スマートフォンのモーションセンサを利用した個人認証アプリケーションの開発に関する研究

総合情報学研究科

知識情報学専攻

マルチメディア情報システムの基礎と実際

15M7112

髙坂 賢佑

要旨

スマートフォンの普及により，従来型のフィーチャーフォンと呼ばれる端末と比較して多種多様なサービスを利用できるようになった．例えば，スマートフォンはパーソナルコンピュータ向けに設計されたWebサイトを閲覧できるフルブラウザが利用できる．これを用いることで，オンラインショッピングを含むサービスをどこでも手軽に利用できるようになった．またスマートフォンは，様々な企業や個人が開発した多種多様なアプリケーションを自由にインストールし利用できる．例えば国内銀行各社からスマートフォン向けアプリが提供されているが，これを用いることでアプリ内で口座残高の確認や入出金明細の確認はもちろん，振り込みや振り替えなども行える．このようなサービスを利用する際には，あらかじめ登録したユーザ名とパスワードを用いた個人認証を行うのが一般的である．だがこれら情報をアプリ内に記憶しておくことで，利用毎に再入力する手間を省けるような仕組みを持つ場合もある．

このようにスマートフォンによって日々の生活がより豊かになった一方で，端末がより多くの個人情報を内包するようになった．このことから，第三者によって不正に個人情報にアクセスされたり，インストールされたアプリを通じて様々なサービスをなりすまし利用されたりといった場合の危険性は高くなった．そのため，端末利用時にはパスコード認証や指紋認証などを用いて本来の端末所有者であるか認証するよう設定することが推奨されている．

現在スマートフォンにおいて個人の認証方式として広く使われている方法として，パスコード認証と指紋認証が挙げられる．しかしながらこれら認証方式にはいくつかの問題点が考えられる．まずパスコード認証では認証作業が煩雑であったり認証に用いる鍵情報の自由度が低いという点がある．また指紋認証では指紋を読み取るためのハードウェアが必要である点や，指紋情報は変更ができないため，何らかの原因でこの情報が第三者に漏洩した場合はその指紋を用いた個人認証ができなくなるという点がある．

そこで本研究ではスマートフォンに一般的に搭載されている加速度センサと角速度センサを利用し，人間の動きを用いて端末を振ることで個人を認証するシステムを開発した．本シス

テムでは，認証時に入力されたデータが本来の端末所有者本人によるものかを識別するために，人工ニューラルネットワークを利用した．端末所有者はあらかじめ認証時に利用したい動きをシステムに入力し，登録処理を行う．登録処理では，まず Denoising Autoencoder を用いて入力されたデータの特徴を学習し，その後 Denoising Autoencoder の後ろに識別を行うニューロンを繋いで学習を行う．認証時には，登録処理で得られた識別器を用いて入力されたデータが端末所有者本人のものであるかを識別する．

評価の結果，云々．本システムにより，パスコード認証が抱える認証の煩雑さと鍵情報の自由度といった問題点を軽減した．また指紋認証における鍵情報が変更できないという問題点を解消し，直感的に個人認証を行うことが可能になった．

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究背景

近年，スマートフォンと呼ばれる携帯端末が急速に普及しつつある．スマートフォンとは，パーソナルコンピュータ向けに設計されたWebサイトを閲覧できる機能を持つフルブラウザを搭載し，様々な企業や個人が開発した多種多様なアプリケーションをインストールし利用できる携帯端末のことを指す[1]．平成28年版の情報通信白書によるとスマートフォンの世帯普及率は2015年末時点で72.0%とあり，また前年比で7.8ポイント増となっている[2]．スマートフォンの普及によりどこでも手軽にオンラインショッピングやネットバンキングをはじめとする多種多様なサービスを利用できるようになった．

その一方で，これらサービスの利用にはユーザIDやパスワード等を含む個人情報を用いた個人認証を必要とする場合が多い．また，利用しているブラウザやアプリケーションによっては，サービスにログインすれば一定期間ログイン状態を保持し再ログインの手間を省くような機能を持つものもある．この機能により，ユーザはサービスを利用するたびに再ログインする手間が無くなることから利便性が向上する．しかしながら，悪意のある第三者がサービスへのログインに必要な情報を知らずとも，本来のユーザになりすましてサービスを利用できてしまうという危険性がある．

このように，スマートフォンは従来型のフィーチャーフォンと比較してより多くの個人情報を内包しており，第三者からのこれら情報への不正なアクセスを防ぐための仕組みが不可欠となっている．現在この仕組みを実現する方法として広く採用されているのが，端末利用時にあらかじめ登録したパスコード情報や指紋情報をもとに，現在の利用者が本来の端末所有者であるかを確認する個人認証システムである．パスコード認証方式では，あらかじめ端末所有者が特定の文字種からパスコードを構築し，これを端末に登録しておく．そして，端末利用時に入力されたパスコードと登録されたパスコードを比較して同一であれば端末所有者であるとみなして，その後の端末利用を許可する．指紋認証方式では，あらかじめ端末所有者が端末に搭載

された指紋スキャナを通じて自らの指紋をスキャンし，これを端末に登録しておく．そして，端末利用時に指紋をスキャンして登録された指紋との比較をし，同一であれば端末所有者であるとみなしてその後の端末利用を許可する．これらの個人認証システムを利用することにより，第三者によって不正に端末内の個人情報へアクセスされる危険性をある程度軽減できる．しかし，これらの認証方式にはそれぞれいくつかの問題点が挙げられる．

まずパスコード認証方式だが，これは個人認証を行う際にスマートフォン画面上に表示されたソフトウェアキーボードを目視し指でタッチして操作する必要があり，ユーザにとっては煩雑である可能性があるという点がある．またあらかじめ決められた文字種の中から一つずつ選んだ文字を並べてパスコードを構築することから，パスコードのパターン数が限られ，認証に用いる鍵の自由度が制限されてしまうという点がある．

指紋認証方式については指紋をスキャンするためのハードウェアをスマートフォンに搭載しなければならないという点がある．また指紋情報は変更ができないため，何らかの原因でこの情報が第三者に漏洩した場合は，今後その指紋を用いた個人認証ができなくなるという点がある．さらに，ドイツのハッカー集団である Chaos Computer Club の生体認証チームが，一般的なカメラで撮影された写真に写り込んだ指から指紋を複製することに成功している [3]．このことから，指紋情報が漏洩する可能性が十分にあり個人認証システムが担う機密性の確保が難しいといえる [4].

