公立はこだて未来大学 2018 年度 システム情報科学実習
グループ報告書

Future University Hakodate 2018 System Information Science Practice
Group Report

プロジェクト名
AI するディープラーニングプロジェクト

Project Name
AI Love Deep Learning Project

プロジェクト番号 / Project No
06

プロジェクトリーダー/Project Leader
1016163 濱口 和希　/Kazuki Hamaguchi

グループリーダー/Group Leader
1016224 三浦 幸泰　/Kouta Miura

グループメンバー/Group Member
1016224 三浦 幸泰　/Kouta Miura
1016082 坂下 夏槻　/Katsuki Sakashita
1016075 和田 孝喜　/Koki Wada
1016061 川村 一世　/Issei Kawamura
1016036 松元 健　/Takeru Matsumoto

指導教員/Advisor
竹之内 高志/Takashi Takenouchi
香取 勇一/Yuuichi Katori
寺沢 憲吾/Kengo Terasawa
片桐 恭弘/Yasuhiro Katagiri
冨永 敦子/Atsuko Tominaga

提出日/Date of Submission
2019 年 1 月 16 日/January 16，2019

概要

近年，ディープラーニングと呼ばれる技術が著しい発展を遂げている．ディープラーニングとは，任意の関数を近似することが可能なニューラルネットワークを利用した機械学習の総称である．観測したデータから未知の事象を推定するという手法はディープラーニング以外にも様々あるが，それらの手法は多次元データに適応しづらい場合が多い. 一方，ディープラーニングは多次元データの推定を得意としている．したがって，画像のようなデータを扱う課題（分類問題や物体検知など）ではディープラーニングが近年最もよく使われている．AI するディープラーニングプロジェクトとは，そのディープラーニングを用いて現実の問題を解決しようと考えている学生が集まったプロジェクトである．

私たち B グループは，ディープラーニングと同じく最近様々な形で利用されているモーションキャプチャと呼ばれる技術に注目した．モーションキャプチャとは，物体の運動を記録する技術のことである．モーションキャプチャには高価な機材が必要になるため，私たちはディープラーニングを用いて比較的安価で可能なモーションキャプチャの開発を試みようと考えた．しかし，ディープラーニングを用いて一つのカメラのみでモーションキャプチャを行うという試みは，既にある程度研究が進められていた. そのため私たちは，さらにその一歩先を行くモーションキャプチャ手法を考案することにした．ディープラーニングを用いたモーションキャプチャを行ういくつかの研究では，カメラは固定するか，もしくはモーションキャプチャの対象者とは別の人がカメラを動かす必要がある．つまり，一人でモーションキャプチャを試みた場合，動くことができる範囲が固定されてしまうという問題がある．そこで私たちは，対象者のみで自由な範囲を動くことができるようなモーションキャプチャ手法を提案することにした．私たちが提案する手法とは，胸部に装着した全方位カメラを用いてモーションキャプチャを行うというものである．具体的には，胸部に装着した全方位カメラで撮影された手や足や顎といった情報からディープラーニングを用いて，全身の動きを推定し記録するという流れである．この提案手法であれば，場所に縛られることなくモーションキャプチャを行うことができる．私たちはこの手法を実現するために二つのグループに分かれた. 一つはデータを収集する環境を整え，実験を円滑に行えるようにするカメラ班，もう一つはニューラルネットワークの実装や実験を行うニューラルネットワーク班として一年間活動をした.

前期は，二つのグループに分かれた後，開発環境の構築や一般的な畳み込みニューラルネットワークの実装，データ収集方法の検討や実装を行った.

後期は，データ収集を行い，一般的な畳み込みニューラルネットワークに比べて表現力の高いモデル（ResNet や DenseNet）を実装し，各々で実験を行った.

結果的に十分な精度は得られず，いくつかの課題が明らかになった.

キーワード ディープラーニング，モーションキャプチャ，姿勢推定，全方位カメラ

(※文責: 和田孝喜)

Abstract

In recent, the technique that is called Deep Learning remarkably advance. Deep Learning is method that estimate something from data using neural network. There are some methods to estimate something from data except for Deep Learning, but these method cannot apply to high-dimensional data such image. Whereas Deep Learning easily applies to high-dimensional data. Thus, Deep Learning is most used to these data in recent. This project solves some real world problems using Deep Learning.
Group B focused on motion capture. Motion capture is technique that records motion of anything. Because of this technique needs expensive equipments, we thought new motion capture technique that is not necessary expensive equipments using Deep Learning. However, motion capture technique using Deep Learning such we thought has already studied, so we decide to propose more advanced method these studying. In these study, There is problem such as camera should be fixed, or anybody move camera to follow subject that wanted to capture motion. In other words, when anyone wants to capture motion oneself, one is restricted range of motion because of camera must be fixed. Therefore, we propose new motion capture method that is not restriction of range of motion. For our method, we capture motion using omnidirectional camera equipped to chest. Specifically, we take a movie with omnidirectional camera equipped to chest, and then we estimate and record motion using it's movie and Deep Learning. This method does not restrict range of motion and not necessary expensive equipment. We divided two groups camera team and neural network team to implement this method. Camera team create an environment for collecting data. Neural network team implement some neural networks and experiment using these neural networks.
In the first semester, we constructed development environment, implemented general convolutional neural network and discussed data collection.
In the second semester, we collected data, implemented more representation power than general convolutional neural network (ResNet and DenseNet) and experimented using these neural network models.
As a result, we could not obtain enough accuracy for the task, and then we found some problems to be solved.
Keyword Deep Learning, Motion Capture, Pose estimation, Omnidirectional camera

(※文責: Koki Wada)

目次

第 1 章　はじめに **1**
1.1　プロジェクトの目的 1
1.2　背景 1
1.2.1　該当分野の現状・従来例 1
1.2.2　現状における問題点 1
1.3　グループの目的 2
1.3.1　本グループにおける目的 2
1.3.2　具体的な手順 2
1.3.3　グループ内での課題の割り当て 3

第 2 章　課題の設定 **4**
2.1　課題設定 4
2.2　前期目標 4
2.3　後期目標 4

第 3 章　課題解決プロセスの詳細 **6**
3.1　技術知識習得 6
3.2　データ収集班 6
3.2.1　開発 6
3.2.2　課題 8
3.3　ニューラルネットワーク班 9
3.3.1　ニューラルネットワーク概要 9
3.3.2　開発 11
3.3.3　課題 13

第 4 章　発表 **14**
4.1　中間発表 14
4.1.1　中間発表に関する活動 14
4.1.2　中間発表における評価シートの内容と考察 15
4.2　最終発表 17
4.2.1　最終発表に関する活動 17
4.2.2　最終発表における評価シートの内容と考察 18

第 5 章　実験および結果 **20**
5.1　実験 20
5.1.1　実験環境 20
5.1.2　モデルの入出力 20
5.1.3　モデルの学習 20
5.1.4　仮実験 20

5.1.5　結果の検証方法 20
5.2　ResNet の結果 21
5.3　全身で学習した DenseNet の結果 21
5.4　手首を除いて学習した DenseNet の結果 21
5.5　関節を上半身と下半身に分けて学習した DenseNet の結果 22

第 6 章　考察　23
6.1　入力に含まれる不要な情報 23
6.2　タスクの大きさ 23
6.3　多重共線性の問題 23

第 7 章　今後の展望　25
7.1　部位ごとの学習 25
7.2　関節の角度の簡略化 25
7.3　角度の表現方法の変更 25
7.4　クォータニオンの特性に合わせたモデルの構築 25
7.5　より精度の高いデータセットの作成 25
7.6　より精度の高いデータセットの作成 26
7.7　リアルタイムモーションキャプチャ 26

第 8 章　グループメンバーの活動　27
8.1　三浦幸泰 27
8.2　坂下夏槻 27
8.3　和田孝喜 27
8.4　川村一世 28
8.5　松元健 28

第 9 章　まとめ　29
9.1　前期の成果 29
9.2　後期の成果 29
9.3　結果の考察 30

参考文献　31

第 1 章　はじめに

この章では，プロジェクトの目的と背景，グループの目的について述べる．

1.1　プロジェクトの目的

世界が情報化社会に移り変わる中，大量のデータを処理するための手法として機械学習は日々発展を続けてきた．その中で，企業が持つビッグデータの存在とその価値が周知されるようになると，機械学習は一気に注目を集め，様々な企業で導入されるようになった．ビッグデータを扱う上で，特に適しているとされたのがディープラーニングである．ディープラーニングは機械学習技術の一つで，ニューラルネットワークという脳の神経構造を模した機械学習モデルのうち，隠れ層という層をいくつか重ねたものを指す．層を重ねることで，データを分析するために必要な要素とその関係を, より多くかつ複雑に蓄積することができるようになっている．ディープラーニングの精度は用意する学習データ量に左右されるため，限られたデータ量でより精度を上げるための研究開発が進められている．本プロジェクトでは，このディープラーニングを用いて新たな問題解決に挑むことを全体の目標とする．

(※文責: 坂下夏槻)

