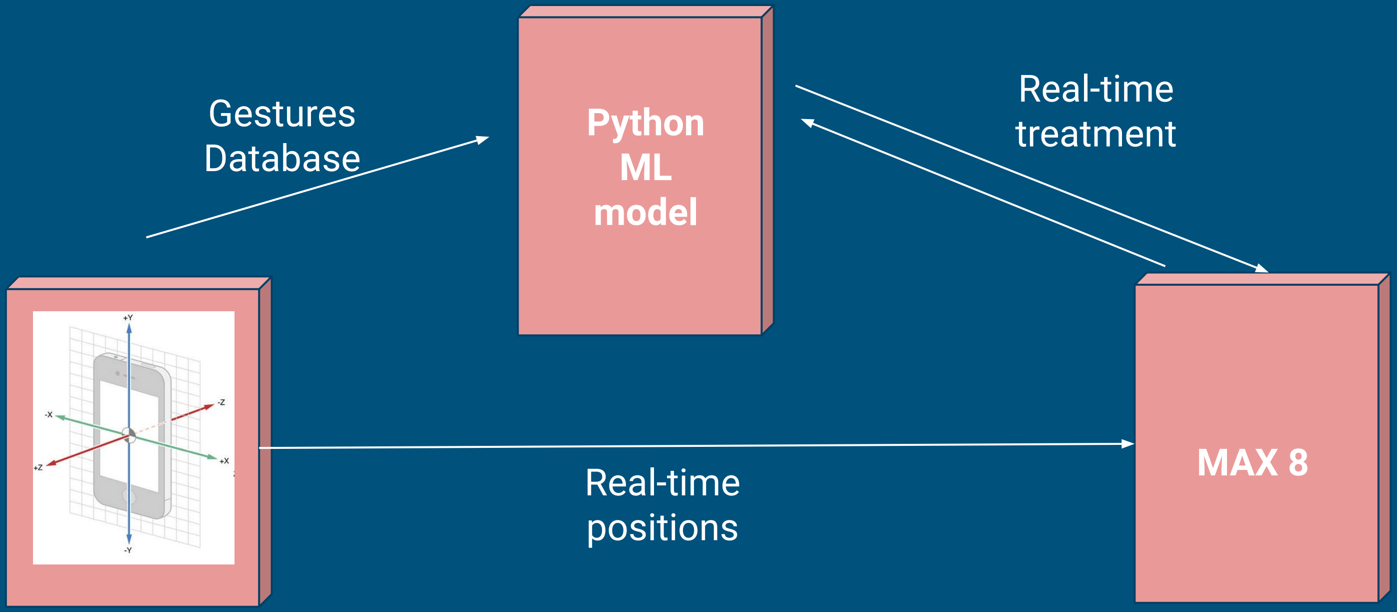
Real-time movement-based sound interaction using smartphones

Matthieu CERVERA
Sebastian DONZIS
Paul-Eloi MANGION
Candice VAN DEN BERGH

Map gestures to sounds

Create your own chord progression with your hand using our algorithm

Framework



Gestures Database

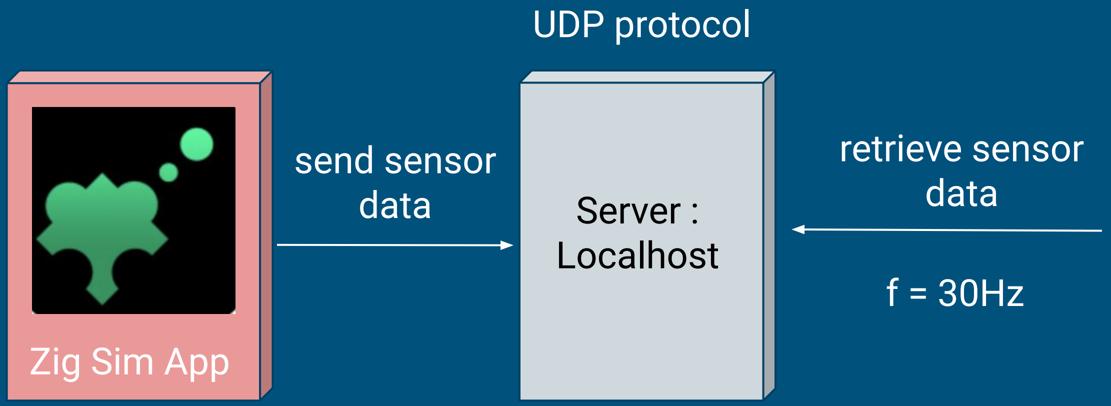
Python ML model

Real-time treatment

MAX 8

Real-time positions

Sending the Data



UDP protocol

Zig Sim App

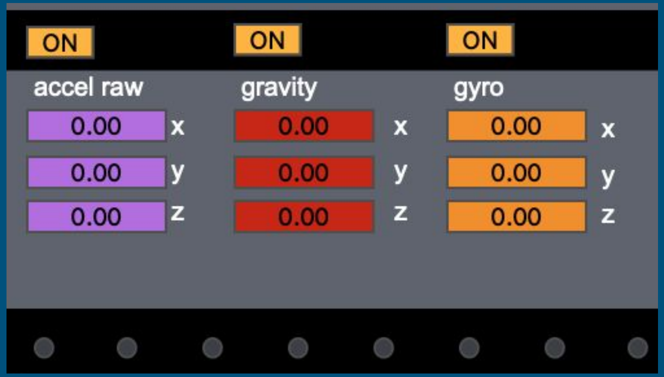
send sensor data

Server : Localhost

retrieve sensor data

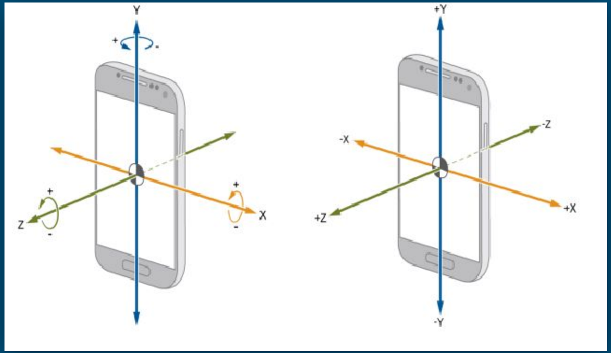
f = 30Hz

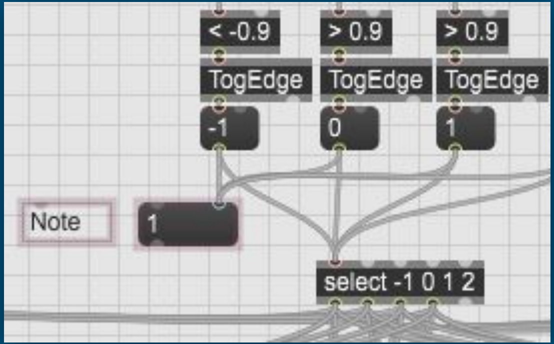
Max OSC Module



ON		ON		ON	
accel raw		gravity		gyro	
0.00	x	0.00	x	0.00	x
0.00	y	0.00	y	0.00	y
0.00	z	0.00	z	0.00	z

