指文字動作が終わった後には手を下げるため，図 11 のように手が動画からフレームアウトまたは一定の座標を下回った時という条件で自動的に分割を行い，それらの動画に対して，指文字で表現している語彙をファイル名とする．

この際，単語数は 300 個ほど確認できたが，同一動画の繰り返しばかりだったため，それでは学習に十分なデータ数とは言えない．そこで，新たに指文字の動画を撮影し，105 単語，3,532 個のデータを用意した．

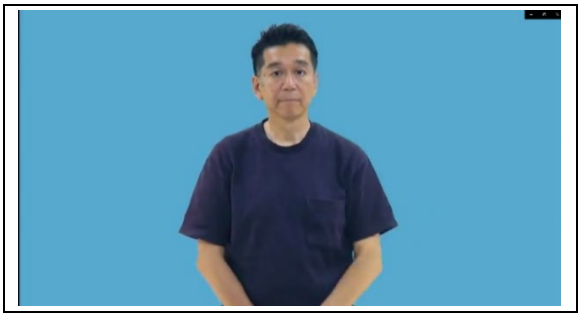


図 11：指文字動作が終わり，手を下げた様子

5.3. データセットの作成

動画データの作成完了後，MediaPipe を用いてデータセットを作成する．MediaPipe-Hands を利用して，指文字動画から特徴点の情報を抽出し，CSV ファイルとして出力する．105 単語 3,532 個のデータのうち 95%を訓練データとして利用し，残り 5%をテストデータとした．この際，指先のみでは手首の角度などの判別が難しいため，MediaPipe-Pose を用いて図 12 に示したキーポイントのうち，11 から 16 までの手首，肘，肩の座標データも取っておく．

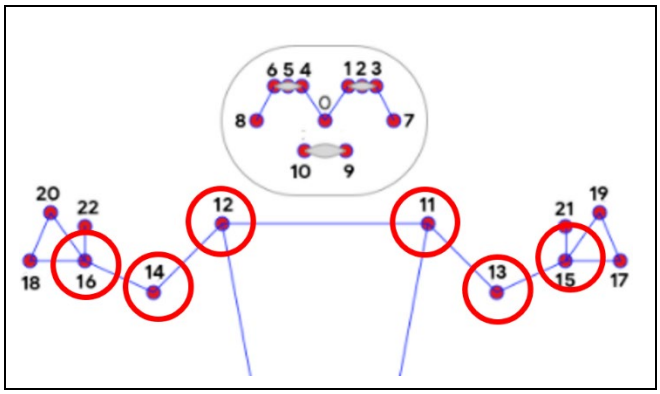


図 12：MediaPipe による姿勢推定

5.4. ラベル作成とアノテーション

次に，作成されたデータに対してアノテーションを行うためのラベル作成を行う．指文字は一つの音に一つの手型や動作が対応するため，50 音に濁音，半濁音，長音，拗音，ブランクなどを加えた合計 83 のラベルを用意し，それぞれのラベルに順に数値を割り当て，図 13 のようなラベルを作成する．完成したラベルを利用してアノテーションを行う．

```
label = { "" : 0,"_" : 1,
          "ア": 2,  "イ": 3,  "ウ": 4,  "エ": 5,  "オ": 6,
          "カ": 7,  "キ": 8,  "ク": 9,  "ケ": 10, "コ": 11,
          "サ": 12, "シ": 13, "ス": 14, "セ": 15, "ソ": 16,
          "タ": 17, "チ": 18, "ツ": 19, "テ": 20, "ト": 21,
          "ナ": 22, "ニ": 23, "ヌ": 24, "ネ": 25, "ノ": 26,
          "ハ": 27, "ヒ": 28, "フ": 29, "ヘ": 30, "ホ": 31,
          "マ": 32, "ミ": 33, "ム": 34, "メ": 35, "モ": 36,
          "ヤ": 37, "ユ": 38, "ヨ": 39,
          "ラ": 40, "リ": 41, "ル": 42, "レ": 43, "ロ": 44,
          "ワ": 45, "ヲ": 46, "ン": 47,
          "ガ": 48, "ギ": 49, "グ": 50, "ゲ": 51, "ゴ": 52,
          "ザ": 53, "ジ": 54, "ズ": 55, "ゼ": 56, "ゾ": 57,
          "ダ": 58, "ヂ": 59, "ヅ": 60, "デ": 61, "ド": 62,
          "バ": 63, "ビ": 64, "ブ": 65, "ベ": 66, "ボ": 67,
          "パ": 68, "ピ": 69, "プ": 70, "ペ": 71, "ポ": 72,
          "ァ": 73, "ィ": 74, "ゥ": 75, "ェ": 76, "ォ": 77,
          "ッ": 78, "ャ": 79, "ュ": 80, "ョ": 81, "ー": 82}
```

図 13：音すべてを網羅したラベル

5.5. ネットワークの構築

4.4 節 4.5 節で述べた通り，モデルには LSTM を，フレームワークには TensorFlow を使用してネットワークを構築する．損失関数には CTC を使用する．

5.6. 評価

5.5 節までで作成したデータセットとネットワークを使用し，学習結果を評価する．完全一致する単語数をカウントし正答率を評価する．この際，間違えた単語や認識されなかった単語に関してもリストにまとめ，どのような傾向があるのか調査・考察を行う．

6. 研究結果

研究結果を以下に記す．研究内容に記した内容から変更があった点や得られた結果は以下のとおりである．

6.1. 動画データの作成

実際に手話技能検定試験の学習 DVD からデータの作成を行ったところ，切り分けを行う過程で OpenCV を使用したため RGB 配列が BGR 配列になってしまい図 14 のように色調が大きく変化した．

図 14：BGR になってしまった指文字動画

当初は色調に変化があっても MediaPipe での推定には影響がないと考えていたが，背景や服との区別がつきにくく認識精度を低下させる要因になっている可能性が浮上したため色調を元に戻す処理を行った．

6.2. データセットの作成

MediaPipe を使用し，特徴点の抽出及び CSV ファイルでの出力を行った．研究期間中に MediaPipe がアップデートされ，指が重なっている場合や手の甲を向けている場合の精度が向上したため CSV ファイルの再出力を行った．

図 15 に示すように，指文字では人差し指と中指を交差させて行う「ら」が存在する．また，指を重ねて表現するかつ手型が類似している図 16 の「ち」「つ」も存在するため，指が重なっている際の精度の向上による影響は軽視できないものであった．実際にアップデート以前では重なりによって座標の値が抜けてしまっていた特徴点のうち，いくつかは値を取得できるようになっていた．


「ら」

図 15：「ら」の指文字表現

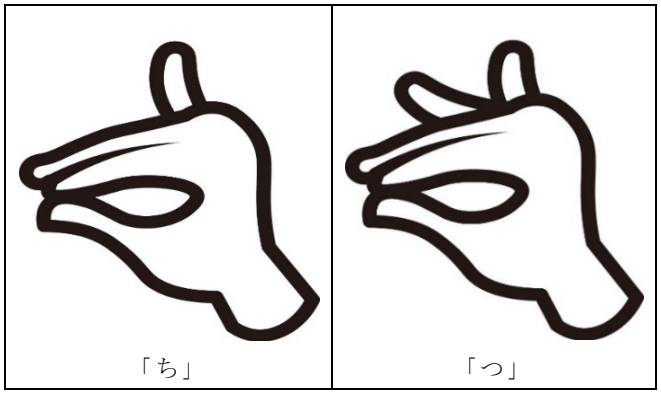

「ち」　「つ」

図 16：「ち」「つ」の指文字表現

6.3. ラベル作成とアノテーション

当初，ラベルを 83 個用意しアノテーションを行っていたが，後述の事情により濁音や半濁音などを含まない 50 音のみや，ア行のみなど細分化したラベル及びデータセットを作成した．

6.4. 評価

ここまでで作成したデータセットとモデルを利用し，学習を行った．しかし，結果としては図 17 のように nan(Not a Number)が出力されてしまい正しい結果を得ることができなかった．この原因の考察と改善のために以下のことを行った．

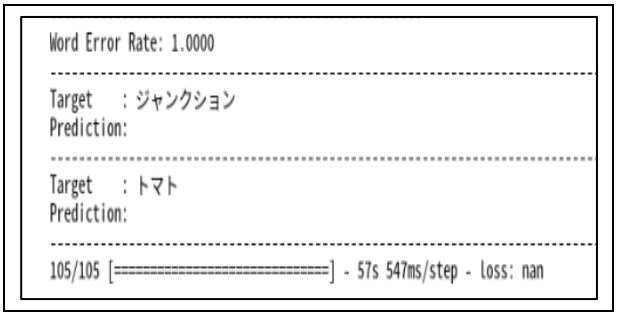


図 17:出力結果

6.4.1. 前提

過学習や勾配の消失によって nan が出る場合は loss の値が緩やかに変化し，学習途中で nan が出力される．今回のように，最初から nan が出てしまう場合はデータまたはモデルそのものにエラーがある可能性が高いと考えられる．

6.4.2. データの確認

まずはデータを検査し，欠損値や大きな外れ値が認められないか確認を行った．また，仮に欠損値を含む場合はデータ読み込みの時点で 0/-1 で埋める処理を追加した．同様にデータのエラーを考慮し，データを単語ごとに小分けにし，学習を行うなどの対策も取った．しかしながら改善は見られなかった．

6.4.3. ラベルと使用データセットの変更

ラベルが多すぎるため音素データが不足している可能性を考慮し，濁音や拗音を取り除いた単語データセットを使用して学習させた．同様に，ア行とカ行のみで表現されている単語など，条件を細かく設定し学習を行ったが，結果は変わらなかった．

6.4.4. その他行った対策

nan 出力の要因として多くあげられるのは 0 除算である．これが発生している可能性を考慮し，微小な値をあらかじめ加算しておく処理を追加した．その他にも学習率やエポック数を上下させての学習や，開発環境の確認，別の OS に変更するなどの対策も行ったが改善は見られなかった．

6.4.5. モデルの変更

モデルが原因である可能性は低いと考えていたが，原因の特定のため，別モデル(conformer[14])を使用し学習を行った．

その結果，単語の正答率(Accuracy)は 96%となった．また，誤認識をした単語は図 18 に示した通りであった．左側が正しい単語，右側が誤って認識された単語になっている．

誤って認識されたもののほとんどは学習済みの単語リスト

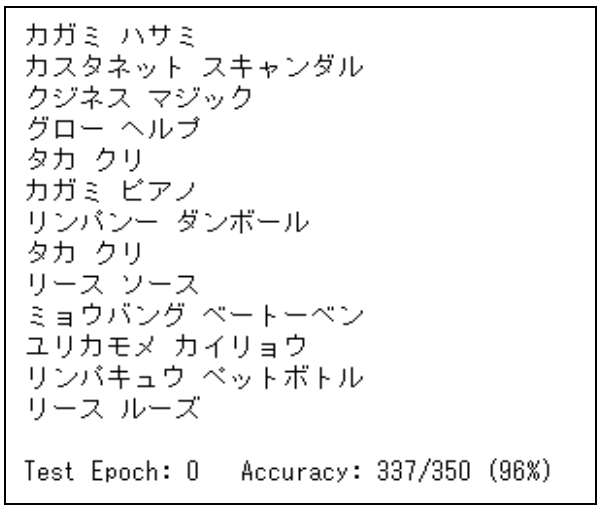


図 18：誤認識単語のリスト

に含まれないものであり，文字数は一致するものの，全く見当違いな回答を出力している．

7. 考察

作成したデータの精度，モデルの変更前，変更後のそれぞれの出力結果について考察する．

7.1. MediaPipe 利用による精度の向上

今回，OpenPose から MediaPipe への移行で特徴点の数とそれぞれの点における Z 軸分の情報量が増えたため，データ 1 つ当たりの持つ情報量は格段に増加した．

過去の研究におけるデータの差し替えで精度比較を行いたかったがフレームワーク移行ができなかった．

7.2. nan 出力について

出力結果が nan になってしまった原因について考察する．データのチェック及びデータに抜けや外れ値があった場合についての例外処理の追加や，ごく少数の単語データに絞っての学習などを試したが改善されなかったため，データに原因があるとは考え難い．

使用したモデルに関して，類似のモデルを使用して出力に成功している研究が存在するため，モデルに原因がある可能性としては低いと考えていたが，変更したところ出力結果を得ることができたため，モデルに原因があったと考えられる．

50 音を扱う指文字では，濁音や拗音を含めて 83 個ものラベルがあるため，当初の予定以上に多く集まった今回の学習データでも，まだデータ数が不足している可能性がある．加えて，指文字では手話のように「体のどの位置で行うか」といった条件が存在しないため，話者や姿勢が変わることによって発生する座標の差が大きい．これにより同一の単語であっても別単語として認識されている可能性がある．

7.3. モデル変更後の出力について

モデルの変更後について，出力には成功しているが完全一致で認識に成功するのは学習済みの単語の場合のみであり，未学習の単語の場合では 1 音も正しく認識できていないことが多い．これは音素単位ではなく単語単位での学習になっていることが原因だと考えられる．

また，長音のみ認識に成功しているケースがほかのテストデータで実行した場合にも見られ，これは拗音の手前に引く動作や半濁音などの上下に動く動作は種類が多いが，長音による横軸の動きは他の動作と識別しやすいからであると考えられる．「の」など動的な指文字を含む場合でも，学習済みであれば誤ることがなかったことから，単語単位で学習しか行えなかったが，学習済みの単語については動きや遷移を含む場合でも高い精度が得られたといえる．

8. 今後の課題

今回の研究では，当初望んでいた結果を得ることはできなかった．前節の考察を基に，今後行うべき課題について述べる．

8.1. モデルの見直し

まず，モデルの見直しを行う必要がある．今回は類似モデルで結果を得られていた研究があったことで，モデルには原因がないと思い込んでおり，別モデルの利用に移行するのが遅れてしまった．しかし，実際にモデルを変えた場合に出力結果を得ることができたので，まずはモデルの見直しから行うのが良いと考えられる．

8.2. データの改善

今回，データに大きな欠損や外れ値は認められなかったが，ラベルの種類数が多いことを加味するとデータ数はまだ不十分であると考えられる．指文字は左右対称ではないため左右反転などでデータ数を増やすことは難しいが，動画自体を増やすことや同一動画でも再生速度を変更するなどしてデータ数を増やすことが望ましい．

また，中心座標を肩幅から決めていたが，指文字は体のどこで行うかが明確に定義されておらず，人によって大きな差が出てしまうため，指文字動作を開始する最初の座標を中心として設定することや，手話のホームポジションのように撮影の際，基準となる位置を指定するなどの対策も有効であると考えられる．

8.3. 別角度からの撮影

MediaPipe の利用により奥行きデータを取得することが可能にはなったが，使用しなかったデータの中には顔や肩と重なる際に特徴点が欠落している場合も見受けられた．そのため，横からなど別の画角から撮影したデータも用意できると，より認識精度が高まると考えられる．

8.4. 学習内容の絞り込み

単語認識について，一文字単位で認識できず単語全体での学習になってしまったが，指文字ごとに切り分けて細分化して学習を行うことや，単語データが十分

に集まれば，ア行限定，カ行限定，50 音限定といった形で絞り込んだ単語で学習を行うことにより，それぞれどの音が認識されやすくどの音が間違いやすいのかを絞り込むことが可能であると考えられる．

8.5. フレームワークやモデルの変更

今回利用したフレームワークは TensorFlow だったが，PyTorch を利用してより高い精度を記録している研究もあるため，PyTorch への移行も考慮する必要があると考えられる．デバッグが容易である点や昨年度の研究分野における PyTorch の使用率は 80%を超えており[15]，情報収集が容易な点からも PyTorch を利用する価値はあると考えられる．同様に，モデルに関しても手話認識において conformer を利用して認識率が上がった事例がある．そのため，モデルの変更も検討の余地があるといえる．

8.6. CTC 不使用の検討

単語全体での学習であれば，CTC を利用せずにブランクや遷移がすべて入ったままでも比較的高精度で単語認識ができる可能性がある．手話と違い手の位置が大きく動かない指文字では CTC 利用による遷移の除去を行わなくとも精度を出せるのか試してみるのもよいと考えられる．

9. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）および電気通信普及財団の助成を受けたものである．

文　献

[1] 木村勉,神田和幸他,"手話認識機能を備えた手話辞書システムの開発",信学技報, vol. 120, no. 419, WIT2020-39, pp. 53-58, 2021]

[2] “CMU-Perceptual-Computing-Lab/openpose”,https://github.com/CMU-Perceptual-Computing-Lab/openpose,2021 年 10 月 21 日　閲覧

[3] “MediaPipe Holistic”, https://google.github.io/mediapipe/solutions/holistic, 2021 年 10 月 21 日閲覧

[4] 磯谷 光,木村 勉,神田 和幸,"ディープ・ラーニングを用いた手話認識に関する研究～手話文からの単語認識　～ ", 信学技報, vol. 120, no.419,WIT2020-38,pp.47-52, 2021].

