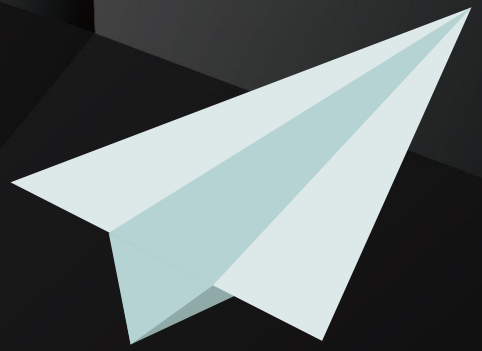


Touch location learning of phone screen basing motion sensors

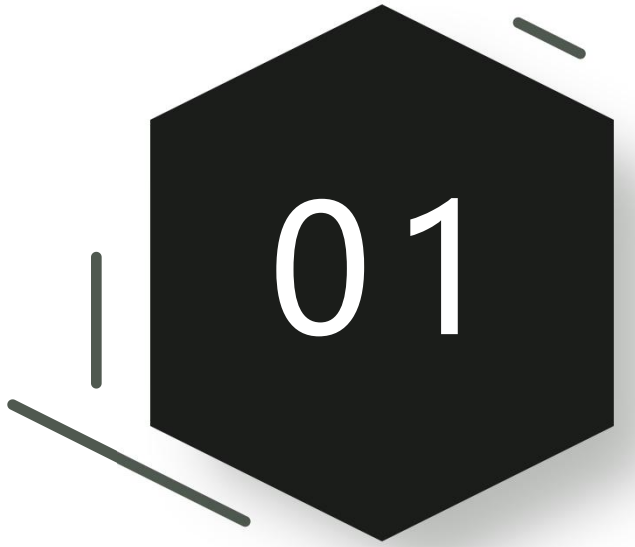
Xiyuan Bao
Sixue Xu
Ziyi Xi

Contents

Former work

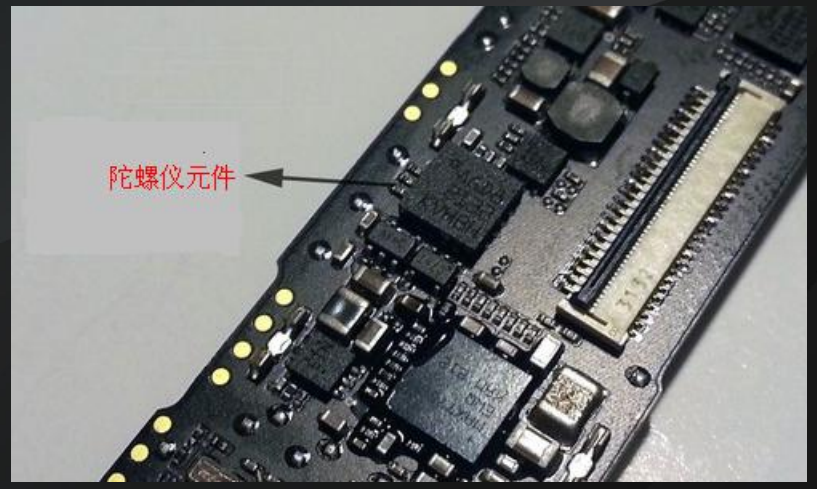
Collect and learn

Future work

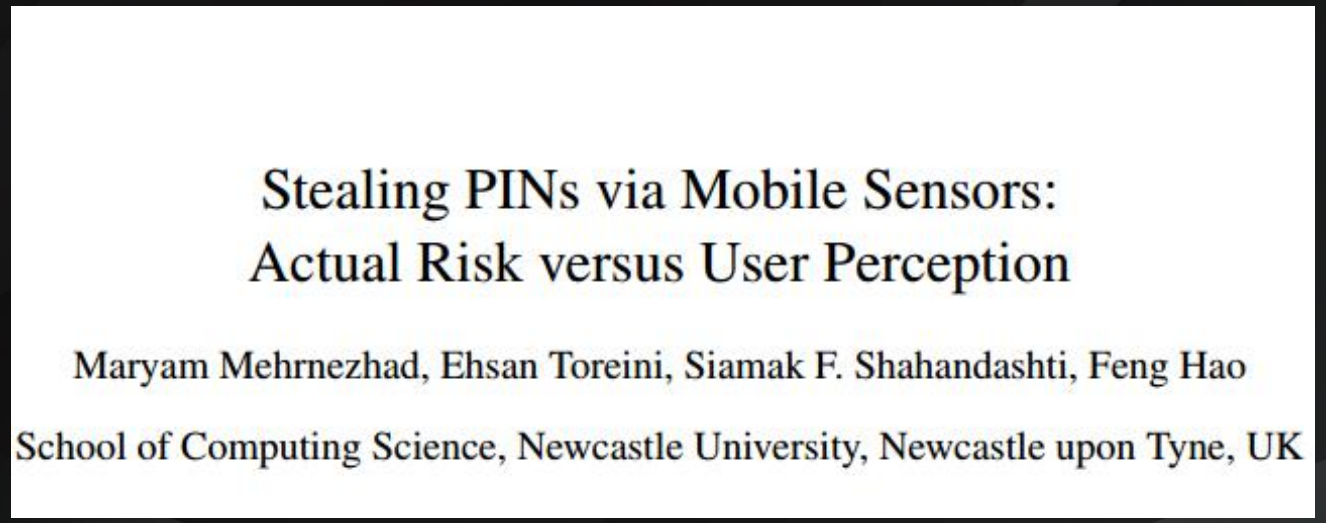


Former work

Motion sensor of smart phone and daily use



- Chasing rotation and direction of wrist using gyroscope to deduce four digit PIN with high accuracy
- Only a few sensors(GPS,camera) as for permission
- App and websites can monitor data from user's sensor freely