Map gestures to sounds





gravity Z < -0.9

Note C

A
B
C
D
E
F
G

gravity X > 0.9

Note E

A
B
C
D
E
F
G

gravity Z > 0.9

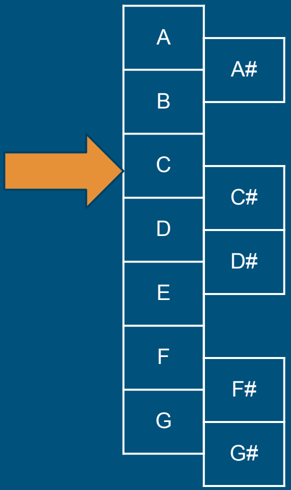
Note G

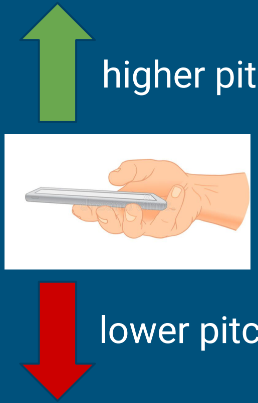
A
B
C
D
E
F
G

abs(gyro X) > 5

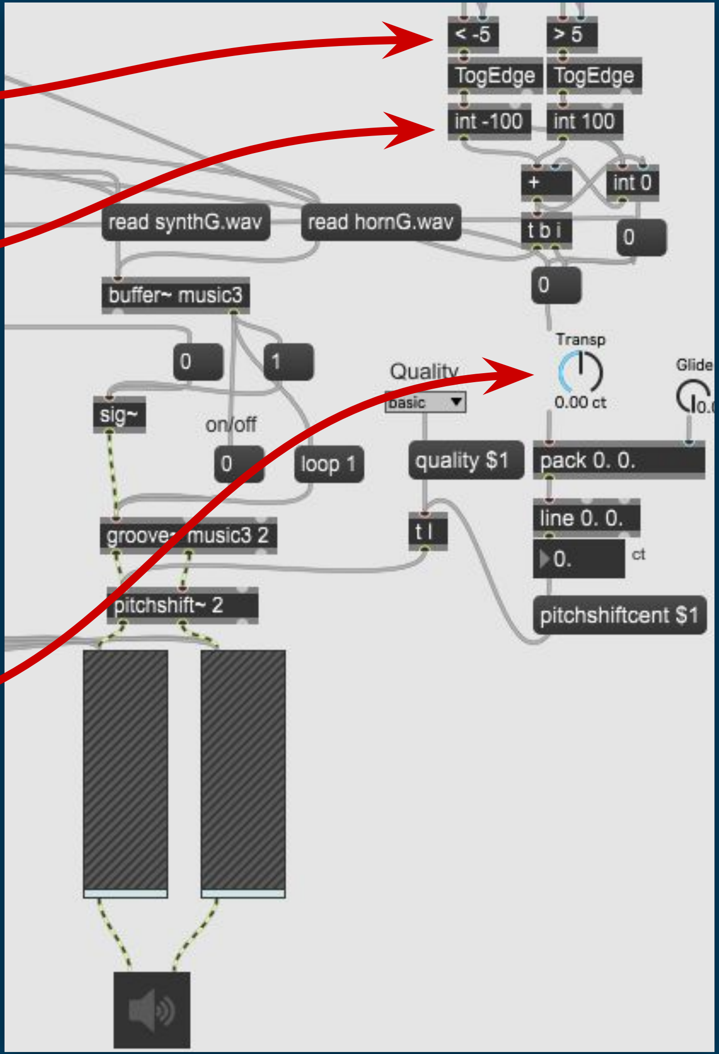
100 = one semitone

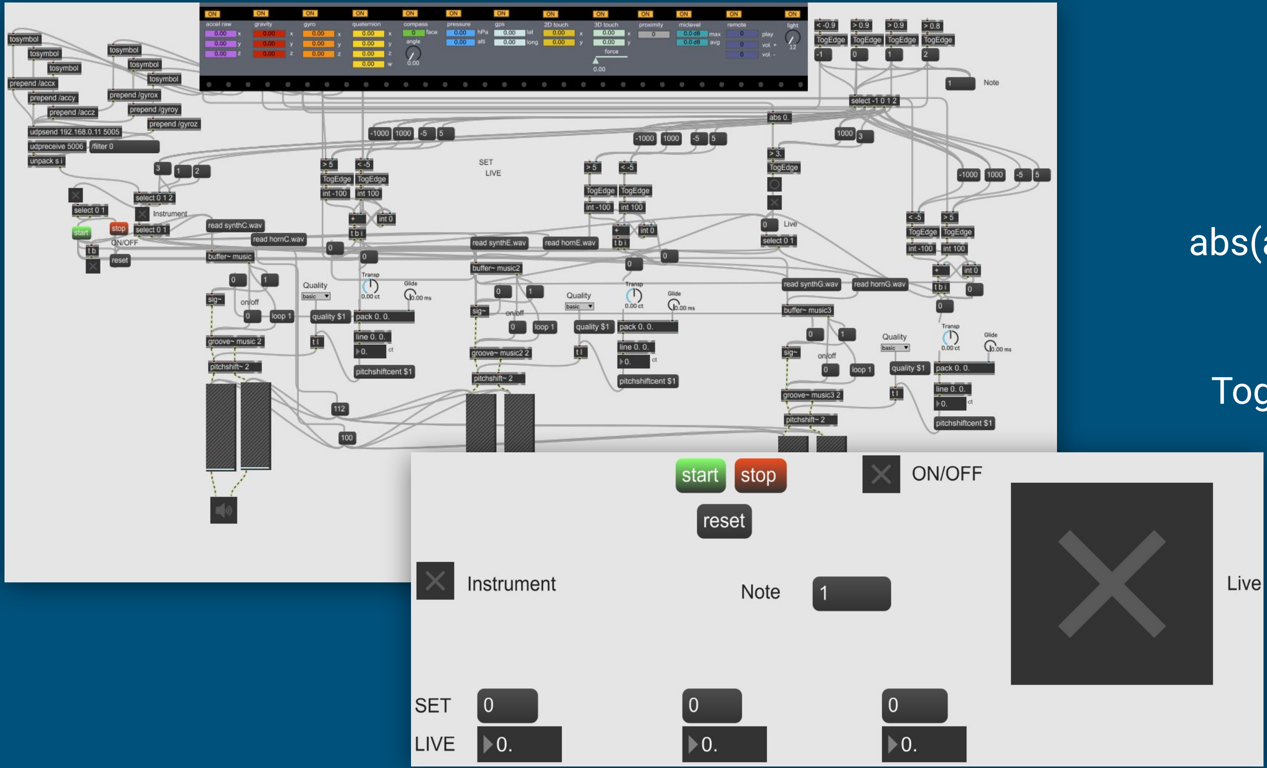
Note C



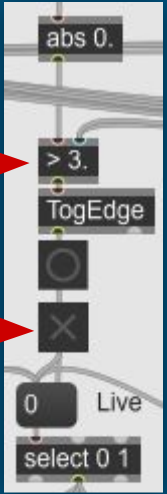

higher pitch

lower pitch


```
< -5          > 5
TogEdge       TogEdge
int -100      int 100
+             int 0
read synthG.wav    read hornG.wav    t b i    0
0
buffer~ music3
0    1
Transp
0.00 ct
Glide
Quality
basic
sig~
on/off
0    loop 1    quality $1    pack 0. 0.
line 0. 0.
groove~ music3 2    t l
0.   ct
pitchshift~ 2
