山口大学教育学部研究論叢　第 72 巻　227 頁～ 235 頁　2023 年 1 月

筋電位を利用した深層学習による手の動作識別について

伊藤　正剛 *・北本　卓也 **

On the Identification of Hand Movements by Deep Learning Using Myoelectric Potentials

ITO Masataka*, KITAMOTO Takuya**

（Received September 22, 2022）

近年の計算機の性能向上やフレームワークの発展により、深層学習が手軽に行えるようになってきた。従来、計算コストの問題や数値化するのが難しかった事象でも、高い精度で回帰や分類などが行えるようになってきている。本稿では、筋電位に注目し手の動作の識別について、深層学習で分類する。実験初期段階では手動で分類していたが、現段階では精度に問題はあるがリアルタイムで分類することも可能となった。データ取得方法として、ワンボードマイコンの一種である Arduino Uno と簡易筋電センサーの MyoWare 筋電センサーを用いる。

このように、生体信号の一つである筋電位を用いて動作する義手を筋電義手と言い、多方面に渡る応用が期待される分野の一つである。また、筋電位を詳しく理解することによって心拍、脳波、脈拍などの他の信号の理解の助けになると考えられる。

キーワード：深層学習、 Arduino、 MyoWare 筋電センサー、筋電位、動作識別

1. はじめに

1.1 現状分析

現在、コンピューター、特に GPU(Graphics Processing Unit) の発展により、手軽に深層学習ができるようになっている。例えば、 Google アカウントがあれば誰でも利用できる Google Colaboratory を用いると、若干の制約はあるもののフリーで GPU や TPU を備えた Jupyter Notebook の環境を使え、簡単にデータ分析が行える。

本論文では、深層学習の計算はプログラミング言語としては Python3、環境として Google Colaboratory を用いている。最後の方で紹介するリアルタイム処理では OS: Windows10 Pro、 CPU: Intel Core i7-8565U、メモリ: 16GB のコンピュータで、 Anaconda をインストールし、Python3 を用いた。

深層学習が容易になった要因であるが、 TensorFlow などの機械学習ライブラリが作られたということがある。TensorFlow の上部で動く Keras は、直感的に理解しやすく、レゴブロックのように、層を積み重ねて、これまた容易にネットワーク構造を作ることができる。Keras は理解しやすいだけでなく、本格的な研究にも十分なほどの機能がある。本研究では Keras を用いている。

主題の手の動作識別についてであるが、事故や病気により失われた手の機能を補う義手の研究が行われている。かつては、義手の研究では主に外観を回復することに重きがおかれていたが、機械学習の発展に伴って、腕の生体信号の一つである筋電位を読み取って、動作を識別することができるようになっている。例えば、手を握ると、コンピューターが何らかの機械学習アルゴリズムにより、手の握りを検知できるようになっている。筋電位を測ってコントロールする義手のことを筋電義手と言い、これから、さらに発展が見込まれる分野である。ビッグデータと深層学習を用いて、高い精度で動作識別を可能にしている研究者もいる [1]。

1.2 研究動機

現状分析であげた筋電義手であるが、手の動作を精度が高く識別することができれば、違和感なく手の役割を補助する道具として使うことができ、事故や病気で失った手の代わりになり得る。さらに発展していけば、健常

＊ 放送大学教養学部（自然と環境コース）
＊＊ 山口大学教育学部　〒753-8513 山口市吉田1677－１，kitamoto@yamaguchi-u.ac.jp

の手の動きより多くの能力を実現できるかもしれない。そのような研究も行われている。さらに、完全な義手として使うことができない段階にあっても、例えば筋電義手と同期させたモニター上の手を動かすことによってリハビリに使えるのではないかと考えている。画面上に手を投影することにより、拡張現実（AR:Augmented Reality）のようなこともできるかもしれない。

このように応用分野は様々あるが、この研究では、最終的な目標として、左手を屈曲させたらカーソルを右に動かす、反ったら左に動かすなど、障がいを持っている人の補助をするプログラムを作成したいことを動機として行った。

2. 計測機器について

この研究では計測機器として、Arduino（アルドゥイーノ）と言うワンボードマイコンと MyoWare 筋電センサーを使っている。
図 1 に Arduino Uno の写真を示す。