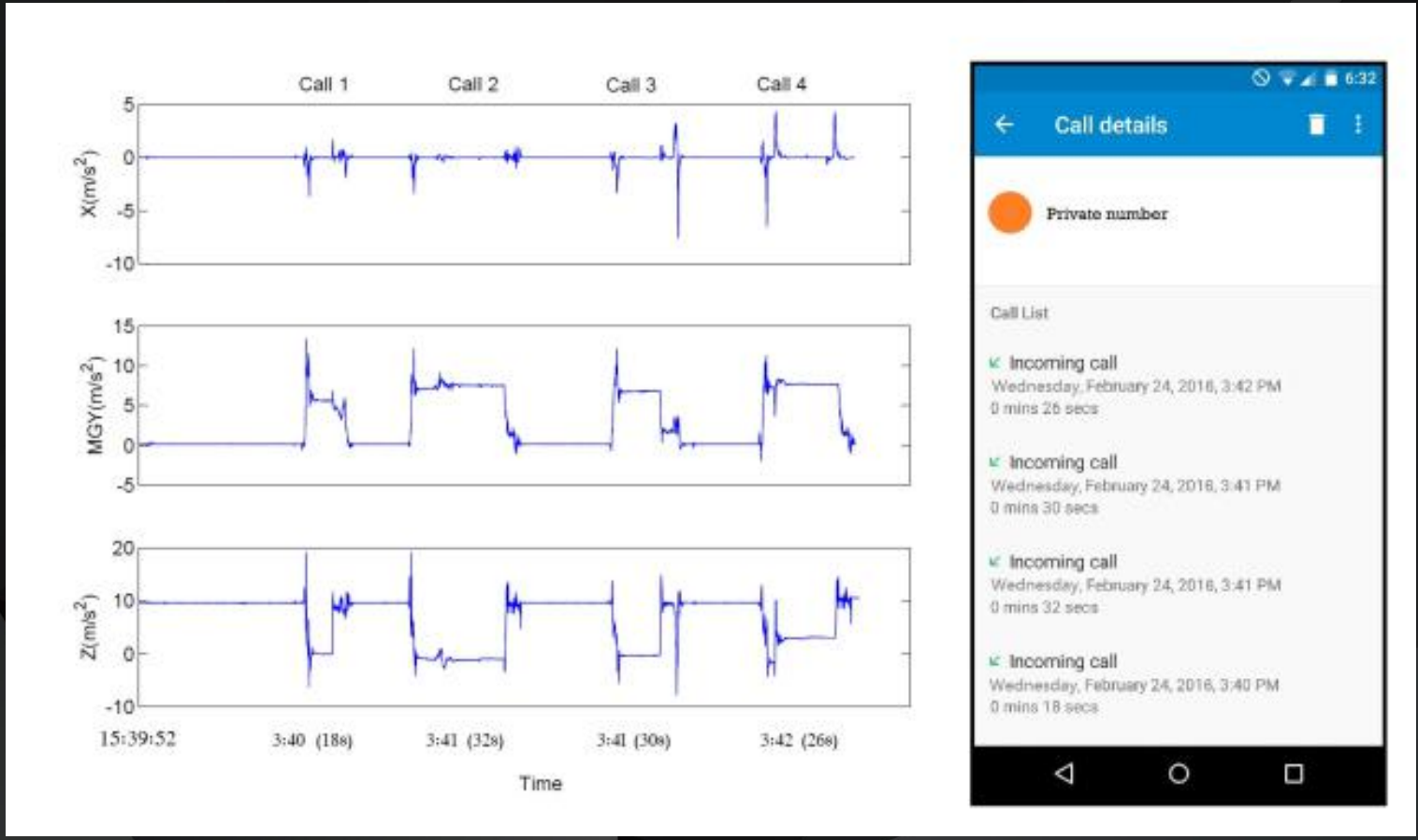
Stealing PINs via Mobile Sensors:
Actual Risk versus User Perception

Maryam Mehrnezhad, Ehsan Toreini, Siamak F. Shahandashti, Feng Hao

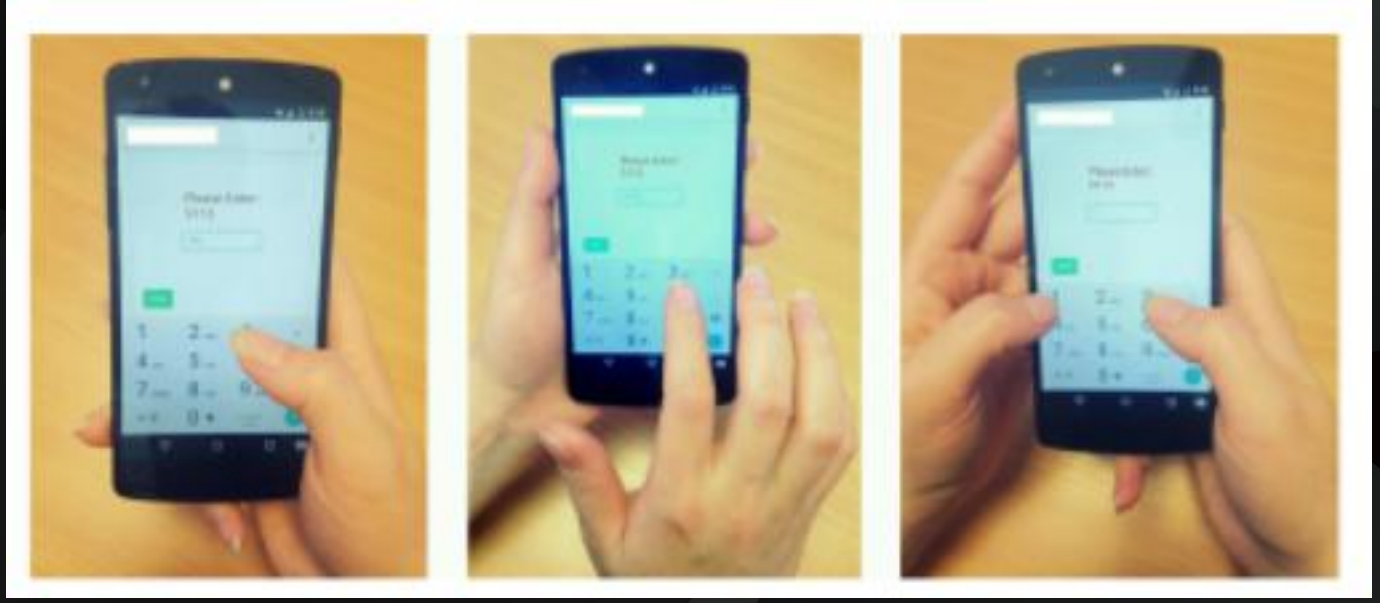
School of Computing Science, Newcastle University, Newcastle upon Tyne, UK

