Prototype for a virtual keyboard based on IMUs and machine learning

Paul Bienkowski & Carolin Konietzny

TAMS, Fachbereich Informatik, Universität Hamburg

2017-05-16

Contents

Introduction

Motivation

Traditional Keyboards

Pros

- easy to learn
- precise
- universal
- cheap

Motivation

Traditional Keyboards

Pros

- easy to learn
- precise
- universal
- cheap

Cons

- poor ergonomics
- depends on motor abilities
- not adjustable to task → „shortcuts”

Motivation

Traditional Keyboards

Pros

- easy to learn
- precise
- universal
- cheap

Cons

- poor ergonomics
- depends on motor abilities
- not adjustable to task → „shortcuts”

https://en.wikipedia.org/wiki/File:Blackberry-Q10-transparent.png

Motivation

Traditional Keyboards

Pros

- easy to learn
- precise
- universal
- cheap

Cons

- poor ergonomics
- depends on motor abilities
- not adjustable to task → „shortcuts"

Alternatives

- voice recognition
- handwriting recognition
- visual methods (eye tracking)

https://en.wikipedia.org/wiki/File:Blackberry-Q10-transparent.png

Vision

1. record finger movements and input while typing

Vision

1. record finger movements and input while typing
2. use machine learning for input prediction

Vision

1. record finger movements and input while typing
2. use machine learning for input prediction
3. remove keyboard

Vision

1. record finger movements and input while typing
2. use machine learning for input prediction
3. remove keyboard
4. type everywhere

Related Work

Sensor Gloves

- gesture detection [5] [19] [8]
- sign language detection [8] [13]
- music generation [14]
- medical applications [3]

→ overview at [1]

Related Work

Alternative Keyboard Inputs

- buttons on glove [12]
- braille gloves [4]
- "The Learning Keyboard" [7] → Kinect

Figure 1: The Keyglove - a wearable, wireless, open-source input device

https://vimeo.com/23269969

Related Work

Commercial Products

- *Gest* [9] ($ 199, 988 on Kickstarter)
- *Project Virtual Keyboard* [16]
- *Hi5 VR Glove* [10] → VR Gaming

`https://gest.co/`

Figure 2: Gest general purpose interaction wearable

`http://hi5vrglove.com/`

Figure 3: Noitom Hi5 VR Glove

Related Work

In Research

no research project combines

sensor + keyboard data → machine learning

Project Goal

Of course we won’t build a fully working keyboard in 2 bachelor theses.

Project Goal

Of course we won't build a fully working keyboard in 2 bachelor theses.

Goal

Design a system for recording characteristic hand movements of typing and the corresponding input.

Define an approach for utilizing machine learning to map the recorded data back to the keyboard input.

Evaluate the quality of such mapping and discuss whether this principle could be turned into a working keyboard alternative.

System design

Desired System Properties

- utilizes machine learning
- detects characteristic values
  - fast (goal 100Hz)
  - accurate
  - independent of pose
- non-obstructive
  - flexible
  - wireless
  - light
- software suitable for fast prototyping
- cheap

Sensors

Possible Choices

Flex sensors

https://www.flickr.com/photos/indiamos/3060497602

Figure 4: Possible types of sensors; *left* resistive flex sensors

Sensors

Possible Choices

Flex sensors

https://www.flickr.com/photos/indiamos/3060497602

Visual system

https://de.wikipedia.org/wiki/Kinect#/media/File:Xbox-360-Kinect-Standalone.png

Figure 4: Possible types of sensors; *left* resistive flex sensors, *center* Kinect for Xbox 360

Sensors

Possible Choices

Flex sensors

https://www.flickr.com/photos/indiamos/3060497602

Visual system

https://de.wikipedia.org/wiki/Kinect#/media/File:Xbox-360-Kinect-Standalone.png

IMUs

https://://organicmonkeymotion.wordpress.com/category/propeller/

Figure 4: Possible types of sensors; *left* resistive flex sensors, *center* Kinect for Xbox 360, *right* InvenSense MPU-9150 IMU

