ニューノーマルに向けた
加速度計データとRFIDタグ
を使用した清掃活動の認識

１．　背景

現在，世界各地で蔓延している COVID-19 は，社会的な問題となっている．変異種の確認もされ，ますます感染の予防が必要となるであろう．感染予防の一つとして「2020 ユーキャン新語・流行語大賞」年間大賞にも選ばれた「3 密」があるが，この 3 密を避けるために IoT 技術が利用されている．例えば，西菱電機株式会社の提供した「換気お知らせパッケージ」がある．これは，室内の二酸化炭素濃度をモニタリングし，換気が必要なタイミングをメールなどで知らせるものである．このように IoT 技術の利用により，人間の活動を監視することで感染を予防することができるようになってきている．今回は，感染予防に不可欠な清潔な環境維持に向け，清掃活動のモニタリングを簡単にかつ低コストで実現することに焦点を当てる．清掃活動のモニタリングをするにあたり，ディープラーニング技術を導入する．このディープラーニング技術によって，人間の知識の範疇に基づく特徴抽出に匹敵しない成果を可能にしているため，センサベースの活動認識に対して自動的に高水準の特徴抽出ができている．ディープラーニングは，短い活動認識に対して，RNN（Reccurent Neural Network）や LSTM（Long Short-Term Memory），長期的な繰り返しの活動認識には，CNN（Convolutional Neural Network）が適しているなど対象となる活動によって適切な方法がある．このディープラーニングの技術を用いて，清掃活動を認識する．今回は，清掃活動の中でも拭き活動に焦点を当てる．特に飲食店や各家庭において必須となるような「テーブルを拭く」といった動作に焦点を当てることによって，不特定多数の使用する共有スペースでの感染防止に役立つのではないかと考えられる．また，清掃した時間や場所を特定するために近距離無線通信を採用する．

以上のように IoT 技術とディープラーニング技術で清掃活動を認識，近距離無線通信で時間と場所を認識する．センサデータを用いた行動認識に関する研究は近年多く行われているが，これまで清掃活動の認識に着目したアプローチがなかったため，清掃活動のモニタリングに必要なパラメータ評価や初期概念を提示し，今後の研究に貢献できると考えられる．