1.2　背景

1.2.1　該当分野の現状・従来例

ここ数年，バーチャルリアリティが大きな流行と発展をみせており，市場規模はこれからさらに大きくなるとみられている．バーチャルリアリティは「より現実に近い体験を求める」という性質から，身体の動きを仮想現実に直接反映するモーションキャプチャを入力機器に用いる．モーションキャプチャは，スポーツのフォームや日常生活の仕草のデータを収集するなどの研究目的にも多く利用され，バーチャルリアリティに関わらず様々な分野を支えている．従来のモーションキャプチャには，専用のスーツを着用しジャイロセンサ等で検知する，正面から深度センサで検知するなどの手法がある．

一方で，機械学習によってカメラだけの情報で姿勢を推定し，モーションキャプチャを行うという試みもさかんに行われている．例として，Zhe Cao らの “OpenPose”[1] は映像に映った人間の各関節の二次元位置を推定することができる．また，Weipeng Xu らの “Mo2Cap2”[2] は，つば付き帽の先に下向きのカメラを付けることで全身を映し，三次元姿勢を推定する．

(※文責: 坂下夏槻)

1.2.2　現状における問題点

従来のモーションキャプチャの手法では，精度を向上させようとするほどセンサを増やす必要があり，費用や管理システムのコストが問題になる．正面からのセンサを用いる場合，少ないセンサ

で全身の動きをとることができるが，移動範囲がセンサの範囲内に制限されてしまう．また，正面からのセンサが全身を認識することができ，かつ対象が十分に動き回れるような比較的広いスペースも必要になる．以上のモーションキャプチャの問題は，一般にバーチャルリアリティが普及する上での一つの障害となっている．前述のように，機械学習を用いて姿勢推定の精度を補助し，カメラでモーションキャプチャできるようにする試みが続いているが，現在実際にモーションキャプチャに利用されることはまだ少ない．

(※文責: 坂下夏槻)

1.3 グループの目的

1.3.1 本グループにおける目的

本グループの目的は，第三者視点のカメラと大掛かりな機材を必要としない，ディープラーニングを用いたモーションキャプチャを行うためのシステムの開発である．従来手法は，モーションキャプチャを行うにあたって撮影するための大掛かりな機材が必要となり，結果的に多くのコストがかかってしまう問題がある．その問題を解決するために，ディープラーニングと全方位カメラを用いることで低コストかつ簡単に姿勢推定ができると考えた．そこで，私たちは全方位カメラを胸に装着して姿勢推定を試みたが，問題が発生することが分かった．全方位カメラは前方 360 度撮影することができるが，全方位カメラの撮影範囲外である体の軸より後ろにある腕や足等を撮影することはできない．そのため，モーションキャプチャを行う前に，姿勢推定をする必要があると考えた．姿勢推定を実現させるためには，全方位カメラによって得られた映像から姿勢を出力するモデルが必要である．そこで私たちは，ディープラーニングを用いることによって全方位カメラの映像から得られた情報をもとにモデルが作成できるのではないかと考えた．この姿勢推定ができるようになることで，全方位カメラの装着者の周囲の状況をモデル化し，全方位カメラに映っていない部分の状況を予測することができる．これらのことから，私たちはディープラーニングを用いてモーションキャプチャのシステムを開発することを目的とする．

(※文責: 三浦幸泰)

1.3.2 具体的な手順

プロジェクトを行うための具体的な手順は，モーションキャプチャや姿勢推定の調査，画像認識の手法である畳み込みニューラルネットワークの構築及びデータの収集である．モーションキャプチャや姿勢推定の知識を得るために先行事例の調査を行う．畳み込みニューラルネットワークはニューラルネットワークの層をより深くするために考えられた ResNet と呼ばれるモデルおよび画像認識の分野で精度が高く，深い層での学習が比較的安定する DenseNet と呼ばれるモデルを構築し，用いる．データ収集には，全方位カメラと KinectV2 を用いる．全方位カメラで動画を撮影し，KinectV2 で動画の各フレーム毎の関節の角度を収集を行う．これらのデータを用いて畳み込みニューラルネットワークの訓練と調整を行う．その後，今回は歩行のみの姿勢推定に限定し，Unity の 3D モデルを用いて，全方位カメラの映像から畳み込みニューラルネットワークを用いた推定を行い 3D モデルに反映させる．

(※文責: 三浦幸泰)

1.3.3　グループ内での課題の割り当て

本グループでは，データを収集する環境の整備や，モデルに学習させるためのデータセットの作成，姿勢推定の実験を行うためのシステム及びデータセット作成のためのシステムの構築を行うデータ収集班，モーションキャプチャおよび姿勢推定の先行事例の調査や，畳み込みニューラルネットワークの実装および実験を行うニューラルネットワーク班として活動する．

(※文責: 三浦幸泰)

第 2 章　課題の設定

この章では，課題設定と目標について述べる．

2.1　課題設定

本グループでは，全方位カメラからの映像のみで姿勢推定を行う．だが，カメラには全身が映るわけではない．また，深度センサーや赤外線も用いない．これらを実現するためにはカメラの映像から姿勢を出力するモデルが必要である．そのモデルを作成するためにディープラーニングを用いる．ディープラーニングを用いて，カメラの映像から関節の角度を出力するような関数を近似する．また，ディープラーニングには大量のデータが必要である．そのため，カメラの映像と深度センサーを用いて測定した，その時々の姿勢を表したデータも必要である．これらから本プロジェクトにおける課題は，全方位カメラの映像から姿勢を推定するモデルの構築，及びその学習に用いるデータセットの作成である．

(※文責: 松元健)

2.2　前期目標

前期のプロジェクト活動の目標は，ディープラーニングに関する知識の取得や，畳み込みニューラルネットワークの構築及びデータの収集である．プロジェクト全体の目標としては，「ゼロから作る DeepLearning」[3] を用いてディープラーニングの知識を取得することとした．また，個別でのディープラーニングに関する学習を行った後にプロジェクトのメンバー全員が理解できているかの確認として，メンバー間で復習のための講義を数週間にかけて行うこととした．開発言語は，データ解析やディープラーニングによく用いられる Python を畳み込みニューラルネットワークのモデルの構成に用いることとした．

それぞれの班の目標としてはニューラルネットワーク班が web 上の画像認識の論文を参考に，一般的な畳み込みニューラルネットワークとプロジェクト特化型の畳み込みニューラルネットワークの 2 種類のモデルの構成，モーションキャプチャや姿勢推定の先行事例の調査および資料の収集，2 種類の畳み込みニューラルネットワークのモデルを構成するための環境の構築とした．

データ収集班は Unity と KinectV2 の接続の確認および設定，データのシステム化や，Unity によって KinectV2 から各フレーム時点での関節の角度を取得するシステムの作成，姿勢推定を行うための全方位カメラ等の機材の確保を目標とした．

(※文責: 三浦幸泰)

2.3　後期目標

前期で構成した一般的な畳み込みニューラルネットワークとプロジェクト特化型畳み込みニューラルネットワークの 2 種類のモデルは，今回のプロジェクトの目標を達成する上でニューラル

ネットワークの層の数が比較的に足りないと判断した．そこで後期のプロジェクト活動の目標は，ニューラルネットワーク班はニューラルネットワークをより深くするために考えられた ResNet と呼ばれるモデル及び画像認識の分野で高い精度を記録しており，また深い層での学習が他のニューラルネットワークに比べて比較的安定する DenseNet と呼ばれるモデルの実装を行うこととした．また，データ収集班から得られるデータセットを用いて DenseNet に学習させ高い精度を出すためのハイパーパラメータの調整，高い精度で学習できているかのモデルの分析をすることを目標とした．

また，前期で用いていた全方位カメラは様々な問題が生じたため今回のプロジェクトの目標の達成には不適であると判断した．そこでデータ収集班は，新たに今回のプロジェクトに適した全方位カメラの確保や，前期に引き続き Unity によって KinectV2 から各フレーム時点での関節の角度のデータを取得し保存するプログラム及びその関節の角度を 3D モデルで再現するプログラムの作成と，モデルを学習させ姿勢推定の実験を行うためのデータセットを作成とした．データの作成においてはニューラルネットワーク班とデータ収集班の合同で行い，被験者は 2 分間のデータ作成のための実験を 50 回行い，計 100 分間のデータセットを作成することとした．

(※文責: 三浦幸泰)