Sensors

Inertial Measurement Units (IMUs)

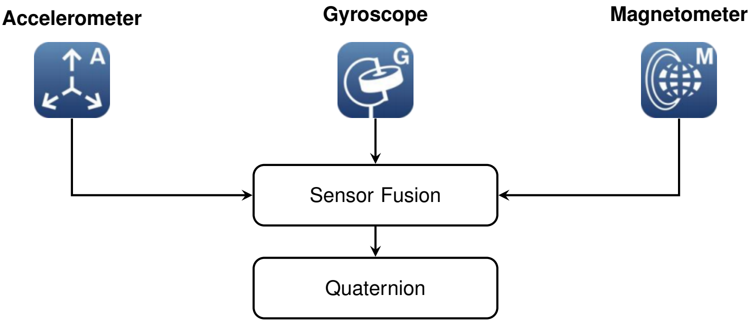
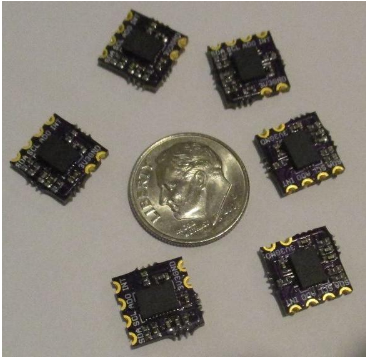


Figure 5: Sensor Fusion Overview

Sensors

Wearable BNO055 Nano Board

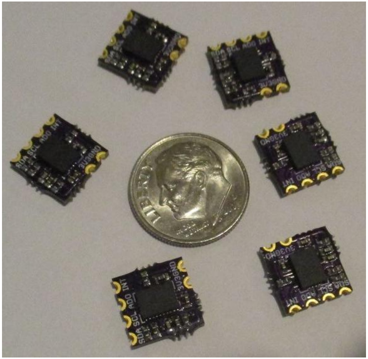


https://www.tindie.com/products/onehorse/wearable-bno055-nano-board/

Figure 6: Wearable BNO055 Nano Board

Sensors

Wearable BNO055 Nano Board



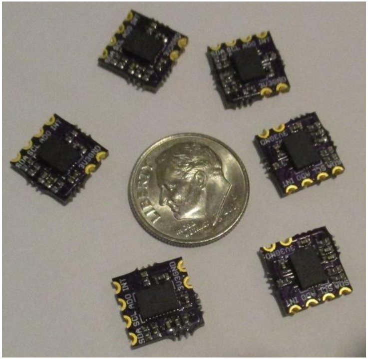
https://www.tindie.com/products/onehorse/wearable-bno055-nano-board/

Figure 6: Wearable BNO055 Nano Board

- 32 bit System-in-Package
- tiny (10 mm × 10 mm)
- easy to use
- good performance (~100 Hz)

Sensors

Wearable BNO055 Nano Board



https://www.tindie.com/products/onehorse/wearable-bno055-nano-board/

Figure 6: Wearable BNO055 Nano Board

- 32 bit System-in-Package
- tiny (10 mm × 10 mm)
- easy to use
- good performance (~100 Hz)

however...

- ca. 24 € each
- ships from USA
- gyro clipping problems

Sensors

Inertial Measurement Units (IMUs)

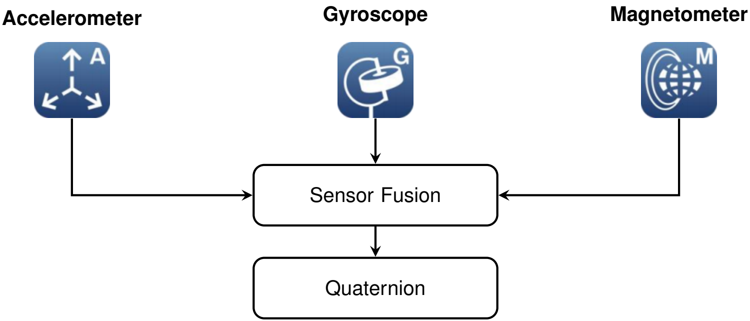


Figure 5: Sensor Fusion Overview

Sensors

Inertial Measurement Units (IMUs)