pitchshiftcent $1
```



Overview + UI



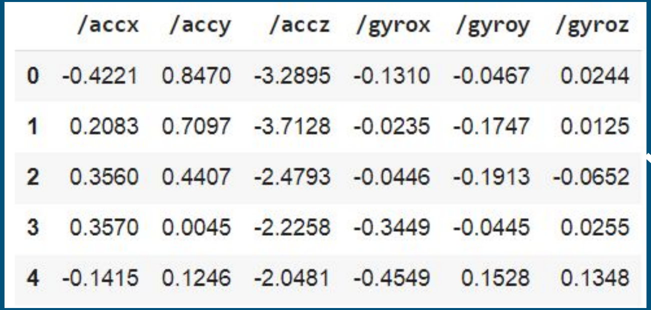
abs(acc X) > 3

Toggle live or off

Bonus Live Feature

Learn gestures : Classification problem

Input

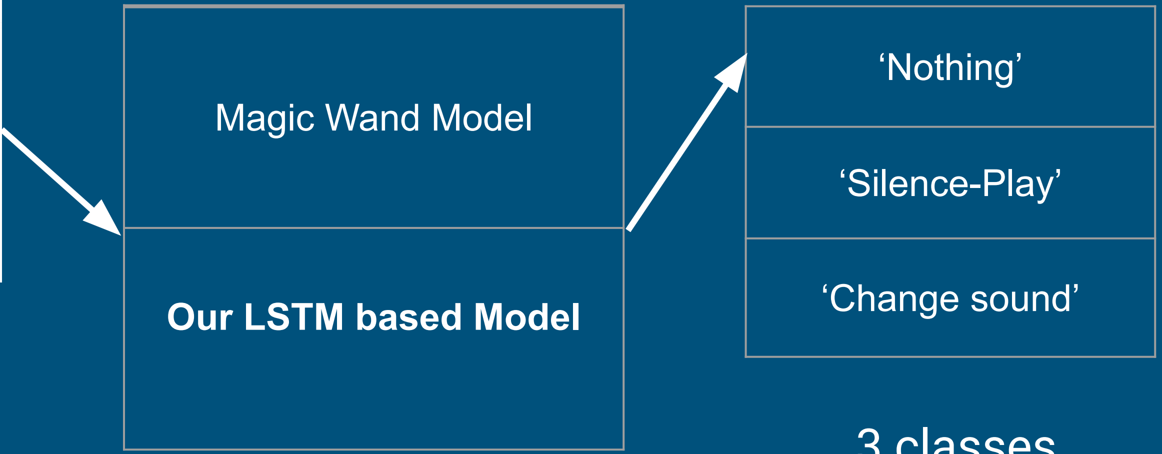


	/accx	/accy	/accz	/gyrox	/gyroy	/gyroz
0	-0.4221	0.8470	-3.2895	-0.1310	-0.0467	0.0244
1	0.2083	0.7097	-3.7128	-0.0235	-0.1747	0.0125
2	0.3560	0.4407	-2.4793	-0.0446	-0.1913	-0.0652
3	0.3570	0.0045	-2.2258	-0.3449	-0.0445	0.0255
4	-0.1415	0.1246	-2.0481	-0.4549	0.1528	0.1348

Time serie of shape (N,6)

N : hyperparameter of the problem

Model



Magic Wand Model

Our LSTM based Model

Output

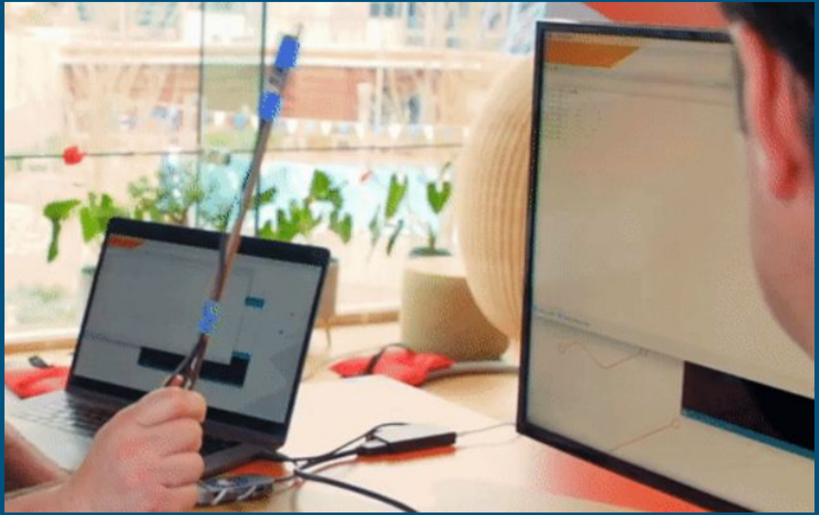
‘Nothing’

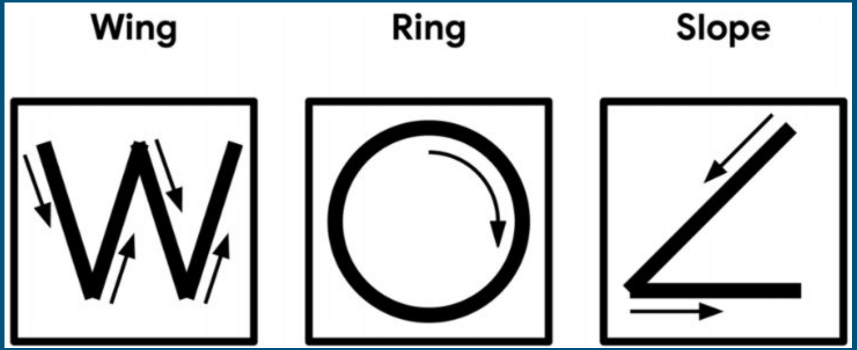
‘Silence-Play’