第 3 章　課題解決プロセスの詳細

この章では，知識取得や班ごとの役割担当について述べる．

3.1　技術知識習得

私たちはディープラーニングをプログラムするために，Python を学ぶことにした．Python にはディープラーニングをプログラムするためのフレームワークがあり，初学者でも比較的簡単に取り組めると考えたからである．Python に加え，ディープラーニングの知識習得をする必要もあったため，「ゼロから作る DeepLearning」を用いて学習することにした．この書籍はディープラーニングを Python でプログラミングしていく内容となっている．しかし，すべてを読み，理解するには時間が多くかかってしまうため，6 章まで進めることを決め，学習に取り組んだ．1〜4 章では基本となるニューラルネットワークとその実装を学び，5 章ではディープラーニングの理論，6 章では学習を行う上でのテクニックを学んだ．この書籍を使用し，学んでいく上で，私たちは次のようにして学ぶことを決めた．基本的には個人で書籍を使用し，学ぶ．その中でわからないことがあれば，講義で集まった時にわかる人に質問をする．また，プロジェクトメンバーの一人がディープラーニングに対しての理解が深いことから，5 章の内容については講義形式の学習も行った．さらに夏休みの途中からニューラルネットワーク班の内二人が，ディープラーニング技術者育成講座である DL4US に参加し，Keras というライブラリを用いたディープラーニング講座に参加した．ここで Keras を学んだ．Keras は他のライブラリと比べると，比較的簡単にモデルを記述できるという特徴がある．そのため，DL4US で Keras の使い方を学んだことから，後期に複雑なモデルを構築するきっかけになった．

KinectV2 及び Unity の学習は，公開されているプロジェクトのコードを参考に動かし方を学んだ．また，データ収集時に利用した RaspberryPi も同様に，公開されているプロジェクトのコードを参考に使い方を学び利用した．

(※文責: 松元健)

3.2　データ収集班

この章では，データ収集班での開発について説明する. データ収集班では，モデルの学習に必要なデータセットの作成を行った．

3.2.1　開発

データセット作成のシステムを考案，構築し，実際にデータセットを作成した．データセットには，足踏みをする人間を対象に，同じタイミングで取得した各関節の角度と全方位カメラの画像のセットが必要になる．

設備

角度の取得には KinectV2 を利用した．全方位カメラには最初，PIXPRO SP360 と Ricoh Theta S を用意したが，前者は無線接続の機能制限の厳しさ，後者は撮影範囲の不足から不適と判断し，ウェブカメラ機能を搭載している PIXPRO SP360 4K を利用することになった．また，ウェブカメラ機能の無線化のために Raspberry Pi で画像を送信する予定だったが，スペック不足による送信画像のノイズが酷かったため，装着者がノート PC をリュックで背負って画像を送信した．二つのデバイスの制御には，三次元座標の処理に優れる Unity を使用した．また，Unity はフレームレート設定することで一定間隔で処理を繰り返すことができるため，各機器の更新タイミングを統一する目的でも利用した．

技術

全方位カメラの制御においては，以下のように OpenCV でカメラ画像の取得と縮小を行い，Python 標準のソケット通信モジュールで画像を送信した．

1. 全方位カメラを認識し，取得する画像サイズを指定する．
2. 画像のリクエストが来たら，全方位カメラから画像を取得する．
3. OpenCV のリサイズ機能で画像を縮小する．
4. 画像を jpeg 形式のバイト列に変換して送信する．
5. リクエストが来るたび 1 から 3 を繰り返す．

Unity 側では，以下のような処理を実装した．

1. 全方位カメラのサーバから画像を取得する．
2. KinectV2 から関節の角度を取得する．
3. 1，2 をフレーム更新の度に行い，配列に格納していく．
4. 指定した時間が経過し次第，配列内の全ての画像を保存する．
5. 関節の角度は対応した画像のファイル名を付随させ，CSV 形式で保存する．

また，学習結果の確認用に，関節の角度を CSV から 3D モデルに再現するプログラムも作成した．

角度の表現について

今回の実験では，角度の表現にクォータニオン [9] を用いた．クォータニオンは x，y，z，w の 4 つの要素を持ち，x，y，z で回転軸を決め，w でその回転軸に沿って回転角度を決める．この表現は 3D グラフィックス技術で多く用いられ，Unity で扱う上でも推奨されているため，今回の実験に採用した．

データセット作成

データセットの対象は全方位カメラを胸に装着し，さらにカメラを接続したノート PC をリュックで背負う．準備が完了し次第，対象はその場で足踏みを開始し，Unity を操作してデータセットの取得を開始する．2 分間のデータセットを 50 回作成し，計 100 分間のデータセットを作成した．図 3.1) は, データセットの作成方法の流れを表したものである.

(※文責: 坂下夏槻)

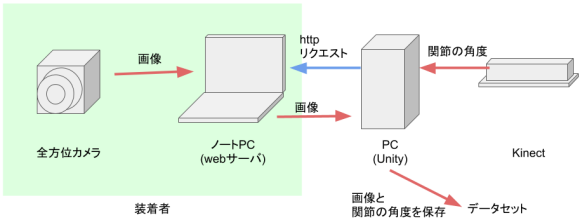

図 3.1　データの作成方法

3.2.2　課題

有効なデータセットを作成するためには，可能な限りノイズとタイミングの遅延を少なくする必要がある．

全方位カメラの装着方法

全方位カメラの装着方法として，ウェアラブルカメラ用の胸部ベルトを用いた．しかし接続用のコードの位置関係から，ハーネスに付属していた固定用のボルトでは長さが足りなかったため，新たに別のボルトを購入した，

画像サイズ

送信する画像のサイズが大きすぎると，送信に時間がかかり遅延が大きくなってしまう．しかし SP360 4K の仕様上，全方位画像を取得する際に大きなサイズの画像しか取得できなかったため，送信前にモデルへの入力に必要なサイズまで縮小する処理を挟んだ．具体的には，1440px × 1440px（図 3.2)）から 128px × 128px（図 3.3)）まで縮小した．これによって遅延の問題は解決したが，画像の送信に Raspberry Pi を利用した場合，送信されたデータにノイズが発生することがわかった．このノイズについて，送信時の縮小処理に対する Raspberry Pi のスペック不足が原因としてあげられた．そのため，縮小処理を行うのに十分なスペックをもっているノート PC を用いることで対処した．

画像の欠損

実験中，およそ 1 分に 1 フレームの頻度で画像の送信に失敗することがあった．これについて，ウェブカメラと Unity の更新タイミングのズレ，縮小処理が間に合っていない可能性があげられた．ニューラルネットワーク班と相談し，前フレームの画像で欠損を補完するようにプログラムを修正した．

KinectV2 の不具合

作成したデータセットの関節の角度を再現すると，左足関節の計測の精度が低いことがわかった．KinectV2 を取り替えると精度が改善したので，新しい KinectV2 を使用してデータセットを作り直した．

(※文責: 坂下夏槻)

図 3.2　縮小する前の画像

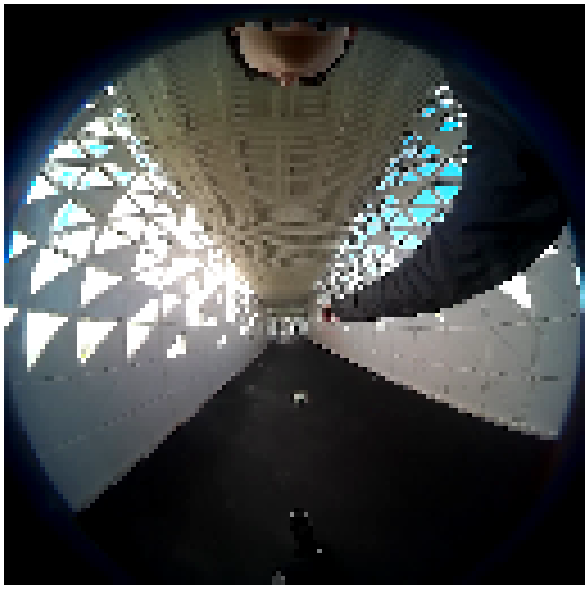

図 3.3　正常に縮小した画像

3.3　ニューラルネットワーク班

この章では，ニューラルネットワーク班で行った開発について説明する. ニューラルネットワーク班は名前の通り，主にニューラルネットワークの実装を行ってきた.

3.3.1　ニューラルネットワーク概要

ここではニューラルネットワークの概要について説明する．ニューラルネットワークについて詳しく学びたい場合は [4] や [3] などの書籍がある．ニューラルネットワークとは，関数を近似するための構造の一つで，隠れ層に十分なニューロンが存在することを条件とすれば，任意の関数を任意の精度で近似することができる (故にニューラルネットワークは万能近似器などと呼ばれたりもする).

ここから具体的な数式を用いてフィードフォワードな N 層のニューラルネットワークについて説明する．まず，以下に現れる記号について説明する．第 n 層第 i ニューロンへの入力の総和を a_i^n，第 n 層第 i ニューロンの出力を y_i^n と表す．また，第 n 層第 i ニューロンから第 n+1 層第 j ニューロンへの重みを w_{ji}^n と表す．その時それぞれの記号の関係は以下のようになる.

$$a_j^n = \sum_{i=0}^{m} w_{ji}^n y_i^{n-1} + w_{j0}^n \tag{3.1}$$

$$y_j^n = f(a_j^n) \tag{3.2}$$

ここで，w_{j0}^n は n 層第 j ニューロンへの接続のしきい値を表す．また，f は活性化関数と呼ばれ，重み付き総和 a を非線型変換するために用いられる．代表的な活性化関数として，シグモイド関数

$$f(x) = \frac{1}{1 + \exp(-x)} \tag{3.3}$$

などがあげられる.