1.2　研究目的

本研究では，一般的なスマートフォンに搭載されている加速度センサと角速度センサを利用し，端末を振る動き（以下，モーション）で個人認証を行うシステムを開発する．これによりパスコード認証方式における認証作業の煩雑さと鍵情報の自由度が制限されるという課題点を軽減し，指紋認証方式における指紋情報が漏洩した場合に鍵情報の変更ができないという課題点を解消した生体認証システムの実現を目指す．

1.3　本論文の構成

第 2 章にて本研究に関連する先行研究について述べる．第 3 章では本研究で開発した個人認証システムを提案するにあたり必要となる知識について説明する．第 4 章では本研究で開発した個人認証システムの実装について，その詳細を述べる．第 5 章では本研究で開発した個人認証システムの評価実験とその結果を示し，第 6 章で結論と今後の課題を述べたあと，本論文を総括する．

第2章　関連研究

坂本の研究 [5] では，ユーザが入力したモーションの数値化に加速度センサを用いた．あらかじめ保存しておいた複数種類のジェスチャパターンと認証時にユーザが入力したモーションデータをパターンマッチング方式のアルゴリズムを用いて比較することで個人認証を行った．しかし，このプログラムは扱うジェスチャによって認証率が高いものと低いものに二分化する傾向が見られるという問題点があった．

濵野らの研究 [6] では，加速度センサに加えて角速度センサを用いたジェスチャ動作による認証手法を提案した．これにより回転動作の取得によるモーションの自由度向上となりすまし認証に対する強度の向上を可能にした．認証手法として単一動作を組み合わせて認証する単一動作組み合わせ認証と，ユーザが自由に考えたモーションを用いて DP マッチングによって認証する一筆書き認証の二つを提案した．このシステムの実証実験は複数日かけて実施されており，一筆書き認証において日を経ることによる習熟度の向上から，本人拒否率が改善したことが確認された．しかし，初日の認証での本人拒否率が高く，さらなる本人拒否率の改善が課題として挙げられていた．

第3章　予備知識

本章では，本研究で開発した個人認証システムで用いた技術について説明する．

3.1　人工ニューラルネットワーク

人工ニューラルネットワーク（以下，ニューラルネットワーク）とは，脳内に存在する多数のニューロンによる，シナプスを介した信号のやりとりからなる情報処理の機能を計算機上に再現することを目指したものである．ニューラルネットワークにおけるニューロンは図 3.1 のように表され，図中の x_1 から x_i からなる入力から，式 3.1 によって y_j を出力する．図 3.1 及び式 3.1 においてそれぞれ w_{j1} から w_{ji}，w_{ji} と表されているものはシナプスの結合荷重（以下，結合荷重）で，対応する入力にどれだけの重みを持たせるかを示している．また，式 3.1 において b_j と表されているものはバイアスと呼ばれ，ニューロンが発火する傾向の高さを示している．式 3.1 における f は活性化関数と呼ばれ，入力とそれに対応する結合荷重の積を総和したものにバイアスを足した出力を正規化するために用いる．活性化関数には恒等写像や ReLU（ランプ関数）など様々なものがある．このようなニューロンを複数個・複数層に重ねることにより，より複雑な問題に対応できる．

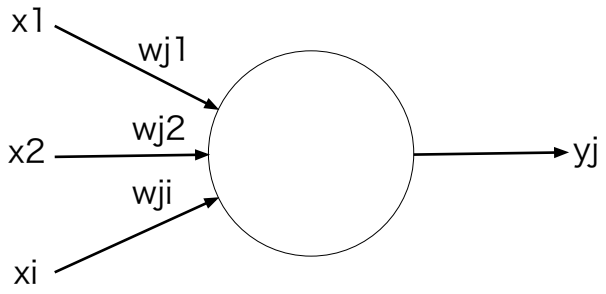

図 3.1: ニューラルネットワークにおけるニューロン

$$y_j = f(\sum_i w_{ji} x_i + b_j). \tag{3.1}$$

ただし，ただニューラルネットワークを構築して入力を与えただけでは，期待した出力が得られることは稀である．そのため，期待した出力が得られるように誤差逆伝播法を用いて結合荷重とバイアスを更新していく，ニューロンの学習を行わなければならない．学習を行うためには，何を目標に結合荷重やバイアスを更新していくのかという基準を設定する必要がある．これは損失関数と呼ばれ，ニューラルネットワークにより得られた出力が，期待する出力（以下，教師信号）とどれだけの誤差があるかを定量的に測るために用いられる．

ニューロンの学習を行う際は，図 3.2 のような縦軸に損失関数（エネルギー関数）E，横軸に結合荷重 w を置いたグラフで，E を最小化するような opt_w に近づくように結合荷重を更新していく（勾配法）．

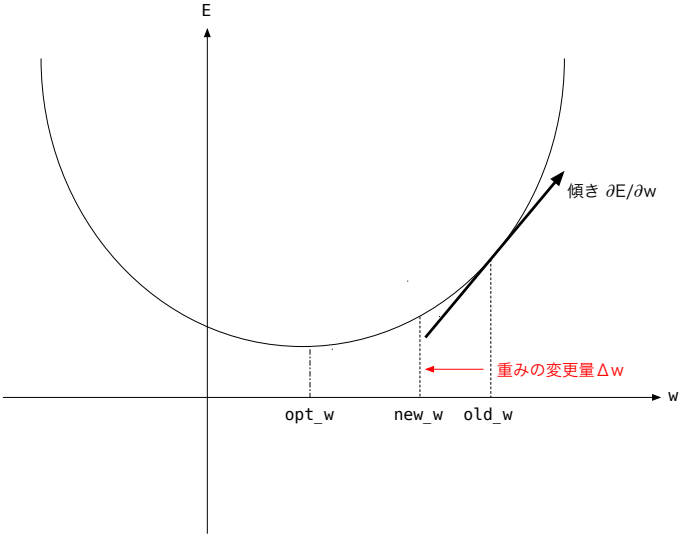

図 3.2: ニューロンの結合荷重の更新

更新した new_w は，式 3.2 で得られる．

$$new\_w = old\_w + \Delta w. \tag{3.2}$$

式 3.2 における Δw が結合荷重の変更量となるのだが，これは式 3.3 で得られる．

$$\Delta w = -\eta \frac{\partial E}{\partial w}. \tag{3.3}$$

η は学習率を表し，どれだけの割合で結合荷重を更新するかを示している．この値を小さくすることで，結合荷重の更新幅が小さくなる．その後ろの $\frac{\partial E}{\partial w}$ は傾きを示す．結合荷重 w が opt_w に近づくためには，傾きが正の場合は Δw が負に，傾きが負の場合は Δw が正になる必要がある．そのため，$\eta \frac{\partial E}{\partial w}$ の結果得られた値の符号を逆にしている．