２．　目的

本実験の目的は以下の 2 点である．

1）加速度センサによる清掃動作の識別

2）RFID タグを用いた清掃場所の認識

３．　システム内容

以下の図 1 に清掃活動の認識シーケンスを示す．

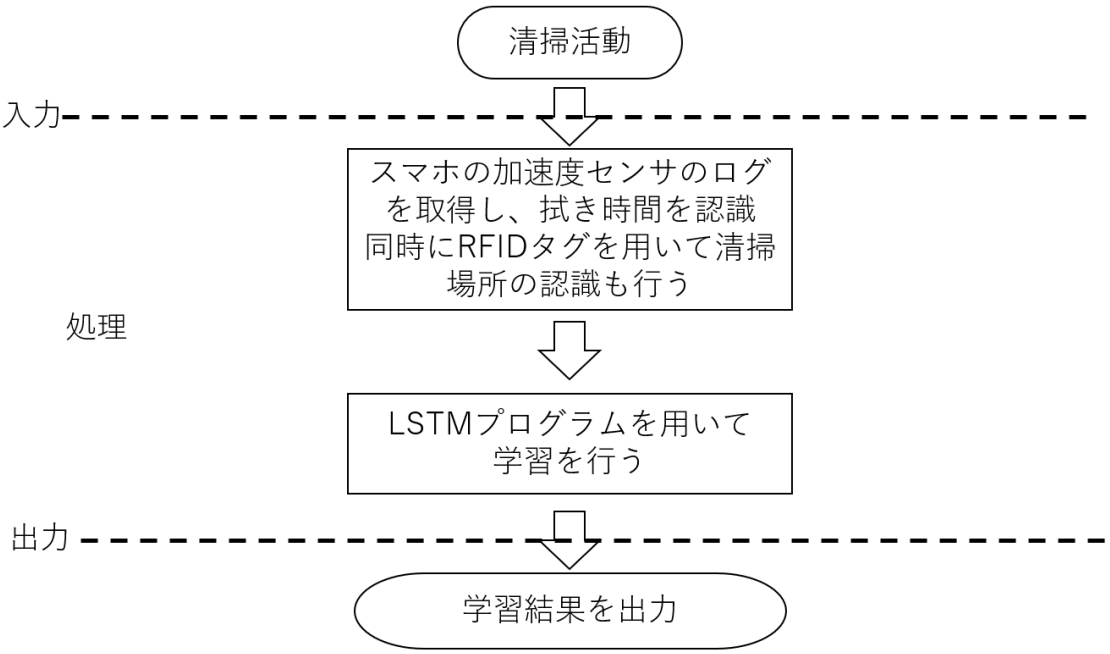

図 1　認識処理の流れ

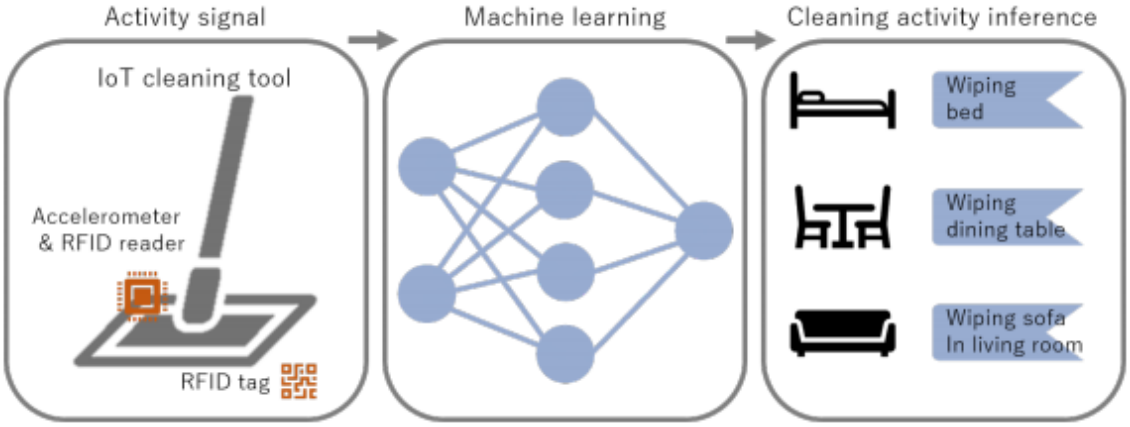

図 2　コンセプト図

上記の図 2 のように清掃活動を簡単に認識できるように清掃用具に加速度センサと RFID リーダーを取り付ける．その後，加速度センサのログからディープラーニング技術を用いることで拭き動作を認識する．ディープラーニングには，LSTM を採用した．

また，以下の図 3 にアプローチの順序を示す．清掃段階と訓練段階の二つの段階に分かれており，訓練段階では清掃活動をモニタリングするために清掃箇所の各ポイント地点に RFID タグを配置する．さらに清掃活動と非清掃活動の加速度データを集めて機械学習モデルをトレーニングする．次に清掃段階ではトレーニングされたモデルに基づいて加速度センサから清掃か非清掃かを識別し拭き動作がされていることを認識する．また，活動場所は配置された RFID タグをリーダーで読み取ることにより認識する．