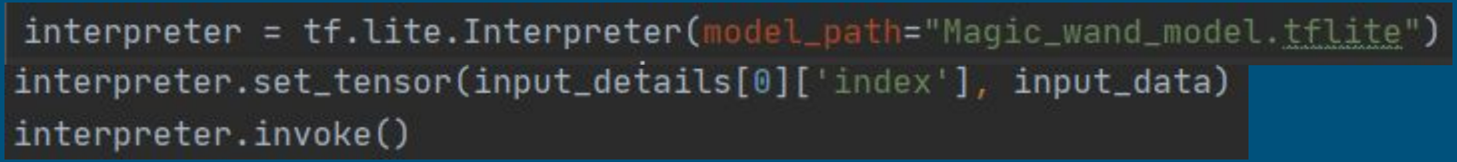
‘Change sound’

3 classes

First steps : Magic-Wand model

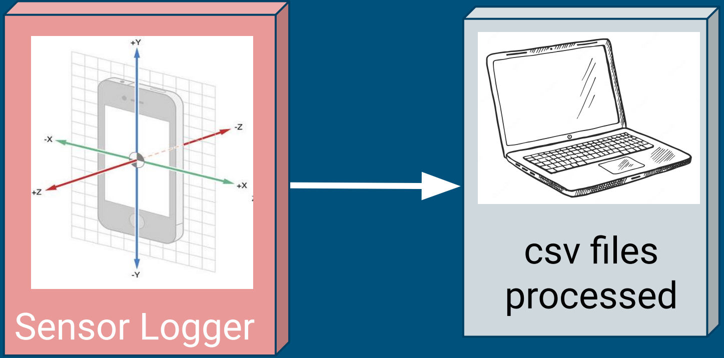






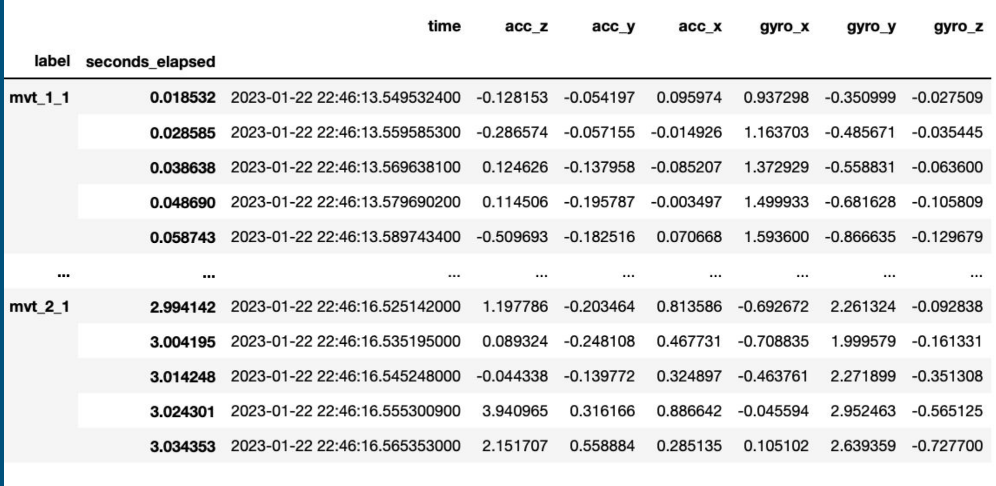
```
interpreter = tf.lite.Interpreter(model_path="Magic_wand_model.tflite")
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()
```

Database generation



Sensor Logger

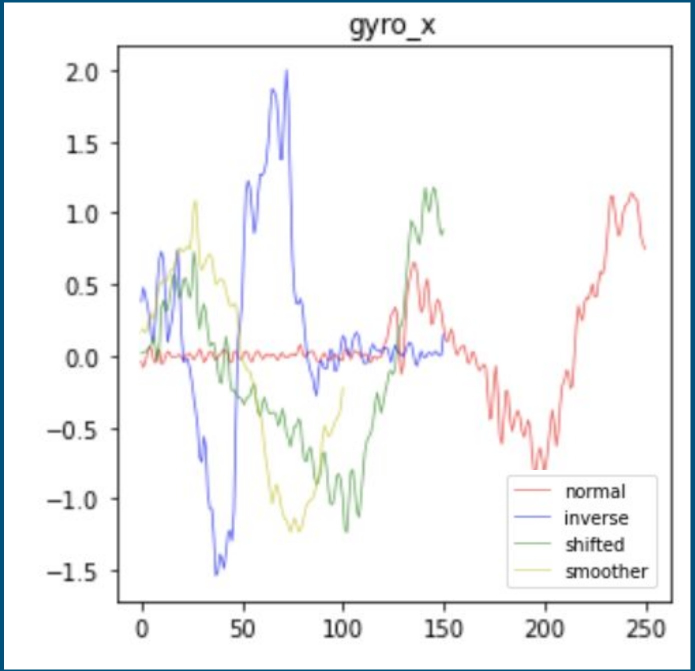
csv files processed



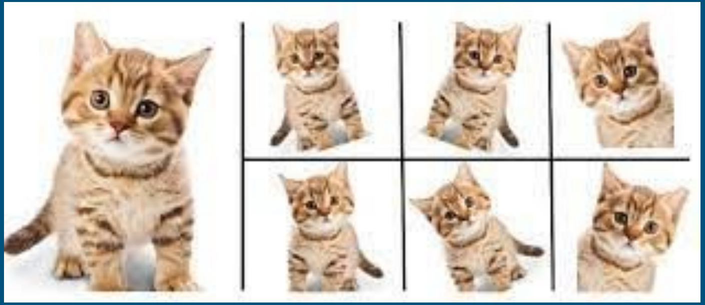
		time	acc_z	acc_y	acc_x	gyro_x	gyro_y	gyro_z
label	**seconds_elapsed**							
mvt_1_1	**0.018532**	2023-01-22 22:46:13.549532400	-0.128153	-0.054197	0.095974	0.937298	-0.350999	-0.027509
	0.028585	2023-01-22 22:46:13.559585300	-0.286574	-0.057155	-0.014926	1.163703	-0.485671	-0.035445
	0.038638	2023-01-22 22:46:13.569638100	0.124626	-0.137958	-0.085207	1.372929	-0.558831	-0.063600
	0.048690	2023-01-22 22:46:13.579690200	0.114506	-0.195787	-0.003497	1.499933	-0.681628	-0.105809
	0.058743	2023-01-22 22:46:13.589743400	-0.509693	-0.182516	0.070668	1.593600	-0.866635	-0.129679
...	**...**	...	...	...	...	...	...	...
mvt_2_1	**2.994142**	2023-01-22 22:46:16.525142000	1.197786	-0.203464	0.813586	-0.692672	2.261324	-0.092838
	3.004195	2023-01-22 22:46:16.535195000	0.089324	-0.248108	0.467731	-0.708835	1.999579	-0.161331
	3.014248	2023-01-22 22:46:16.545248000	-0.044338	-0.139772	0.324897	-0.463761	2.271899	-0.351308
	3.024301	2023-01-22 22:46:16.555300900	3.940965	0.316166	0.886642	-0.045594	2.952463	-0.565125
	3.034353	2023-01-22 22:46:16.565353000	2.151707	0.558884	0.285135	0.105102	2.639359	-0.727700