結合荷重の変更量 Δw を得るために必要な傾き $\frac{\partial E}{\partial w}$ は，式 3.4 で得られる．

$$\frac{\partial E_n}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \frac{\partial E_n}{\partial a_j}x_i. \tag{3.4}$$

式 3.4 は，合成関数の微分の公式を用いて式変形を行っている．x_i は学習中のニューロンに入力された値を示している．ここで，$\frac{\partial E_n}{\partial a_j}$ を δ_j と置く．この δ_j と学習中のニューロンの入力値 x_i を掛けることで傾きが得られ，これに $-\eta$ を掛けることで結合荷重の変更量 Δw が得られる．

前述したように，損失関数を用いて教師信号との誤差を測るために用いるのは出力層の出力値である．ここで，出力層ニューロンの活性化関数と損失関数が特定の組み合わせであれば，δ_j は式 3.5 で得られる．

$$\delta_j = y_j - t_j. \tag{3.5}$$

式 3.5 における y_j は出力層より得られた出力値，t_j は教師信号である．このように極めて単純な計算式で δ_j を得られるため，基本的に出力層では損失関数と活性化関数を合わせて考えるのが一般的である．

まず，ニューラルネットワークで回帰を行う場合は，活性化関数と損失関数にそれぞれ恒等写像（式 3.6）と最小二乗誤差（式 3.7）を用いる．

$$f(a_j) = a_j. \tag{3.6}$$

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(y_{nk} - t_{nk})^2. \tag{3.7}$$

ニューラルネットワークで二値分類を行う場合は，活性化関数と損失関数にそれぞれシグモイド関数（式 3.8）と交差エントロピー誤差（式 3.9）を用いる．

$$f(a_j) = \frac{1}{1 + e^{-a_j}}. \tag{3.8}$$

$$E_n = -\sum_{k=1}^{K}\{t_{nk}\ln y_{nk} + (1 - t_{nk})\ln(1 - y_{nk})\}. \tag{3.9}$$

また，ニューラルネットワークで多クラス分類を行う場合は，活性化関数と損失関数にそれぞれソフトマックス関数（式 3.10）と交差エントロピー誤差（式 3.11）を用いる．

$$f(a_j) = \frac{e^{a_j}}{\sum_{k=1}^{K} e^{a_k}}. \tag{3.10}$$

$$E_n = -\sum_{k=1}^{K} t_{nk} \ln y_{nk}. \tag{3.11}$$

このような組み合わせで出力層を構築することで，前述した式 3.5 で δ_j を得られる．

ニューラルネットワークが入力層を含めて 3 層以上あるような複雑なもので中間層を学習したい場合，この δ_j は式 3.12 で得られる．

$$\delta_j = \frac{\partial f}{\partial a_j} \sum_{k=1}^{K} w_{kj}\delta_k. \tag{3.12}$$

つまり，学習しているニューロンの活性化関数を微分したものと，出力層に一つ近い層の各結合荷重と δ を掛けたものの総和を掛けることで，中間層の δ_j が得られる．

以上で学習に必要な傾きが得られるが，これを用いた結合荷重の更新方法として式（3.2）を改良したアルゴリズムが複数考案されている．

まず，AdaGrad[7] というものがある（式 3.13）．

$$\begin{aligned} new\_g_{ji} &= old\_g_{ji} + (\delta_j x_i)^2. \\ new\_w_{ji} &= old\_w_{ji} - \frac{\eta}{\sqrt{new\_g_{ji} + \epsilon}} \delta_j x_i. \end{aligned} \tag{3.13}$$

これは，過去の傾きの二乗和を結合荷重ごとに覚えておき（new_g_{ji}），その平方根で η を割ったものを学習率とする．これにより，総更新量が少ない結合荷重はより大きな幅で，総更新量が多い結合荷重はより小さな幅で更新がなされる．この方式では，学習率が自動的に調整される [8].

また，Adam[9] というものがある（式 3.14）．

$$\begin{aligned} new\_m &= \beta_1 old\_m + (1-\beta_1)\delta_j x_i. \\ new\_\nu &= \beta_2 old\_\nu + (1-\beta_2)(\delta_j x_i)^2. \\ new\_\hat{m} &= \frac{new\_m}{1-\beta_1^t}. \\ new\_\hat{\nu} &= \frac{new\_\nu}{1-\beta_2^t}. \\ new\_w_{ji} &= old\_w_{ji} - \frac{\alpha}{\sqrt{new\_\hat{\nu}} + \epsilon} new\_\hat{m}. \end{aligned} \tag{3.14}$$

まず，傾きの一次モーメント（平均値）と二次モーメント（分散した平方偏差）の概算値を求める（new_m 及び new_ν）．そしてこれらの偏りをバイアス補正した推定値を計算することで小さくし（$new\_\hat{m}$ 及び $new\_\hat{\nu}$），これらを用いて結合荷重の更新を行う [10].

他にも様々なアルゴリズムが提案されている．

3.2 Autoencoder

Autoencoder とは，図 3.3 のような入力層と出力層のニューロン数を入力データの次元数と同数にし，中間層のニューロン数を一定の割合で削減した 3 層構造のニューラルネットワークにおいて，入力データと教師信号に同じデータを用いて教師あり学習させたものである．入力層から中間層への処理をエンコーダ，中間層から出力層への処理をデコーダと呼ぶ．中間層と出力層の活性化関数は自由に選ぶことができるが，出力層の出力値が教師信号を表現できるような活性化関数を選ばなければならない．

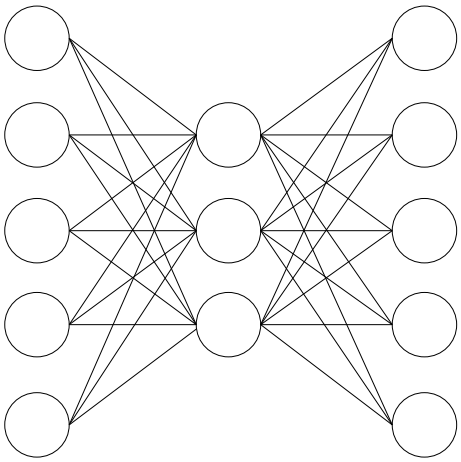

図 3.3: Autoencoder

Autoencoder の学習を進めた結果出力層の出力と教師信号との誤差が無くなった場合，中間層においてより少ないニューロン数で入力データの情報を表現できていると見ることができる．このことから，学習済みの Autoencoder からデコーダの部分を取り除き，エンコーダの出力を別のニューラルネットワークへの入力値として渡すことで，入力データの特徴抽出器として用いることができる．