例えば今 2 層のニューラルネットワークを構築するとき，数式で表すと

$$y_k^2 = f(\sum_{j=0}^{u} w_{kj}^2 f(\sum_{i=0}^{v} w_{ji}^1 x_i))) \tag{3.4}$$

となる．ただし，ここで x は入力で，しきい値は省略した．

ここまで，基本的なニューラルネットワークの構造を数式で表してきたが，ここでの問題は与えられた入力 x に対して最適な y を返すためにはどのようにすれば良いかとなる．具体的には，式 (1) の重み w をどのように学習するかが問題となる．次はニューラルネットワークの学習についてを簡単に説明する．特にここではプロジェクト学習で用いた教師あり学習についてのみ説明する．また，具体例として，回帰問題についてのみ説明する．回帰問題とは，ある入力 X に対して実数値の出力 $f(X) = Y \in R$ を返す問題で，例えば身長を入力として予想される体重を返したり，株価を予測したりするような問題は回帰問題と呼ばれる．B グループが行なった映像から関節の角度を予測する問題も回帰問題となる．教師あり学習では，与えられたデータセットを用いて予測器を学習させ，観測していない入力から正しい出力をする予測器を構築する．具体的には，損失関数と呼ばれる関数を最小化するような予測器 f を求める．代表的な損失関数として，二乗和誤差

$$L = \frac{1}{2}\sum (f(x_i) - \hat{y}_i)^2 \tag{3.5}$$

などがある．これらの損失関数を最小化するための方法の一つとして，勾配降下法と呼ばれる方法が存在する．勾配降下法とは，損失関数 L が減少するような重み w の勾配を計算し，その勾配を徐々に降下することによって損失関数を 0 に近づけるといった手法で，ニューラルネットワークの最適化ではこれが基本となっている．数式で表すと，

$$w_{ji}^{t+1} = w_{ji}^t - \epsilon \frac{\partial L}{\partial w_{ji}^t} \tag{3.6}$$

となる．ただしここで w_{ji}^t の t は t 回目に更新されたことを表す．また，ε は学習率や学習係数と呼ばれ，普通 $0 < \epsilon < 1$ の間で任意に選ばれる．ニューラルネットワークの最適化については他にも誤差逆伝播法や汎化性能についての議論，最適化の速さについての議論など様々あるが，ここでは割愛することにする．

subsection 畳み込み層について

ニューラルネットワークの層の種類には様々あるが，畳み込み層と呼ばれる層は本グループで常に使用していたため，説明する．

畳み込み層は特に画像認識の問題において頻繁に使われている．畳み込み層の特に大きな利点は，画像の中の物体の位置に対してロバストになることで，これによって上記で説明したニューラルネットワーク（これは畳み込みニューラルネットワークに対して全結合ニューラルネットワークなどと呼ばれる）よりも大幅に画像認識の精度を向上させた．畳み込み層の計算を数式に表すと，

$$a_{ij}^n = \sum_{s=0}^{u}\sum_{t=0}^{v} w_{st}^n y_{(i+s)(j+t)} \tag{3.7}$$

となる．ここで，w_{st}^n は n 番目の重み行列 (カーネルと呼ぶ) の (s,t) 要素を参照していて，y_{ij} は前の層の出力 (行列表現であることに注意) を表している．このように，ある一つの層に w_{st}^n というカーネルをいくつも用意して，二次元画像を局所的に計算していくことで物体の画像内の平行移動や回転に対して頑健なニューラルネットワークを構築することができる．この計算以降は全結合ニューラルネットワークと同様に，活性化関数 $f(a_{ij}^n)$ を計算し，また畳み込みの計算をするということを繰り返していく．

3.3.2 開発

前期開発環境

前期でははじめにプロジェクトで使用するコンピュータを用いて環境構築を行った．具体的には最初に Ubuntu17.10 をインストールした後，Anaconda と呼ばれるデータサイエンス向けの Python パッケージをインストールした．そして，本グループではニューラルネットワークの実装に Python を使うこととした．Python とは汎用的なプログラミング言語だが，機械学習やデータサイエンス，科学計算向けに特化されたパッケージが特に多いという特徴がある．この特徴から，今回の目標達成のためには Python を使うことが一番であると判断した．Python のバージョンは 3.6.5 とした．その後，私たちは TensorFlow と呼ばれる機械学習等に用いられるテンソル計算に特化したパッケージをインストールすることにした．数ある機械学習ライブラリから TensorFlow を選択した理由は，このパッケージを使い慣れているメンバーがいたからである．また，当時 TensorFlow に関するとても素晴らしい書籍が出版されたということも選択にあたっての理由の一つである．ここで，私たちは GPU を用いて TensorFlow を動かすために，CUDA および cuDNN をインストールした．CUDA とは，GPU を用いた並列計算を提供するプラットフォームで，cuDNN とは GPU を用いた Deep Neural Network の開発用のライブラリである．CUDA と cuDNN のバージョンはそれぞれ 9.2 と 7.1 である．その後 tensorflow-gpu をインストールして環境構築を終えた．tensorflow-gpu のバージョンは 1.8.0 とした．

畳み込みニューラルネットワークの実装

環境構築を一通り終えた後は一般的な畳み込みニューラルネットワークを実装することにした．畳み込みニューラルネットワークとは，特に画像認識の分野で活躍しているモデルである．私たちは 7 層の畳み込みニューラルネットワークを実装した．しかし，姿勢推定関連の先行研究を調べた結果，7 層では明らかに足りないことがわかったため，別のニューラルネットワークを実装することにして，前期でのニューラルネットワーク班の活動は終了した．

後期開発環境

後期では，前期で挙げられた問題点を改善する前に，開発環境に Keras というライブラリを追加し，以降は Keras を用いてニューラルネットワークを実装することにした．Keras とは，TensorFlow 上で実行される高水準なニューラルネットワーク用のライブラリである．TensorFlow から Keras に変えた理由は単純に Keras の方が実装が簡単に済むからである．前期に挙がった層の数が少なすぎるという問題点を踏まえ，今後複雑かつ深い層のニューラルネットワークを実装することになると予想されることから，より実装が簡単な Keras を利用することにした．また，夏休みを利用して Keras を使用したディープラーニングの仕方を学ぶ機会があり，プロジェクト学習でも十分使えると考えたということも理由の一つである．Keras のバージョンは 2.2.2 とした．

ResNet 実装

Keras をインストールした後，私たちは実際に層を増やすために ResNet[6](モデルの概形は図 3.4) と呼ばれるモデルを実装した．ResNet とは，ニューラルネットワークをより深くするために考えられたモデルである．以前から，浅いニューラルネットワークよりも深いニューラルネットワークの方が比較的良い性能を発揮することが知られていたが，ただ闇雲に深くするだけでは学習が進まないという問題があった．しかし ResNet はその問題をある程度緩和する能力を持ってお

り，ResNet の有用性が示されて以来，ResNet の構造を利用した様々なニューラルネットワークモデルが提案されている．私たちは 26 層の ResNet を実装した．また，RAM を節約するために Global Average Pooling[5] を利用した．一般にニューラルネットワークのモデルでは，最後は全結合層とするが，特徴マップをベクトルに変換して全結合層に入力するという従来の方法ではパラメータの数が膨大になる．一方，Global Average Pooling では一つの特徴マップを平均し，スカラーとして扱う．全ての特徴マップを平均した値を並べることによって特徴マップの数と同じ次元のベクトルを入力となる．この手法を使用すれば精度を維持または改善しつつパラメータの数を大幅に削減することができる．

DenseNet 実装

ResNet を実装した後，実験結果から私たちはさらに層の深いモデルを構築する必要があると考えた．しかし，これ以上層を増やすことは RAM の都合上不可能であったため，DenseNet[7](モデルの概形は図 3.5) と呼ばれるモデルを実装することにした．このモデルは ResNet に比べて一層あたりのパラメータの数を抑えることができるため，より深い層のモデルを構築することができる．私たちは 72 層の DenseNet を実装した．

また前期から後期にかけて全方位カメラから得られたデータをモデルに入力する際の形に変換するパイプラインも実装した．具体的には，全方位カメラから得た映像を 32bitfloat 型の配列に変換し保存するスクリプトを実装した．

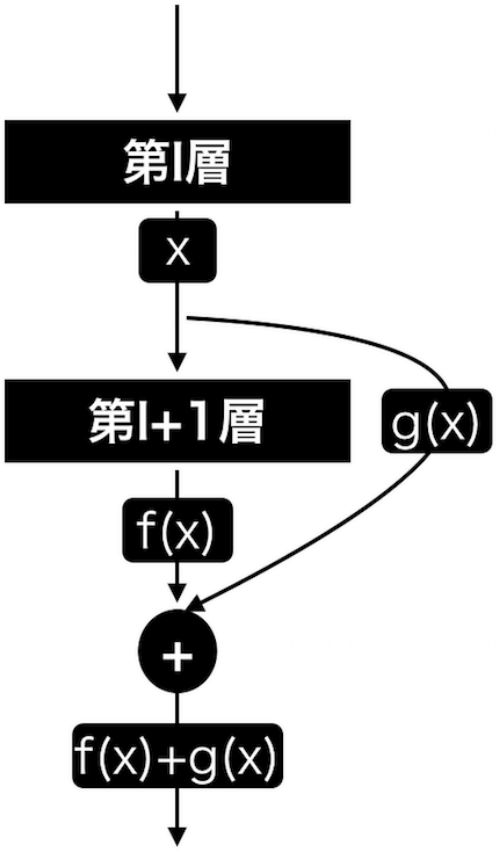

図 3.4　ResNet の第 l 層から第 l+1 層の構造. ただし f(x) および g(x) はそれぞれ非線形変換と線形変換.g(x) のパスが skip connection と呼ばれる.