[5] 小松 周生,白石 優旗,”深層学習を用いたセンサグローブによる指文字認識の検討”,”情報処理学会研究報告”,Vol.2018-AAC-6 No.4,pp.1-5,2018

[6] 池田 聡哉,高橋 正信,”リアルタイム指文字認識システムの研究”,”平成 29 年度電子情報通信学会東京支部学生研究発表会”,講演番号 154,D-11,p154,2018

[7] 横山 和明,森本 正志,”手指形状・動作特徴を用いた LSTM による連続文字 3 クラスインデクシング手法の精度向上”,”情報処理技術者学会第 83 回全国大会”,2 号 pp179-180,2021

[8] 上乃　聖　他，”CTC による文字単位のモデルを併用した Attention による単語単位の End-to-End 音声認識”，”情報処理学会研究報告”，vol.2018-0

[9] “TensorFlow”, “https://www.tensorflow.org”, 2022 年 1 月 18 日閲覧

[10] “MediaPipeHands”,”https://google.github.io/mediapipe/solutions/hands.html”,2021 年 10 月 15 日閲覧

[11] “LSTM ネットワークの概要”, https://qiita.com/KojiOhki/items/89cd7b69a8a6239d67ca,2021 年 10 月 19 日閲覧

[12] “Chainer の特徴”, https://tutorials.chainer.org/ja/#features-of-chainer” ,2021 年 1 月 23 日閲覧

[13] “PyTorch”,https://pytorch.org/,2022 年 1 月 15 日閲覧

[14] “AI-SCOLAR Transfomer × CNN=Conformer” https://ai-scholar.tech/articles/voice-recognition/transformer,2022 年 2 月 20 日閲覧

[15] “Papers with code-Trends” https://paperswithcode.com/trends，2022 年 2 月 20 日閲覧

手話表現中における読話認識に関する研究

豊田工業高等専門学校　大石 啓登，木村　勉

あらまし　本研究では顔の特徴点を検出し，深層学習を用いることで口形から母音読唇を行うことで手話における同形異義語の認識を可能とすることを目的とする．本研究では音声認識の手法である Connectionist Temporal Classification (CTC)に着目し，母音読唇に取り入れることで問題の解決を試みる．昨年度までの研究とは異なる深層学習フレームワーク TensorFlow を使用し，CTC の実装を行い，認識を行った．

キーワード　機械読唇，手話，CTC，GRU

1. 研究背景

豊田高専木村研究室では，これまで深層学習による手話の認識を行ってきた．現在では，手話技能検定試験 6 級に指定されている単語に対して，約 97.4%の認識精度を持つ学習ネットワークの開発に成功した[1]．

しかし，未だに正確に認識ができない単語がある原因として，同じ形で別の意味を持つ「同形異義語」の区別ができないという問題が存在する（図 1）．

現在の手話単語認識システムでは，手話翻訳に「手型」「位置」「腕の動き」という手指動作のみを情報として使用しているが，「同形異義語」の認識には「口の動き」「頷き」「眉上げ」といった非手指動作を考慮する必要がある．

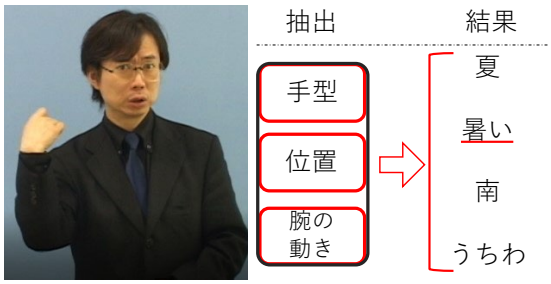


図 1　現状の手話単語翻訳結果

2. 目的

本研究では非手指動作である「口の動き」から母音を抽出し，「同形異義語」の判別を行う．基本的な方針として機械読唇と音声認識の手法に従って母音読唇ネットワークの構築を行った．

まず動画から唇の特徴点を検出するために，Google 社が開発した機械学習向けフレームワークである MediaPipe[2]の FaceMesh を使用する．FaceMesh の使用イメージと抽出できる特徴点を図 2 に示す．

機械読唇の手法は動画を学習させるケースがほとんどである．一方，木村研究室で研究している手話辞書システムや手話翻訳システムでは，手話認識は身体の関節点などから行っている．本研究でも同様に口の特徴点のみでの母音抽出を試みる．またこれらのシステムは，現在は動画を送信しサーバー側で解析を行っているが，将来的に利用者の端末で関節点を抽出して解析することを想定しており，本研究でも特徴点を抽出し，解析する方法で進める．

そして，時系列データを扱うために RNN(Recurrent Neural Network)の一種である GRU(Gated Recurrent Unit)を使用する．

さらに，音声認識や手話文認識の様に読唇においてもどの音素でも無いブランクや音素と音素のつながりによる連音が発生する．これを解決するために音声認識分野において主要な手法である CTC(Connectionist Temporal Classification)を使用した解決を行う．

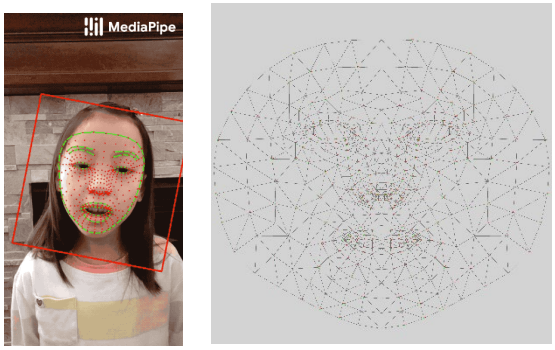


図 2：FaceMesh の使用イメージ [1](左)と
抽出できる座標点 [2](右)

3. 関連研究

機械読唇の研究は日本では 1960 年代から始まった．機械読唇の手法として，オプティカルフローを用いた手法[3]が提案され，その後それらに加え口形変化コード[4]が提案されたことで機械読唇技術が体系化されていった．近年では口唇を中心とした観測領域を切り出し，3DCNN(Three Dimensional Convolutional Neural

[1] Google, “Face Mesh – mediapipe”, https://google.github.io/mediapipe/solutions/face_mesh.html, 2022 年 2 月 18 日閲覧

[2] Google, “GitHub – google/mediapipe”, https://github.com/google/mediapipe, 2022 年 2 月 18 閲覧

Network)[5]や GRU[6]を用いて学習する方法が日本では主流になっていった.

また海外では，LipNet[7]と呼ばれる画期的なネットワークが 2016 年に登場し注目を集めた．これらの手法と本研究で使用した音声認識ネットワークである DeepSpeech2[8]について本章では述べる．GRU の手法は LipNet の説明とともに述べる.

3.1. オプティカルフローを用いた読唇

オプティカルフローは物体やカメラの移動によって生じる隣接フレーム間の物体の動きの見え方のパターンであり，これを用いた機械読唇の手法が，1989 年に間瀬ら[3]によって提案された.

手法として発生中の口唇画像のオプティカルフローをグラディエント法によって抽出し，唇が動いた時点での特徴ベクトルを計算している．この方法によって口唇周りのオプティカルフローによる特徴抽出からから筋肉の動作による変形が認識できるようになり，英単語の認識結果として被験者の一人に対し 93%の成果を上げている.

3.2. 口形変化コード

口形変化コードを使用した手法は宮崎ら[4]によって提案された手法であり，基本口形「あ」「い」「う」「え」「お」「ん（閉唇)」と定義し，日本語発生時の初期に形成される口形を“初口形”，発生した音の母音に相当する口形を“終口形”と呼び，初口形と終口形の各口形に対するコードを“口形コード”として提案している．口形コードの一部を表 1 に示す.

表 1 に示した口形変化コードを次に示す口形変化コード生成規則に適応させることで五十音を判別し，発話時の口形の動きを体系化し,読唇技術にも貢献した.

口径変化コードの生成規則は，次のようになっている.

まず，初口形 C_F と終口形 C_L と口形節番号を以下のように定義する．また「っ」のような促音や「ん」のような撥音は確定することができないため，一時的に「*」で示す.

$$C_F = \{ i, u, x \}$$
$$C_L = \{A, I, U, E, O\}$$
$$s = 1, 2, 3, \ldots$$

口形変化生成規則は 6 つ存在し，前提として s>1 であるとする. 1 つ目は CL(s)=CL(s-1)かつ CF(s)=ϕ である場合, CL(s)は CL(s-1)に吸収される. 2 つ目は, CF(s)≒CL(s-1)である場合，CF(s)は CL(s-1)に吸収され，CF(s)=ϕ となる（≒は口形コードに対する口形が等しいことであり，ϕ はその口形節の音が単口形音であるため，初口形が存在しないこと). 3 つ目は，CL(s)=*かつ CF(s+1)=X である場合，CL(s)=X となり，CF(s+1)は CL(s)に吸収され, CF(s+1)=ϕ となる. 4 つ目は, CL(s)=*かつ CL(s-1)=A または CL(s-1)=E である場合，CL(s)=I となる. 5 つ目は, CL(s)=*かつ CL(s-1)=O である場合，CL(s)=U となる．6 つ目は CL(s)=*かつ CL(s-1)=I または CL(s-1)=U である場合，CL(s)は CL(s-1)に吸収される．また，4 番目から 6 番目の規則は 3 番目の規則と同時に発生する場合があるが，その場合には 3 番目の規則が優先される.

例えば,「明かり（あかり)」という単語に対する口形コードは「AAI」であり, 口形変化コード生成規則を適応すると「AI」となる. このことから,「あ」から「か」に移る発音の際は口形が変化していないことが分かる.

表 1：口形変化コードの一部

※「x」は閉唇

	ア列	イ列	ウ列	エ列	オ列
ア行	A	I	U	E	O
カ行	A	I	U	E	O
サ行	iA	I	U	iE	uO
タ行	iA	I	U	iE	uO
ナ行	iA	I	U	iE	uO
ハ行	A	I	U	E	O
マ行	xA	xI	xU	xE	xO
ヤ行	iA		U		uO
ラ行	iA	I	U	iE	uO
ワ行	uA				uO

3.3. 3DCNN

3DCNN とは，空間情報（2D）と時間情報（1D）をまとめて入力する CNN ネットワークの一種である.

中村ら[5]は，口唇周辺の関心領域（以下 ROI : Region of Interest）を切り出した画像に対し，オプティカルフローを可視化したフロー画像を入力する読唇手法を行っている. また, 3DCNN を用いた学習ではフレームに対してラベル付けを行った後，任意の連続した F フレームを取り出し，その中央フレームを正解ラベルとして与えている.

中村らはこの研究において, 2DCNN やカラー画像のみを使用した場合，ファインチューニングを行った場合というような様々な手法で学習を行っており，3DCNN とカラー画像とフロー画像の両方を使用し, それぞれの画像データの CNN の出力値を統合する late fusion を用いた場合が最も正解率が高くなることが分かった.

この手法では，ALS（筋萎縮性側索硬化症）などの神経難病患者において，77.0%の精度を記録しており有効性が分かっている.

3.4. LipNet

LipNet は 2016 年に Yannis ら[7]によって提案された画期的な読唇ネットワークであり，今でも注目を集め

ている. LipNet は STCNN(Spatiotemporal-CNN)と GRU, CTC を組み合わせた End to End の機械読唇ネットワーク（図 3）である．動画からフレームごとに口唇周辺の ROI を切り取って, CNN と双方向 GRU に入力する．これによって，口形や舌の動きといった読唇に必要な諸要素を学習している．このネットワークは英語の文章読唇において, 単語の誤り率である WER（Word Error Rate : 5.3.2 で詳述）が 6.4%であり，非常に高性能と言える．

しかし, 日本語に対しての LipNet の WER は 61.1%[9]であり，原因として 2 つのことが考えられる．

1 つ目は，英語においては無償で大規模な文章ベースの発話データベースである GRID コーパス[10]が公開されていることに対し，日本語で同じようなデータベースは存在しない．

2 つ目は，英語の母音数は 20 個に対して日本語の母音数は 5 個（図 4）であり，そのうち「あ」と「え」に特徴的な相違がみられず,「う」と「お」が相似で大きさのみにしか相違がないため[11]，コンピュータにとっても判断が難しいことが言えるからである．

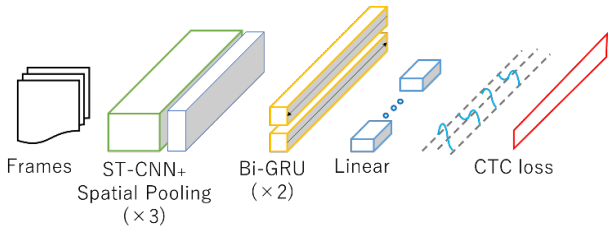


図 3：LipNet アーキテクチャ

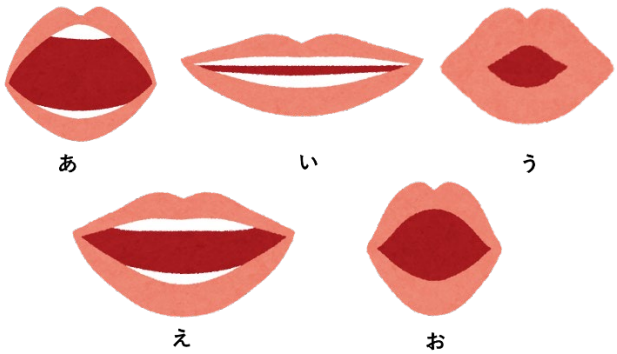


図 4： 5 母音の口形 [3]

3.5. DeepSpeech2

DeepSpeech2[8]のアーキテクチャを図 5 に示す．DeepSpeech2 は CRNN を用いた音声認識ネットワークであり，入力音声をメルスペクトログラム変換した後に CNN と GRU を組み合わせた CRNN を使用して学習し，CTC を用いてテキスト出力を行っている．

また，最初に入力する特徴量だけでなく，ネットワークの各レイヤーに対しても，入力する特徴量をミニバッチごとの統計量を使用して行う Batch Normalization と呼ばれる手法を用いていることも特徴といえる．

このネットワークは英語と北京語という性質が全く異なる言語に対して，それぞれ 9.52%と 5.81%の WER を記録している．

DeepSpeech2 は異なる性質の言語に対しても高い精度を出せる汎用的なネットワークであることや推論が早いことを特徴としており，音声認識において高い性能を示している．本研究ではこのような理由から，DeepSpeech2 ネットワークをベースとして読唇ネットワークの構築を行う．

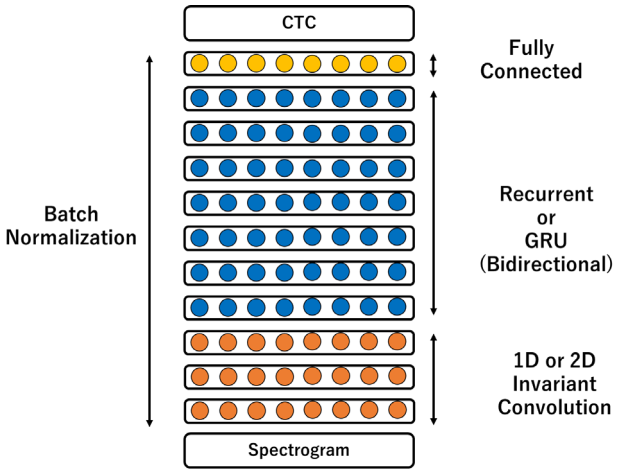


図 5：DeepSpeech2 アーキテクチャ

4. 研究手法

研究で使用した手法について述べる．本研究は機械読唇の手法を基本として, ネットワークの構築を行う．しかし，本研究の目的はあくまで「同形異義語」を区別することであるため，子音の認識を行う必要がないことや，画像ではなく特徴点の時間変化を利用しているため音声認識の手法に近しいという 2 つの理由から, DeepSpeech2 のネットワークを使用する．