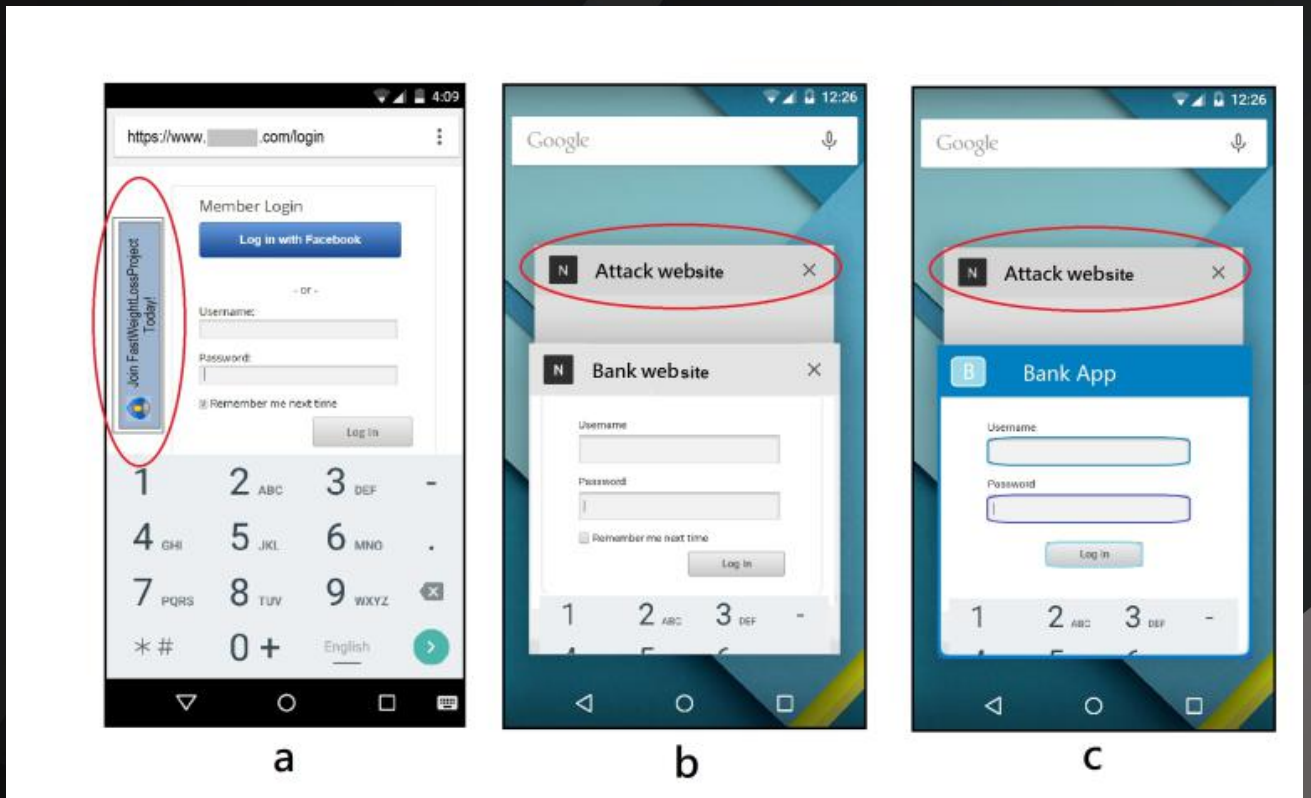
Theory

- Different user/gesture->click number button->different move of cellphone->different wave from sensors



- JS background monitor
- HTML5 GUI shows random PINs for input
- different gesture





- feature extraction using 12 components from 4 sensors
- Consider range,average,energy of the sequence in time and frequency domain(FFT)
- Correlation coefficient

$$R_{AB} = \frac{\mathrm{Cov}(A, B)}{\sqrt{\mathrm{Cov}(A, A) \cdot \mathrm{Cov}(B, B)}}$$

- ANN，2500 records from 10 people X 114 features
- 70%train15%validate15%test
- Matlab with one hidden layer and 1000 nodes

- success rate:one attempts <80%,increase for multiple ,3 times close to100%

Attempts	Multiple-users	Same-user
One	74%	79%
Two	86%	93%
Three	94%	97%

Table 1: PINlogger.js's PIN identification rates in different attempts.

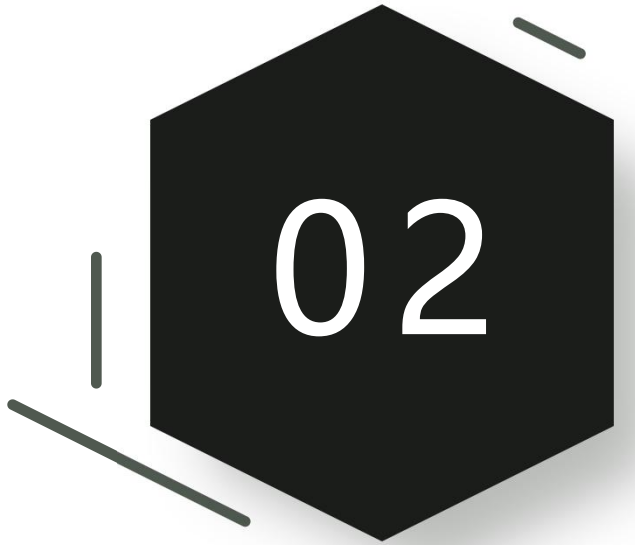
Attempts	Multiple-users	Same-user
One	70%	79%
Two	83%	90%
Three	92%	96%

Table 2: Average digit identification rates in different attempts.

We want to promote

- Study coordinates directly instead of buttons
- recognize more than PIN:complex keys, even all the touch events
- avoid error comparing to buttons