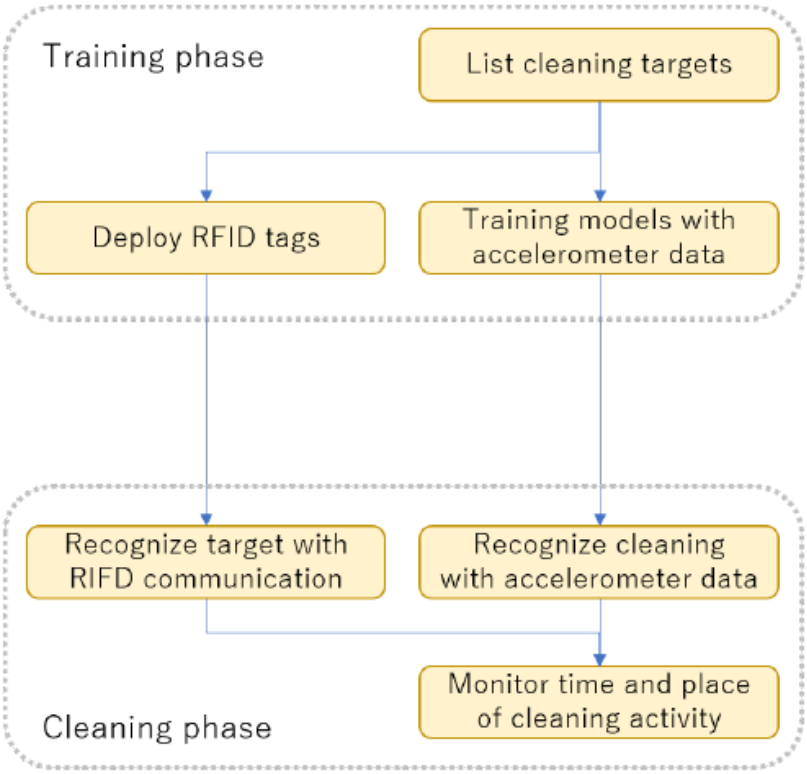

図 3 アプローチの順序

４． 実装

本実験に使用した清掃用具を以下の図 4 示す.

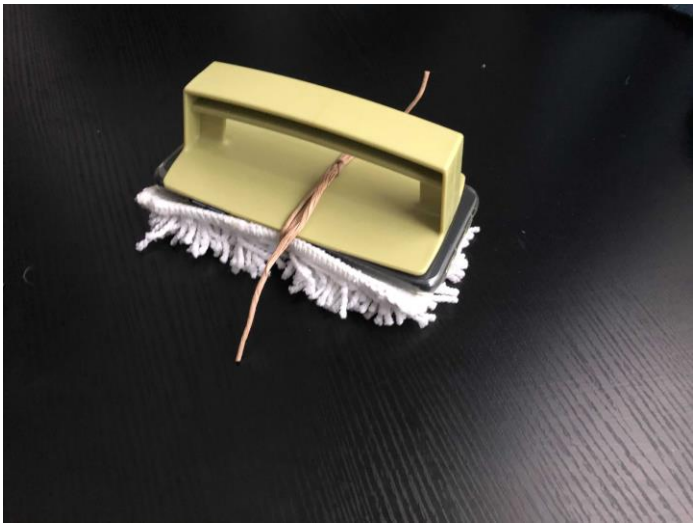

図 4 使用した清掃用具

ハンディワイパーにスマートフォン（Zen Phone4）を装着したものである．スマートフォンは，加速度センサと RFID タグを簡易的に使用するために採用した．加速度センサデータの取得は，Android 用アプリケーションを使用した．付録にこのアプリケーションのソースコードを示す．x，y，z 軸加速度センサのデータとタイムスタンプを取得する．以下の図 5 に動作中のアプリケーションのようすを示す．上から x，y，z 軸の加速度を示す．START ボタンを押すと，加速度の計測が始まり，STOP ボタンを押すと計測が終了する．計測後のデータはファイルに保存される．

図 5　加速度センサアプリの動作

また，以下の図 6 のように実験室のテーブル 4 つに RFID タグを配置し，スマートフォンに搭載した RFID リーダーがタグと通信すると，時刻とタグ ID が記録されるようにした.