4.1. MediaPipe

MediaPipe は Google 社が開発した機械学習向けのフレームワークであり，TensorFlow により実装されている. 本研究で使用する顔の特徴点を取得する Face Mesh という機能を使用するが，その他に身体の関節点を検出できる Pose, 視線を検出できる Iris などが存在する．

このフレームワークは昨年度まで使用していた OpenPose[12]や PoseNet[13]に比べて認識速度が速く，リアルタイムでの特徴点の抽出も正確に行うことがで

[3] いらすとや, “日本語の母音を発音する口の形のイラスト”, https://www.irasutoya.com/2016/01/blog-post_70.html, 2022 年 2 月 18 日閲覧

きる．また，機械読唇分野で主に用いられている dlib[14]の shape predictor に比べて特徴点の数が多いことも利点と言える．

そして，CPU のみで動作が可能なためスマートフォン上でも動作させることができることから，手話の辞書システムや手話のリアルタイム翻訳にも実用が可能だと考えられるので使用する．

4.2. GRU

GRU は RNN の一種であり，長時間の記憶を学習できる LSTM (Long Short Term Memory)のゲート機構を簡略化することで計算効率を上昇させたネットワークである．ゲート機構を簡略化した欠点として LSTM より表現力が低いとされているが，一般的には変わらないと言われている．

また，実際に手話認識において GRU と LSTM の精度が，ほとんど変わらないことが明らかになっており[15]，表現力が落ちたことで特に問題が発生しないと考え，本研究では計算速度が速い GRU を使用する．

4.3. CTC

CTC は損失関数として解析後の出力データに対する損失値の計算に使用されている．音声認識の深層学習では，音素の間に入るとどの音でも無いブランク表現に対して本手法でアプローチを行っている．

例えば CTC による「あいう」の認識を図 6 に示す．同じ意味を持つデータでも，「ああ_い_う」と「あ_い_う_う」と実際には音素やブランクによる差が生じる．CTC はデータに対して，ブランク(“_”)の挿入と音素の連続を表現する．その後，1)同じ音素が連続した場合には一つの音素にする．2)ブランクを削除するという処理を当てはめることで「あいう」という単語を認識することができる．

ブランクの判定には，音の入力がないこと以外にも発話前や対話破綻後の音の入力がない状態，あいまいな音素に対しても判定される．この処理を行うことにより，音素と音素のつながりで発生する連音による誤認識の発生を防ぐ効果がある．

また，学習済みネットワークから単語を取り出すにはデコーダを使用する必要があり，デコーダは文字に対する確率分布を文字列に変換する．CTC ベースのネットワークで採用されるデコーダには,「貪欲デコーダ」と「言語ネットワークの再スコアリングを行うビームサーチデコーダ」（以下，ビームサーチデコーダ）の 2 種類が存在している[16].

「貪欲デコーダ」は各タイムステップの中で最も確率の高い文字を出力する．このデコーダは非常に高速で元の発音に近い文字列を生成することができるが，小さなスペルミスが多く発生する可能性がある．これが原因で，WER の性質上スペルミスによって評価が下がってしまう可能性がデメリットとして存在している．

対して「ビームサーチデコーダ」は一度に多くの複合化をチェックし，与えられた言語ネットワークに従ってより可能性の高い N-gram に高いスコアを割り当てることができる．これはスペルミスの修正という点で非常に役立つが，貪欲なデコーダに比べて処理速度が大幅に低下する．

本研究では最終的に候補の単語との類似度を，レーベンシュタイン距離（5.3.3 に詳述）を用いて計算することで同形異義語を判別し，スペルミスが大きく認識に作用することはないと考え，実行速度を早くするために貪欲デコーダを使用している．

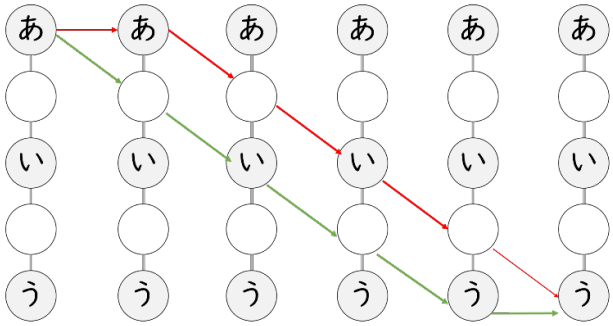


図 6：CTC による「あいう」の認識

5. 研究内容

本研究の一連の流れと各手法の詳細について述べる．

5.1. MediaPipe を用いたデータセットの作成

MediaPipe を用いてデータセットの作成を行う．MediaPipe の Face Mesh を用いて，唇の特徴点計 37 個の発話中の推移を csv データとして保存する．これを 1 単語につき約 45 個，計 2,088 この発話データに対して行った．このうちの約 95%の 1,983 個を訓練データ，残りの約 5%である 105 個をテストデータとして使用する．

また，本来であればこの際に手話表現と発話の両方を行ったデータを使用すべきであるが，データ数が不足しており，日常的に手話を使用している人のデータを収集することが困難であった．そこで検証のみにおいて手話表現と発話のデータを使用し，発話のみを行ったデータを学習データとして使用している．検証に使用した手話表現と発話の両方を行ったデータは，1 単語につき 25 個，計 100 個を用意し，これに対しても同様の処理を行った．

また，GRU に学習させるための処理として鼻を中心点，最大値を肩幅の半分とし全ての特徴点を 0 から 1 の間にスケーリングする正規化を行う．

そしてこのデータに対してアノテーションを行う．本研究では「あ」「い」「う」「え」「お」の五母音と閉

唇である「ん」のラベルを用意する．その後，深層学習においてフレームワークがラベルを理解できる形にするため数値を表 2 のように割り振る．

表 2：各母音と割り振った数値

1	2	3	4	5	6
あ	い	う	え	お	ん

その後，このラベルを使用して各単語に対してアノテーションを行う．本研究では五母音と閉唇の認識を行えればよいので，例えば「暖かい」という単語に対しては「1 1 1 1 2」という形でアノテーションされる．

5.2. 母音読唇ネットワークの構築

母音読唇のネットワーク構築について，GRU のみのネットワークと GRU と 2DCNN を組み合わせた CRNN の構築を行い比較する．使用言語は Python であり，深層学習フレームワークとして TensorFlow を使用する．また，どちらもエポック数は 750，ドロップアウトは 0.45，学習率は e^{-5} としており，最適化関数として Adam を使用した．

5.2.1. GRU を用いたネットワーク

最初に GRU のみでのニューラルネットワークの構築を行う．入力データは，5.1 で述べたデータのうち約 95%（1,983 単語）を訓練データ，残りの 5%（105 単語）をテストデータとして利用する．この時のアーキテクチャを図 7 に示す．

また，CTC は損失関数として利用する．

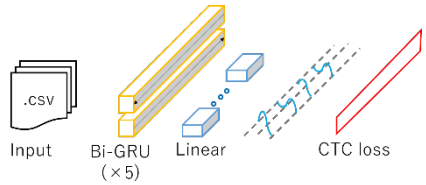


図 7：読唇ネットワークアーキテクチャ(GRU)

5.2.2. CRNN を用いたネットワーク

次に GRU と 2DCNN を組み合わせた CRNN による学習を行う．CNN を使用することで，フレームにおける特徴がより正確に捉えることができるようになる．また，このネットワークは英語の音声認識において，単語の誤り率である WER が 7.89%を記録している DeepSpeech2 で考案されたネットワークを使用している．このネットワークにおいても損失関数として CTC を使用している．この時のアーキテクチャを図 8 に示す．

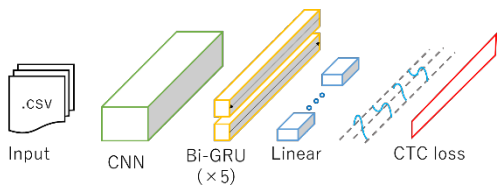


図 8：読唇ネットワークアーキテクチャ(CRNN)

5.3. 評価

5.2.1 と 5.2.2 で構築したネットワークによって学習を行い，それぞれの性能に対して評価を行う．

評価方法として，単語の誤り率である WER と実際に同形異義語を判別する際は，ネットワークの出力と同形異義語の各ラベルに対して類似度を計算して答えを出力するため，レーベンシュタイン距離の平均値も評価基準として使用する．

初めに手話表現をせずに発話のみを行った動画に対しての評価を行い，GRU ネットワークと CRNN ネットワークの比較する．

次に同形異義語の手話表現と発話の両方を行った動画に対して，5.2.1，5.2.2 のうち高い精度を記録したネットワークでの母音読唇を行い，手話表現を行った動画の場合も構築したネットワークが母音読唇をすること可能かどうかを検証する．

評価に使った指標については以下に述べる．

5.3.1. 精度計測における操作

単語や文字の誤りの計算方法として，ある文字列を正解の文字列に変換するまでに，何度「挿入」「削除」「置き換え」の処理を行ったかの回数である「編集距離」が利用される．「編集距離」は各操作の組み合わせのうち最も操作回数が少ない最小距離が適用される．

5.3.2. WER（Word Error Rate）

WER は単語の誤り率で，文章単位での音声認識や読唇で最も多く用いられている指標である．WER は以下の式で計算される．

$$WER = \frac{挿入誤り単語数 + 置換誤り単語数 + 削除誤り単語数}{正解単語数}$$

WER は，ネットワークの出力結果が答えと完全一致していなければ間違いとされ，テストを行った回数に対して不正解だった割合が WER となる．

また河原ら[17]の報告によれば，人間が意味を理解できるレベルは 75%以上，議会の議事録レベルが 85%以上，アナウンサーの原稿読み上げレベルが 95%以上であるため，本ネットワークにおいても 75%以上の WER を目指す．

5.3.3. CER（Character Error Rate）

CER は文字の誤り率で，単語単位での音声認識や読唇で最も多く用いられている指標である．CER は以下の式で示される．

$$CER = \frac{挿入誤り語数 + 置換誤り語数 + 削除誤り単語数}{正解語数}$$

5.3.4. レーベンシュタイン距離

レーベンシュタイン距離は，「挿入」「置換」「削除」のそれぞれにコストを割り当て，各操作のコストと回

数の積を足したものである．今回はレーベンシュタイン距離の計算に python-Levenshtein[18]を使用し，実際のレーベンシュタイン距離の値として利用するのは，以下の数式で表される，正規化した上で，0 が完全不一致，100 が完全一致となるように指標を揃えたものである．Levenshtein は，実際に用いるレーベンシュタイン距離，Insort は挿入回数，Distance は編集距離，ICost は挿入コスト，Replace は置き換え回数，RCost は置き換えコスト，Delete は削除回数，DCost は削除コスト，LongerLength は 2 つの文字列のうち，長いほうの文字列の長さである．これは python-Levenshtein では ratio 関数として実装されており，ratio 関数では挿入コストと削除コストは 1，置き換えコストは 2 として計算されている．

$$Levenshtein = (1 - Cost) \times 100,$$

$$Cost = \frac{Insort \times ICost + Replace \times RCost + Delete \times DCost}{LongerLength}$$

例えば，「あさくら」を「さくら」に変換する時のレーベンシュタイン距離は，すべての操作のコストが 1 であれば「あ」を 1 度削除すればよいので編集距離は 1，使用する文字列の長さは 4 となり，レーベンシュタイン距離は 0.75 となる．

6. 研究結果

各ネットワークの評価結果を表 2，損失曲線を図 9 に示す．

表 2：各ネットワークの性能評価

ネットワーク	WER [%]	CER [%]	レーベンシュタイン距離平均[%]
GRU	23.8	11.6	92.7
CRNN (CNN+GRU)	20.0	11.0	92.3

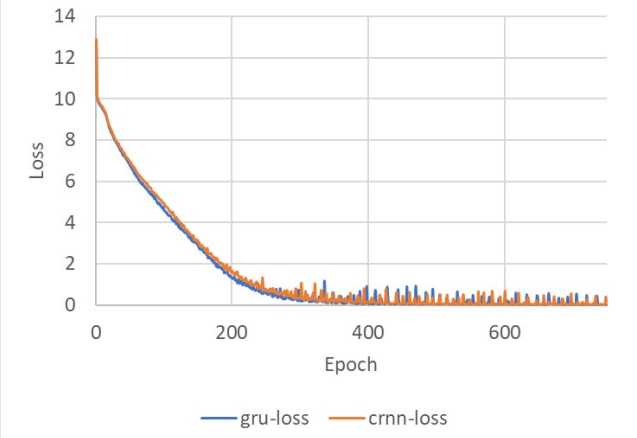


図 9：GRU ネットワークと CRNN ネットワークの比較

6.1. GRU ネットワークの評価

GRU ネットワークを使用した学習における損失曲線（gru-loss）は，図 9 の青線のようになった．学習が進むにつれて CTC 損失値は 0 に近い値に収束しており，最終的な CTC 損失値は 0.003 となったため，学習が成功していることが確認できた．

表 2 に示した結果より，母音読唇において 76.2%の精度を記録した．ネットワークが間違えた単語の一例を表 3 に示す．

表 3 より 3 で挙げた，「あ」と「え」、「う」と「お」のような似た音素の判別と，連続して同じ音素が続く語の認識が苦手であることが分かった．また，母音のみを検出する欠点として連音において「め」(表 1 より口形コード「xE」) のような閉唇を挟む音素の場合誤認識を起こしやすいことが分かった．同じ音や似た音が連続した場合の認識が苦手である原因としては，このような発話を行う時は口形の変化はなく，音素と音素の間にブランクが存在しないため，CTC の連続した音を一つにまとめる，という処理を行う時に実際は連続した音素であっても一つの音素として扱われてしまうからである．

表 3: GRU ネットワークのエラーリスト

※「x」は閉唇

Predict	Answer
aaaai	Aaii
Ai	ui
Exai	eai
Ui	Uui
Au	Aui

6.2. CRNN ネットワークの評価

CRNN ネットワークを使用した学習における損失曲線（crnn-loss）を図 9 の橙線に示す．CRNN ネットワークにおいても GRU ネットワークと同様に学習が進むにつれて，CTC 損失値は 0 に近い値に収束しており，最終的な CTC 損失値は 0.136 になったことから学習が成功していることが確認できた．

表 2 に示した結果より，CRNN ネットワークは GRU ネットワークと比較して，単語の誤り率である WER は 3.8%ほど向上しているが，CER やレーベンシュタイン距離平均ではほとんど変わらない値を示していることが分かる．しかし WER が GRU ネットワークと比較して低いことは言えるので，同形異義語の判別には CRNN ネットワークを使用することにした．

CRNN においても間違えた単語の傾向は，GRU ネットワークと同様であった．

6.3. CRNN ネットワークを使用した同形異義語認識

手話表現と発話の両方を行った手話動画（1 単語につき 25 個，計 100 個）に対して，MediaPipe によって特徴点を csv データにした後に本ネットワークによる認識を行った．その結果を表 4 に示す．

表 4 より，本ネットワークは手話表現を行った動画に適応することが，不可能であることが分かった．

また表 5 より，出力が得られないか，2 文字程度の回答しか得られていないことから，入力したデータが意味のない情報としてブランクとして取り扱われている可能性が高いと考えられる．