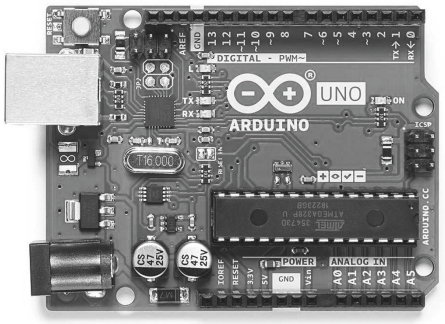

図 1: Arduino Uno R3

Arduino は、誰でもデジタル技術にアクセスし電子工作を行えることをコンセプトとしてイタリア人開発者たちが作ったコンピューターである。他のコンピューターと比べると比較的安価で購入ことができる。2022 年 8 月現在、本研究で使っている Arduino Uno R3 は、値段の変動はあるが 4,000 円前後である。Arduino にセンサーをつなぎ、さらに Arduino をパソコンにつなぐことによって、センサーデータをパソコンに出力することができる。

また、 Arduino IDE という付属ソフトウェアがあり、C/C++をベースにした Arduino 言語により、プログラムを作ることができ、細かくセンサーの出力などを制御することができる。Arduino IDE はシリアルモニタとシリアルプロッタいう機能があり、シリアルモニタでは、シリアル通信でセンサーから得た値を表示することができる。シリアルプロッタは、センサーから得た数値をグラフにしてプロットすることができる。

その他、この付属ソフトウェアを利用して、プログラムを作り、 Arduino に入力しておけば、単独でもコンピューターとして使うことができる。例えば、乾電池などの電源とつなぎ、LED を Arduino とつないで点灯させるなどできる。単独でも簡単に電子機器を作ることができる。

次に、 MyoWare 筋電センサーについてであるが、これは筋電位を測るセンサーである。図 2 に MyoWare 筋電センサーの写真を載せた。

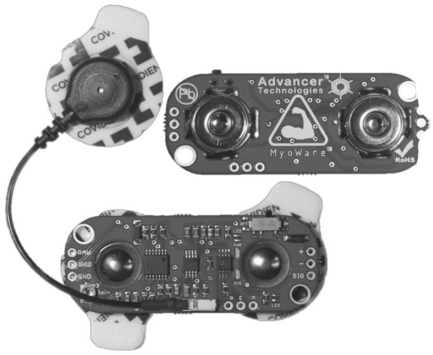

図 2: MyoWare 筋電センサー

2022 年 8 月現在で 6,000 円台である。ただし、現時点で、本研究に使用している MyoWare 筋電センサーのバージョンは販売が終了しており、新しいバージョンも現在手に入りにくい状態である。

もともと、筋電位は医学研究や神経筋疾患の診断に利用されてきた。しかし、センサー部品の小型化・高性能化によって、義手や義足の作成や制御システムにも使われるようになってきている。この筋電位を測るセンサーは、 Arduino や Raspberry Pi などと接続するように構成されており、増幅、整流、統合された信号を出力することができる。MyoWare 筋電センサーから増幅された生波形 (Raw EMG)、整流化された積分筋電図（IEMG:Integrated electromyography）を出力することができる。本研究では、データとして IEMG を用いた。

センサー本体左側に、電気信号を増幅できるゲイン調整のポテンショメーターがあり、出力の上げ下げをする設定することができる。また、高頻度に起こることではないが、パソコンがコンセントとつながっているため、予期せぬ電流の入出力を防ぐためにも、アイソレーター (Adafruit 2107) を使用した。

センサーの腕への取り付け方としては、筋腹付近に取り付ける必要がある。Reference 電極は、電極電位の測定の際、基準となる電極であり、計測したい筋肉とは関係ない筋肉に貼付する必要がある。センサーを取り付ける前に、アルコールで皮脂や角質を落とす必要があるた

め、市販のエタノールを綿に浸して電極貼付部位の皮膚を拭いた。

Arduino の端子には、ピンソケットが取り付けられておらず、そのままではジャンパワイヤと呼ばれる電線をつなぐことができない。そこで、ピンソケットと MyoWare 筋電センサーとはんだ付けを行った。

3. TensorFlow について

TensorFlow とは、 Google 社が開発した、機械学習のライブラリである。使う際に、 Python3 では、ライブラリを読み込むコマンド import 命令を入力する必要がある。

TensorFlow で重要なのが、テンソルと計算グラフである。厳密な定義は異なるが、テンソルは、フレームワークを使った深層学習の計算上、多次元配列として使用することがある。テンソルを図示したものが [7] にあるので詳しく理解したい方は参照してほしい。簡単に言うと、階数 0 のテンソルがスカラー、階数 1 のテンソルがベクトル、階数 2 のテンソルが行列というように前の階数のテンソルが集まったものが次のテンソルとなっている。

計算グラフとは、計算の過程をグラフにしたもので、ノードとエッジからなる。TensorFlow は以前は計算グラフを構築してから値を流すといった処理である Define and run が主に行われていた。現在は、 Eager Execution と呼ばれる、計算グラフの作成と評価を同時に行う命令的なプログラム環境も使うことができる。