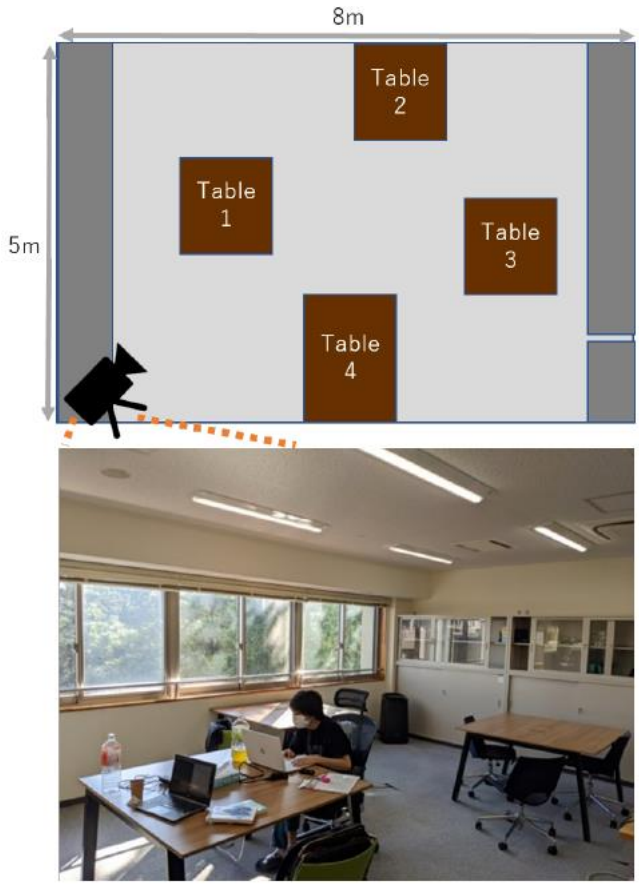

図 6　実験室

実験者がこの清掃道具を持ち部屋を歩き回りながらテーブルを拭いていくという実験シナリオとした．図 7 に拭く動作のサンプルログを示す．往復運動を描いているのが特徴としてみえる．

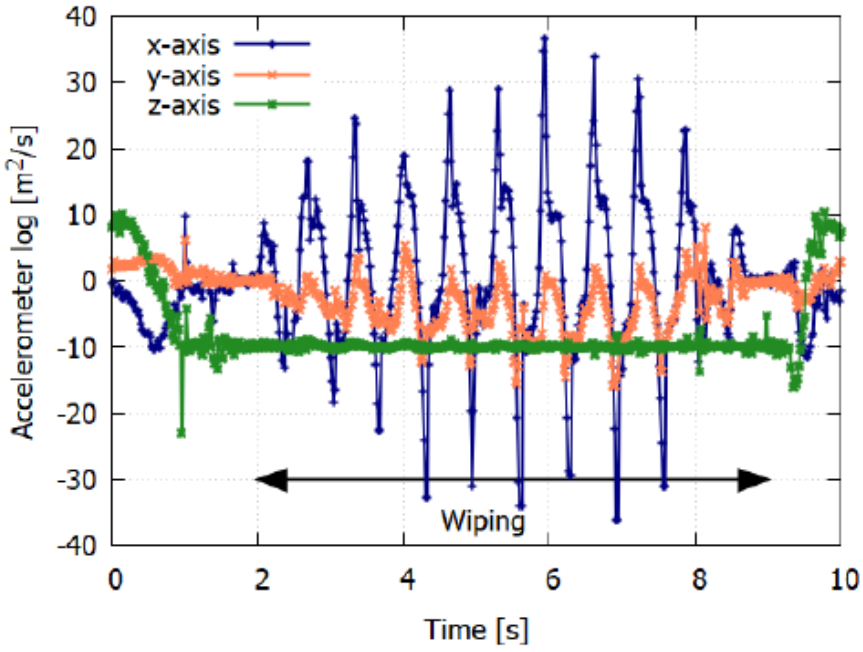

図 7　拭く動作のサンプルログ

加速度センサを利用して収集したデータ 5 時間分にアノテーションをつけた．タイムスタンプに基づき，清掃か非清掃に分類し，時間窓の中でラベルの大部分，すなわち清掃または非清掃を行っているものをアクティビティと定義した．データセットとしてこのアクティビティ 10000 個を取得した．清掃の検出にはアダム最適化を施した LSTM を採用し，隠れ層の数を 100，バッチサイズが 64，エポック数を 30 として行った．データセットを 5 つのグループに分割し，5 分割交差検証を採用した．また，清掃活動を認識するために必要なパラメータ設定を明らかに

するために，サンプリング周波数と時間窓の値を変化させ，機械学習の性能を評価した．LSTM は MATLAB（バージョン:R2020b）を用いて行った．以下に使用した PC のスペックを示す．