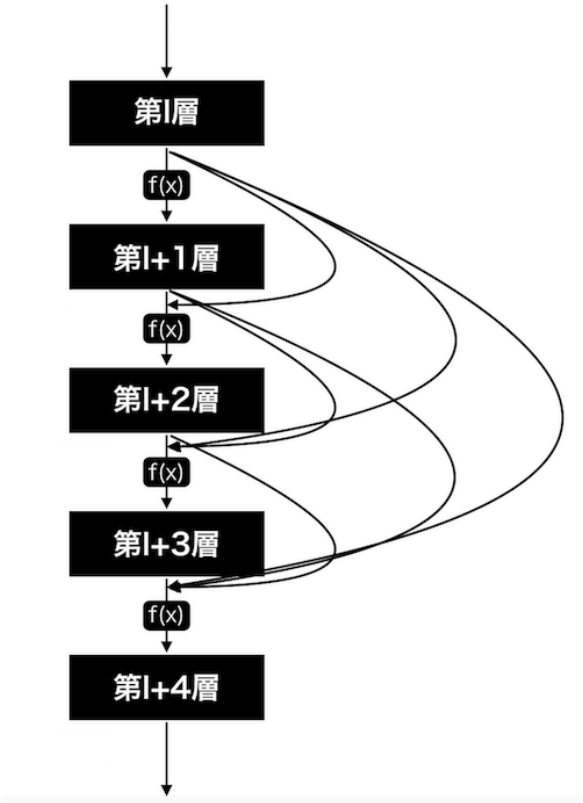

図 3.5　DenseNet の第 l 層から l+4 層の構造. ただし f(x) は非線形変換. ResNet とは違い, それぞれの層に稠密にパスが存在している. 自身の出力以外にパスがない層も存在する.

(※文責: 和田孝喜)

3.3.3 課題

ここでは，ニューラルネットワークを実装する際に問題となったことについて説明する．
ニューラルネットワーク班は主に二つの課題を解決する必要があった．

1. 勾配消失問題 [8]
2. モデルの表現力

勾配消失問題

勾配消失問題とは，ニューラルネットワークの層を深くすると起きる問題で，学習の際に前半の層へ流れる情報が少なくなり，学習が極端に遅くなったり出来なくなったりするというものである．一方で私たちの目標を達成するためには層をより深くする必要があるため，7 層の一般的な畳み込みニューラルネットワークとは別に勾配消失問題に対処しつつ層を大幅に増やすことが課題となった．

私たちは勾配消失を起こさずに層を増やすために ResNet[6] を実装した．ResNet は skip connection と呼ばれるパスをモデルに組み込むことで勾配消失を抑えている．具体的には，第 l 層の出力を第 l+1 層の出力に線形変換の後に加算している．この skip connection を用いて私たちもニューラルネットワークの層を 26 層まで増やすことができた．ここでなぜ 26 層なのかというと，ニューラルネットワークの学習をする際，計算に必要な RAM はすべて一番最初に確保する必要があり，26 層よりも層を多くするとそれ以上 RAM を確保することができずにエラーを出力するからである．勾配消失問題への対処法は他に，活性化関数の選び方やバッチ正規化などが挙げられる．

モデルの表現力

モデルにおける表現力とは，入力と出力の関係をどの程度よく表現できるかを表す言葉である．私たちのタスクでは，映像という奥行きの情報を持たないデータから 3 次元の関節角度を予測する必要があり，また映像自体の質も良いものではないため非常に高い表現力を持つニューラルネットワークのモデルを実装する必要があった．しかし，コンピュータの RAM には上限があるためただ大きいモデルを構築するだけでは学習時にモデルが RAM にすべて読み込まれず，そもそも学習することができない．こういった制限がある中でどれだけの表現力があるモデルを実装することができるかが二つ目の課題となった．

私たちは RAM の消費を抑えつつ層を増やす為に DenseNet[7] を実装した．DenseNet は ResNet をヒントに考えられたモデルである．ResNet との違いは，パスを一つ前の層からだけでなく，以前の層全てから接続することによってさらに流れる情報量を増やしていることが特徴である．このモデルは ResNet と同じ数の層を実装し，同じくらいの精度を出す時でも，ResNet より 50% 近くメモリの消費を抑えることができる．そのような利点から，私たちは DenseNet を実装することにした．さらに，効果が薄いであろう非線形変換を極力減らすことによって，最終的に 72 層の DenseNet を実装することができた．

(※文責: 和田孝喜)

第 4 章　発表

この章では中間発表および最終発表について述べる．

4.1　中間発表

4.1.1　中間発表に関する活動

2018 年 7 月 13 日，公立はこだて未来大学にてプロジェクト活動の中間発表が行われた．中間発表へ向け，ポスターやプレゼンテーション用のスライド資料の製作に B グループ全員で取り組んだ．前期の進捗状況や後期の目標などについてのポスターを作成し，それらをより詳細に説明できるようにスライドを作成した．中間発表では keynote を用い，A 班 B 班合同のポスターと B 班のみでのポスターの計 2 枚を作成した．作成したポスターを図 4.1 に示す，スライドは google スライドを用い，A グループと B グループの 2 つのスライドが統合されたものを作成した．スライドを作成する際には，各々が担当した作業についてのスライドを作成する方向性で決定した．中間発表の時点では実験まで手をつけることができておらず，発表する内容を考えることに時間を使ってしまい，担当教員からレビューをいただくのが遅れてしまったため，スライドの作成も遅くなってしまった．その結果，メンバー全員が発表練習を行わずに中間発表に臨む形となってしまった．

図 4.1　中間発表ポスター

4.1.2 中間発表における評価シートの内容と考察

中間発表の際，傍聴者６３人に発表評価シートを記入してもらった．その結果を表 4.1 に示す．

表 4.1 中間発表での評価

評価	発表技術	発表内容
10	7	8
9	10	7
8	18	17
7	16	15
6	4	5
5	2	3
4	3	0
3	0	2
2	0	0
1	0	0
無回答	3	5
平均点	7.7	8.0

また，中間発表ではいくつか意見や質問が寄せられた．発表技術についての意見と質問を以下に示す．

- 発表が聞き取りやすかった
- 一つ一つ説明があり、わかりやすかった
- 図が多くて見やすかった
- 下敷きが欲しかった
- 専門用語の説明を増やして欲しい

次に発表内容についての意見と質問を以下に示す．

- タスクが面白い
- 目的や手段がはっきりしていた
- とてもわかりやすかった
- 先行研究との違いを明確にしたほうがよい
- カメラを胸につけることによるメリットを書くべき
- CNN の説明がないので、何を行う装置なのか技術的な説明が欲しかった
- 内容が専門的なため、もう少し補足や説明があると良い
- 内容がよくわからない

発表技術については，発表練習を行わなかったにも関わらず，声の大きさやスライドについての高評価が多かった．事前に先生からアドバイスやフィードバックを頂いたことで，メンバー全員の発表技術やスライドの構成が良くなったと考えられる．発表内容については，挑戦する課題が面白いという評価が多かったが，発表内で専門用語が頻出したため，それがわかりにくいという意見も

多かった.