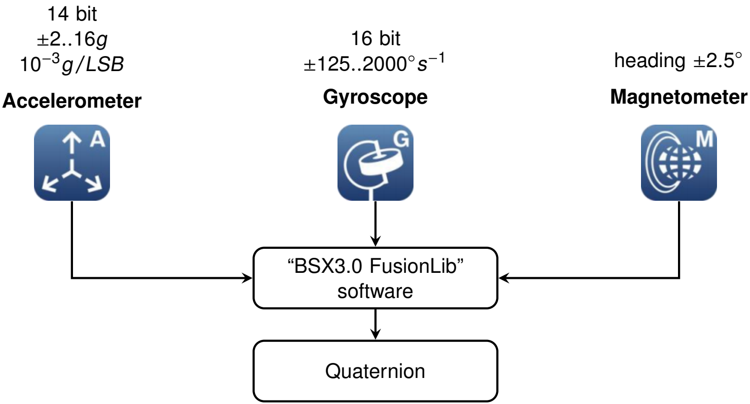
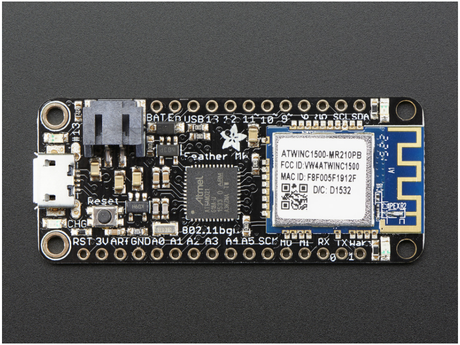


Figure 5: Sensor Fusion Overview

Microprocessor

Adafruit Feather M0 WiFi



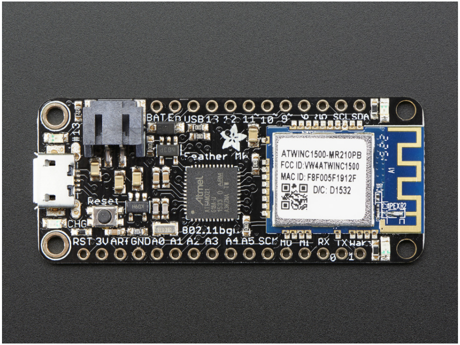
https://www.adafruit.com/product/3010

Figure 7: Adafruit Feather M0 WiFi - ATSAMD21 + ATWINC1500 product image

Microprocessor

Adafruit Feather M0 WiFi

- very small and lightweight (6.1g)
- on-board WiFi
- 6 SERCOMs (SPI/I2C/UART)



https://www.adafruit.com/product/3010

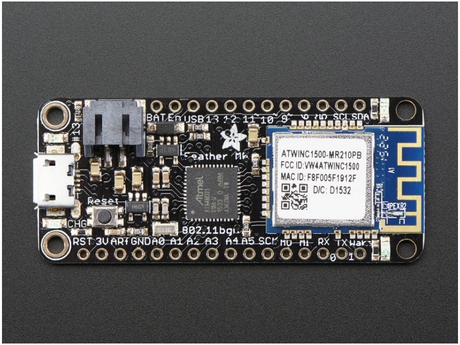
Figure 7: Adafruit Feather M0 WiFi - ATSAMD21 + ATWINC1500 product image

Microprocessor

Adafruit Feather M0 WiFi

- very small and lightweight (6.1g)
- on-board WiFi
- 6 SERCOMs (SPI/I2C/UART)