- 2 gestures + 1 “normal mode” gesture
- 10 people
- 10 takes, ~2-3 seconds, ~ 60-90 lignes per take, f = 30 Hz
- Different size, height, velocity, hand

Data Augmentation



Time Series vs. Image augmentation

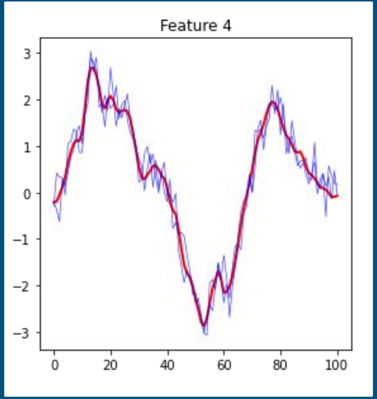


Data Augmentation Techniques

Feature 4: "Gyroscope X"

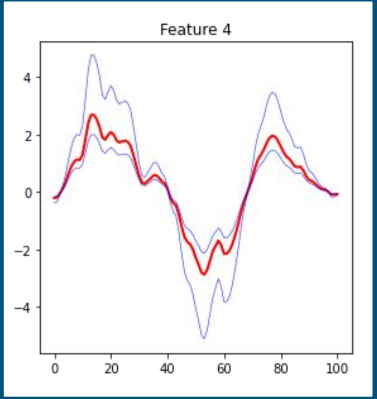
Jittering (adding noise)

mean: uniform: low=-0.1, high=0.1
*max_values
std: uniform: low=0.2, high=0.3

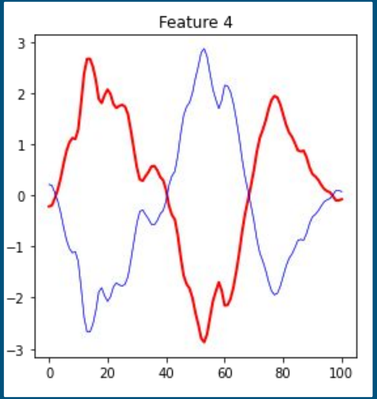


Scaling

uniform: low=0.5, high=2

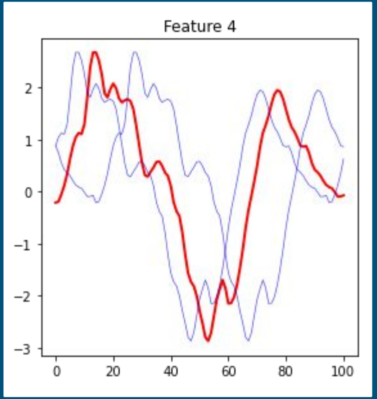


Rotating (flipping)



Permutation/ Shifting

random: low=-20, high=20



Data Augmentation

- Combination of different data augmentation techniques.
- Not always all
  - random

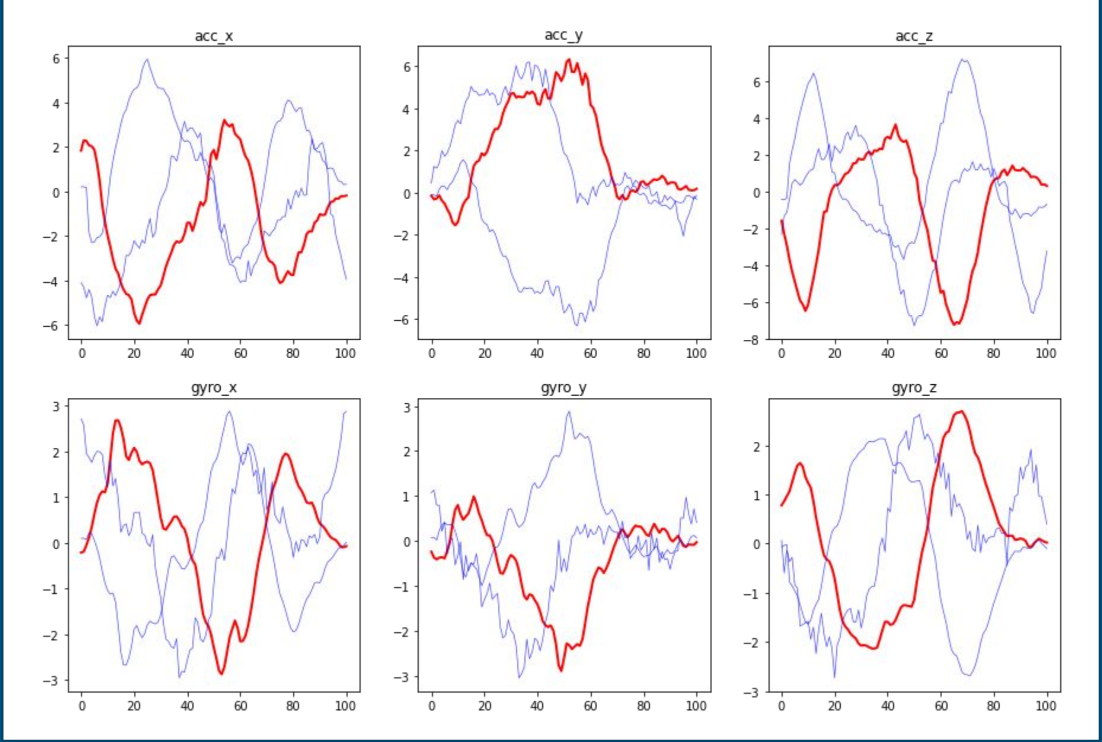
up to ~2000 samples

Shape: [2000, length(i), 6]