表 4：CRNN ネットワークによる手話動画への適用結果

WER [%]	CER [%]	レーベンシュタイン距離平均[%]
100.0	100.0	32.3

表 5: 同形異義語認識のエラーリスト

※「x」は閉唇

Predict	Answer
Aaaai	U
Ao	Uu
Ao	(出力なし)
Au	Uu
Au	u
Uuii	(出力なし)
Uuii	ui

7. 考察

7.1. GRU ネットワークと CRNN ネットワークの性能比較

図 9 の GRU ネットワークと CRNN ネットワークの損失曲線を比較する．

手話認識システムの際に実際に使用する CRNN ネットワークは，GRU ネットワークと比較して CER とレーベンシュタイン距離平均はほとんど変わらず，WER については，3.8%だけ CRNN ネットワークが下回る結果となった．しかし，最終的な CTC 損失値と図 11 のグラフから CRNN ネットワークは最終エポックにおいて軽微な過学習を起こしており，損失値が GRU ネットワークよりも高くなった可能性がある．

しかし，以上の結果を考慮すると CTC 損失はほとんど変わらないということも言えるため，実際の GRU ネットワークと CRNN ネットワークの性能は変わらない可能性がある．よって実用を考慮すると，CNN がないことによってデータ容量が小さな GRU ネットワークを使うことも考える必要がある．

7.2. 発話のみ動画へ適用時の認識率低下の原因と解決策

作成した CRNN ネットワークにおいて WER が 20%になった原因として，3 節で述べたように「あ」と「え」，「う」と「お」といった特徴的な違いを見分けることが困難な母音の判別が難しいとこと，また同じ音や似た音が連続した場合，口形の変化が少ないため音素の数を間違えたり，「あまい」の「ま」の音のように，閉唇を挟んで同じ音や似た音を発したり場合に「ん」の音を認識してしまうことが考えられる．

この問題の解決策として 2 つのことが考えられる．

1 つ目は「動画データで学習を行うこと」により，情報や凹凸といった口形以外の特徴を考慮することで「あ」と「え」，「う」と「お」というような似た音の認識の改善を図り，舌の動きをオプティカルフローで可視化したフロー画像を利用して学習させることで連続した音の認識が可能だと考えられる．

2 つ目は「学習データを増やす」ことであり，本研究に用いたデータは手話表現において同形異義語が存在する単語のみで学習を行ったが，母音抽出を行うという目的に限れば同形異義語が存在する単語のみにデータを限定する必要はない．そのため，現状は 1,983 個を学習データに使用しているが，単語の種類とデータを増やし，GRID コーパスのようにより多くのデータを用意することができれば，さらに正確な学習が可能ではないかと考える．

しかし，1 つ目の方法は「口の特徴点のみで母音抽出を行う」という本研究の目的に反しており，現実的な対策としては 2 つ目の「学習データを増やす」ことが現実的な解決策と言うことができる．

7.3. 手話動画へネットワークの適用ができなかった原因

手話と発話の両方を行った動画に対して読唇が全くできなかった原因として，「①手話を交えない発話」と「②手話を交えた発話」では特徴が大きく異なる点が二つ考えられる．

一つ目は，①と比較して②の発話時間が倍以上になる傾向があり手話を交える場合は，ゆっくりと発話していることが分かる．そのため，②のデータに対しても母音抽出を試みた際に発話がゆっくり過ぎることで，ブランクと認識されてしまうことが考えられる．

2 つ目は②の場合，手話には顔を傾ける動作や頷く動作が存在し，特徴点同士の位置関係が変わってしまったため認識ができなかったという可能性がある．この解決策として 3 つの方法が考えられる．

1 つ目は，現状では CTC を使い音素ベースでの読唇を行っているが，単語ベースでの学習にすることである．

2 つ目は，同じ音素でもいくつかのパターンで異なるラベル付けを行うことである．

3 つ目は，MediaPipe の 3 次元座標を取得できる特徴を生かし，高さ方向と奥行方向の傾きを使用して回転させることで，口を真横にするような正規化をすることである．

1 つ目の方法は単語ベースでの学習を行うと，手話辞書システムの対応する単語が増えるごとに学習をやり直さなければいけないというデメリットが存在する．

また，2 つ目の方法は音素でのパターン分けは体系的にパターンを分けられる保証がないほか，学習データに存在しないパターンに対しては有効でないので，どちらの方法も検討する必要がある．

そして 3 つ目の方法は，MediaPipe の奥行情報の取得機能はまだ発展途上の段階であり，撮影を単眼カメラで行うことから情報の正確性を担保できない可能性がある．

1 つ目の方法が最も確実性があるが，2 つ目と 3 つ目の方法は，手話辞書システムのデータが更新されても柔軟に対応することが可能だと考えられる．

8. 今後の課題

今後，検討して実装していくべきことについて述べる．

1) 手話の動きを考慮した読唇ネットワークの開発

今回の研究では通常の発話に対する母音読唇においては成果を得られたが，手話の動きと発話の両方を行ったデータに対しては，成果を全く得ることができなかった．

よって 7 節で述べたようにデータを増やす，手話動作を考慮したアノテーションを行うことが必要となる．

2) 唇以外の非手指動作への対応

日本手話には「手指動作」の他に眉上げや頷きといった「非手指動作」が存在し，これも手話の認識には重要な要素と言える．本研究では「口の動き」のみから「同形異義語」の判別を試みたが，「頷き」や「眉上げ」といった他の非手指動作も重要な要素であるため，これらも考慮した手話認識を行っていく必要がある．

9. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）および電気通信普及財団の助成を受けたものである．

文　献

[1] 稲田渉, “Conformer を用いた手話単語認識に関する研究”, 豊田高専卒業論文, pp25-32, 2021

[2] MediaPipe, https://google.github.io/mediapipe/, 2022 年 2 月 10 日閲覧.

[3] 間瀬健二, アレックス ペントランド, “オプティカルフローを用いた読唇”, “ITEJ Technical Report”, vol. 13, No. 44, PP. 7-12, VAI’ 89-8(Sep. 1989), 1989.

[4] 宮崎剛, 中島豊四郎, “日本語発話時における口形変化コード化の提案”,FIT2008(第 7 回情報科学技術フォーラム), RK-002, pp.55-57, 2008.

[5] 中村祐哉，齊藤剛史，伊藤和幸，“3D-CNN を用いた神経難病患者の口形認識に関する研究”，信学技報，vol. 120，no. 198，WIT2020-14，27-32，2020.

[6] 白方達也，齊藤剛史,“表情特徴を用いた読唇” FIT2020(第 19 回情報技術フォーラム)，H-027，139-142，2020.

[7] Yannis M. Assael，et al,“LIPNET: END-TO-END SENTENCE-LEVEL LIPREADING,” arXiv:1611.01599 [cs]，2016.

[8] Dario Amodei，*et al.*，“Deep Speech 2: End-to-End Speech Recognition in English and Mandarin”，arXiv:1512.02595 [cs]，2015

[9] 北原瑠伊, 張力峰, “機械学習を用いた日本語読唇における五十音データセット作成の提案”，産業応用工学会全国大会 2021 講演論文集，2021

[10]“The GRID audiovisual sentence corpus”, http://spandh.dcs.shef.ac.uk/gridcorpus/, 2022 年 2 月 10 日閲覧.

[11]河野淳 他,“母音口形図形の知覚,” 音声言語医学, vol. 36，No. 1，1995.

[12]openpose, https://github.com/CMU-Perceptual-Computing-Lab/openpose, 2022 年 2 月 10 日閲覧

[13]”ポーズ推定 | TensorFlow Lite”, https://www.tensorflow.org/lite/examples/pose_estimation/overview?hl=ja, 2022 年 2 月 10 日閲覧.

[14]“Dlib C++ Library”, http://dlib.net, 2022 年 2 月 10 日閲覧

[15]児玉知也 齊藤剛史,”手話認識に有効な骨格情報の検討,”,” IEICE Technical Report “, SP2017-49, WIT2017-45 (2017-10),2017

[16]“DeepSpeech2 – OpenSeq2seq 0.2 documentation”, https://nvidia.github.io/OpenSeq2Seq/html/speech-recognition/deepspeech2.html, 2022 年 2 月 18 日閲覧.

[17]河原達也, “話し言葉の音声認識の進展―議会の会議録作成から講演・講義の字幕付与へ”, Journal of Multimedia Education Research 2012, Vol.9, No.1, S1−S8, 2012.

[18]“python-Levenshtein”, https://github.com/ztane/python-Levenshtein, 2022 年 2 月 10 日閲覧.

オンライン手話辞書システムの開発

豊田工業高等専門学校　情報工学科　鈴木　勇登，木村　勉

あらまし　木村研究室では手話認識機能を備えた辞書システムを Web アプリケーションとして開発してきた．本研究は辞書システムが抱えている問題を解消し，さまざまな状況で使用することができ，かつ使いやすいシステムにすることを目的としている．サーバー上でフレームワークやサーバーアプリケーション等を用いて開発を行い，システムが抱えている問題を解消していった．

キーワード　手話認識，辞書，オンライン，AWS

1. 目的

木村研究室では手話学習者向けに，手話に対応する日本語を検索することのできる手話辞書システム[1]を開発してきた．この理由として，日本語から手話を調べるという形の辞書は多く存在するのに対し，手話から日本語を調べる辞書は数が少ないためである．このシステムは手話学習システムとしての側面もあり，この形態の辞書システムによって手話学習者の学習意欲の向上が期待できる．

しかし，このシステムは様々な問題を抱えている．これらの問題を解消し，手話辞書システムとしてより多くのユーザーが使えるようにすることを本研究の目的とする．

2. システム概要と問題点

ここでは前年度までに開発された手話辞書システム（以下従来システム）を利用する一連の流れについて述べる．

画面の操作メニューを図 1 と図 2 に，録画中の様子を図 3 に，手話の認識後の画面を図 3 に示す．ユーザーは最初に図 1 にあるカメラの形をしたボタンをクリックすると，図 3 のようにカメラのプレビューを表示させる．その後は操作部分が図 2 のように変化し，左下の録画ボタンをクリックすると録画が開始されるので，ユーザーはプレビューを確認しながら手話を録画する．手話を録画した後，停止ボタンをクリックして録画を終了する．録画を終了すると，録画した手話動画のダウンロードが自動で行われる．その後，図 2 左下にある“YOUR FILE“の部分をクリックすると，ユーザーストレージにあるファイル一覧が表示されるので，送信する手話動画を選択する．

図 2 左下にある“send”をクリックすると，選択されている手話動画がサーバーに送信される．送信された動画はサーバー上にある手話認識エンジンによって認識され，その結果の一覧がユーザーに提示される．ユーザーは提示された単語をクリックすることで，見本動画を再生することができる．ユーザーはその動画を確認して，送信した動画の手話と提示された単語が一致しているか確認することができる．ここまでが，従来システムの一連の動作である．

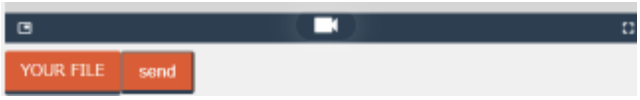


図 1:従来システムの GUI（操作部分）

図 2:従来システムの GUI（ボタン変化後の操作部分）

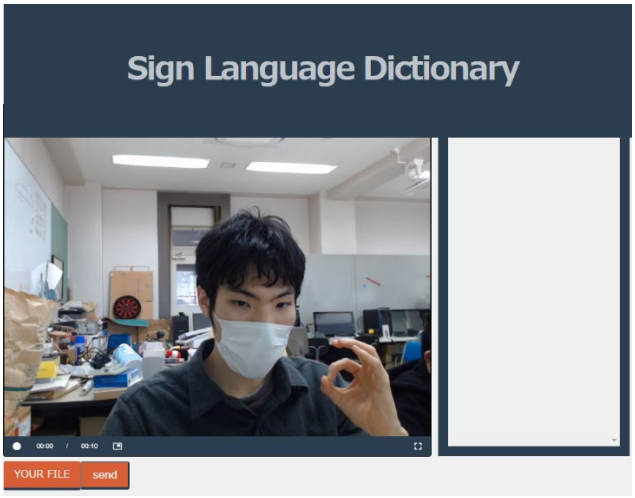


図 3:従来システムの GUI（録画画面）

図 4:従来システムの GUI（認識後）

以下に，従来システムの問題点を列挙する．

2.1. オンラインへの非対応

従来システムの概要図を図 5 に示す．従来システムは図 5 のユーザー，サーバー，ストレージが一体化されている．この構造では，システム全体を所有するユーザーでなければ従来システムを使用することができない．サーバーの機能を動作させるには，膨大な量の見本動画を抱えた上で，デバイスに負荷のかかる認識機能を持つ必要がある．汎用的なシステムという観点から見れば，ユーザーの負担を減らすため，従来システムのサーバー部分をオンライン上で動作させるべきである．しかし，ユーザー部分とサーバー部分が一体化されていることが原因で，オンラインには対応できていない．

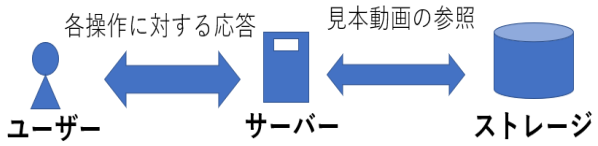


図 5:従来システム概要

2.2. 録画操作による認識率の低下

手話動画を撮影する際，録画開始ボタン，および録画終了ボタンを押すユーザーの操作を必要とする．従来システムの仕様上，操作をした直後に録画が開始されるので，録画された動画に手話前後の操作が混入してしまう．手話動画として不完全な状態で認識を行った場合，認識率が低下すると考えられる．

2.3. 録画した動画の送信方法

録画した動画を送信する際に，録画した手話動画を一度ユーザーのローカルストレージに保存させる．その後，保存した動画をファイル一覧から選択してサーバーへ送信している．この方法ではユーザーのローカルストレージを圧迫する上，録画した動画を送信するまでの手間が増えるので，手話認識の効率が下がると考えられる．

3. システム概要

本システムは次のようなシステムを想定している．

3.1. 外部仕様

本研究では，上述の問題点を踏まえ，下記のような外部仕様を満たす手話辞書システムを開発する．図 6 に外部仕様の概要図を示す．

1） 手話動画の録画（図 6 の 1，2，3 間の部分）

録画開始ボタンをクリックすると，ユーザーのデバイスに接続されているカメラに対して映像取得の許可を要求して手話を録画する．その際，2.2 の問題を解決するために，録画を開始する操作と実際に録画が開始されるまでに数秒のカウントダウンを設ける．ユーザーはカウントダウン中に表示されるカメラのプレビューを確認しながら自分の位置を調整し，カウントダウン終了と共に手話動画の録画を開始する．

録画を終了する際にも 2.2 の問題は起こりうる．それを解決するため，録画前にユーザーが設定した時間が経過した時点で録画を自動終了させる．

2） 手話認識（図 6 の 3，4，5 間の部分）

録画した動画を自動的にサーバーに送信し，動画に対して手話認識が行われる．2.3 の問題を解決するために，撮影した動画のダウンロードはしない．

3） 認識結果の表示（図 6 の 5，6，7 間の部分）

ユーザーに対して手話動画の認識結果を一覧表示する．表示する単語は一致する確率の高い上位 10 語である．ここに表示された単語をクリックすると，単語の手話動画が再生される．

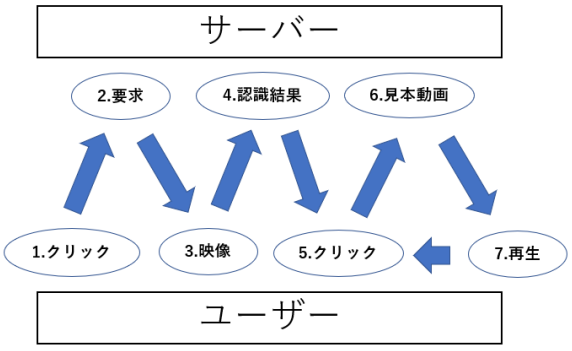


図 6:外部仕様の概要図

3.2. 内部仕様

ここではシステムの内部仕様についての概略を述べる．図 7 に各要素の通信概要図を，図 8 に内部仕様の概要図を示す．

今回開発するシステムは Web 上で動作する Web アプリケーションを想定している．そのため大きく分けて，図 7 のようにクライアントとサーバーの 2 つのサブシステムとそれらを繋ぐ通信から成り立つ．