3.3 Denoising Autoencoder

Denoising Autoencoder とは，Autoencoder で用いる入力データに対し一定の割合でランダムなノイズを付与し，ノイズを付与する前の元データを教師信号として与えて教師あり学習させたものである．入力データにランダムなノイズを付与して学習させることで，入力データの特徴を抽出することに加えてノイズの除去についても学習しなければならなくなり，ノイズに強くより汎化能力の高い特徴抽出器となる．

ノイズの付与方法として，入力データのうちランダムに選択したデータを 0 にするマスキングノイズや，0 か 1 にする胡麻塩ノイズなど様々なものがある．

3.4 Dropout

Dropout とは，図 3.4 のように一定の割合でランダムに一部のニューロンを無視して学習を行うものである．学習後のニューラルネットワークで推定を行う際は，Dropout した層における各ニューロンの出力値に 1 から Dropout した割合を引いた値を掛ける．

ニューラルネットワークの学習時にランダムで一部のニューロンを無視することから，複数の独立したニューラルネットワークを学習しているとみなすことができる．そして推定時にはそれらニューラルネットワークから得られた出力の平均値を求めていると考えられるため，汎化能力がより高くなる．

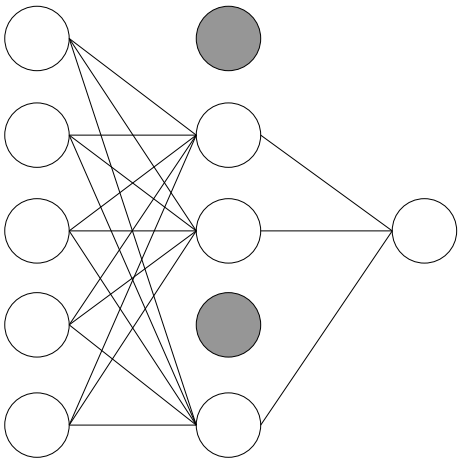

図 3.4: Dropout

3.5 モーションデータの取得

本システムは, モーションデータの取得に Android 端末で一般的に搭載されている加速度センサ（TYPE_LINEAR_ACCELERATION）と角速度センサ（TYPE_GYROSCOPE）を用いる．モーションセンサの座標系は図 3.5 のようになっており，直線で示した方向に端末を動かすことで正の加速度が得られ, 曲線で示した方向に端末を回転させることで正の角速度が得られる [11]. センサ登録時に指定できるモーションデータの取得間隔は SENSOR_DELAY_FASTEST を指定しており，研究で使用した LG エレクトロニクス社と Google 社によって開発された Nexus 5 では，およそ 5 ミリ秒間隔でモーションデータが取得されていることを確認した．ただし，SENSOR_DELAY_FASTEST などによる取得間隔の指定はあくまでシステムへのヒントであり，システムによって変動する可能性がある [12].

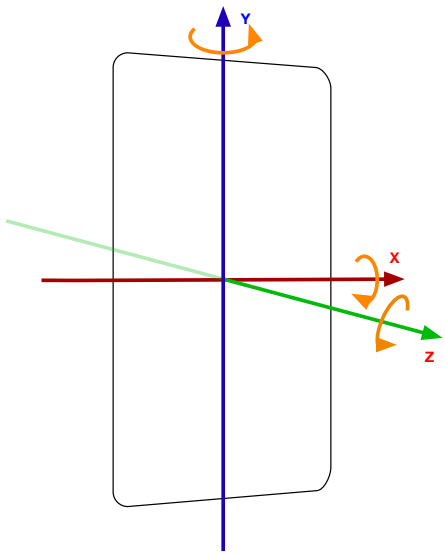

図 3.5: モーションセンサの座標系

第4章　提案システム

本章では，本研究で開発したスマートフォンのモーションセンサを利用した個人認証システムについて説明する．

4.1　システムの概要

本システムは，Android 端末に一般的に搭載されている加速度センサと角速度センサを用いてモーションデータを収集し，Denoising Autoencoder とその後ろに識別用ニューロンを繋げた識別器を用いて個人認証を行う．本システムには，登録モードと認証モードという二つの動作モードがある．登録モードでは，入力されたモーションデータの一部をランダムにデータの最大値及び最小値でノイズを付与したものを用いて Denoising Autoencoder で特徴を学習させる．その後識別用ニューロンを繋いで入力データに対する教師信号として 0.0 を，入力データと同じ値域・次元数で乱数生成したダミーデータに対する教師信号として 1.0 を与えて識別器の学習を行う．認証モードでは，入力されたモーションデータを学習済みの識別器に入力し，得られた出力を用いて個人認証を行う．

4.1.1　動作モード選択

Android アプリケーションを起動すると, 図**??**のスタート画面が表示される．画面下部にある “Start” ボタンを押すことで, 図**??**の動作モード選択ダイアログが表示される．“Registration” を選択すると登録モードに，“Authentication” を選択すると認証モードに遷移する．

4.1.2　登録モード

登録モードに遷移すると，図**??**のユーザ名入力画面が表示される．ここでは登録するユーザ名を入力する．ユーザ名を入力して画面下部にある “OK” ボタンを押すことでユーザ名が正しく入力されたか確認する処理が行われ，確認できれば図**??**のモーション入力画面が表示される．ユーザ名が入力されていない，もしくは空白文字しか入力されていない場合は図**??**のように Android システムに標準で用意されている通知機能である Toast を用いてユーザにエ

ラーを通知する．

モーション入力画面では，画面下部の “モーションデータ取得” ボタンを押している間，加速度センサと角速度センサそれぞれにデータを取得するための専用スレッドが起動して各センサからデータが取得・蓄積される．また，データ取得用スレッドの起動と同時に時間計測用スレッドも起動し，1 秒経過毎に端末をバイブレートさせることでユーザにモーション入力の経過時間を伝える．モーション入力中に任意のタイミングで “モーションデータ取得” ボタンから指を離すことで，モーション入力を終了できる．モーション入力はデフォルトでは 3 回となっているが，画面右上のハンバーガーメニュー，もしくは端末に搭載されたメニューボタンを押すことで表示される図**??**のメニュー画面から，“データ取得回数設定” を選択することで回数を変更できる．“データ取得回数設定” を選択すると，図**??**のダイアログが表示される．画面上にあるスライダーを操作して左右に動かすことで，データ取得回数を増減でき，画面下部の “OK” ボタンを押すことで設定が反映される．また，先ほどのメニュー画面から “リセット” を選択すると，図**??**のダイアログが表示される．ここで “YES” を選択することで，モーションデータの取得状態をリセットし，一からモーション入力をやり直すことができる．