また、 TensorFlow には Tensorboard という構築した計算グラフを可視化できるツールがある。計算グラフは、複雑になると把握しにくくなるため、このようなツールを活用して、計算グラフが目的のものになっているか、確認することができる。今回使用した Keras は、TensorFlow がバックエンドとなったものを用いている。

4. 筋電位による動作識別

4.1 先行研究

筋電位の動作識別については、すでに先行研究があり、深層学習などを用いた学習型と単に閾値を用いる閾値型の動作識別の手法がある。学習型では、細やかな動作が可能になるが、学習する時間が必要でありデータ量が多くなると義手を装着する人に負担をかけてしまう。閾値型は学習は必要なく装着してすぐに使用可能だが、細やかな姿勢を実現するのが難しい。また、学習型の場合は、取り外しによって信号特性が変化してしまうために、再学習が必要になる。ここにビッグデータを利用し、取り外しによる信号の変化である時変性の問題を克服しようという研究もある [1]。

また、 Raspberry Pi と MyoWare 筋電センサーを用いてデータを取得し、深層学習を行ったという Web ページ [2] があったので、研究の最初の頃はこの先行研究を参考にし、データを取得するなど行った。また、深層学習のモデル作成もこのページを参考にした。他にも、YouTube で、実際に手を動かすと動作を識別し、簡易的な義手が動くなどの動画が公開されており、それも将来どのような方向で研究を進めて行けばよいのかと言うことのヒントとなることがあった。

4.2 データの取得方法

データの取得として、 2 章であげたマイコン Arduino と計測機器 MyoWare 筋電センサーを使い、センサーの数を一つに限定して、動作識別がうまく行えるかをデータをとりながら試してみた。

データの取得は、手のひらを 0.5 秒間程度、握った後力を抜いたもの、 0.5 秒間程度、背屈し力を抜いたもの、0.5 秒間程度、掌屈し力を抜いたもの、手首に何も力を加えなかったものの 4 動作計測した。動作区分については、4.4 で述べる。

研究当初は、 10 秒おきに握り動作といった力を入れる動作と力を抜く動作を 3 動作繰り返した。動作無しは力を入れる動作はないので、センサーをつけたまま、力を抜きデータを記録した。研究途中から時間間隔を、 10 秒おきから、 5 秒おきに変更した。なお、時間を測るにあたって、Web 上にタイマーがあり、それを使って力を入れるタイミングを計った。

データの測定は、試行錯誤しながらデータの取り方やセンサーパッドの貼る位置など変化させていかにデータがうまく取れるか試した。その際に、深層学習は、データが重要であることが分かった。また、筋電位のデータを取り、実験を繰り返すことによって、深層学習の特徴が少しではあるが分かってきた。

今回は、ビッグデータを用いずに少ないデータ数で判定を行った。ビッグデータの方が、特徴を良く捉えておりそちらの方が有用なこともあるが、少ないデータを使用することにもメリットがある。比較的少ないデータでは、学習の際に時間が短くなることである。用いるコンピュータとデータ数によって学習時間が異なるので一概に言えないが、この研究で用いたデータ数では、 Google Coraboratory で GPU を使用すると学習は 1 分以内で計算できた。データが大きいと、他にも、時変性などの問題をクリアできるメリットがあるが、学習するのに時間がかかり、実際に筋電義手を動かすまで時間がかかる場

合がある。

学習だけでなく、実際、手を失った場合その部位によっても、筋電義手を皆が同じ位置に装着できるかどうかが問題となってくる。そのため、いろいろな位置にセンサーを着け、部位を変えてみると言うのは、有用なことだと考えられる。

4.3 筋電位発生の仕組み

この研究では筋電位を取り扱っているため、筋電位発生の仕組みをある程度知っておく必要がある。[4] を参考にした。筋電位とは、筋繊維が収縮し、そのときに発生する活動電位のことである。筋肉収縮の流れとして、脳からの運動の司令が、運動ニューロンにより伝えられ、神経筋接合部を経て、筋肉を収縮させる。神経の活動電位が筋収縮と筋活動電位を起こす仕組みを少し詳しく述べる。

1. 神経の活動電位が神経筋接合部に到達する。
2. アセチルコリンがシナプス間隙に放出される。
3. アセチルコリンがアセチルコリン受容体に結合することにより、アセチルコリン受容体が活性化する。
4. 電気化学的濃度勾配によって Na^{+} が流入し、脱分極が生じる。活動電位は、筋形質膜に沿って伝わり横行小管に入る。
5. 横行小管を介し、筋小胞体から Ca^{2+} が放出される。
6. 筋繊維が収縮し活動電位が発生する。