- Arduino® compatible
- 256KB FLASH, 32KB SRAM
- LiPo charger



`https://www.adafruit.com/product/3010`

Figure 7: Adafruit Feather M0 WiFi - ATSAMD21 + ATWINC1500 product image

Microprocessor

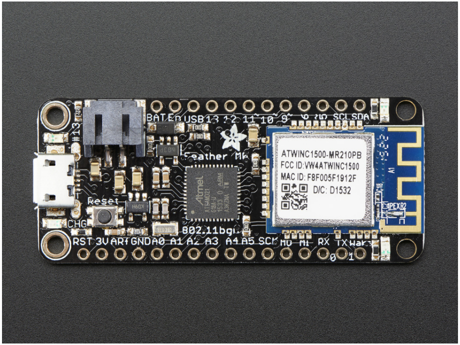
Adafruit Feather M0 WiFi

- very small and lightweight (6.1g)
- on-board WiFi
- 6 SERCOMs (SPI/I2C/UART)

- Arduino® compatible
- 256KB FLASH, 32KB SRAM
- LiPo charger

however...

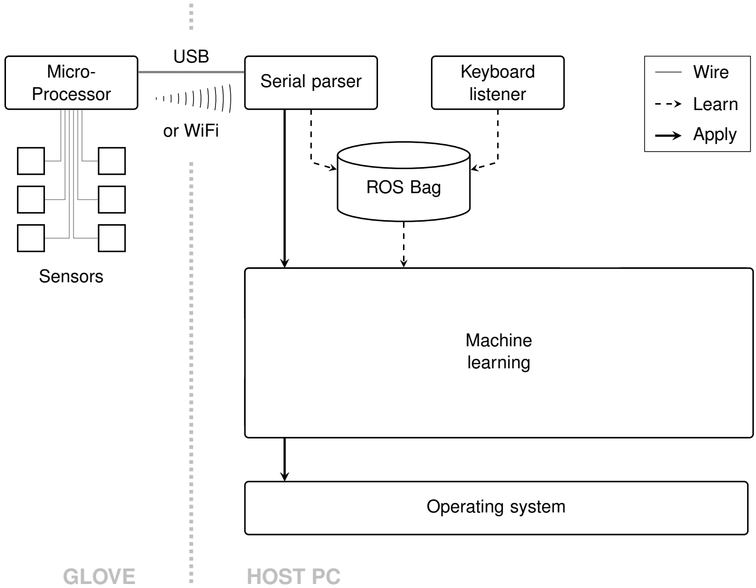
- no EEPROM
- ca. 40 € each



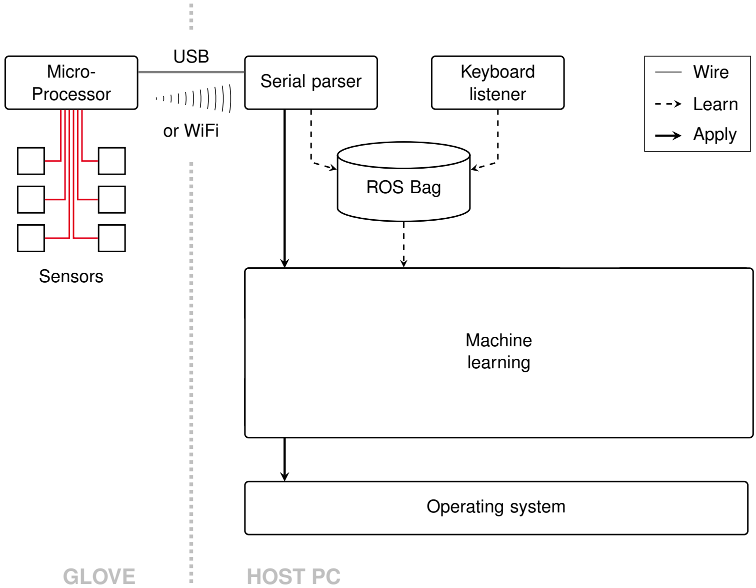
https://www.adafruit.com/product/3010

Figure 7: Adafruit Feather M0 WiFi - ATSAMD21 + ATWINC1500 product image

Architecture



Architecture



I2C Bus

Requirements

$$\frac{6 \text{ IMUs}}{2 \ \frac{\text{addresses}}{\text{IMU}}} = 3 \text{ buses}$$