モーション入力が終わるたびに，取得したモーションデータのデータ数が確認される．加速度センサ及び角速度センサから得られた X 軸データのいずれかのデータ数が 10 個を下回っていた場合，図**??**のダイアログを表示し，ユーザに再度モーションを入力させる．設定回数分のモーション入力が終わると図**??**のプログレスダイアログが表示され，後述するモーションデータの加工及び識別器の学習を行うスレッドが起動する．

モーションデータの加工及び識別器の学習が終わると図**??**のダイアログが表示される．“OK” ボタンを押すことでユーザ名と暗号化した学習済み識別器のパラメータ，データの次元数を他アプリからの読み書きができない形で端末に保存し，スタート画面に遷移する．何らかの原因で識別器の学習ができなかった場合は図**??**のダイアログが表示される．“OK” ボタンを押すことでモーションデータの取得状態がリセットされるので再度モーションを入力する．

4.1.3　認証モード

認証モードに遷移すると，図**??**のユーザ名入力画面が表示される．ここでは認証するユーザ名を入力する．ユーザ名を入力して画面下部にある “OK” ボタンを押すことで登録モードで保存されたユーザ名のリストから入力されたユーザ名と合致するものが存在するか検索され，存在していた場合は図**??**のモーション入力画面が表示される．存在していなかった場合は，図**??**のように Toast を用いてユーザにエラーを通知する．

モーション入力画面では，画面下部の “モーションデータ取得” ボタンを押している間，加速度センサと角速度センサそれぞれにデータを取得するための専用スレッドが起動して各センサからデータが取得・蓄積される．また，データ取得用スレッドの起動と同時に時間計測用スレッドも起動し，1 秒経過毎に端末をバイブレートさせることでユーザにモーション入力の経過時間を伝える．モーション入力中に任意のタイミングで “モーションデータ取得” ボタンから指を離すことで，モーション入力を終了できる．モーション入力は 1 回となっており，登録モードと違い変更はできない．

モーション入力が終わると，取得したモーションデータのデータ数が確認される．加速度センサ及び角速度センサから得られた X 軸データのいずれかのデータ数が 10 個を下回っていた場合，図**??**のダイアログを表示し，ユーザに再度モーションを入力させる．設定回数分のモーション入力が終わると図**??**のプログレスダイアログが表示され，後述するモーションデータの加工及び識別器による個人認証を行うスレッドが起動する．

モーションデータの加工及び個人認証が終わると，認証結果をダイアログで表示する．認証に成功すれば図**??**のダイアログが表示され，“OK” ボタンを押すことでスタート画面に遷移する．認証に失敗すれば図**??**のダイアログが表示され，“OK” ボタンを押すことでモーションデータの取得状態がリセットされ，再度個人認証を行える．

4.2　モーションデータの加工

登録モードと認証モードのいずれも，モーションセンサから得られたデータはニューラルネットワークで用いる前に，“データ数の均一化”・“フーリエ変換を用いたローパスフィルタ”・“角速度から変位，角速度から角度への変換”・“変位データを角度データで回転” という四つの加工を行う．

4.2.1　データ数の均一化

本システムでは，モーション入力を任意の時間で行える．この際，登録モードにおけるデフォルトでは 3 回のモーション入力によるモーションデータ，認証モードでは登録時に用いたモーションデータと新たに入力されたモーションデータ間でデータ数の差異が生じる場合がある．本システムで用いたニューラルネットワークでは入力されるデータにおける次元数のばらつきは許容できないため，事前にデータ数を均一化する必要がある．

登録モードでは，最も入力時間の長かったデータを基準に他のデータに対して末尾にゼロを補填する方法を用いる．認証モードでは，登録時に用いたデータの長さを基準に新たに入力さ

れたデータが短い場合は末尾にゼロを補填し，長い場合は末尾を切り落とす方法でデータ数を均一化している．

4.2.2　ローパスフィルタ

モーションを入力している際に生じる手の震えなどによるモーションデータへの影響を抑えるために，フーリエ変換を用いたローパスフィルタ処理を実装している．時間軸領域で表されるデータをフーリエ変換を用いて周波数領域に変換すると，モーション入力中に生じた手の震えなどによるデータが高周波成分として現れる．この高周波成分を取り除いた上で元の時間軸領域で表されるデータに逆変換するローパスフィルタ処理を行うことで，手の震えなどによる影響の少ないデータを得られる．

フーリエ変換の実装には，CERN の Colt Project[13] で開発された科学技術計算用ライブラリである Colt をマルチスレッド化した Parallel Colt[14] に含まれている，JTransforms[15] を用いた．

この処理をソースコード 4.1 に示す．

ソースコード 4.1: ローパスフィルタ

```
1  public double[][][] LowpassFilter(double[][][] data, String sensorName,
       String userName) {
2    log(INFO);
3
4    DoubleFFT_1D realfft = new DoubleFFT_1D(data[0][0].length);
5
6    // Execute forward fourier transform
7    for (double[][] i : data) for (double[] j : i) realfft.realForward(j);
8
9    // Low pass filtering
10   for (int time = 0; time < data.length; time++)
11     for (int axis = 0; axis < NUM_AXIS; axis++)
12       for (int item = 0; item < data[time][axis].length; item++)
13         if (item > 30) data[time][axis][item] = 0;
14
15   // Execute inverse fourier transform
16   for (double[][] i : data) for (double[] j : i) realfft.realInverse(j,
       true);
17
```

```
18    return data;
19  }
```