(※文責: 川村一世)

4.2 最終発表

4.2.1 最終発表に関する活動

2018 年 12 月 7 日，公立はこだて未来大学にてプロジェクト活動の最終成果発表が行われた．発表へ向け，ポスターやプレゼンテーション用のスライド資料の製作に B グループ全員で取り組んだ．1 年間の成果などについてのポスターを作成し，それらをより詳細に説明できるようにスライドを作成した．最終発表では keynote を用い，A 班 B 班合同のポスターと B 班のみでのポスターの計 2 枚を作成した．中間発表での反省を生かし，余裕を持って 1 ヶ月前に作成を始めたが，完成したのが最終発表前日となってしまい，計画性の欠如を実感した．作成したポスターを図 4.2 に示す．スライドは google スライドを用い，A グループと B グループの 2 つのスライドが統合されたものを作成した．スライドを作成する際には，中間発表のスライド構成を参考にし，各々が担当した作業についてのスライドを作成する方向性で決定した．しかし，実験が終了するのが遅く，実験結果の部分のスライドの作成作業に手をつけられなかったため，スライドの完成が大幅に遅れてしまい，担当教員からレビューをいただく時期も遅くなってしまった．また，担当教員からいただいたレビューを反映させたスライドを完成させることができたのが最終発表の直前となり，メンバー全員が発表練習を行わずに最終発表に臨む形となってしまった．

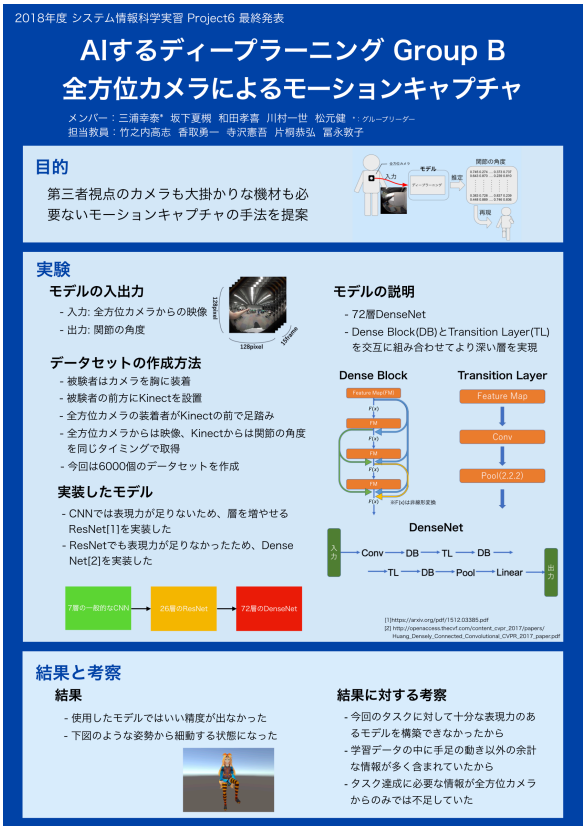

図 4.2　最終発表ポスター

4.2.2　最終発表における評価シートの内容と考察

最終成果発表の際，傍聴者 43 人に発表評価シートを記入してもらった．その結果を表 4.2 に示す．

表 4.2　最終発表での評価

評価	発表技術	発表内容
10	6	6
9	6	3
8	13	21
7	14	9
6	3	3
5	0	0
4	1	0
3	0	0
2	0	0
1	0	0
無回答	2	3
平均点	7.9	8.0

また，最終発表でもいくつかの意見や質問が寄せられた．発表技術についての意見と質問を以下に示す．

- スライドがわかりやすかった
- どんな活動をしてきたのかわかりやすかった
- 聴衆の目を見て発表できていた
- 声が聞き取りやすかった
- 重要な部分は声を大きくして強調していた
- 画像が挿入されていてイメージしやすかった
- 発表が後半にいくに連れて，声が小さくなっていった
- 専門用語が多く，理解に時間を要することがあった
- デモンストレーションがあった方がわかりやすい
- 文字数が多すぎるスライドがあり，読みにくかった
- もっとイラスト等を効果的に使うべき

次に発表内容についての意見と質問を以下に示す．

- 目標が面白い
- 知識がなくてもわかりやすい
- しっかり考察されていて良い
- 成果が見える形で発表されていてよかった
- 目標を設定した背景をもっと詳しく知りたい
- 設定した目標が大変だったと思う

- ディープラーニングについての説明がもっと欲しい
- モーションキャプチャの利点がわからない
- スケジュール管理をしっかりするべきだった
- ひざの設定を再考するべきだった
- データを収集する際の環境が複雑すぎた

発表技術，発表内容ともに，発表練習を行わなかったにも関わらず，前期の評価の平均点を上回っていた．発表技術については，声の大きさやスライドについての高評価が多かった．事前に先生からアドバイスやフィードバックを頂いたことで，メンバー全員の発表技術やスライドの構成が良くなったと考えられる．しかし，専門用語が多くてわかりにくかったという意見もあった．発表内容については，ディープラーニングについての説明が足りていなかったという意見が多数あった．技術・内容共に中間発表での意見を反映できていなかったと言えるだろう．

(※文責: 川村一世)

第 5 章　実験および結果

この章では，実験環境およびその実験の結果について説明する．

5.1　実験

5.1.1　実験環境

コンピュータのスペックはメインメモリ 64GB，プロセッサは Intel Core i7-7700k，GPU は GeForce GTX1080Ti である．OS は Ubuntu17.10 を使用した．

5.1.2　モデルの入出力

モデルの入力は映像とし，出力は quaternion の形式の関節角度とした．具体的な numpy-ndarray の shape は以下の通りである．

入力　[15，128，128]([フレーム数，縦 pixel 数，横 pixel 数])
出力　[15，44]([フレーム数，各関節数の 4 倍])

それぞれを言語化すると，入力の一次元目は 15 フレーム分，二次元目および三次元目はそれぞれ映像の縦と横のピクセル数である．出力の一次元目は 15 フレーム分，二次元目は 11 の関節 (腰，肩，ひじ，手首，足の付け根，膝) の角度を quaternion で表現しているため 44 となっている．つまり，出力は一要素あたり [-1.0，1.0]（[a，b] は閉区間）の値をとる 15×44 の行列となる．

5.1.3　モデルの学習

私たちは ResNet およびいくつかの DenseNet を学習させた．全てにおいて epoch 数は 500 とした．データは 6000 個集め，5000 を学習用データ，500 を検証用データ，残りの 500 をテスト用データとした．ResNet の学習時間は約 14 時間，DenseNet の学習時間は約 41 時間となった．

5.1.4　仮実験

私たちは本実験を行う前に，50 個のデータを使い仮実験を行い，モデルの細かい調整を行った．モデルの出力は quaternion を利用することから一要素あたり [-1.0，1.0] となるため，出力層の活性化関数として hyperbolic tangent 関数を使用したが，一切 loss が低下しなかったため恒等関数に置き換えた. その結果，loss が低下するようになった．また，計算量が多いことから学習にかかる時間が長くなってしまい，十分な実験を行うことができなくなる可能性が挙げられたため，pooling の範囲や stride の幅を調整することによって学習時間の短縮を図った．

5.1.5　結果の検証方法

私たちは実験結果に対して二種類の検証をした．一つ目は tensorboard を利用して誤差を定量的に測る方法，二つ目は学習済みのモデルが出力した関節のデータを Unity で読み込み，Unity

上のモデルをその関節データに従って動かしてどれだけ自然かを主観的に判断する方法である．tensorboard とは TensorFlow や Keras で実装したモデルの様々なデータを可視化するためのライブラリである．tensorboard ではトレーニングデータと検証用データそれぞれの平均絶対値誤差と Loss を epoch 毎にプロットして監視を行った．

(※文責: 和田孝喜)

5.2 ResNet の結果

ResNet を使って学習した結果，loss が収束した．loss が収束するとは図 5.1 のように loss の曲線がほぼ一定の値に収束することである．また，実験を行った全てのモデルにおいて同様の現象が見受けられたが，各クォータニオンの誤差の値が大きいことから，ResNet では上手く学習できていないことがわかった．さらに，loss の値が収束したため，現状これ以上学習を続けても精度の向上は見込めない．そのため，学習に使うデータまたはモデルに対しての改善が必要である．

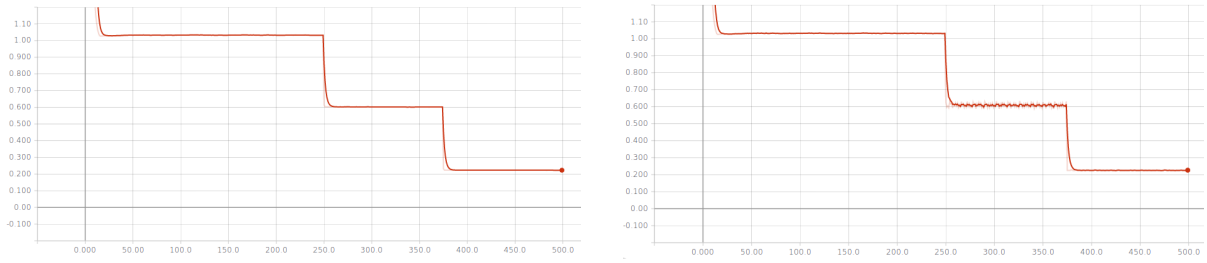

図 5.1 loss が収束している状態のグラフ

5.3 全身で学習した DenseNet の結果

ResNet と比較して，層を深くすることのできる DenseNet を用いて学習した．だが，ResNet と比較して精度が少し改善されたが良い精度ではなかった．出力の値を分析した結果，手首の精度が著しく低いことがわかった．原因として考えられることは，Kinect から取得されたデータの精度が低いということである．期待する出力として与えているデータの精度自体が低いため，ネットワークから出力されるデータの精度も低くなる．また，Kinect で取得しているデータを可視化して，動かしている映像を見たときに手首が安定せず，常に回転したり，本来動かない方向に曲がっていたことから，手首の関節の値を安定して取れていないことがわかった．そのため，歩行というタスクにおいて，手首の動きは重要なパラメータでないと判断したため，予測するパラメータから手首を除いて学習することとした．こうすることで，改善することの難しい手首の予測に使用していたパラメータを他の関節の予測に使用できると考えた．これにより，精度が改善されると考えた．