I2C Bus

Requirements

$$\frac{6 \text{ IMUs}}{2 \ \frac{\text{addresses}}{\text{IMU}}} = 3 \text{ buses}$$

	Primary pads				Alternative pads				
SERCOM	0	1	2	3	0	1	2	3	Used by
0	4	3	1	0	A3	A4	8	9	Serial1
1	11	13	10	12					
2	22		2	5	4	3	1	0	
3	20	21	6*	7*	11	13	10	12	Default I2C
4	22		23*	24*	A1	A2	2	5	SPI
5	A5*		6	7	20	21			Debug Port

* need to be configured as *SERCOM alt*

Table 1: Available SERCOM pin pads on Adafruit Feather M0 WiFi

I2C Bus

Requirements

$$\frac{6\ \text{IMUs}}{2\ \frac{\text{addresses}}{\text{IMU}}} = 3\ \text{buses}$$

	Primary pads				Alternative pads				
SERCOM	0	1	2	3	0	1	2	3	Used by
0		3	1	0	A3	A4		9	Serial1
1	11	13	10	12					
2	22			5		3	1	0	
3	20	21	6*		11	13	10	12	Default I2C
4	22		23*	24*	A1	A2		5	SPI
5	A5*		6		20	21			Debug Port

* need to be configured as *SERCOM alt*

Table 1: Available SERCOM pin pads on Adafruit Feather M0 WiFi

I2C Bus

Requirements

$$\frac{6\ \text{IMUs}}{2\ \frac{\text{addresses}}{\text{IMU}}} = 3\ \text{buses}$$

	Primary pads				Alternative pads				
SERCOM	0	1	2	3	0	1	2	3	Used by
0		3	1	0	A3	A4		9	Serial1
1	11	13	10	12					
2	22			5		3	1	0	
3	20	21	6*		11	13	10	12	Default I2C
4	22		23*	24*	A1	A2		5	SPI
5	A5*		6		20	21			Debug Port

* need to be configured as *SERCOM alt*

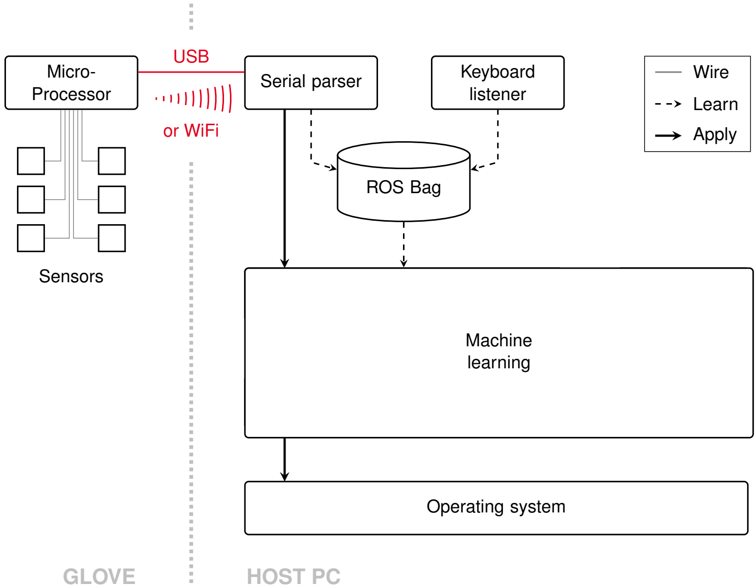
Table 1: Available SERCOM pin pads on Adafruit Feather M0 WiFi