of great importance for secure user's information



Collect and learn

Collecting program

Machine learning

Input test

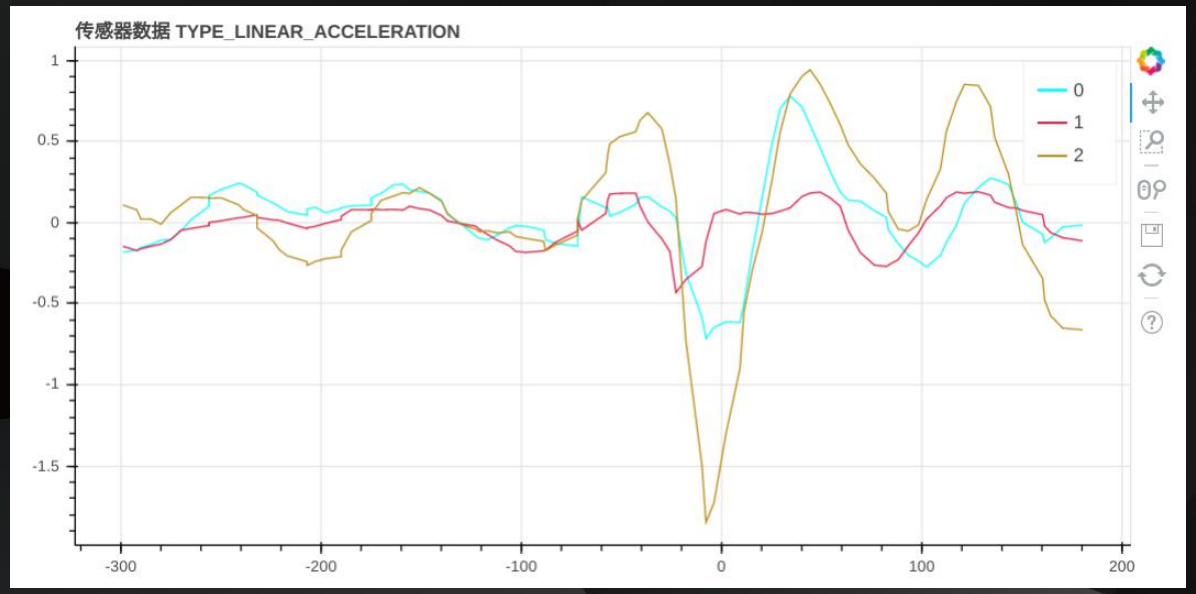
Collecting procedure

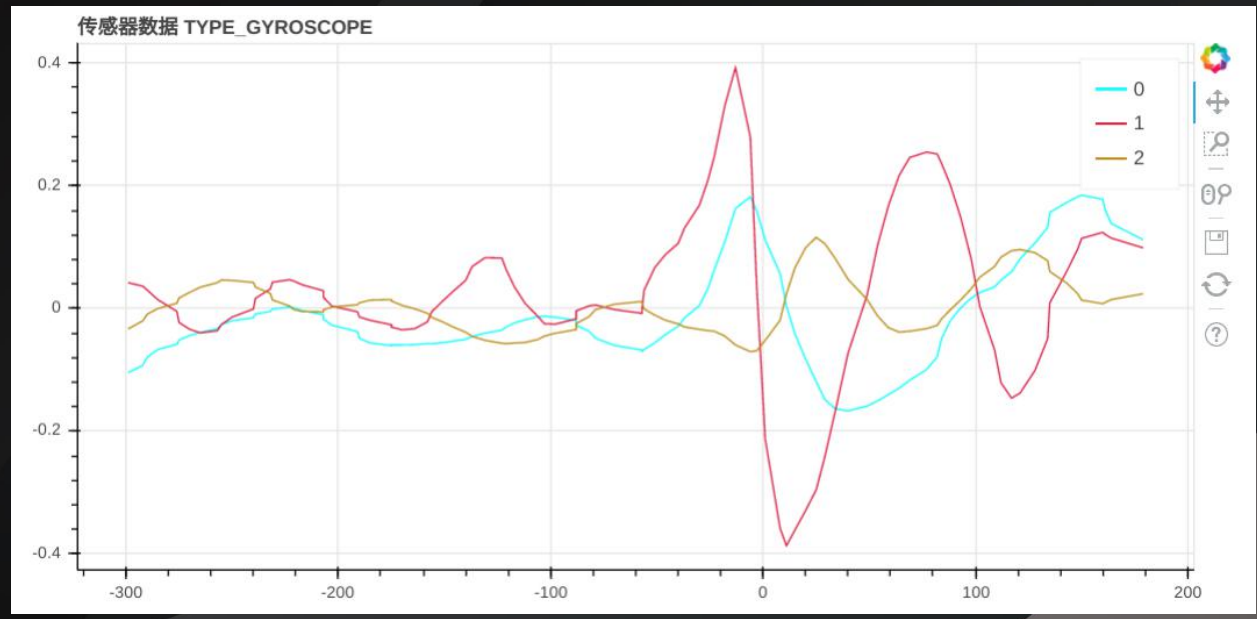
Collecting program:typical touch event

acceleration and rotation of barycentre

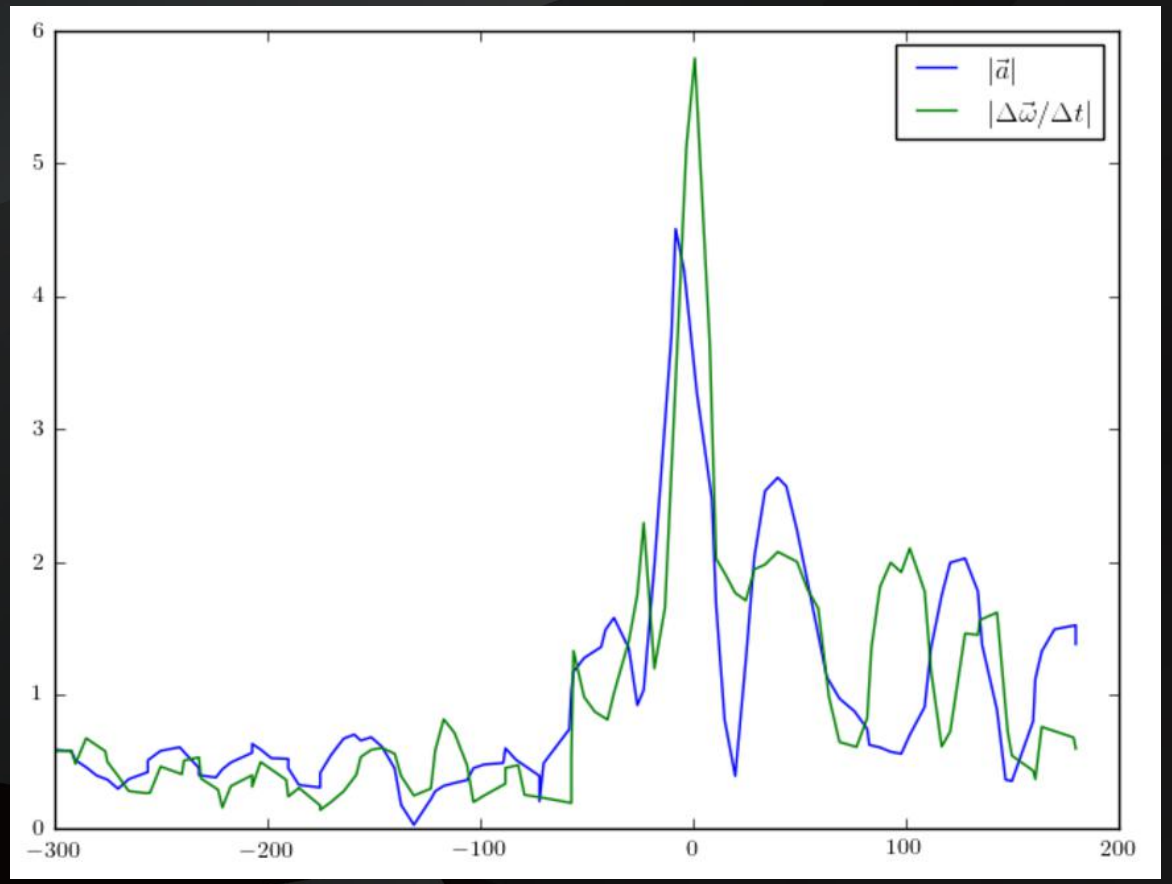
peak at the touching moment:

use theroshold to judge an event



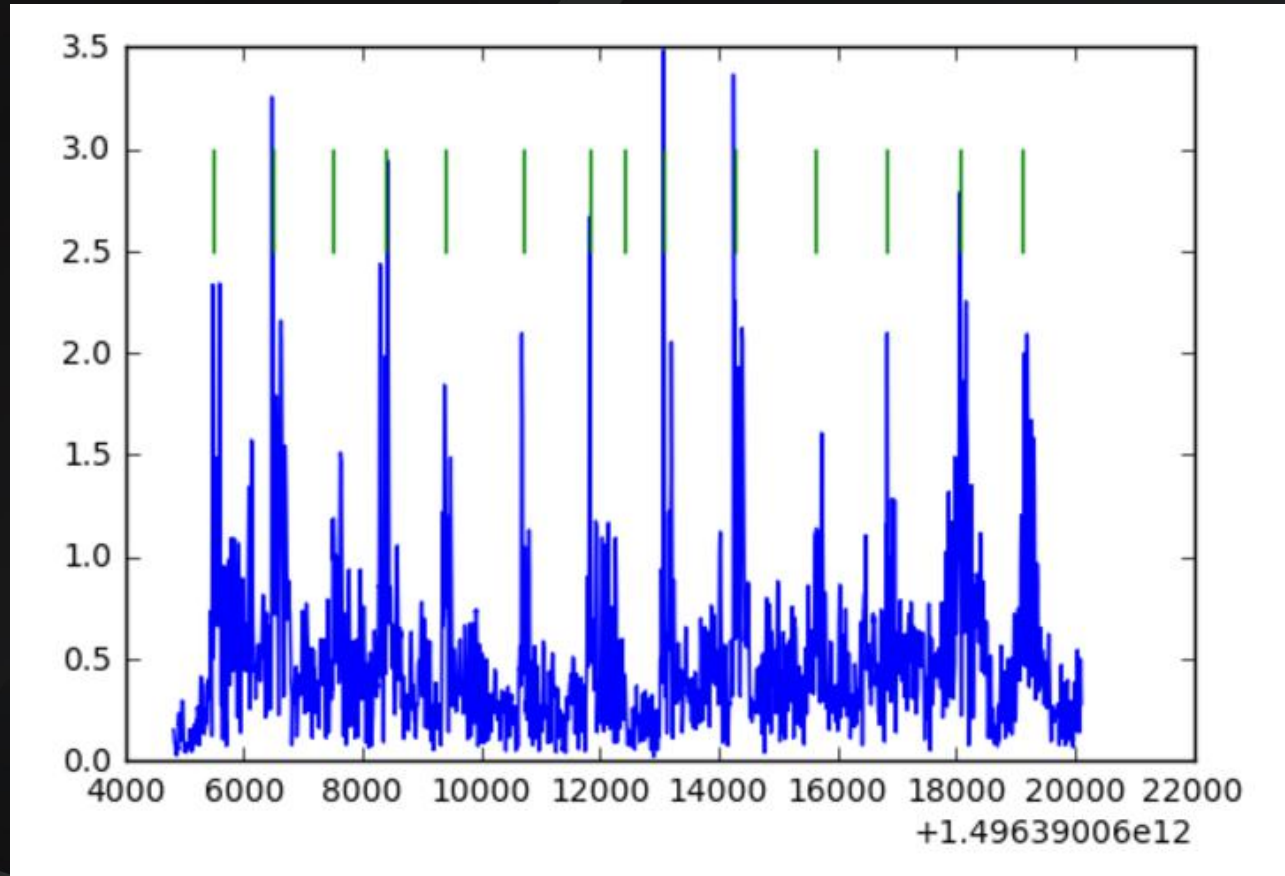


Collecting program:typical touch



time range

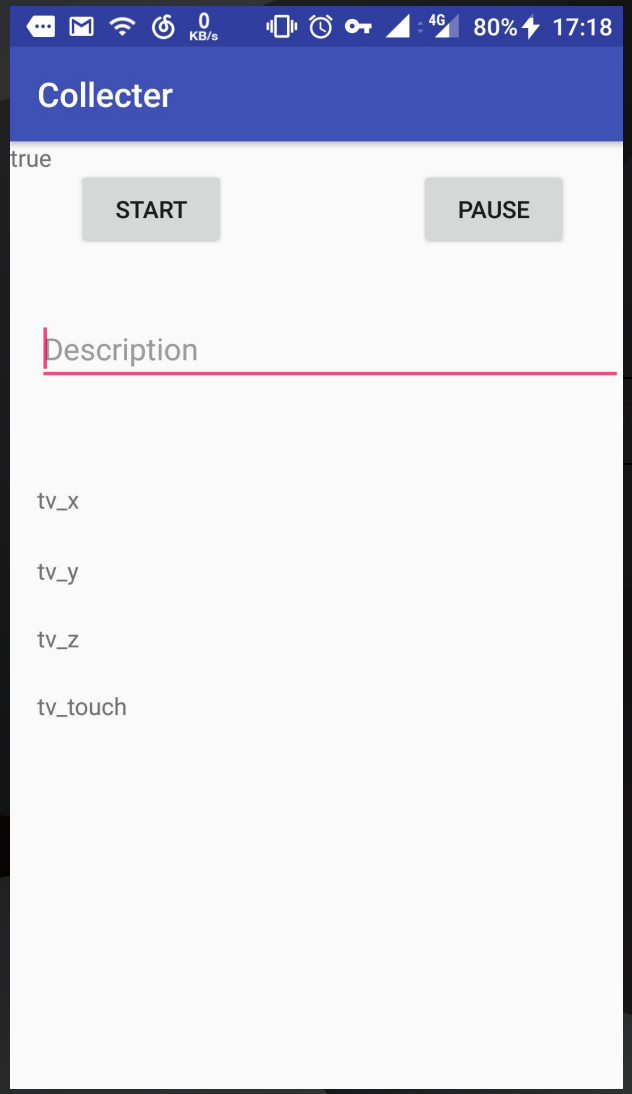
500ms



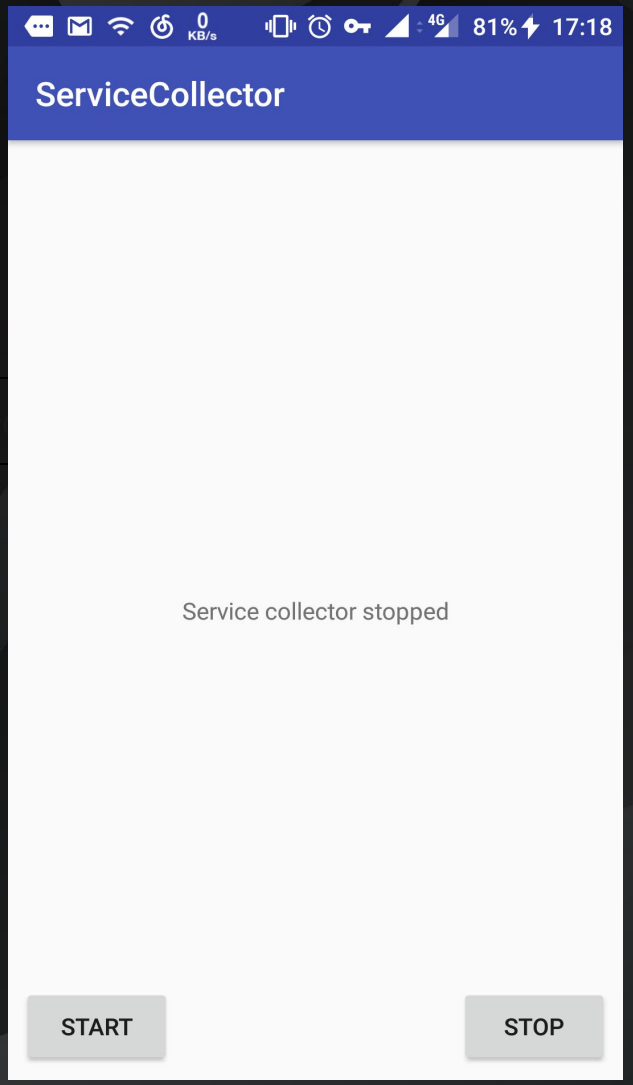
Collecting program：select the sensor

Sensor	Sensor event data	Description	Units of measure
TYPE_ACCELEROMETER	SensorEvent.values[0]	Acceleration force along the x axis (including gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (including gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (including gravity).	
TYPE_GYROSCOPE	SensorEvent.values[0]	Rate of rotation around the x axis.	rad/s
	SensorEvent.values[1]	Rate of rotation around the y axis.	
	SensorEvent.values[2]	Rate of rotation around the z axis.	
TYPE_LINEAR_ACCELERATION	SensorEvent.values[0]	Acceleration force along the x axis (excluding gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (excluding gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (excluding gravity).	
TYPE_ROTATION_VECTOR	SensorEvent.values[0]	Rotation vector component along the x axis (x * sin(θ/2)).	Unitless
	SensorEvent.values[1]	Rotation vector component along the y axis (y * sin(θ/2)).	
	SensorEvent.values[2]	Rotation vector component along the z axis (z * sin(θ/2)).	
	SensorEvent.values[3]	Scalar component of the rotation vector ((cos(θ/2)).[1]	

Collecting program@js:front and back



START
to coolect,
PAUSE
to write,
0,0
at left corner
of top.



Collecting program:code

```
// 第 tap_index 个触碰事件
object[<tap_index>][0].tap_t // 触碰时刻，long
object[<tap_index>][0].tap_x // 触碰坐标x，tap_y 坐标y