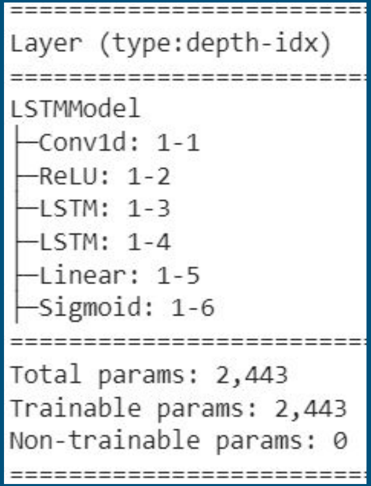
samples

length

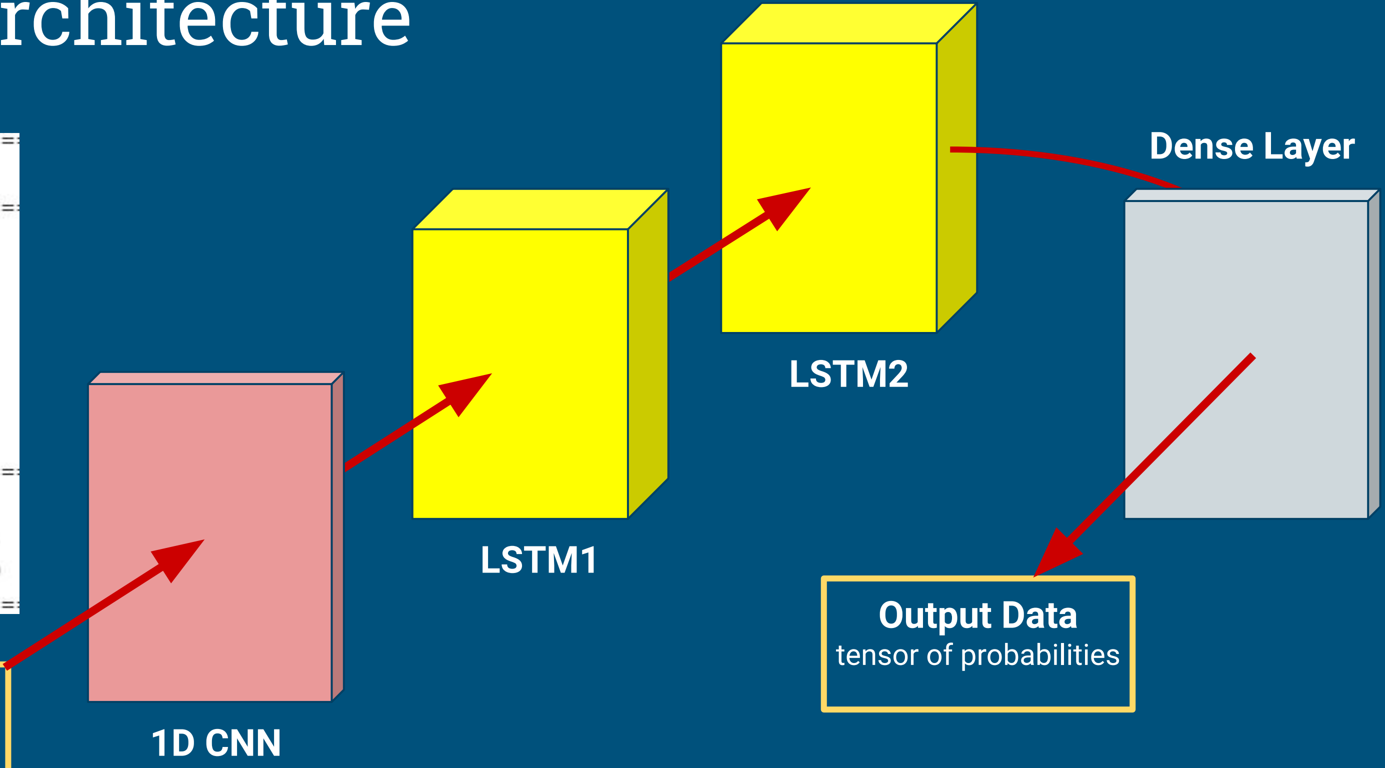
features



Model architecture



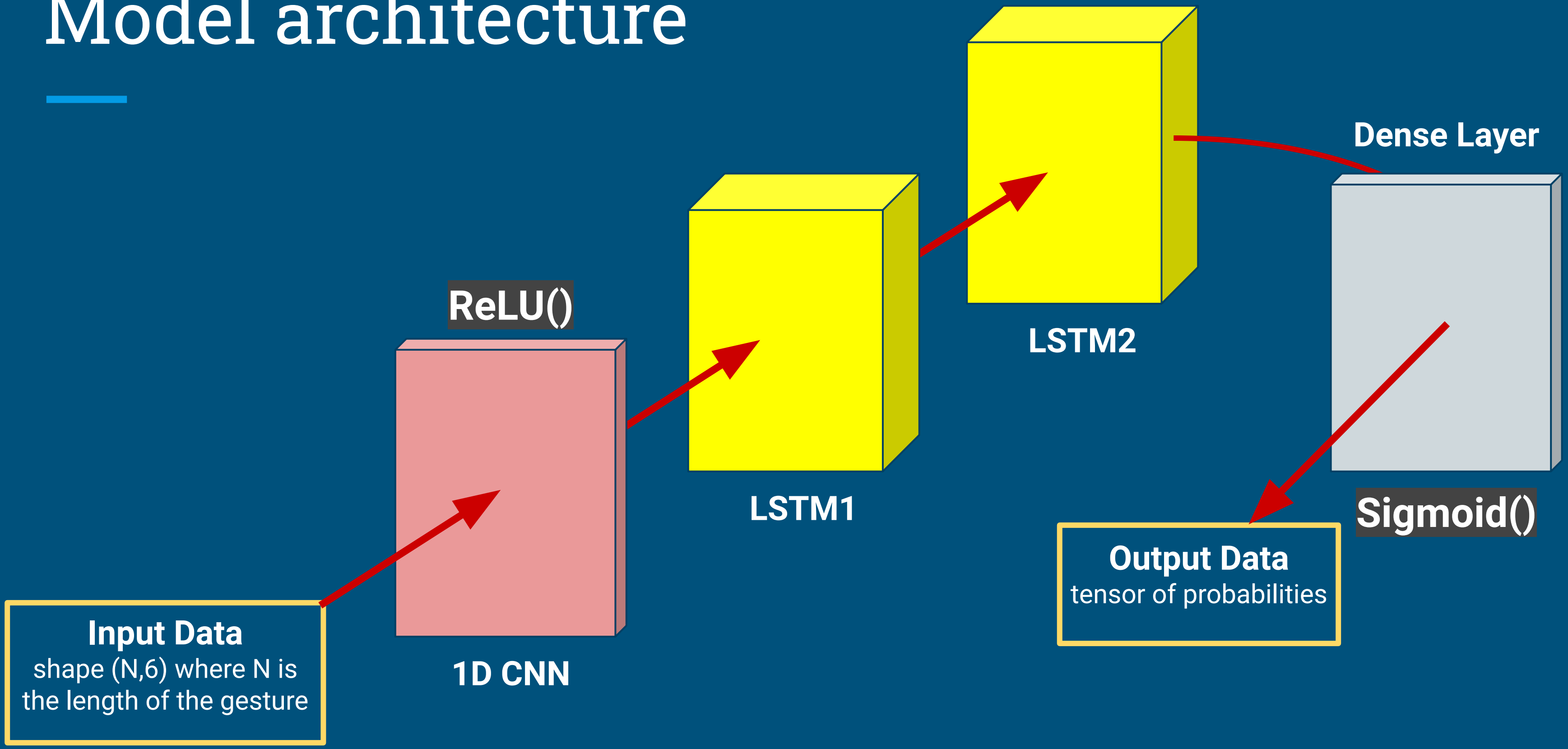
```
========================
Layer (type:depth-idx)
========================
LSTMModel
├─Conv1d: 1-1
├─ReLU: 1-2
├─LSTM: 1-3
├─LSTM: 1-4
├─Linear: 1-5
├─Sigmoid: 1-6
========================
Total params: 2,443
Trainable params: 2,443
Non-trainable params: 0
========================
```