表 1 使用した PC のスペック

OS 名	Microsoft Windows 10 Home
バージョン	1909
CPU	Intel(R) Core(TM) i5-7200U CPU @2.50GHz,2712 Mhz,
搭載メモリ容量(実装 RAM)	8.00GB
HDD 空き容量(C ドライブ)	20GB

５． 実験結果

以下に実験結果を示す．

表 2 実験結果

sampling rate [Hz]	Time-window size [s]	Accuracy	Recall	Precision	F1-measure
10	0.1	0.973	0.981	0.965	0.973
	0.5	0.988	0.992	0.984	0.988
	1.0	0.991	0.993	0.988	0.990
	2.0	0.991	0.988	0.994	0.991
20	0.1	0.986	0.989	0.983	0.986
	0.5	0.990	0.992	0.987	0.990
	1.0	0.992	0.991	0.993	0.992
	2.0	0.988	0.993	0.983	0.988
50	0.1	0.990	0.992	0.987	0.989
	0.5	0.993	0.993	0.992	0.992
	1.0	0.992	0.994	0.989	0.992
	2.0	0.982	0.976	0.988	0.982
100	0.1	0.992	0.995	0.988	0.992
	0.5	0.993	0.993	0.993	0.993
	1.0	0.990	0.993	0.988	0.990
	2.0	0.986	0.989	0.983	0.986

上記の表は，サンプリング周波数と時間窓の値を変化させたときの評価値を示している．評価値には，Accuracy（精度），Recall（再現率），Precision（適合率），F1-measure（F 値）の 4 つを算出した．それぞれ予測が正しかった割合，網羅できた割合，正確に予測できた割合，適合率と再現率の調和平均を示す．上記の表 2 の結果から適切なサンプリング周期と時間窓のパラメータを設定することで 99%以上の評価を得られることが示された．サンプリング周波数は，20Hz 以上において，時間窓の値に関わらず精度，F 値が 98%以上を保つことができていると考えられる．時間窓に関しては，1 秒程度に設定すると 99%近くの評価が得られていると考えられる．ゆえに今回の

清掃活動の認識には，サンプリング周波数を 20Hz まで下げても認識が十分に可能であり，かつ精度の確保のために時間窓は，1 秒程度に設定することが適切であると考えられる．

６．　考察

今回のレポートでは初期のコンセプトと加速度計データと RFID タグを用いた清掃活動認識の準備段階での結果を提示した．示したアイディアの最終目標は COVID-19 パンデミックの後のニューノーマルな世の中に向けて清潔な環境を維持するために清掃活動のモニタリングを簡単に低コストで行うことである．清掃活動を効率的に認識するために機械学習技術と近距離無線通信を活用した．典型的な清掃活動として，レストランやコーヒーショップ，図書館などの公共な場で感染を防ぐために重要である「拭く」という清掃動作に着目した．今回は，拭くという動作に着目し PoC 実験結果を報告した．さらなる実験のための必要なパラメータを明確にするために，Accuracy とサンプリングレート及び時間窓のサイズの関係を評価した．このアイディアは，ニューノーマルな時代のライフスタイルを確立するために清掃活動を効率的に監視することに大きく貢献すると思われる．これは，将来的に様々な清掃用具や大規模なデータセットを用いてさらに包括的な評価を行うことの第一歩となるだろう．

今回は，拭き動作のみの認識であり，基本的に z 軸方向の加速度は清掃時には一定であった．ゆえにテーブルを拭くなどといった動作に関しては，z 軸方向が一定である箇所を見れば清掃を認識できると考えられる．しかし，壁に沿って清掃するなどといった条件では，3 軸すべてが影響する清掃であるため認識が難しくなるのではないかと考えられる．また，手を動かすテーブル拭きと異なるモップや掃除機を使った床に対する清掃の認識も今後の課題といえると考えられる．さらに，様々な清掃動作が認識できるようになれば，図書館やカフェテリアなどだけでなく監視カメラによって監視が不可能な場所であるトイレの掃除などもモニタリングすることができるようになると考えられる．