object[<tap_index>][1-4].sensor //传感器类型名，String，"TYPE_ACCELEROMETER","TYPE_LINEAR_ACCELERATI
object[<tap_index>][1-4].timestamp //采集时间点，long [采集次数]
object[<tap_index>][1-4].data //采集的数据，float [采集次数][单次采集数据长度]
```

主体代码

```
app/src/main
├── AndroidManifest.xml
├── java
│   └── com
│       └── example
│           └── hzxusx
│               └── collecter
│                   ├── CollectingActivity.java
│                   └── SensorBuffer.java
└── res
    ├── drawable
    ├── layout
    │   ├── activity_collecting.xml
```

Collecting procedure

- Limited source:data from one people using right hand
- spread evenly in the keyboard area
- 5000 valid touch，data:coordinates(x,y),13 motion sensors

Machine learning@python3 Preprocessing

```
def getall(filedir,cutplace,crosslabel):
    feathers=[]
    labels=[]
    jsonFiles = list(filter(lambda name:name.endswith(".json"), os.listdir(filedir)))
    print(jsonFiles)
    for item in jsonFiles:
        temp1,temp2=myinitprocess(filedir+item)
        feathers.append(temp1)
        labels.append(temp2)
    temp3,temp4=np.concatenate(tuple(feathers)),np.concatenate(tuple(labels))
    train=np.zeros((np.shape(temp3)[0],100-start,13))
    trainlabel=np.zeros((np.shape(temp3)[0],2))
    for i in range((np.shape(temp3)[0])):
        train[i]=temp3[i].T
        trainlabel[i]=temp4[i]
    test=0;testlabel=0
    if(crosslabel==True):
        train,test,trainlabel,testlabel=cross_validation.train_test_split(train,trainlabel,test_size=cutplace, random_state=
    return train,test,trainlabel,testlabel
```

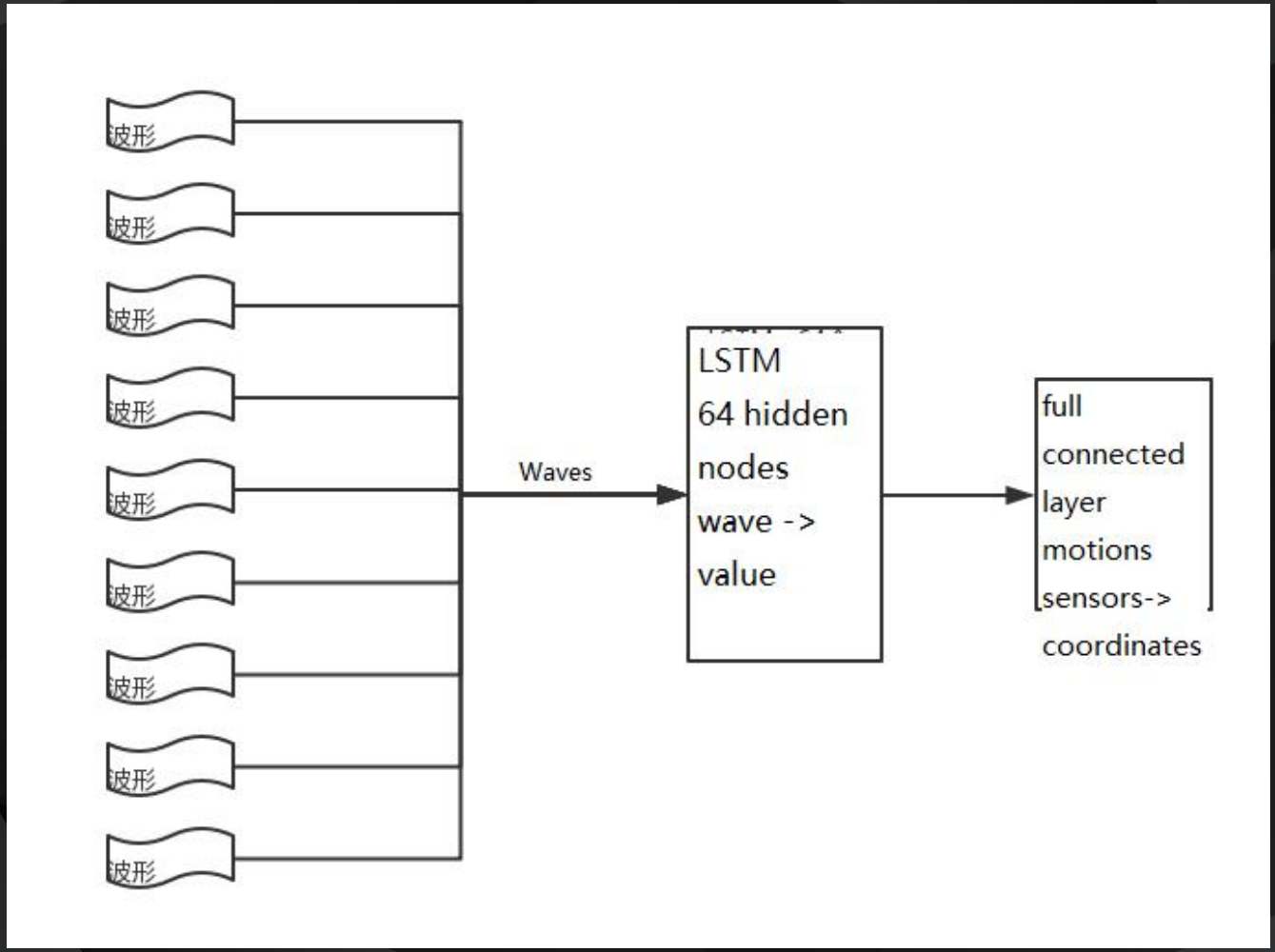
Machine learning:establish neural network