この様に、発生した筋繊維の活動電位が複数集まり、その和が筋電位（複合活動電位）として計測される。

4.4 動作区分

動作区分として、握り動作、背屈、掌屈、動作なしがある。今回は、センサーをつけながら右手でマウスやコンピューターの操作を行ったため、貼付部位は左腕とした。したがって、図 3 は左手の写真となっている。

教師ラベルとして、握り動作は 0、背屈は 1、掌屈は 2、動作なしは 3 とした。動作を行う時間は 0.5 秒程度である。また、どの動作も、机の上に腕を置いて動作を行っている。動作無しは、力を抜いてただ机に腕を置いただけである。これら、 4 つの姿勢を区別することを今回の実験の目的とする。

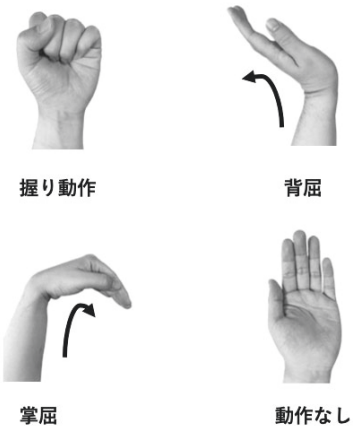

図 3: 動作区分

4.5 筋電位データの取得と訓練データの準備

筋電位データとして、 2 章であげた IEMG を用いた。実験途中から分かったことだが、SVM(Support Vector Machine) の場合、 IEMG が例えば Raw データなどより高い識別率を与えることが分かっている [3]。

適切な筋電位データ取得のアルゴリズムとして、今回は筋電位がピークになるときのインデックスから 600 個データ取得してそれを学習データとした。先述した、先行研究の Web ページでは時間を基準にデータを切り出しているようだったが、MyoWare 筋電センサーのデータの 1 つ 1 つの値の間隔の時間を計算してみると、6 ミリ秒から 7 ミリ秒と常に一定ではなく、時間を基準としてデータを取得すると、値のずれが発生する可能性があった。したがって、ピークを検出するアルゴリズムを使うこととした。また、複数のプログラムを動かすと実行時間が変わってくるという点もあってそのようにした。

ピークを求める関数としては、 scipy.signal にある argrelmax() 関数を用いてピークを検出した。argrelmax() 関数では order という引数で感度を変えることができ、今回でもほとんどの場合で order=500 と設定した。目測できちんと計測されていると思われるデータを選んで学習した場合、最終的にはある程度高い精度で動作識別ができるようになった。

以下に、ピークを求めるのに必要なコードを示す。今、data という、実際の筋電位データが与えられた時のコードである。

```
import numpy as np
import matplotlib.pyplot as plt
time = np.arange(len(data))
from scipy import signal
maxid = signal.argrelmax(data,order=500)
```

```
peak_plot(time,data,maxid)
```

コード 1: ピークを求めるコード

maxid は、最大値を与えるインデックスが入る。なお、ピークを求め、その結果を視覚化するための関数は以下である。

```
def peak_plot(x,y,ID):
    plt.plot(x,y)
    plt.plot(x[ID],y[ID],"ro")
    plt.xlabel("time")
    plt.ylabel("")
    plt.legend("upper␣right")
```

コード 2: peak_plot 関数

これらを使って、 1 つの姿勢に対して、ピークを与えるインデックスから 600 個の NumPy データを 5 つ取得した。したがって、 1 つの姿勢に対しての形状は NumPy 配列で (5,600) つまり 5 行 600 列の行列データとなる。姿勢の数は 4 姿勢なので、訓練データの NumPy 配列の形状として (20,600) つまり 20 行 600 列の行列データとなる。この (20,600) の形状を持つデータを**コード** 3 で書かれている x_train のデータとした。以降、データの形状を (20,600) のように表記するので、留意されたい。

補足説明として、図 4 を見てもらいたい。図の左は、握り動作のデータを 1 つのピークが見えるように取り出したものである。右側は背屈のデータを同様に取り出したものである。少し分かりにくいが、動作によって波形が異なっている。

ピークを与えるインデックスとは、図のピークから、真下に向かっておろした線と時間軸との交点の番号（インデックス）である。

具体的には、図の左側、握り動作においては、 ピークを与えるインデックスは、この場合は 845 を指す。右側、背屈においては、 816 を指している。別の言い方をすれば、握り動作においては、 845 のときにピークになっており、背屈では、816 においてピークになっている。また、データとデータの間隔は 1 になっている。