今回は周波数 10Hz でも高めの精度が得られたため，今後はもっと少ない周波数で試してみるのがいいと思われる．また，非清掃動作には装置を持って歩くという動作が大半を占めており精度が全体的に高くなったと考えられるため，今後は立ち止まってよそ見をしたり座ったりする動作などいろいろなパターンの動作を含めてより一般的な動作データを集めていくのも今後の課題である．また今回 4 人でデータを集めた事を考えると 1 人が集めたデータをテストデータにし，他の 3 人が集めたデータを学習データにするというやり方を用いると識別率はより高くなると考えられる．今後はそのような方法を用いて検証していきたい．

さらに，どの動作がどの清掃用具を用いた清掃なのかの判別，認識も今後の検討課題となると考えられる．これらの認識ができてくると，より簡易で低コストな方法の考案をし，一般的に活用できるモニタリングができることを期待する．

付録

加速度センサのアクティビティソースコード

```
package com.example.myapplication;

import android.os.Bundle;

import android.hardware.Sensor;

import android.hardware.SensorEvent;

import android.hardware.SensorEventListener;

import android.hardware.SensorManager;

import android.util.Log;
```

```
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Timer;
import java.util.TimerTask;
import androidx.appcompat.app.AppCompatActivity;
public class MainActivity extends AppCompatActivity implements SensorEventListener {
    private Sensor accSensor;
    private TextView mX;
    private TextView mY;
    private TextView mZ;
    private SensorManager manager;
    public int button_flag = 0;
    public int count = 0;
    public void onAccuracyChanged(Sensor sensor, int n) {
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        manager = (SensorManager)this.getSystemService(SENSOR_SERVICE);
        accSensor = this.manager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        Button start_btn = (Button) findViewById(R.id.button);
        Button stop_btn = (Button) findViewById(R.id.button2);
        start_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                button_flag = 1;
            }
        });
        stop_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                button_flag = 0;
                count = count + 1;
            }
        });
        mX = (TextView)this.findViewById(R.id.text_view1);
        mY = (TextView)this.findViewById(R.id.text_view2);
```

```
        mZ = (TextView)this.findViewById(R.id.text_view3);
    }
    @Override
    protected void onPause() {
        super.onPause();
        this.manager.unregisterListener((SensorEventListener)this, this.accSensor);
    }
    protected void onResume() {
        super.onResume();
        this.manager.registerListener((SensorEventListener) this, this.accSensor, 10000);
    }
    public void onSensorChanged(SensorEvent sensorEvent){
        if (sensorEvent.sensor.getType() ==  Sensor.TYPE_ACCELEROMETER) {
            if(button_flag==1){
                Log.d("SENSOR_DATA", "TYPE_ACCELEROMETER1 = " + String.valueOf(sensorEvent.values[0]));
                Log.d("SENSOR_DATA", "TYPE_ACCELEROMETER2 = " + String.valueOf(sensorEvent.values[1]));
                Log.d("SENSOR_DATA", "TYPE_ACCELEROMETER3 = " + String.valueOf(sensorEvent.values[2]));
                mX.setText(String.valueOf(sensorEvent.values[0]));
                mY.setText(String.valueOf(sensorEvent.values[1]));
                mZ.setText(String.valueOf(sensorEvent.values[2]));
                try{
                    FileWriter fw = new FileWriter(getFilesDir().getPath()+"/test" + count + ".csv", true);
                    PrintWriter pw = new PrintWriter(new BufferedWriter(fw));
                    //内容を指定する
                    pw.print(String.valueOf(sensorEvent.values[0]));
                    pw.print(",");
                    pw.print(String.valueOf(sensorEvent.values[1]));
                    pw.print(",");
                    pw.print(String.valueOf(sensorEvent.values[2]));
                    pw.print(",");
                    pw.print(String.valueOf(0));
                    pw.println();
                    //ファイルに書き出す
                    pw.close();
                    //終了メッセージを画面に出力する
                    System.out.println("Complete Outputs");
                } catch (IOException ex) {
                    //例外時処理
                    ex.printStackTrace();
                }
            }else{
                mX.setText("STOP");
                mY.setText("STOP");
```

```
                mZ.setText("STOP");
            }
        }
    }
}
```