Input Data
shape (N,6) where N is
the length of the gesture

PyTorch

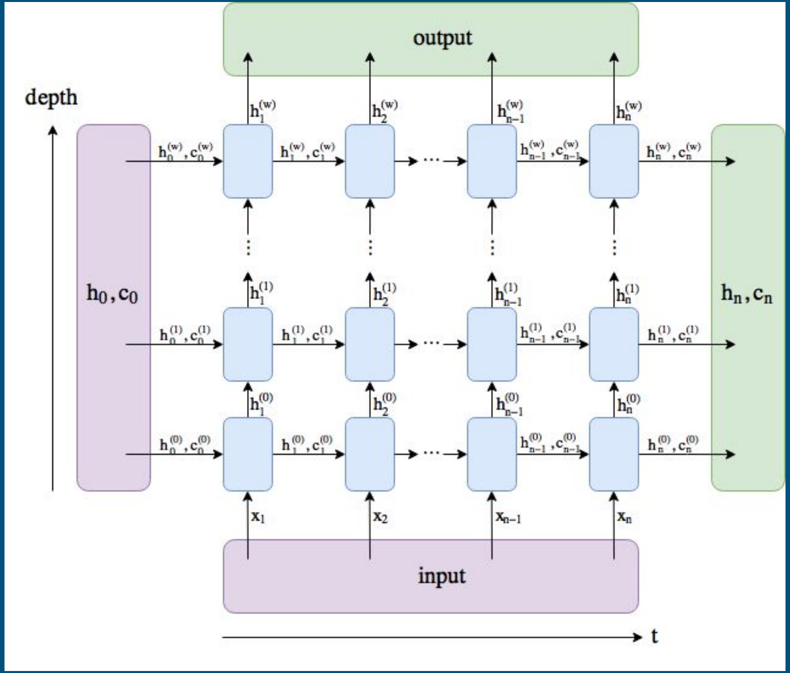
Model architecture

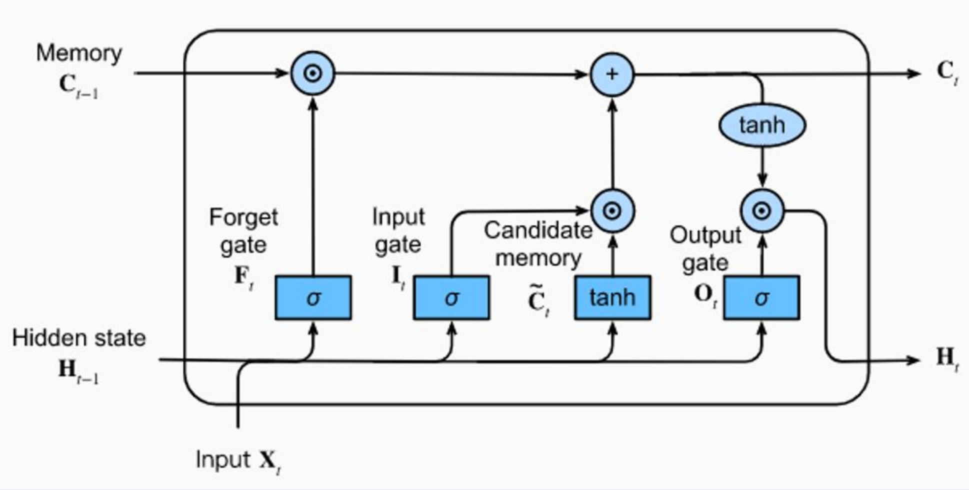


Input Data
shape (N,6) where N is
the length of the gesture

Output Data
tensor of probabilities

Model architecture - Why LSTM ?



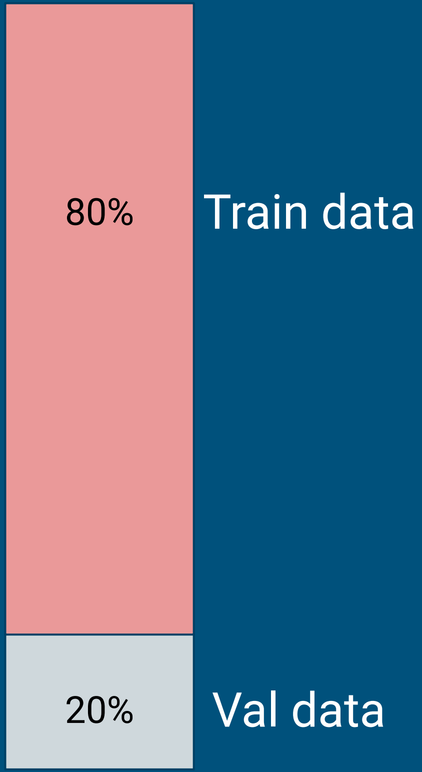


LSTM Architecture

Data preprocessing

1. All gestures need to have the **same length** :
   - choose **optimal length**
   - **pad** shorter gestures **with last value**
   - **truncate** longer gestures to chosen length