クライアントでは，手話動画の撮影とサーバーへの送信を行い，サーバーからの認識結果をユーザーに提示する．

サーバーでは，クライアントから送信されてきた手話動画を，手話認識エンジンを用いて認識を行い，その結果をクライアントに返す．

下記に各サブシステムの仕様について述べる．

1） クライアントの仕様

クライアント側は，図 8 のように Web ページとして HTML と JavaScript を用いて実装する．

2） サーバーの仕様

サーバーは AmazonEC2[2]サービスから提供される Linux 搭載のサーバーに Python API である Flask[3]を用いて以下の仕様を満たすように実装する．サーバーが処理を行うのは，図 8 のように動画送信から認識結果

の表示ページへ遷移する部分である．クライアントから送信された手話動画をサーバーに備えた手話認識エンジンによって認識する．手話認識には同研究室で今年度開発された手話認識エンジンを用いる．手話認識エンジンによって選出された候補となる 10 個の単語を見本の手話動画の URL とともにクライアント側に返す．

図 7:各要素の通信

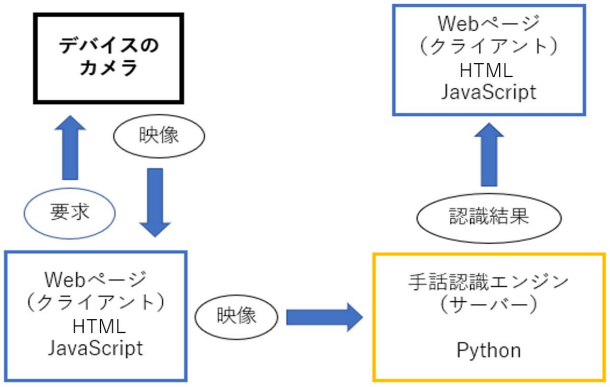


図 8:内部仕様の概要図

3.3. 手話認識エンジン

本システムで使用する手話認識エンジンは，前年度で開発された再帰型ニューラルネットワークを用いた手話認識システムを用いる．手話認識システムはゲート付き回帰型ユニット（Gated Recurrent Unit：GRU）[4]を用いて構築されたニューラルネットワークである．ニューラルネットワークとは，人間の脳の仕組みから着想を得た，脳機能の特性のいくつかをコンピュータ上で表現するために作られた数学モデルである．GRU は再帰型ニューラルネットワークが長期の時系列を学習する際に用いられる手法であり，時系列を持つ手話動画のベクトルを学習させている．

手話認識エンジンは，手話動画を姿勢推定アプリケーションである MediaPipe を使って姿勢推定を行い，動画の縦と横の長さを用いて正規化された動作データを入力する．システムでは手話技能検定試験 6 級 101 単語が学習されており，入力される動作データが，学習済みである手話単語にどれだけ一致しているかを一致率として出力する．この一致率の高い順から，認識結果として提示する．

4. 研究内容

4.1. 開発環境

本研究で用いる開発環境を表 1 と表 2 に示す．サーバーに対してリモート開発環境を用いて開発を行ったので，サーバー側の環境と開発側の環境の両方を示す．

表 1:サーバー側の環境

サーバー	Amazon EC2 t3.large
サーバーの OS	Ubuntu 20.04.3
CPU	CPU : Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz RAM : 8192 MB
姿勢推定	MediaPipe
サーバー用フレームワーク	Flask v2.0.2
Web サーバー	Apache2 v2.4.41
アプリケーションサーバー	mod_wsgi

表 2:開発側の環境

OS	Windows 10
CPU	CPU : AMD Ryzen 5 2600X 3.6GHz RAM : 16GB GPU : NVIDIA GeForce RTX 2060 (6GB)
リモート開発環境	Visual Studio Code v1.63.2

以下，上記の表に表記されているフレームワーク等のサービスについて述べる．

1) Flask

Flask[3]は Python 用のウェブアプリケーションフレームワークであり，最小限の機能だけを所持しているので，小規模の Web アプリケーションを作る事に適するとされている．フレームワークの役割は，HTTP 通信を制御してクライアントとサーバーの間で円滑なデータ交換を行えるようにすることである．

2) Apache2

Apache2[5]は外部に Web サイトを公開するための Web サーバーである．開発したシステムを外部アクセスでも使えるようにするために使用する．主な役割は，クライアントのリクエストに応答し，必要なデータを渡して Web ページ上で表示することである．

3) mod_wsgi

mod_wsgi[6]は Web サーバーと Web アプリケーションを接続するためのインターフェースである．今回は，Apache2 と Flask を接続するために使用する．

4.2. 実装方針

本研究では,3.1 で述べた機能を以下の方針で HTML と JavaScript を用いて実装する.

1) カウントダウン付きの録画機能

録画開始までの待ち時間および録画時間の入力，および録画開始ボタンを表示するページを作成する．録画開始ボタンが押されると，待ち時間だけカウントダウンを行う処理を実装する．この処理は現在時刻を取得し，待ち時間との差を計測することで実現する．

また，カウントダウン中はカメラのプレビューを表示する．これは video タグを用いて実現する．このタグは Web ページ上で動画を再生するためのタグである．

カウントダウンが終了すると，カメラから取得した映像の録画を開始する．録画開始から録画時間として設定された時間が経過すると，映像の録画を終了させて動画の送信へ移行する．

2) 動画の送信

1)によって録画した動画をサーバーへ送信する．その際，2.3 の問題を解決するためには従来のシステムに使われていたファイル送信以外の方法で送信を行う必要がある．録画した動画をユーザーのローカルストレージを介さずに直接かつ自動で送信する機能の実装は，1)で作成された録画データを直接 POST（送信）機能によってサーバーへ送信することで実装する．

3) 認識結果の表示

手話認識エンジンは，認識結果をリストで返す仕様になっている．手話認識エンジンで得られた認識結果のうち，一致率の高い上位 10 語を取り出して，結果表示用の Web ページに渡す．この時，結果の単語とは別に，結果に対応する見本の手話動画のパスも渡す必要がある．よって，見本動画へのパスを認識結果のリストと共に渡す．渡された結果を Web ページ上で表示し，クリックされた単語の見本動画を，video タグを用いて再生する．

5. 研究結果

ここでは，研究結果について述べる．

5.1. 実装結果

図 9 にディレクトリ構造を示す．なお，図 9 について，拡張子の記述がない部分は全てディレクトリである．

ディレクトリ movie には，認識する動画が保存される．また sample には手話の見本動画を格納する．

ローカルデバッグ上で“すずしい”の手話動画を録画し，認識を行った．その様子を図 10，図 11，図 12 に示す．図 10 では，待ち時間と，録画する秒数を入力できる．図 11 はカウントダウンを行い，その間カメラのプレビューを表示する画面となっている.図 12 は認識結果が表示され,単語をクリックした後の画面である．“すずしい”が候補に表示されていることが分かる．図 9 の sample ディレクトリ内にある手話動画へのリンクが認識結果のページに候補単語とともに埋め込まれ，図 12 にある候補単語をクリックすることで再生される.オンライン上で実行するとほぼ同じ挙動となるが，図 12 において,候補単語をクリックしても後述の理由から見本動画の再生ができていない.

なお，待ち時間と録画時間をそれぞれ処理する HTML ファイル（time.html と recorde.html）に送る際，HTML の記述内で js ファイルのパスを指定して埋め込む形の場合だと,うまく数値が読み込めない.そこで，timeget.js と indexx.js は対応する HTML ファイル（time.html，recorde.html）に直接スクリプトとして記述した[7]．そのため，図 9 の timeget.js と indexx.js は実際のシステムでは使用されていない．

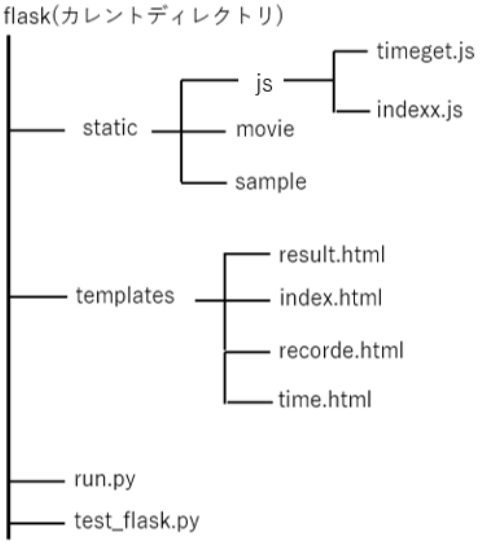


図 9:ディレクトリ構造

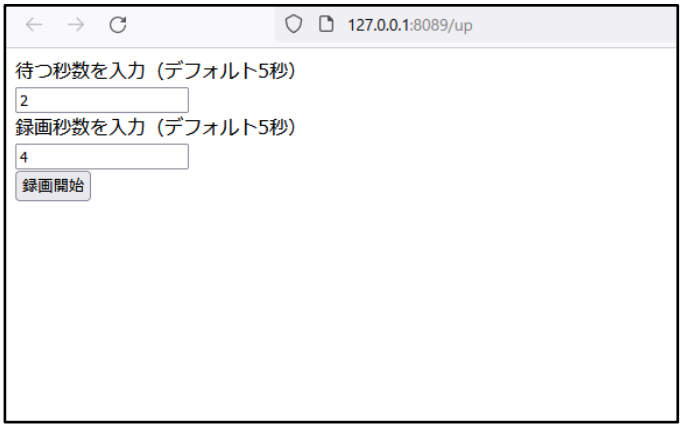


図 10:設定画面

図 11:カウントダウンの画面

図 12:認識結果表示の画面

5.2. 動作の詳細

図 13 に画面遷移の流れを示す．表 3 に作成したプログラムと，その機能について示す．また，表 4 に使用する主な関数を示す．ここではこの関数表を交えて具体的な動作を述べていく．

1) 録画開始画面

図13に示すとおり，最初に表示するドキュメントは，index.htmlである．図10の設定画面において，待ち時間と録画時間をそれぞれ入力して録画開始ボタンをクリックすると，それぞれの値がサーバーに送信される．サーバーは，render_template()[9]の引数に待ち時間と次の処理画面であるtime.htmlを指定してクライアントに対して画面遷移を行う．

2) カウントダウン処理

time.htmlで，待ち時間を引数としてtimeget.jsのCountdownTimer()[10]に与えてカウントダウンを行う．ここで，CountdownTimer()はmsec単位でカウントするので，引数はあらかじめ1000倍しておく．また，time.htmlのページが読み込まれると同時に，Navigator.mediaDevices.getUserMedia()[9]がカメラ映像の取得の許可を要求する．この処理には非同期処理を使用しているので，許可が下りるまではカウントダウン処理が行われないようになっている．許可が下りた場合は，カウントダウンと同時にvideo.play()[9]でページ上にカメラのプレビューを表示する．

CountdownTimer()は，引数に指定された時間と現在時刻が等しくなった時に行われる処理を指定することができるので，カウントダウンが終了すると同時にサーバーに対してページの遷移を要求する．

ページ遷移を要求されたサーバーは，録画時間と共にrender_template()に次の処理画面であるrecorde.htmlを指定してクライアントに対して画面遷移を行う．また，後の工程でファイルをダウンロードする際，ファイルを上書きする形になると処理が一時的に停止してしまうので，movieディレクトリ内の動画と画像を全て削除する．

3) 録画中の処理

recorde.html の処理は time.html と似ている．timeget.js とは別に，indexx.js に CountdownTimer()を定義し，引数には上述の数値に待ち時間ではなく録画時間を使用して録画終了までのカウントダウンを行っている．

recorde.html が読み込まれると，カメラ映像の取得許可を要求し，許可が下りれば MediaRecorder.start()[9]によってカメラ映像の録画が開始される．1/30 秒毎にカメラから取得した画像に canvas.toDataURL[9]で DataURL を付与し，用意した配列に DataURL を追加する．

4) 録画した動画の送信

設定された録画時間が過ぎると，MediaRecorder.stop()[12]によって録画が終了し，配列に格納した DataURL を fetch()[9]によってサーバーへ送信する．この時，サーバー側は受け取った DataURL に対応する画像を download_file()[9]によってサーバーへダウンロードする．ファイル名は imageN.jpg としている（N にはダウンロードした回数が入る）．

非同期処理である DataURL の送信が全て終了すると，location.href()によってサーバーに画面遷移を要求する．サーバーはこの要求を受け取るとダウンロードした連番画像を cv2.VideoWriter()[9]によって動画ファイルにする．

5) 認識結果の表示

作成した動画ファイルは手話認識エンジンに入力され，返ってきた認識結果のリストから一致率の高い，上位 10 個の単語名を抽出する．抽出した単語に見本動

画のパスを追加して，次の画面処理である result.html と共に render_template()の引数として画面遷移を行う．

result.html では，図 14 のように video タグを 10 個用意して，それぞれに”result1”~”result10”（認識結果の上位 10 単語が割り当てられる），onclick（クリックされた時に動作する関数）を設定している．単語がクリックされると，クリックされた場所に対応する id の動画が再生される．（例えば，図 14 の 3 行目の aplay()なら 21 行目の avideo が再生される）

表 3:作成したプログラム

プログラム名	機能
test_flask.py	3.2 の 2)に示すようなサーバー処理を行う．
run.py	ローカルデバッグ用のポートを開く
index.html	録画開始前のページ用のドキュメント
time.html	カウントダウンを行うページ用のドキュメント
recorde.html	録画を行うページ用のドキュメント
result.html	認識結果を表示するページ用のドキュメント
timeget.js	time.html に付随する JavaScript
indexx.js	recorde.html に付随する JavaScript

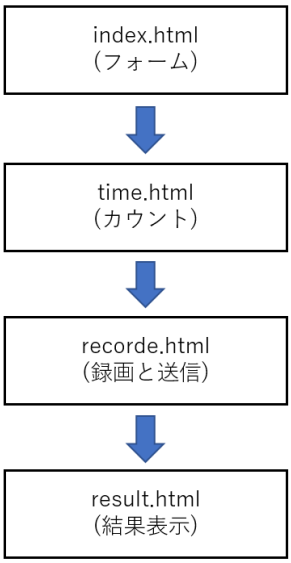


図 13:画面遷移の流れ

表 4:主な関数一覧

関数名	機能
Navigator.mediaDevices.getUserMedia() (JavaScript)	ユーザーのカメラにアクセスする．引数に応じて映像と音声を取得する．
CountdownTimer() (JavaScript)	第二引数で与えた時刻までカウントダウンを行う
video.play() (JavaScript)	カメラのプレビューを html から与えられた video 形式の要素に表示する
MediaRecorder.start() (JavaScript)	カメラ映像の録画を開始する．
MediaRecorder.stop() (JavaScript)	カメラ映像の録画を終了する．
location.href (JavaScript)	サーバーに対して画面遷移を要求する．
canvas.toDataURL() (JavaScript)	画像に対して DataURL を付与する．
fetch() (JavaScript)	引数で指定した URL に対して指定したデータを送信する．
render_template() (Python)	第一引数に指定したドキュメントをクライアントに返す．第二引数以降はドキュメントに付随するデータを渡すことができる．
download_file() (Python)	第一引数の URL からファイルをダウンロードし，第二引数のパスに保存する．
cv2.VideoWriter() (Python)	動画ファイルを作成する

5.3. 実装によって解決した問題

1) オンラインへの非対応

AWS 上でサーバーを立ち上げた．表 1 に示したような環境を導入し，Web サーバーとして動作していることを確認した．サーバーが実行されている状態でかつ apache2 が running 状態であれば，今回，公開用に取得したドメイン名「www.jsldic.net」をブラウザに直接入力することでシステムにアクセスすることができる．これにより，オンラインへの対応が実装され，それに伴ってユーザーへの負荷が減少したと考えることができる．

しかし，ローカル上とオンライン上ではやや挙動が違う点が存在することが判明した．これについては後述する．

2) 録画操作による認識率の低下

このシステムは，index.html, time.html, recorde.html, result.html の順番で画面遷移が行われる．index.html と result.html は図 10 と図 12 のような画面でユーザー