4.6 筋電位データ学習と判定のアルゴリズム

筋電位データの学習であるが、今まで説明してきたように、深層学習を用いて学習を行った。以下に、深層学習のネットワーク構造を示す。全結合層から構成されている。

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,
  Activation, Dropout

model = Sequential()
model.add(Dense(1200, input_dim=600,
  kernel_initializer='he_normal'))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1200))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(4))
model.add(Activation('softmax'))
model.compile(optimizer='Adagrad', loss='
  categorical_crossentropy', metrics=['
  accuracy'])

print(model.summary())

history = model.fit(x_train, t_train, epochs
  =1000, batch_size=8)
```

コード 3: Keras を用いた学習のプログラム

このような、ネットワーク構造で学習を行った。Keras を使ってコードが作られており、比較的簡単にコード作成を行うことができる。

この学習アルゴリズムを用いて model を作成し、その model を用いて予測を行う。予測を行う場合は、predict() メソッドを使うことによって、テストデータ x_test から、簡単に予測することができる。

```
y_test=model.predict(x_test)
```

コード 4: 予測を行うプログラム

4.7 実験の流れと結果

4.7.1 実験初期（訓練データ取得）

研究初期では、 MyoWare 筋電センサーの位置を一箇所に決めデータ収集を行った。計測位置は図 5 である。

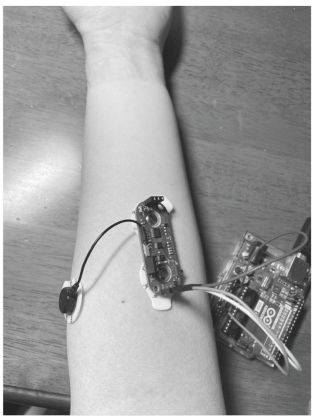

図 5: 研究当初のセンサー位置

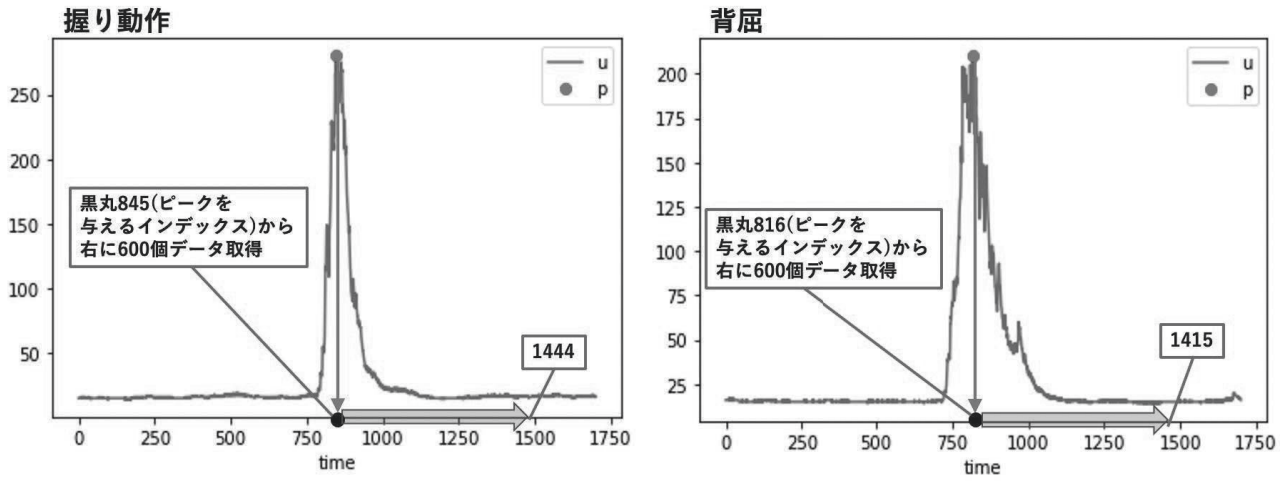

図 4: IEMG データ取得概要

当初は、筋肉の部位によって、特徴データが変化すること分からなかった。そのため、握るなど力を込め、もう一方の手で触って筋肉の収縮を確かめ、そこにセンサーパッドを貼るということをした。また、 Web も参考にした。