加速度センサのレイアウトファイルソースコード

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="#afe"
    tools:context="com.example.myapplication.MainActivity">
    <TextView
        android:id="@+id/text_view1"
        android:layout_marginTop="20dp"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"
        android:textSize="20sp"
        android:textColor="#00f"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/text_view2"
        android:layout_marginTop="20dp"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"
        android:textSize="20sp"
        android:textColor="#00f"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:id="@+id/text_view3"
        android:layout_marginTop="20dp"
        android:layout_marginStart="20dp"
        android:layout_marginLeft="20dp"
        android:textSize="20sp"
        android:textColor="#00f"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="95dp"
        android:text="START" />
    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="95dp"
        android:text="STOP" />
</LinearLayout>
```

LSTM のソースコード

```
N=600; %5 分で N 個
Skip=4; %何行飛ばし
P=20;
Line=floor(30000/(Skip+1));
a=floor(Line/N);
X1Train = zeros(P*N,3*a);
Y1Train = zeros(P*N,1);
current = 1;
for i = 1:P
    filename = append('data/clean',num2str(i),'.csv');
    M1 = csvread(filename,0,0,[0,0,30000,2]);
    M1= M1(1:Skip+1:30000,:);
    M2 = csvread(filename,0,3,[0,3,30000,3]);
    M2= M2(1:Skip+1:30000,:);
    for kk = 1:N
        clean = 0;
        unclean = 0;
        for jj = 1:a
            % input
            for ii = 1:3
                X1Train(current,(jj-1) * 3 + ii) = M1((kk-1) * a + jj,ii);
            end



            % label
            if M2((kk-1) * a + jj,1) == 1
                clean = clean + 1;
```

```
            else
                unclean = unclean + 1;
            end

        end
        if clean > unclean
            Y1Train(current,1) = 1;
        else
            Y1Train(current,1) = 0;
        end
        current = current + 1;
    end
end

X2Train = zeros(P*N,3*a);
Y2Train = zeros(P*N,1);
current = 1;
for i = 1:P+2
    if i == 4||i==11
        continue
    end
    filename = append('data/unclean',num2str(i),'.csv');
    M1 = csvread(filename,0,0,[0,0,30000,2]);
    M1= M1(1:Skip+1:30000,:);
    M2 = csvread(filename,0,3,[0,3,30000,3]);
    M2= M2(1:Skip+1:30000,:);
    for kk = 1:N
        clean = 0;
        unclean = 0;
        for jj = 1:a
            % input
            for ii = 1:3
                X2Train(current,(jj-1) * 3 + ii) = M1((kk-1) * a + jj,ii);
            end

            % label
            if M2((kk-1) * a + jj,1) == 1
                clean = clean + 1;
            else
                unclean = unclean + 1;
            end

        end
```

```
        if clean > unclean
            Y2Train(current,1) = 1;
        else
            Y2Train(current,1) = 0;
        end
        current = current + 1;
    end
end



XTrain = [X1Train;X2Train];
YTrain = [Y1Train;Y2Train];

% % % define LSTM

XTrain = num2cell(XTrain,2);
YTrain = categorical(YTrain);

data=crossval(@gakushuu,XTrain,YTrain,'KFold',5);
data=mean(data)

function data=gakushuu(xtrain,ytrain,xtest,ytest)
numFeatures = 1;
numHiddenUnits = 100;
numClasses = 2;

layers = [ ...
    sequenceInputLayer(numFeatures)
    lstmLayer(numHiddenUnits,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];

% train
miniBatchSize = 64;

options = trainingOptions('adam', ...
    'ExecutionEnvironment','cpu', ...
    'MaxEpochs',50, ...
    'MiniBatchSize',miniBatchSize, ...
    'GradientThreshold',2, ...
    'Shuffle','every-epoch', ...
    'Verbose',false, ...
```

```
    'Plots','training-progress');
net = trainNetwork(xtrain,ytrain,layers,options);
YPred = classify(net,xtest);
accuracy = sum(YPred == ytest)/numel(ytest)*100;
recall=sum(YPred=='1'&YPred==ytest)/sum(ytest=='1');
precision=sum(YPred=='1'&YPred==ytest)/sum(YPred=='1');
data=[accuracy,recall,precision];
end
```