の操作を要求するが，その間の time.html，recorde.html ではユーザーに対して一切の操作を求めないようなシステムになっている．これにより，ユーザーの操作が手話動画に混入しないようになり，認識率の低下を防ぐことができる．

```
<br>
    <p1 style="text-align: right" onclick="aplay()">{{result1}}</p1>
    <p2 style="text-align: right" onclick="bplay()">{{result2}}</p2>
    <p3 style="text-align: right" onclick="cplay()">{{result3}}</p3>
    <p4 style="text-align: right" onclick="dplay()">{{result4}}</p4>
    <p5 style="text-align: right" onclick="eplay()">{{result5}}</p5>
    <p6 style="text-align: right" onclick="fplay()">{{result6}}</p6>
    <p7 style="text-align: right" onclick="gplay()">{{result7}}</p7>
    <p8 style="text-align: right" onclick="hplay()">{{result8}}</p8>
    <p9 style="text-align: right" onclick="iplay()">{{result9}}</p9>
    <p10 style="text-align: right" onclick="jplay()">{{result10}}</p10>
    <video id="avideo" controls width="480" height="360"></video>
    <video id="bvideo" controls width="480" height="360"></video>
    <video id="cvideo" controls width="480" height="360"></video>
    <video id="dvideo" controls width="480" height="360"></video>
    <video id="evideo" controls width="480" height="360"></video>
    <video id="fvideo" controls width="480" height="360"></video>
    <video id="gvideo" controls width="480" height="360"></video>
    <video id="hvideo" controls width="480" height="360"></video>
    <video id="ivideo" controls width="480" height="360"></video>
    <video id="jvideo" controls width="480" height="360"></video>
</br>
```

図 14:結果動画を表示する result.html

3) 録画した動画の送信方法

動画を HTML のファイルインプット機能を使わずに直接送信する機能は，動画データをそのままサーバーに送信することはできなかったものの，画像データを連続で送信して，サーバー側で画像を連結させて動画にすることで間接的に実装することができた．

動画自体を送信できなかった理由としては，録画データに付与できる URL 形式が BlobURL[13]（ブラウザ内部で作成されるオブジェクトへの参照 URL）のみであり，download.file()等のファイルをダウンロードするモジュールに対応していなかったことがあげられる．このような URL からダウンロードするモジュールは複数あり，FFmpeg[14]等の同じようなモジュールも使用したが，BlobURL からデータを取得することはできなかった．その他，URL を付与せずにデータを直接送信する手段も試してみたが，うまくいかなかった．

その一方，画像データはデータに対して付与できる DataURL[15]（データそのものを文字列で表現した URL）がこれらのモジュールに対応していたので送信することができた．最終的に download.file()を使用した理由は，生成される DataURL の文字列が長すぎて FFmpeg では扱えなかったからである．

6. 考察と今後の課題

オンライン手話辞書システムとしての最低限の機能を実装することができた．

しかし，ローカル上と Web 上では挙動が異なっていることが影響を及ぼしている点がある．

表 4 に示した関数のうち，カメラの映像を取得する Navigator.mediaDevices.getUserMedia()は，ローカル上もしくは URL スキームが「https」の通信でないと動作しない．この問題は SSL 証明書を発行して Web サイトに暗号化通信を導入した結果，Web サイトとの通信に URL スキーム「https」を使えるようになったことで解決した．

その他の問題として，見本動画のファイル名がすべて平仮名であることが原因で，ファイル読み込みに対してローカル上では正しく実行されるが，Web 上の場合は 404 エラーを返す．これは現状未解決であるが，見本動画のファイル名を全て半角英数字にし，ひらがなで構成された認識結果のリストに対応する半角英数字で構成されたリストを用意し，そのリストを使用してパスを渡せば解決できると考えている．

これらのローカルとオンライン上での挙動の違いは拡張機能を追加するたびに現れる可能性があるので，その都度対応していく必要がある．

他の観点からの問題として，複数のクライアントによる同時アクセスを検証していないので，単一のクライアントには対応できたとしても，複数のクライアントには対応できない可能性がある．特に懸念されるのは，サーバーに手話動画をダウンロードして保存する際，ファイルが他のクライアントのものと混じってしまい，手話動画が正しく生成されない可能性がある．

7. 終わりに

今回の研究では，手話辞書システムをオンラインに対応させることで様々な状況で使用できるようになった．新たな姿勢推定アプリケーションである MediaPipe を使用した手話認識エンジンを導入することができた．また，手話認識率の低下に繋がるとされる問題を解決することで，手話認識エンジン以外の観点から認識率を上げることができた．

課題を解決すれば，多くの人が使用できる手話辞書システムとして，手話学習をはじめとした様々な用途に貢献できると考えている．

8. 謝辞

本研究は科研費（18K18517：代表者木村勉，18K18518：代表者神田和幸）および電気通信普及財団の助成を受けたものである．

文　献

[1] 木村 勉他，”手話認識機能を備えた手話辞書システムの開発”，信学技報，vol. 120, no. 419, WIT2020-39, pp. 53-58 2021 年.

[2] “AmazonEC2” https://aws.amazon.com/jp/ec2/?nc2=h_ql_prod_fs_ec2，2022 年 1 月 28 日閲覧

[3] “Flask” https://flask.palletsprojects.com/en/2.0.x/，2022 年 1 月 28 日閲覧

[4] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, Yoshua Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling",arXiv:1412.3555, (2014)

[5] "Holistic-mediapipe" , https://google.github.io/mediapipe/solutions/holistic , 2022 年 2 月 21 日閲覧

[6] “Apache” https://httpd.apache.org/ 2022 年 1 月 28 日閲覧

[7] “mod_wsgi” https://modwsgi.readthedocs.io/en/master/ 2022 年 2 月 20 日閲覧

[8] “Python の変数を html(javascript)に渡す” https://christmas-cookies.hatenablog.com/entry/2018/08/06/010709 2022 年 2 月 20 日閲覧

[9] “MDN Web Docs Web API” https://developer.mozilla.org/ja/docs/Web/API/MediaDevices/getUserMedia 2022 年 2 月 21 日閲覧

[10] “副業研究所　[コピペで超簡単]カウントダウン・タイマーを設定表示する方法” https://tryk-magazine.com/200918-html/ 2022 年 2 月 19 日閲覧

[11] “note.nkmk.me Pyhton で Web 上の画像などのファイルをダウンロード” https://note.nkmk.me/python-download-web-images/ 2022 年 2 月 21 日閲覧

[12] OpenCVDocumentation VideoWriter Class Reference https://docs.opencv.org/3.4/dd/d9e/classcv_1_1VideoWriter.html 2022 年 2 月 21 日閲覧

[13] “BLOB URL とはなにか” https://devsway.net/2019/08/11/blob-url%E3%81%A8%E3%81%AF%E3%81%AA%E3%81%AB%E3%81%8B%EF%BC%9F/ 2022 年 2 月 21 日閲覧

[14] “FFmpeg” https://www.ffmpeg.org/ 2022 年 2 月 21 日閲覧

[15] “MDN web Docs　データ URL” https://developer.mozilla.org/ja/docs/Web/HTTP/Basics_of_HTTP/Data_URIs 2022 年 2 月 21 日閲覧

音声データからの自動アノテーション付記法の提案
－手話の機械認識研究の１ステップ－

神田 和幸
国立民族学博物館 〒565-8511 大阪府吹田市千里万博公園 10-1
E-mail: kandak@minpaku.ac.jp

あらまし 機械学習を用いた手話認識の研究において膨大な手話データを深層学習させる段階で，従来の方法では手話動画を見ながら人力で日本語ラベルを与えるアノテーション作業が必要であった・本論では手話動画録画の際に同時に音声を録音し，その音声を機械認識させ，文字化する過程を加えることで時間と労力を節約することを提案する・この手法の正当性を考察するに当たり，手話という言語のバイモダリティという特徴と近年バイリンガル教育などで話題となっているトランスランゲージングという概念を応用している・
キーワード 手話認識，言語交差，バイモーダル，自動アノテーション

A Proposal of Automatic Annotation System with Audio Data
－One Step for Machin Sign Recognition－

KANDA Kazuyuki

‡ National Museum of Ethnology 10-1, Senri Expo Park, Suita, Osaka 565-8511 Japan
E-mail: kandak@minpaku.ac.jp

Abstract In the research for sign recognition though machine learning, annotation by human effort was thought to be necessary process. The new idea is proposed here to annotate automatically signing using the audio data recorded simultaneously in the video, which would reduce the time and huma effort. The idea includes the concept of the peculiarity of sign language as bi-modality and translanguaging in bilingualism.

1. はじめに

手話がどういう言語かという言語観は研究の方向性を決める・本論ではこれまでの手話言語観を改め，近年バイリンガル教育で浸透しつつあるトランスランゲージングの考え方を参考に新たな手話言語観を提言する･手話は古くから言語観が議論の元になっており，聾教育においては口話主義という国語によるモノリンガル教育から，手話を採り入れたバイリンガル教育への変換が行われてきたので，新しいバイリンガル教育の概念との親和性も高い・

本論ではまず手話にとって新しい言語観であるトランスランゲージングについて概説し，次に手話に対する言語観を歴史的に概観し，本論の提案の正当性を確認した上で，新たな手話観に基づいた手話の工学的処理，ことに手話の機械認識の研究手法における処理過程の変更を提案する・この手話に対する概念の変換により従来問題となっていた日本語対応手話と日本手話の扱いや手話データのアノテーションの手間について一定の解決案になることを期待している・

2. translanguaging

トランスランゲージング(以下必要に応じて TL と省略) の基本概念について大山万容（2019）は次のようにまとめて説明している(1)･

・Languaging とは 言語の動的な使用 を表す・
・「トランスランゲージングは，これまで言語に付与されてきた不均衡な政治的･社会的権力に抵抗し変革する 言語実践を指す」（ García & Li 2015, Ch.2）
・「言語」という分類を使う こと自体を拒否する

また猿橋・坂本(2020)(2)では次のように説明する・少々長いがそのまま引用する・

「近年，バイリンガルやマルチリンガルを個別の言語能力の併存と捉えず，包括的な言語レパートリーを言語資源として有する存在と見なすトランスランゲージング（Garcia & Wei, 2014; 加納，2016a, b） の概念が提唱されている・トランスランゲージングは，多言語話者の言語能力について，言語間の境界線を超越した，第一言語（L1）でも第二言語（L2）でもない，ひとつの独自の言語システムから成るものと想定し，話者は必要に応じて自分の持ち備えた言語システム全て

を駆使して言語活動にあたるとする・ひとつの繋がった言語システムの存在を前提とするトランスランゲージングは,L1 と L2 を行き来すると考えるコードスイッチング（Myer-Scotton, 1993）と一線を画す立場を取る・このように，言語の存在論の転換を含むトランスランゲージングという発想は，言語に対する態度や，話者間の関係性にも影響を及ぼし得る・すなわち，母語話者を，言語規範を持つ者と見なすことや，多言語環境で「どっちつかずの言語」や中間言語と見なされてきた言語コードへの見直しを促す・それは，ホスト社会の言語を習得するうえで控えられる傾向にあった移民の継承語使用について，言語資源としての活用を見出そうとする・さらに，言語学習者のエンパワーメントにも繋がり得ることが期待されている・人々が言語レパートリーを駆使し，必要と状況に応じて言語を使用するという見方は，人の移動をめぐる社会言語学的諸課題と密接に関連し，早くも言語指導の場に応用されている・他方で，トランスランゲージングは理論的にも方法論的にもまだ検討すべき面が多くある・トランスランゲージングは,「別言語として社会的に構築された言語特性を，ひとつの言語レパートリーとして使用する」（Garcia & Wei, 2014, p. 2, 筆者試訳）ことを指す・名付けられた言語を閉じられた体系として捉え,2 つの言語体系を行き来することをバイリンガル，3 つ以上の場合をマルチリンガルとする従来の言語観からの脱却を出発点とする概念である・言語名の多くが国や地域の名を冠しているのは，共同体を構成する上で地理的な条件が大きな決定要素となってきたことの反映である・近年のグローバル化は，人や物，情報の大量移動を加速させ，デジタル通信技術の発達は，対人関係と情報伝達における物理的な隔たりを凌駕するかに見える・それに伴い，人々の相互作用や，言語と共同体の結びつけられ方，帰属意識やアイデンティティの形成過程にも変化が生じていると考えられる・そのような変容の只中において，国や地域と言語を分かちがたく結びつけ，閉じられ，安定した言語世界を前提とした従来の言語観では，人々のコミュニケーション実践を的確に捉えることは難しいのではないか・トランスランゲージングは，このような問題意識から提案された概念のひとつと言えよう・複雑かつ活発な可動性を持ち，著しく多様化する社会において，開かれ，動的に交叉することが言語の常態であるとするのが，トランスランゲージングの言語観である・すなわち,トランスランゲージングとは,X 語や Y 語といった言語の範疇にとらわれずに，過去の経験を通して蓄積された言語レパートリーを言語資源として用い，相互作用の中で意味づけを行い，言語資源を更新していくという言語実践である・そこでは，既存の何語にも分類し得ないような表現や用法が生まれ，語義は元来のものとかけ離れることもある・それならば，トランスランゲージングは従来から見られる言語実践とも言えよう・このような言語実践は,主に言語接触,言語変容,言語混淆などの研究領域において注目され，説明されてきた・ただし，そこでは学術的にも社会的にも，X 語と Y 語とそれぞれの言語世界を所与として説明され，変容や混淆は言語の乱れや亜種，創造と見なされてきた・すなわち，単一言語世界が無標で，言語接触によって他の言語の影響を受けることが有標とする見方を基底としていた・トランスランゲージングの考え方は，グローバル化に伴う現象を捉えた他の社会言語学からの提案と軌を一にしている・個々の言語を，閉じられた体系とする言語観の転換を促すという面では，crossing（ Rampton, 1995），polylingualism（ Jørgensen, 2008），metrolingualism（ Pennycook & Otsuji, 2015）等の概念にも通じるという（Garcia & Wei, 2014; Blommaert & Rampton, 2016）・各方面から類似の提案がなされるなかで，トランスランゲージングが特に注目する点は，不均衡な力関係と国家主義的イデオロギーに関連した言語的意味づけの転換の促しにあるという（Garcia & Wei, 2014, p. 43）・」

以上を乱暴に簡略化をすれば,TL も言語の 1 つというよりは言語分類の垣根をとっぱらってしまう，という思想といえる・本論では手話も TL であり，日本手話とか日本語対応手話などの言語変種という分類をやめて，すべてを trans-sign languaging（TSL）と考えようという提案である・

3. 手話の言語観の変遷

手話を言語としてどう取り扱うかという考え方には歴史的な変遷がある・手話が言語とみなされなかった時代は長く，ジェスチャーだと思われていた時代が長く続いた・言語とみなされるようになった現代でも，どういう言語かについては議論が続いている・まずその言語観の変遷を以下にまとめる・

3.1. ダイグロシア

神田(1984)[3]は当時の手話がダイグロシアの状態にあったことを指摘している・ダイグロシア diglossia は日本語訳では**言語兼用**が定着している・ダイグロシアとは簡略化すると同じ，言語内に公的な言語である H 言語（変種）と私的な言語である L 言語（変種）が存在し，話者が状況により使い分けている状態をいう・典型的なのはアラビア語のようにフスハーと呼ばれる正則アラビア語とアンミーアと呼ばれる口語があり，宗教の場や学校教育では正則アラビア語，普段の生活は口語が使われている・こうした例は世界中にある・日本語でも文語と口語があり，公的な場では文語が使

われている・また一部地域では標準語と称されている共通語がH変種，方言がL変種になっていることが普通に観察される・

手話においては手話教室などで習う日本語に近い手話変種と地域の聾者同士の会話では手話方言変種が使われており，ダイグロシア状態であるという指摘である・日本語でも標準語が「正式な」あるいは「きれいな」というニュアンスがあるが，手話でも講座で教えられる手話が標準で，普段の会話の手話は「汚い」と思っている聾者が昔はかなり観察された・