- tensorflow&tflearn&sklearn
- X,netlabel :input training data and placeholder of labels
- tflearn.lstm netwrok:default

```
tflearn.layers.recurrent.lstm (incoming, n_units, activation='tanh', inner_activation='sigmoid', dropout=None, bias=True, weights_init=None,
forget_bias=1.0, return_seq=False, return_state=False, initial_state=None, dynamic=False, trainable=True, restore=True, reuse=False,
scope=None, name='LSTM')
```

- lstm dropout ratio:0.7 ->avoid overfitting
- return_seq=False:return value for every wave（64nodes）
  tflearn.fully_connected:full connected layer
- least RMS for coordinates and training

Machine learning:establish neural network



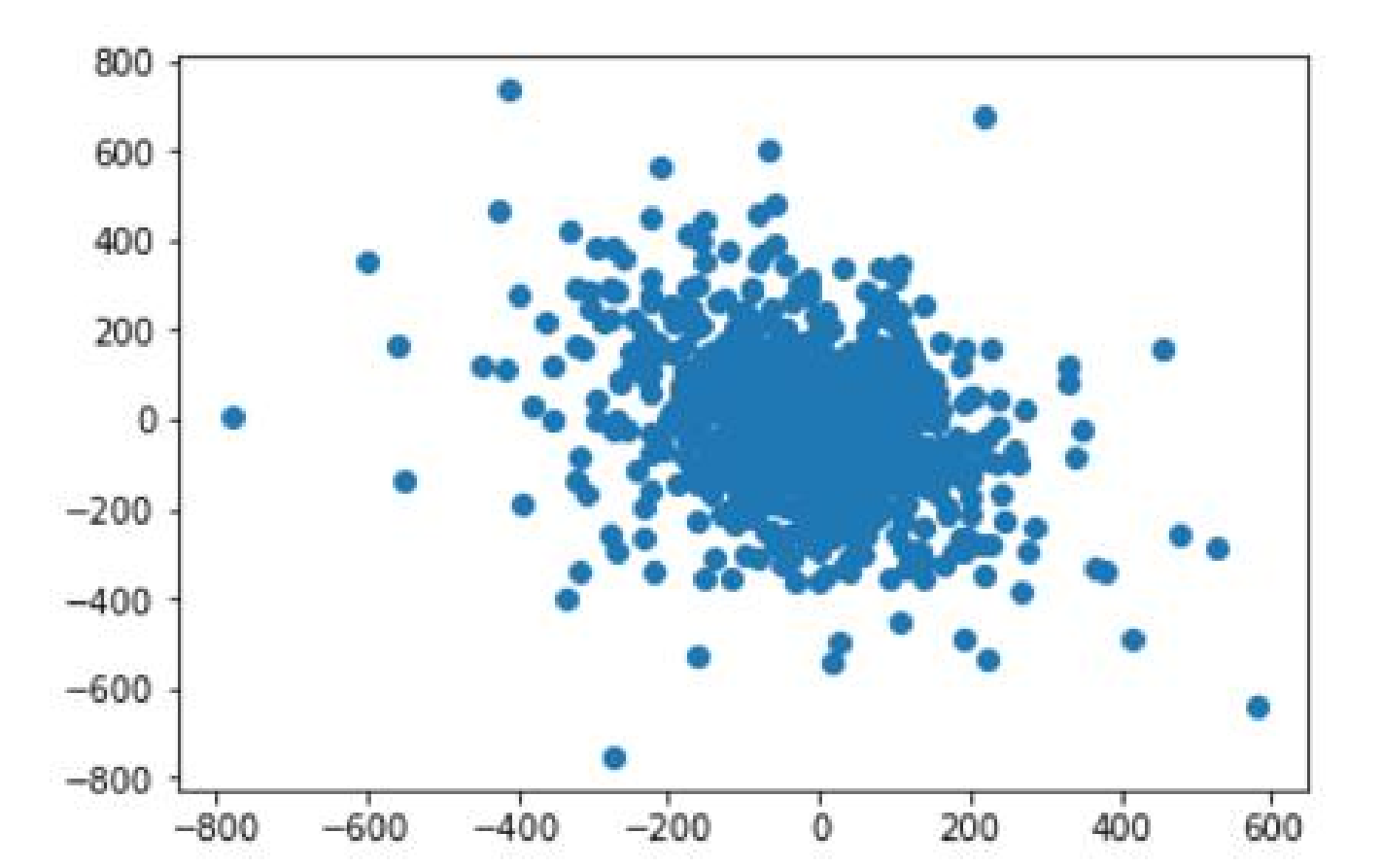
Machine learning:train neural network

- xaccuracy&yaccuracyphased accuracy on test set
- cost:loss

```
xaccuracx:0.171226 yaccuracy:0.002026 step: 0 cost 0.0652756
xaccuracx:0.678825 yaccuracy:0.530902 step: 40 cost 0.00792426
xaccuracx:0.722391 yaccuracy:0.547112 step: 80 cost 0.00719253
xaccuracx:0.736575 yaccuracy:0.552178 step: 120 cost 0.0070219
xaccuracx:0.730496 yaccuracy:0.551165 step: 160 cost 0.00688715
xaccuracx:0.739615 yaccuracy:0.551165 step: 200 cost 0.00671018
xaccuracx:0.738602 yaccuracy:0.642351 step: 240 cost 0.005837
xaccuracx:0.749747 yaccuracy:0.630193 step: 280 cost 0.00597559
xaccuracx:0.767984 yaccuracy:0.722391 step: 320 cost 0.00486835
xaccuracx:0.796353 yaccuracy:0.725431 step: 360 cost 0.00439627
xaccuracx:0.800405 yaccuracy:0.730496 step: 400 cost 0.00404178
xaccuracx:0.799392 yaccuracy:0.727457 step: 440 cost 0.00404576
xaccuracx:0.794326 yaccuracy:0.693009 step: 480 cost 0.00408823
xaccuracx:0.808511 yaccuracy:0.726444 step: 520 cost 0.00333054
xaccuracx:0.801418 yaccuracy:0.714286 step: 560 cost 0.00331399
xaccuracx:0.798379 yaccuracy:0.712259 step: 600 cost 0.00317743
xaccuracx:0.800405 yaccuracy:0.709220 step: 640 cost 0.00310957
xaccuracx:0.815603 yaccuracy:0.732523 step: 680 cost 0.00265303