ローパスフィルタ処理によるデータの変化を示したグラフを図 4.1 に示す．

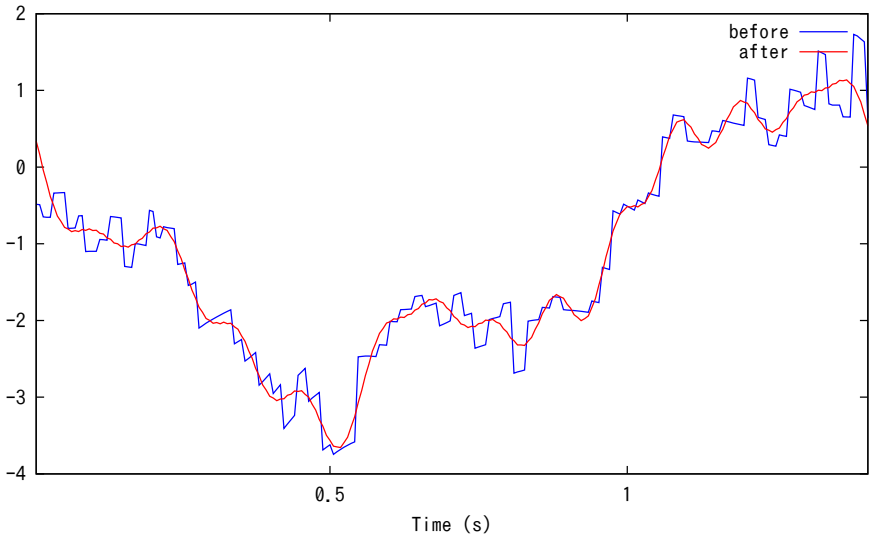

図 4.1: ローパスフィルタ処理によるデータの変化

青色で示した線がローパスフィルタ処理前のグラフ，赤色で示した線が処理後のグラフである．

4.2.3　加速度から変位，角速度から角度への変換

ローパスフィルタ処理したデータに対して，次に加速度から変位，角速度から角度に変換する処理を行う．加速度から変位への変換は式 4.1 で行う．

$$x = \frac{1}{2}at^2. \tag{4.1}$$

x は変位を，a は加速度を，t は加速度データの取得間隔を表している．なお，本システムでは初速度を考慮しない．この処理をソースコード 4.2 に示す．

ソースコード 4.2: 加速度から変位への変換

```
1  public double[][][] accelToDistance(double[][][] inputVal, double t) {
2    log(INFO);
3
4    double[][][] returnVal = new double[inputVal.length][NUM_AXIS][
       inputVal[0][0].length];
5
```

```
6     for (int time = 0; time < inputVal.length; time++)
7       for (int axis = 0; axis < NUM_AXIS; axis++)
8         for (int item = 0; item < inputVal[time][axis].length; item++)
9           returnVal[time][axis][item] = (inputVal[time][axis][item] * t *
              t) / 2;
10
11    return returnVal;
12  }
```

また，角速度から角度への変換は式 4.2 で行う．

$$y = gt. \tag{4.2}$$

y は角度を，g は角速度を，t は角速度データの取得間隔を表している．その処理をソースコード 4.3 に示す．

ソースコード 4.3: 角速度から角度への変換

```
1  public double[][][] gyroToAngle(double[][][] inputVal, double t) {
2    log(INFO);
3
4    double[][][] returnVal = new double[inputVal.length][NUM_AXIS][
       inputVal[0][0].length];
5
6    for (int time = 0; time < inputVal.length; time++)
7      for (int axis = 0; axis < NUM_AXIS; axis++)
8        for (int item = 0; item < inputVal[time][axis].length; item++)
9          returnVal[time][axis][item] = (inputVal[time][axis][item] * t);
10
11   return returnVal;
12 }
```

4.2.4　変位データを角度データで回転

加速度から変位，角速度から角度へ変換したデータに対して，次は変位データを角度データで回転させる処理を行う．この処理をソースコード 4.4 に示す．

ソースコード 4.4: 変位データの角度データを用いた回転，合成

```
1  /**
```

```
2   * 与えられた軸・データ長の距離及び角度配列を，
        combineメソッドを利用して回転させる
3   * @param displacement 軸・データ長の距離配列
4   * @param angle 軸・データ長の角度配列
5   * @return 軸・データ長の距離を角度を用いて回転させたベクトル配列
6   */
7  public double[][] rotate (double[][] displacement, double[][] angle) {
8    log(INFO);
9    double[][] rotated = new double[displacement.length][displacement[0].
         length];
10   double[] combined;
11
12   double angleX = 0.0, angleY = 0.0, angleZ = 0.0;
13
14   for (int item = 0; item < angle[0].length; item++) {
15     angleX += angle[0][item];
16     angleY += angle[1][item];
17     angleZ += angle[2][item];
18
19     combined = combine(displacement[0][item], displacement[1][item],
           displacement[2][item],
20          toRadians(-angleX), toRadians(-angleY), toRadians(-angleZ));
21     rotated[0][item] = combined[0];
22     rotated[1][item] = combined[1];
23     rotated[2][item] = combined[2];
24   }
25
26   return rotated;
27 }
28
29
30 /**
31  * 与えられた距離データを回転させる
32  * 回転は反時計まわりが基本のようなので，ピッチ等の値を正負逆転させている
33  * 参考：http://usi3.com/
```

```
     Position_estimation_by_using_acceleration_sensor.html
34   * @param displacementX 回転させる距離データ（x）
35   * @param displacementY 回転させる距離データ（y）
36   * @param displacementZ 回転させる距離データ（z）
37   * @param pitch ピッチ
38   * @param roll ロール
39   * @param yaw ヨー
40   * @return 回転させた距離データ
41   */
42  private double[] combine(double displacementX, double displacementY,
       double displacementZ,
43                           double pitch, double roll, double yaw) {
44      log(INFO);
45      double sinPitch = round(sin(pitch));
46      double cosPitch = round(cos(pitch));
47      double sinRoll = round(sin(roll));
48      double cosRoll = round(cos(roll));
49      double sinYaw = round(sin(yaw));
50      double cosYaw = round(cos(yaw));
51
52      double vx = displacementX, vy = displacementY, vz = displacementZ;
53      double tmpX, tmpY, tmpZ;
54
55      // X軸周りの回転
56      tmpY = vy;
57      tmpZ = vz;
58      vy = tmpY * cosPitch - tmpZ * sinPitch;
59      vz = tmpY * sinPitch + tmpZ * cosPitch;
60
61      // Y軸周りの回転
62      tmpX = vx;
63      tmpZ = vz;
64      vx = tmpX * cosRoll + tmpZ * sinRoll;
65      vz = -tmpX * sinRoll + tmpZ * cosRoll;
66
67      // Z軸周りの回転
```

```
68    tmpX = vx;
69    tmpY = vy;
70    vx = tmpX * cosYaw - tmpY * sinYaw;
71    vy = tmpX * sinYaw + tmpY * cosYaw;
72
73    return new double[]{vx, vy, vz};
74  }
```