5.4 手首を除いて学習した DenseNet の結果

手首を除いて学習した DenseNet は手首を含んで学習したモデルと比較して，少し精度が改善された．図 5.2 は各関節の誤差をグラフで表したもので，手首を除いて学習することで改善が見られたことがわかる．ここから，学習の難しいと判断されるパラメータを除いて学習することで，他の関節の精度が少し向上することがわかった．だが，全体的な精度としては依然低いままである．現在の精度では，姿勢を推定できていない．著しく精度の低いパラメータを除いても，精度があまり

改善されていないため，モデルの表現力を増す必要があると考えた．モデルの表現力を増すことで，一つの関節あたりに使用できるパラメータを増やすことができる．一つの関節あたりのパラメータ数が増えることで，精度が向上すると考えた．そこで，各関節あたりに使うパラメータを増やすために関節を上半身と下半身に分割する．これにより，各関節あたりのパラメータ数が上半身は 1.8 倍，下半身は 2.25 倍になる．

5.5　関節を上半身と下半身に分けて学習した DenseNet の結果

上半身と下半身に関節を分割して学習した．図 5.3 は手首を除いて学習したモデルと，下半身のみで学習したモデルの誤差を比較したグラフである．この図から，関節を分割した場合も精度が改善されることはなく，むしろ悪化していることがわかる．

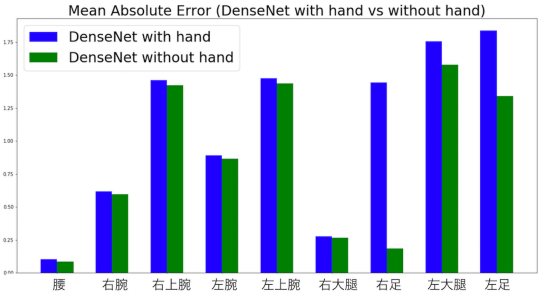

図 5.2　手首を含んで学習したものと手首を除いて学習したものの出力の誤差の比較

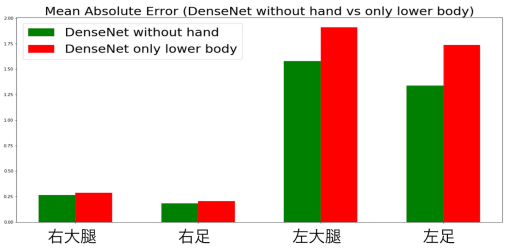

図 5.3　手首を除いて学習したモデルと下半身のみで学習したモデルの誤差の比較

(※文責: 松元健)

第 6 章　考察

数回の実験のいずれにおいても十分な結果が得られなかったため，その原因を考察した．

6.1　入力に含まれる不要な情報

私たちの取り組んだタスクでは，モデルに入力として与えるカメラの画像に手足以外の不要な情報が多く含まれている．先行研究 [2] ではカメラが主に全身を映している．それに対し，今回の方式では部分的にしかカメラに体が映らない．私たちのモデルへの入力画像に対する人の映っている領域は，全体の約 17% である．それに対し，先行研究 [2] では，入力画像に対する人の映っている領域は，約 30% である．これは私たちのモデルでは，入力画像から得られる人の情報が先行研究 [2] と比べておおよそ 45% 少ないということである．そのため，カメラの装着者の身体以外の領域の不要な情報にモデルが反応してしまい，思うように学習が進まなかったと考える．したがって，カメラの映像を直接入力とするのではなく，入力領域内に映っている人の部位を見つけ，マークするような前処理が必要だったのではないかと考える．そうすることで，人の映っていない画面領域に対してモデルが反応することもなく，各関節に対してのみモデルを反応させることができるため，実現すると考える．

6.2　タスクの大きさ

私たちは学習に使う関節の吟味やモデルの選択など，複数の実験の結果から，タスク自体が難しいのではないかと考えた．精度の著しく低い手首を予測の対象から除いて，一つの関節あたりに使用するパラメータを増やして学習した場合には精度が改善された．しかし，ボーンに当てはめて可視化した関節の角度を観察すると，精度が不足しているため歩いているようには見えず，モデルに表現力が足りないと考えた．次に，上半身と下半身に分けて学習することで，手首を取り除いて学習させた時と同様に，各関節あたりに使用できるパラメータを増やし，モデルの表現力を増して学習させた．これによって，手首を取り除いて学習した場合には精度が改善されたため，同様に精度が改善されると考えたが，出力を分析した結果，精度は改善されることなく精度が悪化した．このことから私たちは，モデルの表現力に問題があると考えていたが，モデルの表現力を増しても精度が向上しなかったため，タスク自体が難しいのではないかと考えた．

6.3　多重共線性の問題

私たちのタスクでは，入力画像から対象者の関節の角度を回帰モデルを用いて予測するものである．私たちが本タスクを実現できると考えた理由の一つとして，右腕が前方にある時には左足も前方にあり，左腕，右足は後方にあるという相関が考えられる．だが回帰モデルには，要素同士に強い相関がある場合には上手く予測することができなくなってしまう，多重共線性という問題がある．私たちは，要素に対して強い相関があることで予測できると考えたが，強い相関があることによって多重共線性の問題が発生し，上手く予測できないと考えた．また，私たちのタスクと比較して，先行研究 [2] では，予測を行うプロセスが異なる．先行研究 [2] では，関節の角度ではなく，画

像に映る人間の関節の位置を予測するというプロセスを経て予測する．そのため，私たちのプロセスとは異なり，関節同士の強い相関による多重共線性の問題が発生することがない．それに対し，私たちのタスクでは関節同士の強い相関を用いて予測するため，多重共線性の問題が発生してしまい，高い精度で予測をすることができなくなってしまうことが考えられる．

(※文責: 松元健)

第 7 章　今後の展望

この章では，結果及び考察から話し合った，仮説や未実施の追加実験について述べる．

7.1　部位ごとの学習

実験で行った上半身と下半身に分けた学習では十分な結果が得られなかった．そのため，一つの部位だけを出力するモデルを作成し，さらに学習にパラメータを十分に使える状態で，学習が進むかを確認する必要がある．

7.2　関節の角度の簡略化

今回の実験では，全ての関節があらゆる方向に 180 から-180 度回転できる設定で学習した．しかし人間の関節には，膝などの一定の軸にしか回転しない部位が存在する．実際に，全てのデータを通してほとんど角度が変動しない関節の軸が存在した．また，回転の範囲も一般的に 180 度ほどが限界である．このことを考慮して，それぞれの関節の角度の最大値を決め，その範囲の値の変化を学習させることで，タスクの大きさを減らすことができると考えられる．

7.3　角度の表現方法の変更

関節の角度の表現には x，y，z で表すベクターと，x, y, z, w で表すクォータニオンがあり，今回の実験ではクォータニオンを使用した．しかし，ベクターに比べ，クォータニオンは要素が 1 つ多い点，各要素が同質でない点で学習に適さなかった可能性がある．関節の角度をベクター表現に直したデータセットと，それに対応したモデルを再構成し，クォータニオンで学習したものと比較して，どちらが関節の角度を表現するのに適しているか調べる必要がある．

7.4　クォータニオンの特性に合わせたモデルの構築

前述の通り，クォータニオンの各要素は同質でなく，実際の回転の大きさを決めるのは w 値である．今回の実験の結果では回転そのものの大きさが小さくなっており，w 値の変化が小さ過ぎることがわかる．そのため，w 値を効率的に学習させるためのなんらかの修正を加えることで，出力が改善する可能性がある．

7.5　より精度の高いデータセットの作成

KinectV2 は赤外線センサーによって関節の角度を検知しているため，精度の信頼性に難がある．装着式のセンサーなど，より精度の高いモーションキャプチャデバイスでデータセットを作り直すことで，より精度が改善する可能性がある．

7.6　より精度の高いデータセットの作成

KinectV2 は赤外線センサーによって関節の角度を検知しているため，精度の信頼性に難がある．装着式のセンサーなど，より精度の高いモーションキャプチャデバイスでデータセットを作り直すことで，より精度が改善する可能性がある．

7.7　リアルタイムモーションキャプチャ

今回は CSV の出力にとどまったが，モデルの出力を Unity に直接送信すればモーションキャプチャデバイスとしてコンテンツに利用可能である．モデルの精度さえ改善できれば，既存のモーションキャプチャと十分差別化できる手法だと考えられる．

(※文責: 坂下夏槻)

第 8 章　グループメンバーの活動

この章では，グループメンバーの活動について述べる．

8.1　三浦幸泰

前期ではディープラーニングについて理解を深めるために「ゼロから作る Deeplearning」を用いて学習し，理解を深める上で Python の学習も並行して行った．また，csv ファイルから時間型の情報と画像の関節の角度を分別するシステムを Python で構築を行った．後期では，ローカル上にある csv ファイルを Unity から受信するシステムの構築の支援を行った．また，モデルに学習させるためのデータの作成を行った．反省点としては，プロジェクトのグループリーダーとしての役割をしっかりと果たすことが出来なかったことと，まだまだ学習不足なところがあり今後もディープラーニングについての知識を深めていく必要がある．