I2C Bus

Arduino Setup

```
#include <Wire.h>
#include <wiring_private.h>

TwoWire wire0(&sercom0, A3, A4);
TwoWire wire1(&sercom3, 11, 13);
TwoWire wire2(&sercom5, 20, 21);

void setup() {
    wire0.begin(); wire0.setClock(400000L);
    wire1.begin(); wire1.setClock(400000L);
    wire2.begin(); wire2.setClock(400000L);
    delay(100);

    pinPeripheral(A3, PIO_SERCOM_ALT); // SERCOM0.0 (alt)
    pinPeripheral(A4, PIO_SERCOM_ALT); // SERCOM0.1 (alt)
    pinPeripheral(11, PIO_SERCOM_ALT); // SERCOM3.0 (alt)
    pinPeripheral(13, PIO_SERCOM_ALT); // SERCOM3.1 (alt)
    pinPeripheral(20, PIO_SERCOM_ALT); // SERCOM5.0 (alt)
    pinPeripheral(21, PIO_SERCOM_ALT); // SERCOM5.1 (alt)
}
```

Glove ↔ PC connection



Glove ↔ PC Connection

WiFi

```
// Setup
WiFi.setPins(8, 7, 4, 2);
WiFi.begin();

// Scan for networks (optional)
uint8_t ssidCount = WiFi.scanNetworks();
for (uint8_t i = 0; i < ssidCount; i++) {
    printf("- %s\n", WiFi.SSID(i));
}

// Connect to WPA2 network
uint8_t status = WiFi.begin(MY_SSID, MY_PASSPHRASE);
while (status != WL_CONNECTED) {
    delay(500);
    status = WiFi.status();
}

// Send data via UDP
WiFiUDP wifiUdp;
wifiUdp.begin(8080);
wifiUdp.beginPacket(TARGET_IP, TARGET_PORT);
wifiUdp.write(buffer, length);
wifiUdp.endPacket();
```