手話通訳養成制度により日本語を手話に翻訳する必要ができ，全国統一的な標準手話を普及させる必要が出てくると，手話辞典や手話テキストが多数発行され，その結果，いわゆる日本語対応手話がH変種，聾者手話がL変種となっていった・一方でろう運動などの聾民族主義が広がるにつれ，聾者手話が日本手話で日本語対応手話は手話ではない，という極端な主張がでてくるに従い，手話の変種の分類に混乱が生じた・手話通訳者は日本手話つまり聾者手話を学習すべき，という理念から，日本語対応手話を排除しようという主張である・ここにはダイグロシアという概念はなく手話にはダイグロシア的な変種はない，という前提条件が含意されている・理念はともかく現実には通訳手話というH変種があり，日常会話としてのL変種があるので，すべてを含んだ日本手話という変種のない単一言語とするには無理がある・現実に存在する，いわゆる日本語対応手話をどう扱うのか，という問題が解決されないまま残ることになってしまっている・中にはその解決として，日本語対応手話を聴者の手話と位置づける，という考えもあるようだが，そうなると手話は聾者の言語，という主張と矛盾することになる・

3.2. ピジンとクレオール

手話変種の分類として，聾者手話（日本手話）と手指日本語があり，その混淆形として中間型手話があるという考えがある･アメリカのASL-PSE－MCE連続体という概念を援用したもので，この概念によれば，ASLは手話でMCEは英語であり，PSEはピジン言語である，とする・この概念の問題点はピジン言語をどう考えるか，である・社会言語学的には２つの言語が接触することで混淆した言語，すなわちピジンが発生することは多くみられる・そしてそのピジンを習得し，母語とする人が登場した場合をクレオールとしている・すなわち言語使用者を母語という観点から分類したものである・別の見方をするとピジン言語の母語話者はいない，としている・すると母語話者のいない言語というのが本当に存在するのか，という言語と人間と関係における根本的な問題が再起することになる・

手話の問題として，ほぼ生得的に手話を獲得しているいわゆる手話母語話者は例外的で，手話獲得は失聴時期との関係でいろいろな段階と変種がある・生得的言語獲得の学習臨界期は6歳頃というのが定説になっているが，失聴時期と言語環境が理由で6歳以降に手話を獲得し母語として使用している聾者の数は多い・いわばクレオールに近い状態にある･しかも失聴時期，失聴度は個人差が大きく，単純に年齢で分類することもできない・現状では本人の自覚つまりアイデンティティの問題として扱われ，言語変種として研究されることはほとんどない・この手話クレオール自体，聾者手話に近い存在から日本語に近い存在まで連続的な変種を形成しており,「純粋な」手話との区別は不可能である・そもそも「純粋な」手話とは何か，実際に存在するかの確認すらむずかしい・

3.3. バイリンガリズム

言語接触の結果起こる言語現象として，バイリンガリズム bilingualism という概念がある・似たような概念としてコードミキシング code mixing，コードスイッチング code switching という名称が使われてきた・バイリンガリズムの日本語訳は**言語併用**である・言語の母語話者という観点からすると，言語接触の結果起きた言語習得において，どちらかの言語だけではなく，混淆した状態であることはよく観察される･その場合，両言語が流暢に使いこなせているように思われることが多いが，多くは偏りがあるだけで，両言語を完全に習得していることはまずない・クレオールの場合はどちらの言語にも属さない，いわば「第３の言語」を獲得するのだが，二言語併用の場合の第３の言語には概念上，該当しない・

バイリンガリズムは言語現象というよりは語学教育で積極的に活用されてきた・語学の目標は学習者を母語話者にすることではなく，実用性がある程度の習得で十分である・何が実用かという問題もあるが，それは学習者の使用目的であり個人差があって一様に定義はできない・

聾教育において，音声言語習得だけを目的とした口話教育の欠点が指摘されるようになり，一方で手話だけの聾教育も手話の文語がないため困難であることから，二言語併用による聾教育が主張された・しかも口話と手話が同時に使用されるので，二モード併用でもあるため，bimodal-bilingual education バイバイ教育として知られるようになった・しかし実際問題として発話者は音声と手話を併用するので音声モードと運動モードが併用されるが，受信者（聾児）は視覚だけでありバイモーダルといえるか疑わしい・音声チャンネルと映像チャンネルというのがより正確であろう・この時点で発信者と受信者の立場の違いも明確になり，バイバイ教育は教える側の論理であることがわかる・

3.4. 言語概念の変遷

以上見てきた言語接触に対する分類思想の歴史的変化があった･変種の分類では,ダイグロシア diglossia が ia という語尾があるように,「状態」であることを示し，バイリンガリズム bilingualism という ism という「思想」が示された・当初に上げたトランスランゲージング translanguaging は「～であること」という現状を示している・このように言語接触という古くからある言語現象の取り扱いには歴史的変遷があるが，日本語ではすべて名詞になってしまい，その違いが明確ではないので，きちんと分類しておきたい・

加えてダイグロシアはほぼ同一言語内の変種であるのに対し，バイリンガリズムは異言語間の問題であることも指摘しておきたい・

変種についても，H 変種と L 変種のような羅列にすると力関係が不明瞭になりがちだが，実際には力関係が大きく作用している･H と L は high と low が語源だから社会的な意識が反映されている・公的と私的という以上の社会的評価の違いがある・翻訳は H 言語間の言語交換であり，通訳は L 言語同士の言語交換になる場合もある・日本の英語教育はほぼ H 変種教育だが，会話場面では L 言語使用が多いため「学校で習った英語が役立たなかった」ということになるのも自然なことである・そこで英会話教室が流行り，英語の L 変種を教えている・H 変種はほぼ統一的であり，故に標準になるのだが，L 変種は非統一であり種類も多い・そのため地域差や個人差が大きく，誰もが自分の変種が「正しい」と信じている・L 変種は方言という更に細かく下位分類されうる変種であることも認識しておきたい・

ピジンは混淆という現象を表すのだが，言語そのものを表すこともある・そのためピジン言語とピジン化と分けることもある・ピジン化は原則 L 変種同士の混淆であり,そのため H 変種とはかなり異なってくるが,実際のコミュニケーションの場では，互いになんとか理解しあえるところまで発達していく・L 変種そのものの種類が多いため，ピジン言語にも多くの種類が存在する・そしてピジンの特徴として言語間の優劣が明確であることがある・状況や場所にもよるが，ほぼ支配と被支配の関係にあることが多い・多数と少数という場合もあるが，ここでいう多数とは実際の人数のことではなく社会的な支配力のことを指す・たとえば日本に戦後やってきた進駐軍は優勢であるから，米軍と接する日本人は英語と日本語を混淆させたピジンを作り出す・これをピジン英語という・具体的には特定の単語を日本語順に並べていく・発音は日本語である・それでも米人には何とか通じる・通じるというのは日本語のままよりも，という比較してのことである・このピジン化を定式化すると「優勢言語の語彙と劣勢言語の簡略化した文法」ということになる・

語学教育においては，学習過程において混淆化が起こるのが自然である・これを間違いとして否定せず，「中間言語」として肯定する考え方も多い・学習において自然に生成されるものを否定してしまうと学習動機を下げてしまうからである・語学的には，どの学習者もほぼ同じような“間違い”をするのであれば，そこには何か原因があり，それを現象としてとらえ，発達段階としてとらえることで，もし法則性が発見できれば教授者側にとって有効なカリキュラム作りの材料となる･中間言語も 1 つの変種だと考えるわけである･しかし語学の目標とする目標言語と中間言語には優劣の差の意識があることは変わりない・その意味ではピジンと同じ扱いになる・

ピジンは母語話者が出現することでクレオールになる・この現象をクレオール化と呼んでいる・クレオールはそのまま固定化するのではなく，ピジンやクレオールが発生する環境では優勢言語との接触は続くのが常態である・語学における中間言語もさらに学習を続けることで目標言語に近い変種が次々と現れる・そもそも言語そのものが固定的ではなく流動的であり，同一言語内においても古語があり，死語があり新語がある・比較的変化速度が遅い文法も変化する・語彙も文法も変化してしまうと別の言語のように理解できなくなる・古文がそうである・クレオール言語も優勢言語との接触が長くなると次第に優勢言語に近づいていく変種ができていく・この言語現象を**脱クレオール化** decreolization と呼んでいる・

日本の手話を観察してみると，学校教育や手話通訳が一般化するまでの時代は独自の語彙と独自の文法をもった言語であったと推定される・学校教育の過程で日本語の H 変種と L 変種のみの手話が接触し,まずダイグロシアの状態になる・授業時間は日本語の読み書き，会話は手話という使い分けである・この段階で日本語に対応する手話語彙が生成されて行ったであろうから，この時点で「日本語対応手話」が発生した・この時期にいわゆる「手真似」が「手話」に変わったといえる・聾学校内という限られた社会内ではその社会ごとにピジンが生成されていったため，聾学校ごとの手話変種つまり手話方言が形成されていったと考えられる・とくに寮生活が中心であった頃は普段の生活は手話ですることになるから幼児はこの手話変種が母語化するため，クレオール手話が形成されていく・これが現代でいう「日本手話」である・

手話通訳の出現は学校教育よりもさらに日本語との混淆が強く進むことになった・一方で手話通訳者養成のための手話学習では先生が聾者，生徒が聴者という

社会的立場から，新たなピジン化が発生する・手話と日本語の優劣関係が逆転し，簡略化した日本語文法に手話語彙を乗せるという変種である・こうしたピジン化は音声言語のピジン化ではあまり起こらないが，手話において発生したのは手話学習という特殊な場が成立したからである・優勢言語の名称を用いるピジン英語になぞらえれば「**ピジン手話**」ということになる・いわゆる日本語対応手話と命名した人の意識には「手話を日本語に対応させた」という考えがあったのだろうが，実際には「手話に日本語を対応させた」のであったといえる・この現象をさらに明確化したのが「日本語ラベル」という命名で，手話に日本語訳のラベルをつけたのだから，手話を優先しているといえる・なぜそのようなことをしたかは明確で，手話に文字がなく記述するには特殊な記号を開発するか，音声言語の対訳を利用するしかなく,後者が選択されたのであり，その習慣は今でもどこの手話でも利用されている・このラベリングという行為は分析やその応用である教育には不可欠でもある・だが欠点としてラベル化に利用した言語の意味に引っ張られやすいことがある・英語と日本語の関係からいえば，英語に日本語ラベルを付けて借用すると，いわゆるカタカナ英語になり，そこで固定化された意味がそのまま日本語の中で使用されていく・あるいは同じカタカナ語に複数の意味が含まれるようになり混乱をきたす・手話学習においても日本語ラベルの付いた手話表現は対訳が固定化され，日本語話者は日本語の意味で理解するようになる・この固定化が「対応」と呼ばれる現象の正体である・カタカナ英語は英語話者から見ると「英語ではない」と思うのと同じように,日本語対応手話は「手話ではない」と思うのは自然な感覚といえる・この感覚は，ピジン英語は正式な英語ではない，という言語感覚と同じになり，ピジンを低く見る現象につながっていくが，同様に手話者もピジン手話を低く見ているであろう・実際,手話教室においては中間言語を認めることはなく,誤りであるとして訂正される・

3.5. 手話の脱クレオール化

ピジン言語が母語話者を獲得しクレオール化するのだが，クレオールもやがて脱クレオール化し優勢言語にどんどん近づくことが多い・優勢言語との接触がなくなれば脱クレオール化しないが，戦争による占領でもないかぎり，たとえば英語のように国際言語化すると英語教育により脱クレオール化が進行する・ここで注意が必要なのは，ピジン化はL変種同士の混淆であったものが，英語教育はH変種教育なので，脱クレオール化は英語のL変種に接近するのではなく，H変種に近づいていく，ということである・

手話の場合，学校教育で発生したピジンが聾学校内でクレオールとして発達していったため，日本手話が最初からクレオールとしての特徴があることを見逃してはならない・クレオール化した段階ですでに手真似と手話が別変種になったことを理解する必要がある・同時法的手話[(4)]の考案者たちは伝統的手話，同時法的手話に中間型手話という命名をしているが，中間型が日本語と手話のピジンという観点は正しいが，伝統的手話には手真似と日本手話が混在していることには着目していない・その彼らが同時法的手話を日本語対応手話と変化させたことも混乱の一因であろう・この日本語対応手話の実態は日本語語彙に合わせて多くの手話語彙を造語したものであり，手話通訳の要請に従ったものである・その意味では全日本ろうあ連盟が今も「新しい手話」を作り続けている思想と共通するものがある・使用場面が異なるため当然変種が生まれ，淘汰されていくことになる・日本語対応手話とは１つの思想のことで，実態は多くの変種を含んでいる・

手話の脱クレオール化というのは日本語対応手話の影響はむしろ少なく，日本手話そのものが日本語のH変種に近づいているという言語現象である・日本語対応手話の中でも講演などの手話通訳場面で使用されるタイプ(変種)は手話のH変種といえるかもしれない・これも新たなピジン化であり，優勢言語は日本語，劣勢言語は日本手話（クレオール）である・日本手話が昔の手真似の文法から簡略化が進み，すでに日本語文法に近づいていたのが，さらに簡略化され日本語に近づいていく・言語混淆も流動的に継続される・

手話文法の簡略化を概略すると,ほとんどの日本手話文法論では表情による文法表示が強調されている・文法の一般論では，基本文法は標準言語の主語述語関係や語順などが型として示され，統語構造が示される・音調や抑揚などの音声現象は口語であるL変種にしか現れないのでH変種が主体の標準言語の文法には含まれない・誤解を解いておきたいが，アナウンサの発話は標準語ではなく口語である・文章を読んだ場合でも微妙な差異がある・しかし日常会話のような語彙や文法の変化は見られないので,標準語とみなされている・実際，テレビタレントの発話はほとんど方言のままである・

日本語では助詞は文法の重要な部分だがL変種である口語では省略されることが多い・現在の手話文法論は口語文法の特徴をもち，手話がL変種しか存在しないという仮定に立てばそうならざるをえないが，日本語対応手話が手話におけるH変種として存在しているという立場に立てば，手話文法はむしろその中にもある，と考えるのが妥当であろう・

「日本語対応手話の文法」について筆者以外の手話研究者の指摘を見ないが，その原因は日本語対応手話は

手話ではないという思想に立っているからと思われる・それならば日本語対応手話は何なのかという回答が必要だが，日本語対応手話は日本語である，と断定するだけでは偏見である・日本語話者が理解できないから日本語ではないという論理も成り立つ・日本語とのピジンであれば日本語話者がある程度は理解できるはずだが，まったく理解できない・一方で手話者はある程度は理解できる・つまり日本語対応手話はその名のごとくピジン日本語ではなく，ピジン手話である・

そこで現段階における手話と日本語の混淆状態を整理しておく・

1.学校教育以前に手真似と呼ばれた変種があっ

2.学校教育において日本語とのピジンが発生した

3．寮生活によりクレオール化した

4．手話通訳制度により再度ピジン化が発生

それぞれの段階を命名すると1．手真似，2．日本手話，3．聾手話，4．日本語対応手話，となる・ただ日本手話に1から3まですべてを含める人もいる・4の手話通訳の影響には厳密にいうと手話サークルなどによる聾者と聴者の接触機会が増えたことによる影響も考慮しなくてはならない・手話サークルという日本独自の手話普及制度により，聾者は日本語との接触機会が格段に増えた・手話サークルの聴者の多くは言語的には，教室では劣勢の学習者でも授業後のアフターでは優勢の立場になることが多く,「日本語の話に手話を合わせる」状態になり，知っている手話語彙を日本語の口話と併用する，同時法よりもさらに日本語に近い変種を使用するようになる・このタイプの変種を**手話付口話** Sign Supported Speech(SSS)という・手話を学習した難聴者はこの変種の使用が多いため難聴者手話と呼ばれることもある・これも一種のピジン化で優勢言語が日本語でクレオール手話語彙が使われるため，より日本語に近い・この変種は日本語話者にも理解できるので**ピジン日本語**と分類するのが適当かもしれない・ただ手が動くのを見るため，これも手話だと認識する人が多く，日本語対応手話に分類する人もいるため，混乱がさらに拡大することになる・この変種は日本語の口話と併用されるため，口話方言にも対応しやすく，L 変種との混淆も進む・手話通訳の手話が日本語の H 変種との言語交換であるのに対し，SSS は L 変種との言語交換であると分類するのが理解しやすい・SSS はまだピジンの段階にあるが，クレオール化することは単純には考えにくい・聾学校教育における寮生活のような言語環境は想定しにくく，手話通訳者家庭に育った聾児がそうなるかどうかは未見である・ただ聾児のほとんどは聴者の家庭に生まれ，今の両親は口話だけに拘るのではなく，手話を学習して子供とのコミュニケーションを図る努力をするし，手話通訳もよく見かけるようになってきた・そして聴者の子供が手話を学習することも多くなってきたので，聾児が日本語対応手話に接する機会は昔に比べると圧倒的に多い・そのため今の聾児がピジン日本語から新たにクレオール化する可能性は否定できない・