(※文責: 三浦幸泰)

8.2　坂下夏槻

前期は全員で「ゼロから作る DeepLearning」を読み，ディープラーニングの概論を学習した．その後，テーマの選定とグループ分けを行い，主にデータセット作成のためのシステム構築を担当することに決まった．特に全方位カメラの接続は，仕様の確認不足が多く苦戦した．また，ノイズや画像の抜けなどの障害が発生したため，対応にメンバーに協力してもらった．システムが完成した後は，メンバーで協力してデータセットの作成に努めた．

(※文責: 坂下夏槻)

8.3　和田孝喜

私は本グループの目標を達成するための畳み込みニューラルネットワークの提案をした．また，Ubuntu や CUDA，cuDNN，tensorflow-gpu 等のインストールを行い，開発環境の構築を行った．また，プロジェクト全体でディープラーニングについて学ぶ際は，私が主導してメンバーに教えた．その際，理解しづらいであろう内容はできるだけ全員が理解できるように，記号で抽象的に式を表すだけでなく具体的に計算してイメージさせることができるようにした．さらに，畳み込みニューラルネットワークの実装を TensorFlow を用いて行った．そしてその実装で姿勢推定のタスクをすることの問題点として層の深さが明らかに足りないということを提案し，ResNet を実装した．その後 ResNet を用いたいくつかの実験を行い，その結果からモデルの表現力が足りていないと考え，さらに層を深くすることを提案し，RAM などを考慮した上で DenseNet を使用することを提案し，実装した．そして DenseNet を用いたいくつかの実験を行い，様々な工夫をした．実験結果をグラフ化し，考察を行った．これらすべてのことをするために様々な書籍や多くの論文を読み，プロジェクト学習に役立てた．

(※文責: 和田孝喜)

8.4 川村一世

「ゼロから作る DeepLearning」を用いて，ディープラーニングについての学習を行い，グループ分けを行い，全方位カメラによる姿勢推定グループに所属した．グループのポスター，スライドの作成を主導した．ポスター，スライドの雛形を作成し，その後修正を行った．また，プロジェクトで使用するマシンの環境構築のサポートを行った．後期もグループのポスター，スライドの作成を主導した．また，学習に使用するデータの作成，支援を行った．

(※文責: 川村一世)

8.5 松元健

前期の最初はディープラーニングへの理解を深めることを目標に活動した．プロジェクトメンバー全員で「ゼロから作る DeepLearning」という本を使って学習した．Python 自体は元からできるため，ディープラーニングを中心に学習した．その後グループに分かれ，姿勢推定のグループに所属した．グループの中では，Python に慣れているためニューラルネットワーク以外の部分のコーディングを行った．具体的には，取得した画像データをネットワークに入力できる形に変換するようなコードを記述した．また，プログラムや TeX で記述する報告書のアドバイスを積極的に行った．他にも，github の管理やリファクタリングも積極的に行った．後期は，ニューラルネットワークの形が何度か変わったため，それに合わせてデータを整形した．同時に，前期では理解の進んでいなかったニューラルネットワークも，個人で学習を続けニューラルネットワーク班のメンバーと議論を行い，改善策を考えた．

(※文責: 松元健)

第 9 章　まとめ

本グループの目的は，モーションキャプチャを行うためのシステムの開発である．既存の手法では，多くのコストが必要となってしまうというような問題がある．そこで私たちのグループでは，ディープラーニングと全方位カメラを用いることによって，低コストで簡単にモーションキャプチャが出来ると考えた．しかし，全方位カメラを用いてモーションキャプチャを試みた結果，撮影範囲外である体の軸より後ろにある腕や足等を撮影することは出来ないため，モーションキャプチャを行う前に姿勢推定をすることが必要であると考えた．姿勢推定を実現させるためには，全方位カメラによって得られた映像から姿勢を出力するモデルが必要である．そこで私たちは，ディープラーニングを用いることによって全方位カメラの映像から得られた情報をもとにモデルが作成できるのではないかと考えた．この姿勢推定が出来るようになることで，全方位カメラの装着者の周囲の状況をモデル化や，全方位カメラに映っていない部分の状況を予測することが出来る．これらのことから，私たちのグループでは，ディープラーニングと全方位カメラを用いたモーションキャプチャを行うためのシステムの開発を行う．

9.1　前期の成果

前期の活動では，ディープラーニングに関する知識の取得や，姿勢推定のためのデータの収集，一般的な畳み込みニューラルネットワークとプロジェクト特化型の 2 種類のモデルを構築した．しかし，プロジェクトを達成する上で必要とする畳み込みニューラルネットワークの層の数が比較的少ないと判断したため，2 種類の畳み込みニューラルネットワークのモデルの構築を一時的に中断した．他に，各関節の角度と全方位カメラの画像のデータセットを作成し保存する為のシステムの構築を行った．また，KinectV2 と Unity の接続の設定とデータのシステム化を行った．前期で用いた全方位カメラは，無線で操作する場合専用のソフトウェアの準備および無線での通信に機能制限があることが分かり，また撮影範囲が不足しているため不適であると判断した．

9.2　後期の成果

後期の活動では，前期で構築した 2 種類の畳み込みニューラルネットワークの層の数が学習する上で少ないと判断したため，開発環境に Keras のライブラリを追加し，ResNet と呼ばれるニューラルネットワークをより深くするために考えられた畳み込みニューラルネットワークのモデルおよび DenseNet と呼ばれる他のニューラルネットワークよりも深い層での学習が比較的安定する畳み込みニューラルネットワークのモデルを構築した．また，今回のプロジェクト達成に適した全方位カメラの確保を行った．他に，前期から引き続き構築した各関節の角度と全方位カメラの画像のデータセットを作成し保存するシステムを用いて ResNet や DenseNet に学習させるためのデータの収集を行った．その後収集したデータを ResNet や DenseNet に学習させる実験を行い，学習の精度の向上を図った．全身で学習したモデル，手首を除いて学習したモデル，関節を上半身と下半身に分けて学習したモデルは，いずれも Loss は収束したものの，時間的な問題もありプロジェクトの目標を達成できる精度まで向上させることはできなかった．

9.3 結果の考察

今回このような問題が発生した原因を考察すると，全方位カメラから得られる装着者の身体の情報が少ないので身体以外の不要な情報が学習を妨げている．そのため前処理として手や足等にマークしておくことによって，各関節に対してのみモデルを反応させることが出来たと考える．また，左手と右足，右手と左足に強い相関があるため，回帰モデルの場合に発生する多重共線性の問題が生じてしまい，上手く予測できず良い精度を出すことができなかったことも考えられる．他に，精度の向上のために一つの関節あたりに使用するパラメータを増やすことやモデルの表現力の改善を試みたが，著しい精度の向上が見られなかったため，今回行われた実験の結果からタスク自体が難しいことも考えられる．したがって，十分な結果を得るには入力部分の学習の妨げとなる不要な情報を取り除き，多重共線性の問題の解決や，タスクに必要な情報を取得し出力データに制限をかけて関節の角度の簡略化を行うことによりプロジェクトの目標を達成できると考えた．今後の展望としては，部位ごとの学習にパラメータを十分に使える状態にすることで学習が進むか，今回は角度の表現方法としてクォータニオンを使用したがベクターとクォータニオンのどちらが学習に優れているか，クォータニオンの特性に合わせたモデルを構築することによって出力が改善されるかを調べる必要がある．

今回のプロジェクト活動を通してディープラーニングを用いたモーションキャプチャを行うためのシステムの開発という目標を達成することはできなかったが、この活動を通して学習してきたことによって、ディープラーニングや画像認識に対する多くの知識を習得することができた．

(※文責: 三浦幸泰)

参考文献

[1] Zhe Cao，Tomas Simon，Shih-En Wei，　and Yaser Sheikh．Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields．arXiv:1611.08050．2016.

[2] Xu Weipeng,Chatterjee Avishek,Zollhoefer Michael,Rhodin Helge,Fua Pascal,Seidel Hans-Peter and Theobalt Christian．Mo2Cap2: Real-time Mobile 3D Motion Capture with a Cap-mounted Fisheye Camera．arXiv:1803.05959．2018.

[3] 斎藤康毅．ゼロから作る DeepLearning．O'Reilly Japan．2016.

[4] C.M.Bishop．Pattern Recognition and Machine Learning.

[5] Min Lin, Qiang Chen and Shuicheng Yan Network in Network. arXiv:1312.4400.

[6] Kaiming He,Xiangyu Zhang,Shaoqing Ren and Jian Sun Deep Residual Learning for Image Recognition. arXiv:1512.03385. 2015.

[7] Gao Huang,Zhuang Liu,Laurens van der Maaten and Kilian Q. Weinberger Densely Connected Convolutional Networks. arXiv:1608.06993. 2016.

[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke and Andrew Rabinovich Going Deeper with Convolutions. arXiv:1409.4842. 2014.

[9] "Unity - Scripting API: Quaternion", Unity, https://docs.unity3d.com/2018.3/Documentation/ScriptReference/Quaternion