Capabilities

- WEP & WPA2
- Scan networks
- UDP, TCP, SSL
- HTTP Client
- HTTP Server

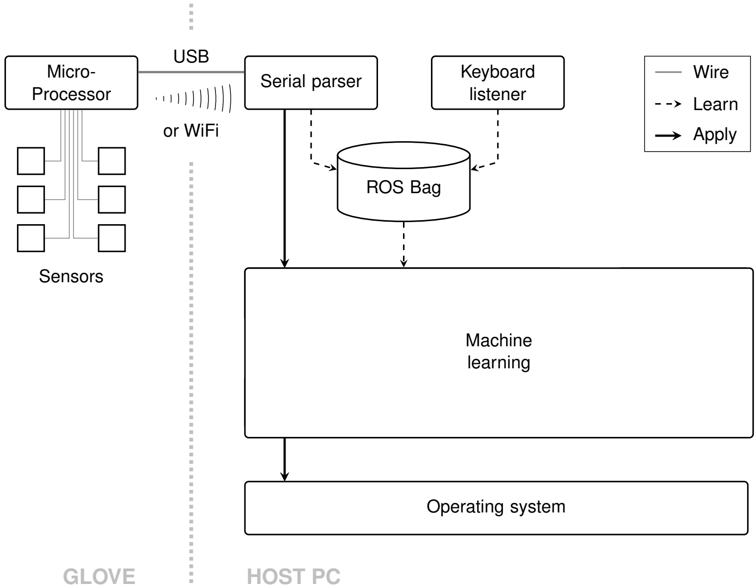
System design

Other Considerations

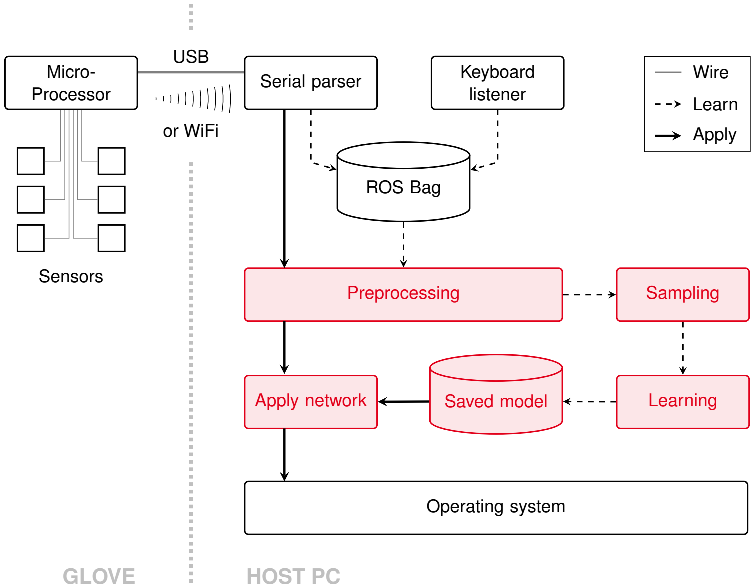
- serial protocol for data transmission
- attachment to the hand
- use ROS for recording & data analysis

Machine Learning

Machine Learning



Machine Learning



Neural Networks

Strengths and Weaknesses [18]

Pros

- general-purpose
- many variations
- fast to apply once learned
- able to detect complex relationships

Cons

- requires large dataset
- blackbox[1], difficult to "understand"
- slow to learn
- can overfit

[1] there are some rule-extraction algorithms [17]

Recurrent Neural Networks

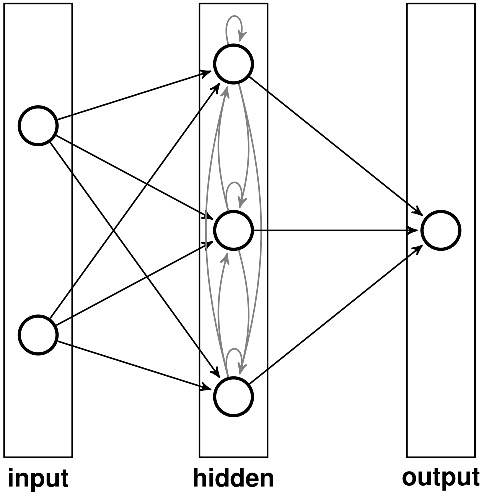


Figure 8: Simplified Recurrent Network

Problems

Vanishing Gradient Problem (Hochreiter [11])

Problem

Deep networks require a lot of training

Problems

Vanishing Gradient Problem (Hochreiter [11])

Problem

Deep networks require a lot of training

- during backpropagation, error is lost with each layer
- first layers receive slowest updates
- unrolled RNNs are very deep

Problems

Imbalanced Data

Problem

Only 2% of our samples are keystrokes (positive class)

Problems

Imbalanced Data

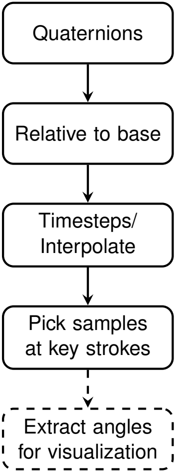
Problem

Only 2% of our samples are keystrokes (positive class)

Possible solutions [2]

- gather lots of data and train a lot
- resampling
- penalize
- generate synthetic data