最初は、 MyoWare センサーデータが、力をこめた状態ではない場所で、上昇してしまうという問題に悩まされた。

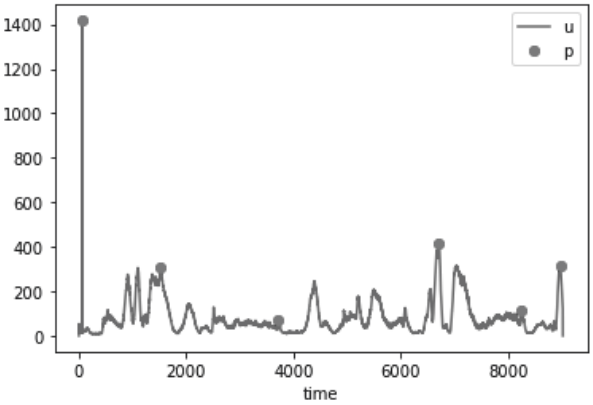

図 6: 研究当初の握りデータ

図 6 のデータの横軸は時間、縦軸は筋電位の値である。この握りデータには、多くの問題がある。まず、インデックスが 0 の付近で、大きなピークがあり明らかに外れ値となっている。そして、外れ値のところでピークが検出されている。

また、見た目にも、どこが握ったときのピークで、どこが関係ないときのピークかがわからない。握ったときと関係なく力が入ったところがピークとして検出されてしまっている。このデータからは、形状が (5,600) の特徴データを得ることができなかった。

このようになってしまった原因として、計測していない時に力を完全に抜かなかったためと考えた。また、目的の動作以外に腕に力がかかる姿勢で計測すると、このように値が乱れると考えた。そこで、次の計測から姿勢に気をつけることと目的の動作以外は腕の力を抜くということを行った。また、 Arduino IDE のシリアルプロッタを見ながら値の上下が落ち着いたときから計測を始めた。

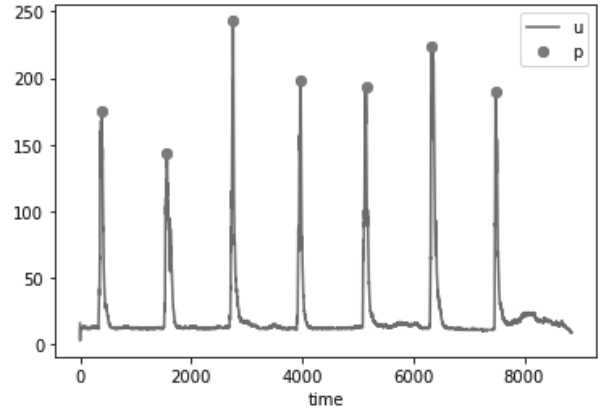

図 7: 注意をして測り直した握りデータ

次に、先程のことに気をつけながら計測した、握り動作の筋電位データを図 7 に示す。このデータでは、ピークの点がちょうど、握った時、筋電位が上がった位置と重なっている。ただし、このような特徴を捉えたデータが、いつも得られるというわけではなかった。

先程の argrelmax() 関数で見つけられた最大値のインデックスから 600 個ほどデータを取り、(5,600) つまり、特徴量が 600 個のデータ 5 個分とした。これを学習データに使った。ここには握ったときのデータしか示してないが、同様に、 (5,600) の背屈したデータ、(5,600) の掌

屈したデータ、(5,600) の動作なしのデータを計測し、4 つのデータを結合した。これで合計 (20,600) のデータが作られ、訓練データとした。

テストデータとしては、新しく取得したデータに argrelmax() 関数を適応し、学習データと同じ様に求めた。ただし、argrelmax() 関数で毎回 5 つのピークを得られない場合やそれ以上のデータが得られた場合は、その数のデータを使った。

4.7.2 実験第 2 段階（データ形式の変換）

先行研究に沿って、このデータをそのまま高速フーリエ変換した。高速フーリエ変換のコードは以下である。