変位データを回転させる combine メソッドには，変位データの他にモーションデータ取得開始時点からどれだけ回転したかを保持する angleX・angleY・angleZ のラジアンを渡している．この際，各軸を中心に反時計回りで変位データを回転させるために，それぞれの正負を逆にした上でそのラジアンを求めている．

combine メソッドでは，渡されたラジアンについてそれぞれ sin と cos を求め，式 4.3 の回転行列を用いて X 軸・Y 軸・Z 軸の順に回転させている．

$$
\begin{aligned}
R_x(\theta) &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}. \\
R_y(\theta) &= \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}. \\
R_z(\theta) &= \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}.
\end{aligned} \tag{4.3}
$$

4.3　ニューラルネットワークによる学習と識別

本システムは，図 4.2 のような Denoising Autoencoder と識別用ニューロンを繋げたニューラルネットワークにより個人認証を行う．

本システムでは，基本的に NVIDIA 製 GPU を搭載し CUDA[16] を利用した GPGPU による高速演算が可能な計算機上で動作するプログラム（以下，サーバ）でニューラルネットワークに関する処理を実行する．Android 端末上で動作するプログラム（以下，クライアント）とサーバでデータをやりとりする際には TCP ソケットを用いる．

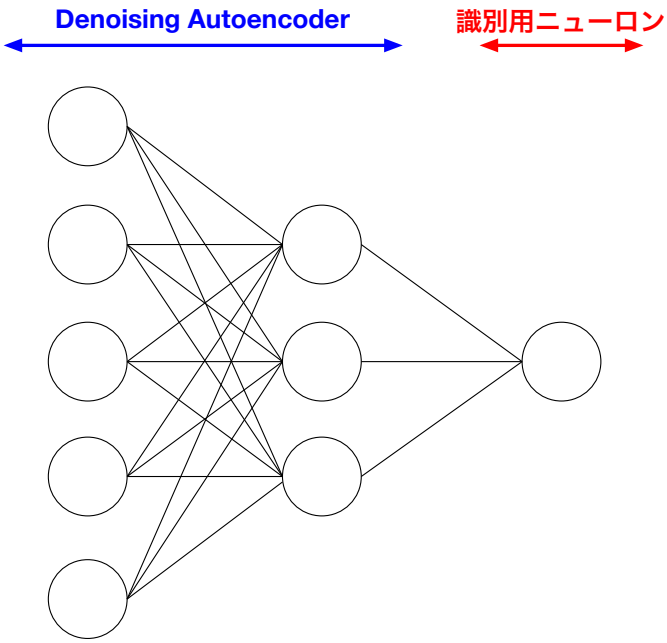

図 4.2: 識別に用いるニューラルネットワーク

4.3.1　登録モード

登録モードでは，動作モード値・ユーザ名・前述の加工を行ったモーションデータを TCP ソケットを用いてサーバに送信する．サーバ側ではこれらデータを Java で書かれたプログラムで受信し，JNI[17] を用いて C++で書かれたニューラルネットワークプログラムに処理を移す．

ニューラルネットワークプログラムでは，まず Denoising Autoencoder の学習を行う．まず，受け取ったモーションデータを平均が 0，分散が 1 になるように正規化する．この処理をソースコード 4.5 に示す．

ソースコード 4.5: モーションデータの正規化処理

```
1  void normalize(vector<double> *input) {
2    // 一つのセットにおける平均値を求める
3    unsigned long input_size = (*input).size();
4    double avg = 0;
5    double sum = 0;
6    for (int data = 0; data < input_size; ++data) sum += (*input)[data];
7    avg = sum / input_size;
8    // 偏差の二乗の総和を求める
9    sum = 0;
10   for (int data = 0; data < input_size; ++data) sum += pow(((*input)[
```

```
      data] - avg), 2);
11  // 分散を求める
12  double dispersion = sum / input_size;
13
14  // 標準偏差を求める
15  double standard_deviation = sqrt(dispersion);
16
17  // 正規化し，得たデータで上書きする
18  for (int data = 0; data < input_size; ++data)
19    (*input)[data] = ((*input)[data] - avg) / standard_deviation;
20 }
```

そして，正規化した各モーションデータのそれぞれ 40%に，そのデータの最大値あるいは最小値で上書きするノイズ加工を行う．この処理をソースコード 4.6 に示す．

ソースコード 4.6: dA 学習時のノイズ加工

```
1  /**
2   * データごとに0.0以上1.0未満の乱数を生成し，
        rate未満であればそのデータを0.0にする
3   * @param input ノイズをのせるデータ
4   * @param rate ノイズをのせる確率
5   * @return ノイズをのせたデータ
6   */
7  vector<vector<double>> add_noise(const vector<vector<double>> &input,
       const float rate) {
8    random_device rnd;
9    mt19937 mt;
10   mt.seed(rnd());
11   uniform_real_distribution<double> real_rnd(0.0, 1.0);
12   double rnd_value = 0.0;
13
14   vector<vector<double>> result(input);
15
16   for (unsigned long i = 0, i_s = result.size(); i < i_s; ++i) {
17     double max = *std::max_element(result[i].begin(), result[i].end());
18     double min = *std::min_element(result[i].begin(), result[i].end());
19     for (unsigned long j = 0, j_s = result[i].size(); j < j_s; ++j) {
```

```
20        rnd_value = real_rnd(mt);
21        if (rnd_value < rate) {
22          if (rnd_value < rate / 2) {
23            result[i][j] = min;
24          } else {
25            result[i][j] = max;
26          }
27        }
28      }
29    }
30
31    return result;
32  }
```

ノイズ加工を行った後，中間層のニューロン数を入力層や出力層のニューロン数から 30%削減し，中間層の活性化関数にシグモイド関数，出力層の活性化関数に恒等関数を用いた Denoising Autoencoder を構築する．訓練データにノイズ加工を行ったモーションデータを，教師信号にノイズ加工を行う前のモーションデータを与え，損失関数に最小二乗誤差を用いて 500 回の学習を行う．また，訓練時に中間層ニューロンのうち 50%をランダムに Dropout させる．