5．ピジン日本語との接触による再クレオール化

現状を見ると学校教育を受けていない非就学聾者(5)の手話は手真似を残しているが，高齢化が進み，近いうちに消滅すると予想される・

筆者の見解では，現在，混淆化は次の段階に進行していると思われる・それはスマートフォンやパソコンによるソーシャルネットワーク（SNS）による影響である・学校生活や通訳の利用が手話者の言語生活を大きく変容させたように SNS が大きな影響を与えている・そして日本語対応手話のクレオール化が進んでいるとしたら，手話はさらに日本語に接近しているといえよう・この言語現象を脱クレオール化と呼ぶか，再クレオール化と呼ぶか，新クレオール化と呼ぶか，名称は人により異なるであろうが，言語変化としては新たな手話変種が生まれつつあるのは間違いない・この調査はまだ行われていないようである・

4. アノテーション方法の変更

手話を記述する方法であるアノテーションにも歴史的変遷がある・言語を最初に記述するのはほとんど言語学者なので，音声言語であれば IPA を用いる・しかし実用目的で接触する人は語彙集を作る・目的は宣教であり貿易であり教育である・語彙集は聞いたままを自国語の音韻文字で記述し，それに意味を書き込む・日本語ならカタカナだが，多くはアルファベットとなる・手話については IPA に相当する記述法が開発されそれがストーキー法であり，HamNoSys ハムノーシスである・あるいは写真やイラストを用いた・しかし一般には普及しないのと，手間のため，手話を利用しようとする人は音声言語で表記しようとする・それがラベリングである・そのため手話辞書は日本語の見出し語に日本語で動作を表現する方法や写真，イラストを載せる手話辞典が発行されてきた・

しかし手話辞典の欠点は語彙集であるため,文の記述ができない・手話表現を全部日本語で書くのは翻訳になる・手話辞書類にはわずかに文章も掲載されているが,「役に立つ」範囲で限定された文章になっている・翻訳には語彙だけでなく文法知識が不可欠なのだが，手話文法は未完成であり，現在手話文法を解説する書籍は日本手話文法すなわちクレオール手話文法になるのだが，実際にはクレオール手話に残存する手真似の文法の一部を書いたにすぎない・文法論の基本となる項関係や品詞分類などはほぼ手つかずの状態にある・

そうこうしているうちに手話はどんどん変化していき，変種が次々と生まれていく・しかし記述法は旧来のままで不完全でもある・

こうした現状を鑑みると，現行の変種を記述する方法は手真似の時代の手話の記述に拘った表情の表記法や古典的な記述法に拘るよりも，いっそ日本語の翻訳文を利用する方が実用的であると思われる・それも標準日本語というH変種ではなく，L変種である口語が適していると考えられる・現在の手話変種は手話通訳のようなH変種もあるが，手話会話に用いられるL変種は脱クレオール化ないし再クレオール化した，かなり日本語口語に近い変種であるからである・

具体的な場面を想定すると，手話表現を見て，その場で通訳した音声がアノテーションとして適切である，ということである・従来はアノテーションといえば文字記述を指していたが，それは処理上の都合であって現在ならば音声データのままでもよいはずである・

4.1. 通訳音声データを用いた手話文の記述

電子技術的には音声データの処理は以前よりも楽になったので，それを記号化するのはむずかしくない・従来はつねに文字に変換してから利用することが多かったし，言語音声については文字変換という概念から抜け出ることが難しかった・現時点でも音声データは簡単に文字化できるので，それを利用することは差支えないが，それは論文にするなどの文字化が必要な場合だけであって，翻訳のような場合に必須ということではない・

単純化すれば，手話動画が直接日本語音声になれば，翻訳機としては成立するし，むしろその方が実用性が高いであろう・

手話表現も録画もしくはライブによる動画像であってもよいが，動作そのものをデータ化したものであってもよいかもしれない・手話は視覚言語である，という観念から離れれば，動作だけの方が冗長性は低くなる・人間が手話を認識する場合，手話者は動画に表れる多くの視覚情報を無視しているか，潜在的に認識しているにすぎない・相手手話者の服装，背景，美貌度などは非言語情報であり，コミュニケーションにまったく関与しないとまではいえないが，言語情報には関係がないと考えられる・実は表情の多くも文法性よりは抑揚のような超分節音的機能の方が多い・表情の文法性についてもモダリティ（法）やアスペクト（相）に関するものがほとんどであり，項関係や格関係，数などの基本文法は手や腕の動作で表現されている(6)・そうした手話の言語的特徴を前提として，手話から日本語への翻訳は手話動作を認識して直接音声に変換することが情報ロスも少なく正確性も高まると考えられる・なぜならそこに意味が介在しないからである・従来の日本語ラベルを用いる方法はどうしても意味が介在してしまう・とくに日本語の同音異義語や手話の同形異義語において誤訳が発生しやすい・音声言語の通訳において，L変種間の翻訳は誤訳のリスクがなくなるわけではないが文字化するよりも，音声の受け手は普通の口語を聞く感覚で適宜文脈から正しく理解するため，誤訳リスクは減らせる・文字化された翻訳では読み手は翻訳語の意味に拘束されてしまい，それが別の同義語であることがわかるためには原語を理解できる能力が必要であり，それならばそもそも翻訳の必要はないことになる・現実の翻訳機はH変種同士の翻訳システムにL変種からH変種への翻訳が付加されているようである・

4.2. 手話における動作の優位性

手話の音韻的要素については従来から，手の形，位置，動きであることが指摘され，外国においては掌の方向などが提案され，その要素を動素と呼んだり手話音素と呼んだりパラメータと呼んだりしている・名称についてここでは議論しないで音韻的要素ということにして話を進める・これらの要素は等位関係にはなく認識上は優劣があることが指摘されている(6).手話学習者や初歩的手話研究者は手の形に注目しやすいが，現実の手話認識では手の形が重要な指文字を除くと手の形がない「ドラえもん手話」でも理解できることから，手の形は劣勢要素であることがわかっている・位置についての検証例は未見だが，位置は身体との相対関係であることは明白であり，身体との物理的距離ではなく，位置空間という抽象的なものである・従って位置だけを示すことは不可能で，何かその空間上に物理的存在が必要で，それが手ということになる・

動きについても同様で空間上での運動であり，時間との関連で距離と速度で決まる・言い換えると手などの物理的存在と空間と時間がないと存在できない，さらに抽象的なものである・抽象度についていえば

動き＞位置＞手の形

であり，実は言語としての重要度もこの順である・重要度を直接論証したわけではないが，神田(2010)は音素記述された手話語彙を検索するには，この順で検索結果がでやすいことが示している・翻訳においては検索速度も重要な要因であるから，まず動作に着目したい・手の形については特定の形に頻度が偏っていることも示されているので，出現の冗長度を活用できる・なお本論では議論からはずしてあるが，片手か両手かという分類も重要で，両手の場合は動作に制限が起きることは古くから観察されてきた・手という動く物理的実体が複数化すると観察しにくくなるのが自然で，制限的になるのが自然である・動きが位置との相関関係が深いことはいうまでもない・

5. 実験手順

以上のような社会言語的考察と手話工学的考察を重ね合わせて，次のような手話アノテーション法を提案する・

5.1. 手話文の口語通訳

手話通訳者は本来日本語のＨ変種への変換を心がけているはずであるが，それでも録音してみるとそのまま文字化できないことが多い・そのため講演録は音声の文字化ではなく校正を行い,「日本語の間違い」として語彙を修正したり，助詞や接続詞を加筆したり，文体そのものを文語に変えてしまう・これは日本語の議事録でも同様で発言をそのまま載せることは少なく，文語というＨ変種に変換する作業がある・こうした変種変換はダイグロシアがある以上,日常的に行われる・このため手話表現のアノテーションはＬ変種が日本語のＨ変種に変換されるという二重変換が同時に行われてきた・そのため原表現の意味が損なわれることもしばしばあった・それ故，原表現の意味をできるだけ残した翻訳をするには日本の口語による通訳が適しているといえる・

5.2. 口語通訳の音声処理

音声データをそのまま利用することは少し前までかなり面倒なことであったが，音声認識や音声合成の研究が進んだ結果，いろいろな方法が開発されている・原始的な方法としては音声のフォルマントを視覚化して，その変化をとらえる方法である・現在は人間の判定だけでなく機械による判定も可能になった半自動型も増えてきて，音声を仮名にすることは比較的容易である・仮名表記になっていれば，現行の手話辞典との整合性もとりやすく，日本語ラベルをつけやすい・またこの段階で翻訳する必要がなければ同音異義語の処理は不要で仮名の連続であっても差支えない・無論ローマ字表記であってもよい・

音声の表記は仮名である必然性はないので，フォルマントの連続変化パターンを音素に変換する古典的な方法の方がより正確な情報として応用がきくであろう・

仮名表記あるいは音素表記する場合には，音声の遷移 transition の処理が問題になるが，同様に手話動作の遷移も問題になってくる・手話動作の動きのパターンと音声のフォルマント変化パターンの直接マッチングをすれば，遷移処理を省略できるかもしれない・そして従来とは異なる手話辞書ができる可能性もある・その辞書は日本語ラベルによる手話辞書の欠陥を補うことにもなる・

6. まとめと考察

上記ではまず言語変種としての手話の変種とその歴史的変化について現状で１手真似の時代，２ピジン化の時代，３クレオール化の時代　４再ピジン化，５再クレオール化の時代があることを述べた・また言語変種には同一言語内にＨ変種とＬ変種があり，翻訳は基本的にはＨ変種間の交換だが，手話にはＨ変種が存在しなかったため変則的な翻訳になった・ピジン化は本来Ｌ変種同士の変換だが，手話の場合は手話通訳により日本語対応手話というＨ変種が創成され，従来主張されてきたような日本語―ピジン手話―日本手話というような単純な連続体ではないことが示された・一方で手真似のような手話口語が学校教育によりほぼ絶滅・クレオールである手話変種が日本手話となった・手話ピジンにもSSSのようなより日本語に近いものと中間型と呼ばれるより手話に近いものがあり，こうした手話変種分類は社会言語学的研究目的以外には手話トランスランゲージング TSL としてまとめるのが妥当であるというのが筆者の見解である・

6.1. TSLの認識

TSL のようなバラエティに富んだ変種を手話としてまとめて処理するには，できるだけ翻訳や文法解析などを介在させない方がよく，表現動作そのものを物理的な記号に変換した手話データと，翻訳のためにはＨ変種である文字記述ではなく，Ｌ変種である通訳音声データを使用することが，より意味が正確に伝わると仮定される・

6.2. 手話動画認識

手話動画を機械的に認識するには現状，モーションキャプチャによる方法とオープンポーズなどの光学的手法が中心になりつつある・データグローブやデータスーツなどの磁気データも有効とは思うが，装置の大きさや移動が困難という点と，手話者の精神的負担も大きく，場面や文脈が固定化されるため，ビデオ録画やインターネット動画などの方がより自然に近い TSL が獲得できる・

録音についても，手話の現場で録音する手法と録画像を見て通訳する方法が考えられるが，これは通訳者の熟練度と手話者との関係により文脈に依存した語彙選択の変換に差が出ることが予想される・手話者に近い関係にあれば，１つの語からでも言いたいことが想像できるが，第３者であればある程度の語彙連鎖がないと想像できない・これは文脈に依存する口語の特徴でもある・また上述のようにＬ変種には多くの下位変種がある・

従って研究手順としては手話のＨ変種の認識実験から始めれば，その翻訳音声は手話動作との意味的誤差が少ない・それでは翻訳の意味がない，という批判はもっともだが，最初から完全な実用を目指すのは難しく，限定的な内容から始めるべきである・

手話のH変種いわゆる日本語対応手話は当然ながら手話通訳者が一番上手である・意味がないように思われるかもしれないが，仮に手話通訳者が手話表現をして本人が読み取って音声にすればズレが出る可能性は相当低い・これは深層学習における教師データとして最適である・実際問題として同じ手話動画でも別の手話通訳者が読み取った場合，音声データが異なる可能性がある・同一動画を多くの手話通訳者に見せて音声データを収録して分析しても 100%の正答率になるとはいえない･それがH変種とL変種の違いでもある･それが第２段階の実験になる・この差を十分理解した上で，多くの手話のH変種を集め，その読み取り音声データとのマッチングペアの確率を分析すると正答率が分かるが，多量のデータを深層学習すれば機械による自動化も無理ではない・

社会実装を想定すると，現状で一番よく見るのはH変種であり，L変種は直接の接触がないかぎり出会う機会が滅多にないので，まずはH変種の音声への自動変換を目標とすることが実現性と有用性が高い・

音声言語の翻訳機でもL変種同士の変換はまだできていないと思われる・一見会話翻訳機に見えるが，話す方も聞く方もH変種であることを認識していないだけのことである・ある意味，ダイグロシアは普段は認識されていないのが普通で，使い分けがわかるのは外国語の場合だけである･研究上においては TSL を目標言語とするのだが，日本語にH変種とL変種があることは理解しておきたい・

7. 今後の課題

上記提案のように上級手話通訳者の手話表現を自分で読み取った音声と比較することは一見容易に見えるが，実はさまざまなハードルがあることが想定される・まずどのような形式のデータに変換するかという課題である・ビデオ画像は２チャンネルが同時収録されるので収集過程は問題がない・そのままではマッチングができないので，どうしてもセグメント化や記号化のアノテーションをしたくなる・それをどう避けたらよいかである・

要点は当人であっても，読み取る場合には意味の介在がある点に着目することである・発話の時間のずれもあるであろう・手話と日本語では句の切れ目とリズムが異なるので，データの塊り方が異なってくる・読み取り作業であれば，音声のリズムやブレスが通常の発話と異なることが予想される･その計測をするには，一旦読み取って発話したものを文字化し，それを朗読してもらうと違いがはっきりする・これが言語干渉である・本論ではそこまで言及できなかったが，その予備実験として言語干渉の測定とその方法の確率が必要かもしれない・

謝　辞

本研究は文部科学省科学研究費補助金課題番号課題番号 18K18518（代表者神田和幸）による研究成果の一部である.

文　献

[1] 大山万容，トランスランゲージングと複言語教育―言語能力観から検討する―，母語・継承語・バイリンガル教育（MHB）学会，2019

[2] 猿橋順子･坂本光代,トランスランゲージングの遂行性：国際的なトークショーのディスコース分析を通し，青山国際政経論集 105 号，2020 年 11 月

[3] 神田和幸，手話のダイグロシア,日本福祉大学研究紀要 62 号，1984

[4] 田上・森・立野,同時法的手話,栃木聾学校,

[5] 神田・木村，未就学聾者

[6] 神田和幸,手話の言語的特性に関する研究,福村書店,

[7] 木村勉，神田和幸　他，”手話認識機能を備えた手話辞書システムの開発”，信学技報，vol1.120，no.419，WIT2020-38，pp.53-58，March.2021.

[8] 磯谷光，木村勉，神田和幸，”ディープ・ラーニングを用いた手話認識に関する研究”，信学技報，vol1.120，no.419，WIT2020-38，pp.47-52，March.2021.

手話コミュニケーション研究会 2020・21 論文集
2022 年 5 月 1 日　　初版発行

編集者 神 田 和 幸
発行所 東京都中央区日本橋小舟町 6-13
日本橋小舟町ビル 5F
特定非営利活動法人
手話技能検定協会