```
import numpy as np

dt=0.005
N=600

F = np.fft.fft(x_train)
freq = np.fft.fftfreq(N, d=dt)

Amp = np.abs(F/(N/2))

x_train=Amp[:, 1:int(N/2)]
```

コード 5: 高速フーリエ変換のコード

サンプリング周期 dt は 0.005、サンプル数 N は、600 とした。高速フーリエ変換では意味のあるデータは、0 番目の 0Hz のデータを除いた 1 番目から N/2 までとなる。従って、全データ数は 299 個となる。結果は、4 つから 6 つのテストデータに対して、握り動作が 83.3%、背屈の精度が 0%、掌屈の精度が 66.6%、動作無しの精度が 100%であった。

この実験では、背屈が全部掌屈と識別されてしまった。握り動作が掌屈、掌屈が握り動作と誤識別されるケースがあった。動作無しは、全てのテストデータの値に対して識別確率が 50-60%台であったが、他の 3 つの動作の識別率がそれほど高く無かったため、動作なしは全てが正しく識別された。この実験での問題点は、背屈のデータが全く正しく識別されていないこと、掌屈の精度が 66%程度と低いこと、握り動作と掌屈動作がお互い誤識別を起こしていたことであった。

次に、握り動作、背屈、掌屈、動作無しが含まれている研究当初の訓練データを分析することによって次のことが分かった。動作なしは他のデータと波形が明らかに異なっており、識別率が高くなるであろうことが分かった。残りの 3 動作は、それぞプロットした波形の見た目が異なっていたので、識別率が高くなることが予想され、高速フーリエ変換を行わずに、動作の識別を行ってみた。

この条件だと、4 つから 6 つのテストデータに対して握り動作は 100%、背屈は 25%、掌屈は 50%、動作なしは 100%の動作識別率であった。

つまり、高速フーリエ変換を施すより、何も処理としないほうが精度が高かった。

この何も処理しないデータと高速フーリエ変換を施したデータを結合させると特徴量が増え、識別率がさらに上がるのではないかと考えたためデータ結合を行った。600 個の特徴量の時間データの後に、299 個の特徴量の高速フーリエ変換の値を結合させた。1 つの動作あたり合計 899 個の特徴データで、(20,899) のデータを学習データを用いて学習を行った。

結果として、4 つから 6 つのテストデータの対して、握り動作は、100%の識別率であった。背屈は 25%の識別率であった。掌屈は 33.3%で握り動作との誤識別が多かった。動作無しは 100%の識別率であった。全体で見ると、高速フーリエ変換した場合より、同等かそれ以下の識別率であったため、データ量のコストを考えてこの方法を使うのは中断した。

4.7.3 実験第 3 段階（データ取得位置）

その後、考えた結果、センサパッドを貼る位置が関係しているのではないかと言う考えに至り、センサパッドを筋肉の書籍 [5] や関連論文 [6] を参考に、3 動作である握り動作、背屈、掌屈に主に関連している筋肉を調べた。

握り動作では、研究当初から貼付した場所で高い動作識別精度が得られていたので、図 5 で示した位置にセンサパッドを貼った。掌屈については、先程の筋肉の書籍では、橈側手根屈筋が最も手首の中で強力な屈筋とあったが、関連論文では、浅指屈筋が手根関節の屈曲に関わる筋肉として一番寄与度が高いとあった。解剖学書では、浅い層の筋肉として、橈側手根屈筋があげられていたので、ここでは、橈側手根屈筋の部位に貼っているとする。背屈では、総指伸筋の部分にセンサパッドを貼った。なお、動作なしは、どの筋肉でも同様な筋電位であるので、3 動作のいずれかと一緒に測った。図 8 に筋肉の部位の概略とセンサーの貼付位置を示す。

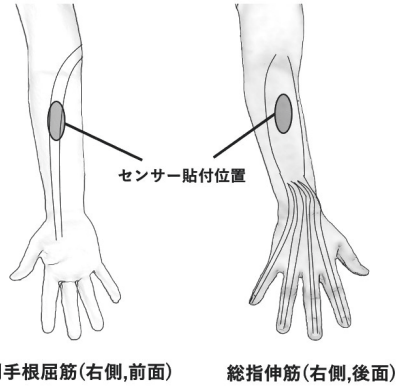

図 8: 筋肉部位とセンサー貼付位置

上のとおり書籍を調べて位置を変更し、識別率の変化を見た。ただし、握り動作と動作無しでは位置を変更しなくても精度が高いのでここでは言及しない。5 つのテストデータに対して、背屈では 100%の動作識別率、 4 つのテストデータに対して掌屈では、75%の動作識別率となった。この時の時点で、センサパッドの位置と余計な力を入れすぎずに測るということに気をつけて測れば、前よりは高い精度が得られた。

4.7.4 実験第 4 段階（データ取得位置）

センサパッド貼付位置を変更して、何回も計測しているときに気がついたのだが、センサの取り外し前と後でデータが異なるということに気づいた。これは、論文 [1] であげられていた時変性だと考えた。そのため、センサーで学習データを計測し、その後、センサーをつけたまま、時間 (3 分、 20 分、 30 分) を変えて計測した。

握り動作では、 3 分経ってセンサーを外さずに計測した識別率は 100%、20 分経ったときは 100%、 30 分たったときでは 80%であった。背屈では、 3 分後、 20 分後、30 分後全てにおいて 100%の識別率だった。掌屈では、3 分後が 100%、 20 分後が 60%、30 分後が 80%となった。動作なしは、一定の信号なので、計測しなかった。