Denoising Autoencoder の学習が終われば，Denoising Autoencoder のパラメータを固定して，この後ろに活性化関数にシグモイド関数を用いたニューロンを繋げる．そして，正規化する前のモーションデータにおける最大値と最小値の範囲内で生成したランダム値からなるダミーデータを生成し，正規化する．訓練データに正規化したモーションデータとダミーデータを，教師信号にそれぞれ 0.0 と 1.0 を与え，損失関数に交差エントロピー誤差を用いて誤差が 0.1 未満になるまで最大 10000 回の学習を行う．この際 Dropout を無効にし，Denoising Autoencoder の出力に対して 1 から先ほどの学習時に適用した Dropout 率を引いた 0.5 を掛ける．

学習が終われば，ニューラルネットワークにおける各層ごとのニューロンが持つパラメータを文字列として連結したデータをクライアントに送る．

クライアント側はこのデータを受け取り，暗号化した上で他アプリからの読み書きができない形で保存する．

4.3.2　認証モード

認証モードでは，動作モード値・ユーザ名・前述の加工を行ったモーションデータ・登録モードにおいて保存した学習済みニューラルネットワークのパラメータを TCP ソケットを用いてサーバに送信する．サーバ側ではこれらデータを Java で書かれたプログラムで受信し，JNI を用いて C++で書かれたニューラルネットワークプログラムに処理を移す．

ニューラルネットワークプログラムでは，まず受け取ったモーションデータを平均が 0，分散が 1 になるように正規化する．次に，受け取った学習済みニューラルネットワークのパラメータをもとにニューラルネットワークを構築する．構築できたらこれに正規化したモーションデータを与え，出力値を得る．そして，得られた出力値をクライアントに送る．

クライアント側はこの値を受け取り，値が 0.2 未満であれば認証成功とし，0.2 以上であれば認証失敗とする．

また，認証モードに限り，端末が何らかの理由でサーバに接続できない場合は，ニューラルネットワークの処理に多少時間がかかるが端末のローカルでサーバと同等の処理が行えるようにしている．

第5章　評価

本章では，本研究で開発した個人認証システムの認証精度と登録及び認証処理にかかる時間の評価を行う．

5.1　認証精度の評価

認証制度の評価について，端末を持ち上げるモーションを対象とする．本システムを用いてあらかじめ 6 名の被験者に端末を持ち上げるモーションを 4 回入力してもらい，モーションデータの収集を行った．各被験者ごとに，最初の 3 回分を登録モードにおける訓練データとして用い，最後の 1 回分を認証モードにおける入力データとして 10 回認証を試行した．また，それぞれの試行時になりすまし認証データとして筆者自身が同様のモーションを入力して得たモーションデータも入力した．

この実験により得られた出力を表**??**に示す．

5.2　登録及び認証の処理時間の評価

本システムについて，登録及び認証の処理時間を計測した．Android のログ出力用 API として用意されている Log クラス [18] を用い，モーション入力が終了し計算処理中であることをユーザに示すプログレスダイアログが表示される部分と，登録及び認証処理が終わり処理結果が表示される前にプログレスダイアログが非表示となる部分にログ出力を行うコードを挿入した．このログから出力される時刻情報を用い，登録及び認証をそれぞれ 10 回試行した平均処理時間を求めて評価を行った．また，認証モードについては端末ローカルでの計算処理も行えるため，こちらの平均処理時間も求めることとする．

実験結果を表**??**に示す．

第6章　結論

本研究では，一般的なスマートフォンに搭載されている加速度センサと角速度センサを利用し，端末を振る動きで個人認証を行うシステムを開発した．そして，複数の被験者を対象にした認証精度の評価と登録及び認証の処理時間計測を行った．

参考文献，参考URL等

[1] 総務省｜電気通信サービスＦＡＱ（よくある質問）｜スマートフォンとはなんですか？，http://www.soumu.go.jp/main_sosiki/joho_tsusin/d_faq/faq01.html，2017年1月27日確認.

[2] 総務省，“平成28年版情報通信白書”，2016年.

[3] CCC ｜ Fingerprint Biometrics hacked again，https://www.ccc.de/en/updates/2014/ursel，2017年1月27日確認.

[4] Chaos Computer Club claims to have“cracked”the iPhone 5s fingerprint sensor Naked Security,https://nakedsecurity.sophos.com/2013/09/22/chaos-computer-club-claims-to-have-cracked-the-iphone-5s-fingerprint-sensor/，2017年1月27日確認.

[5] 坂本翔，“ユーザの直感的な入力をとらえるための3軸加速度センサによるジェスチャ認識の研究”，2009年度公立はこだて未来大学卒業論文.

[6] 濵野雅史，新井イスマイル，“加速度センサ・ジャイロセンサを併用したスマートフォンの利用認証手法の提案”，情報処理学会研究報告，Vol.2014-MBL-70，No.17，Vol2014-UBI-41，No.17，2014.

[7] John Duchi, Elad Hazan, Yoram Singer, “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”, Journal of Machine Learning Research, 12, 2121-2159, 2011.

[8] 実装ノート・tiny-dnn/tiny-dnn Wiki，https://github.com/tiny-dnn/tiny-dnn/wiki/実装ノート，2017年1月29日確認.

[9] Diederik P. Kingma, Jimmy Lei Ba, “Adam: A Method for Stochastic Optimization”, The International Conference on Learning Representations, San Diego, 2015.

[10] 勾配降下法の最適化アルゴリズムを概観する｜コンピュータサイエンス｜POSTD,http:

//postd.cc/optimizing-gradient-descent/，2017 年 1 月 29 日確認.

[11] Sensor Coordinate System | Android Developers, https://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-coords，2017 年 1 月 30 日確認.

[12] Monitoring Sensor Events | Android Developers, https://developer.android.com/guide/topics/sensors/sensors_overview.html#sensors-monitor，2017 年 1 月 30 日確認.

[13] Colt Project, https:/dst.lbl.gov/ACSSoftware/colt/，2017 年 1 月 31 日確認.

[14] Parallel Colt, https://sites.google.com/site/piotrwendykier/software/parallelcolt，2017 年 1 月 31 日確認.

[15] JTransforms, https://sites.google.com/site/piotrwendykier/software/jtransforms，2017 年 1 月 31 日確認.

[16] CUDA Zone | NVIDIA Developer, https://developer.nvidia.com/cuda-zone，2017 年 1 月 31 日確認.

[17] Java Native Interface Specification, http://docs.oracle.com/javase/7/docs/technotes/guides/jni/spec/jniTOC.html，2017 年 1 月 31 日確認.

[18] Log | Android Developers, https://developer.android.com/reference/android/util/Log.html，2017 年 2 月 3 日確認.