xaccuracx:0.815603 yaccuracy:0.720365 step: 720 cost 0.00261896
xaccuracx:0.824721 yaccuracy:0.724417 step: 760 cost 0.0023168
xaccuracx:0.828774 yaccuracy:0.728470 step: 800 cost 0.00217134
xaccuracx:0.823708 yaccuracy:0.728470 step: 840 cost 0.00212671
xaccuracx:0.818642 yaccuracy:0.716312 step: 880 cost 0.00193261
xaccuracx:0.798379 yaccuracy:0.720365 step: 920 cost 0.00192573
xaccuracx:0.812563 yaccuracy:0.717325 step: 960 cost 0.00188861
```

Machine learning:evaluation

- Standard：half finger point，±150px
- Error distribution：centered around orgin point



Machine learning:evaluation

- Very small average(compared to 150):accurate enough

```
print(np.average(a[:,0]))
print(np.average(a[:,1]))
```

```
-17.2657368359
-8.93085081647
```

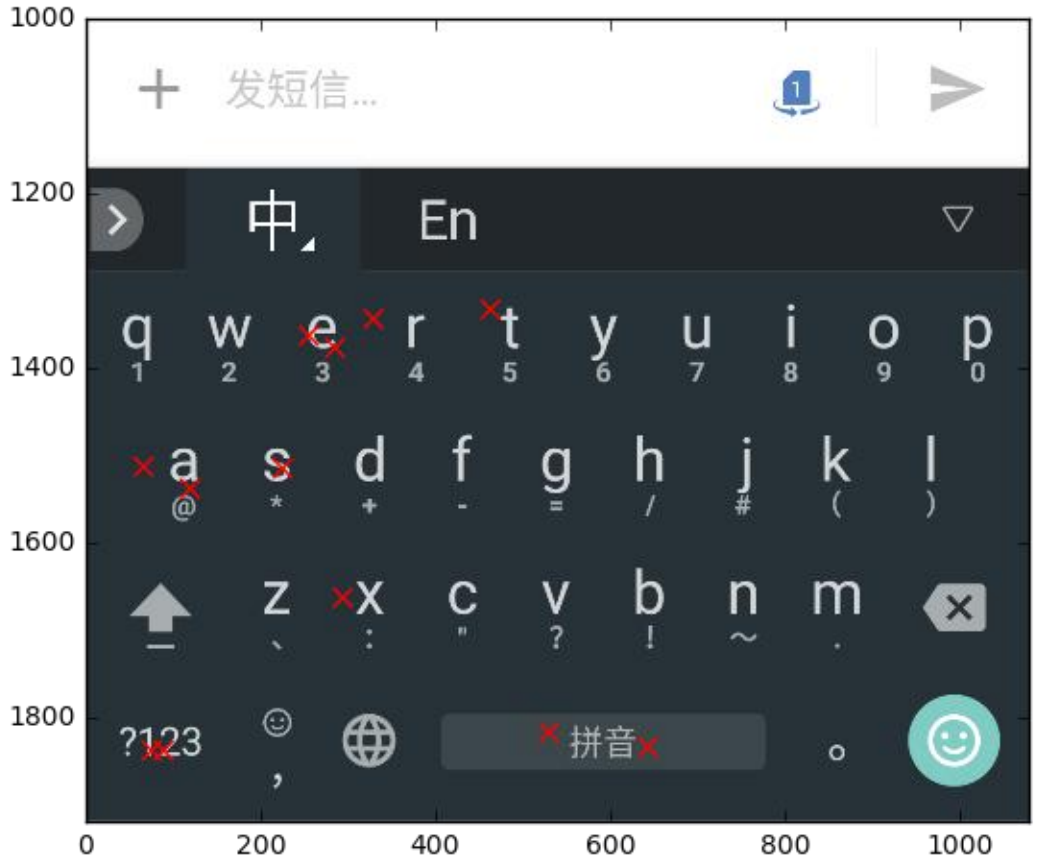
- ultimate accuracy

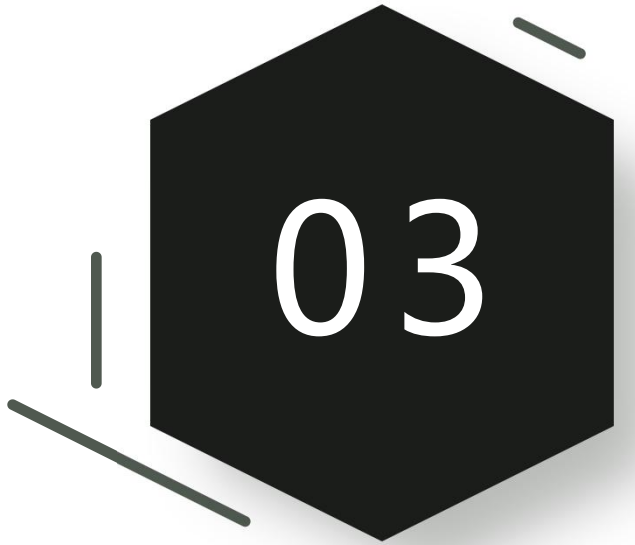
```
countx=0
county=0
for i in a:
    if(abs(i[0])<150):
        countx+=1
    if(abs(i[1])<150):
        county+=1
print('x:',countx/np.shape(test)[0])
print('y:',county/np.shape(test)[0])
```

```
x: 0.8014184397163121
y: 0.7223910840932117
```

Input test

- Input a sentence and recognize letters in sequence
- error :1/12，accuracy:91.7%





Future work

Shortcomings

- limited time and computaion abilities
- gpu:840m
- time costing collecting work
- not general enough(only 1 people, 1 gesture)

Improvement in the future

- Shrink collecting time, get more features , using less data
- better neural network
- better computer
- more people,more gesture
- from coordinates to contents
- study more than coordinates

From coordinates to contents

- PIN
- slide PIN
- full QWERTY,Pinyin 9 keys

More:

- APP history
- web history
- shopping history

Study more than coordinates

- pedometer,gait analysis
- optimal path in racing games
- using phone sensor to control aircrafts

More:

- dectect conditions of patients with epencephalondisease
- path analysis, guidance for the old and the kids

THANKS