4.7.5 実験第 5 段階（リアルタイムの実験）

この研究の最終目標は、筋電位を用いて動作識別をし、カーソルを動かすことである。そのためには、動作をリアルタイムで識別する必要がある。手動でデータの特徴を捉え、入力するとなると、時間がかかるため装着者に、負担をかけてしまい利便性が損なわれるからである。重要な点として、手作業によるデータ収集では、目で見てデータのよさを見極めていたが、リアルタイム処理では、プログラムを使い、学習データを切り出してくる点にある。図 6 のようなデータであると、正しい識別は難しいが、図 7 のようなデータだと難易度は減る。

リアルタイムで学習データを取得するプログラムを作ったところ図 6 のようなデータに対してうまくいかなかった。そこで、データの中で最大を与える点の値の半分を閾値として、それ以下のピークはひろわないプログラムを作った（ピーク検出の関数として argrelmax() 関数から、 find_peaks() 関数に変えた）。このプログラムを使うと、計測のときに気をつけてさえいれば、学習がうまくいくようになった。

その後、得られたデータから動作識別を行う場合、ThreadPoolExecutor を用いての並列処理やシリアル通信と予測を行うプログラムを分離して、動作識別を行ってみた。しかし、うまくいかなかった。そこで、分離したプログラムを一つにして、動作識別ができるか確かめてみたがこれもうまくいかなかった。プログラムを調べたところ、 predict() メソッドの実行時間がかかるということが影響していることがわかり、model だけで予測するようにしたら、若干の遅延と精度に難点があるが、リアルタイムで動作識別ができるようになった。

学習データ取得にかかる時間は約 2 分で、ニューラルネットワークの学習には約 20 秒となっている。ただし、ニューラルネットワークの学習にかかる時間はコンピューターによって異なる。

4.8 実験のまとめ

以上が今回行った実験と研究である。この実験の全てにおいて言えることだが、筋電位を測るときは余計な力を入れずに測らないと、余分な波形が入ってくる。ただ、上で説明した方法だと、いわば、測定方法のポイントを知っている人は比較的きれいな筋電位を取ることができる。しかし、慣れてない人が測定した場合、動作識別率が高いような結果は得られないかもしれない。そこで、今後の課題として、より実用上のため、計測に慣れてない人が測った波形から、意味のある波形を取り出していくといったことが重要になると考えられる。

また、他にもネットワーク構造を変化させたとき、どうなるかの研究も必要になると考えられる。この研究では過学習を抑制するため、ランダムに選びだしたニューロンを消去する Dropout を用いている。他の研究では、Dropout を用いると筋電位の有用な情報まで消されてしまうという報告があり、今後の研究では Dropout を行わないかもしれない。

他にもハードウェアのことになるが、 Arduino センサの回路について、今回深く学ぶことができなかったので、

その部分もフォローしておきたい。

最後に、今回はピークを使ってデータを切り出したが、時間によってデータを切り出して行くのも研究したい。

5 まとめ

ワンボードマイコンの一種である Arduino Uno と簡易筋電センサーの MyoWare 筋電センサーを用いて、手の動作識別を筋電位を用いて行った。深層学習を用いることにより「握り動作」「背屈」「掌屈」「動作無し」の 4 つの動作を高精度で識別することが可能となった。これまでも同様の研究は行われてきたが、本研究の特徴は少ないセンサー数で効率的な動作識別を目指している所である。今後は動作識別の結果を、パソコン操作などにつなげ、身体に障害がある方への支援へつなげていきたい。

参考文献

[1] 山野井祐介 (2018 年度)『筋電義手制御のための信号特徴の時変性を考慮したビッグデータを用いた手指動作推定法』横浜国立大学大学院工学府システム統合工学専攻機械システム工学コース博士論文

[2] 筋電位動作識別参考ページ
https://mou-tsukareta.hatenablog.com/entry/2019/07/19/180948
（2022 年 9 月 1 日アクセス）

[3] 乾大祐、伊藤聡、佐々木実 (2013)「筋電位からの手動作推定における特徴量と SVM カーネルパラメータについての実験的考察」日本機会学会論文集（C 編）79 巻 808 号 (2013-12)

[4] Gerard J.Tortora/Bryan Derrickson 桑木共之・黒澤美枝子・高橋研一・細谷安彦 編訳 (2019)『トートラ 人体の構造と機能 第 5 版』丸善出版

[5] 河合良訓監修 原島広至 (2004)『肉単』NTS

[6] 田中岳 (2012)『表面筋電位による動作事前推定』大阪大学基礎工学部システム科学科生物工学コース卒業論文

[7] テンソルの基礎,
https://www.tensorflow.org/guide/tensor?hl=ja
（2022 年 9 月 1 日アクセス）