2. **Separate** the dataset into **training** and **validation** sets



Training and results

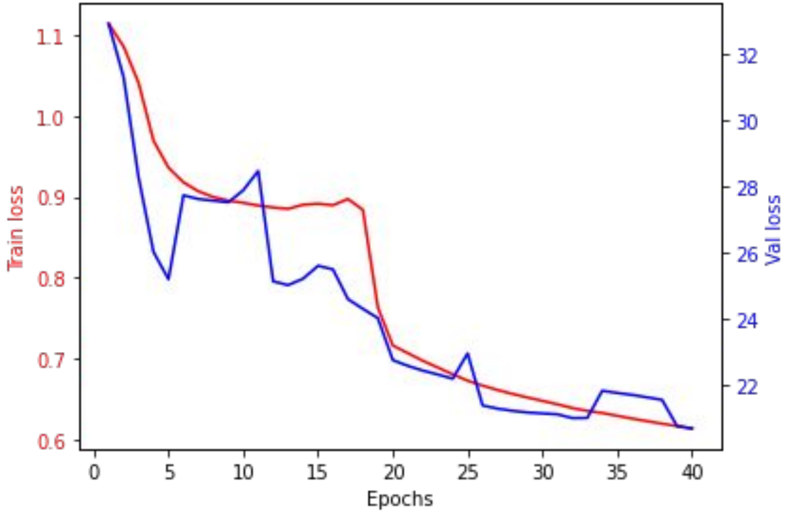
Useful tricks

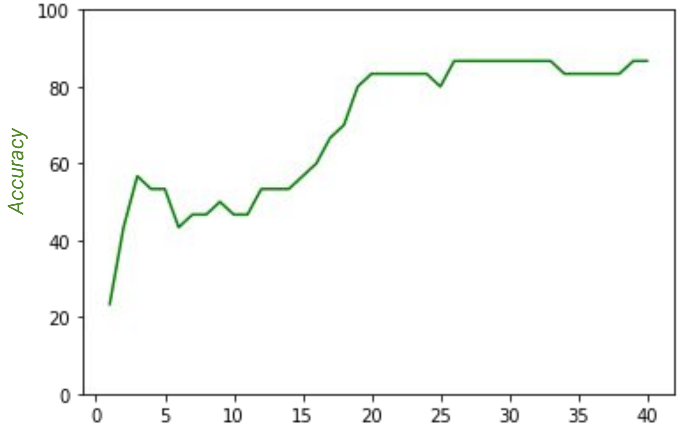
Add **Drop-out**

Add L2 Norm **Regularization**

Choose hidden layers, learning rate and other hyperparameters wisely

Change optimizer





Training and results

Limits

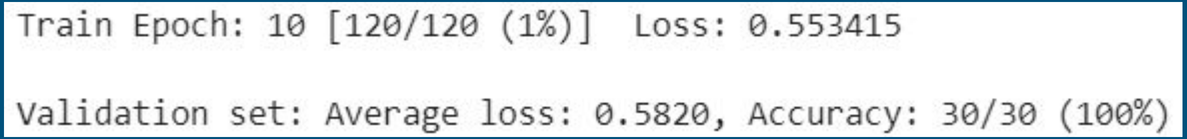
Low quantity of data

Complex networks are **overfitting**

Live results can be **wrong** because of overfitting

Black-Box effect

Random weights initializations affect LSTMs

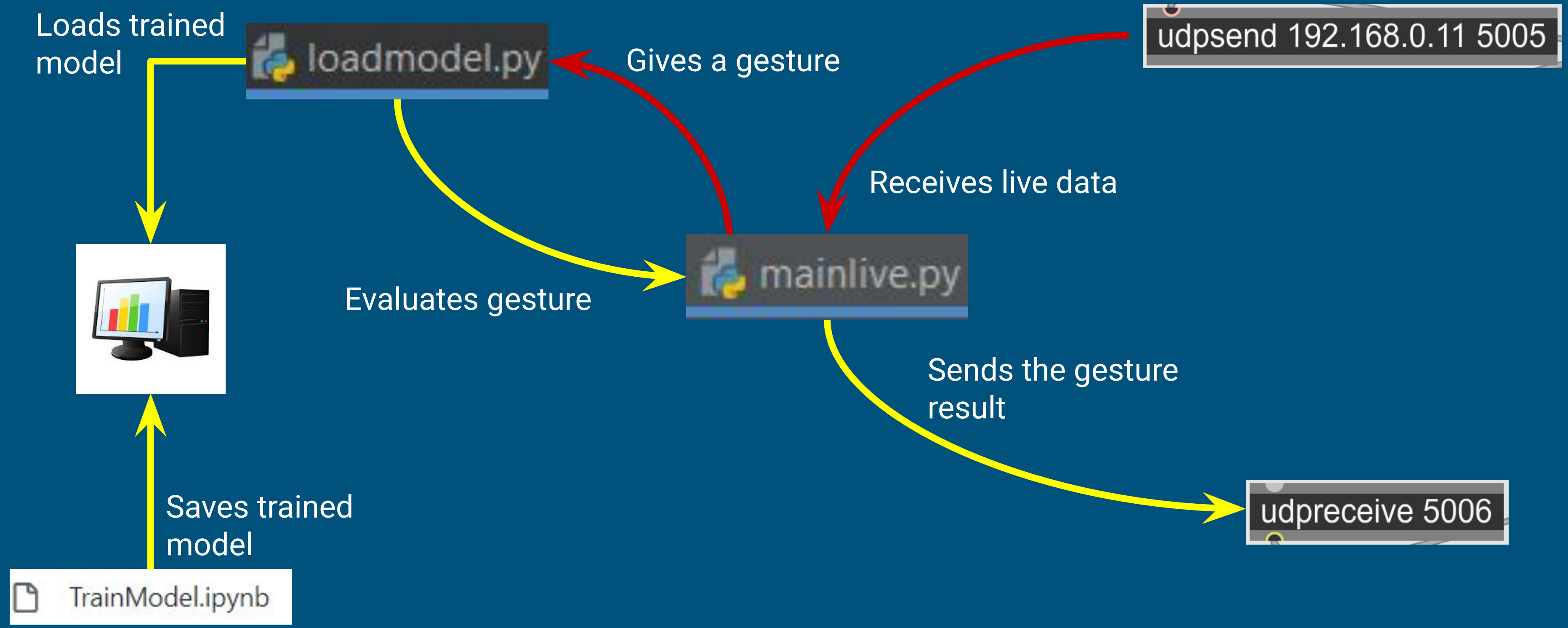


```
Train Epoch: 10 [120/120 (1%)]  Loss: 0.553415

Validation set: Average loss: 0.5820, Accuracy: 30/30 (100%)
```

Link to Max/MSP

Max/MSP



Loads trained model

loadmodel.py

Gives a gesture

udpsend 192.168.0.11 5005

Receives live data

mainlive.py

Evaluates gesture

Sends the gesture result

udpreceive 5006

Saves trained model

TrainModel.ipynb

Thank you

Any questions?