Preprocessing and Sampling



Preprocessing and Sampling

Quaternions

↓

Relative to base

↓

Timesteps/ Interpolate

↓

Pick samples at key strokes

⇣

Extract angles for visualization

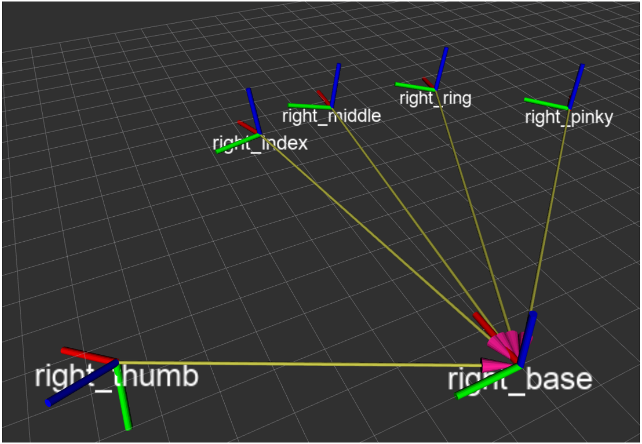
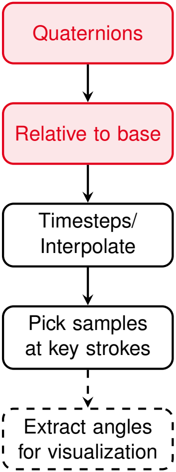


Figure 9: Visualization of relative quaternion rotations, idle pose

Preprocessing and Sampling



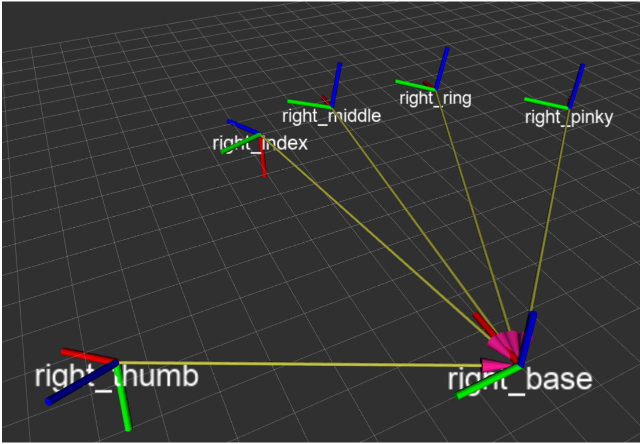
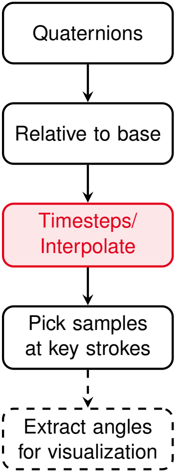


Figure 10: Visualization of relative quaternion rotations, index finger bent

Preprocessing and Sampling



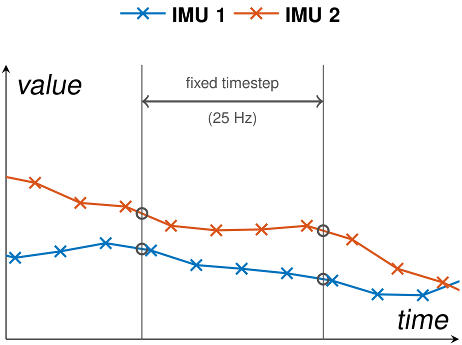
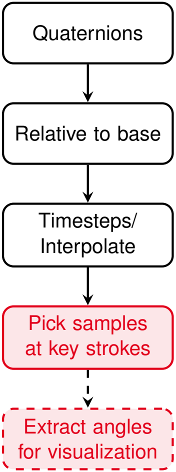


Figure 11: Interpolation of the IMU data (simplified)

Preprocessing and Sampling



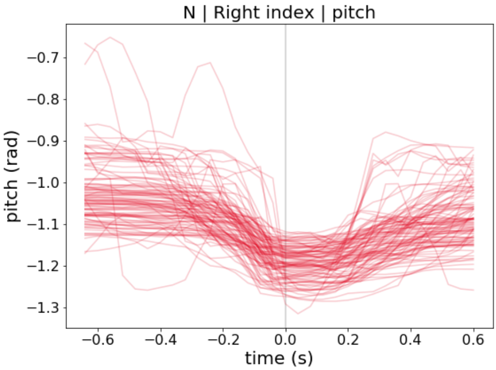


Figure 12: Multiple repetitions of the N key stroke, overlayed at the moment of pressing the key (center line); value plotted is extracted relative pitch angle of right index finger.

Preprocessing and Sampling

Real Preprocessed Data

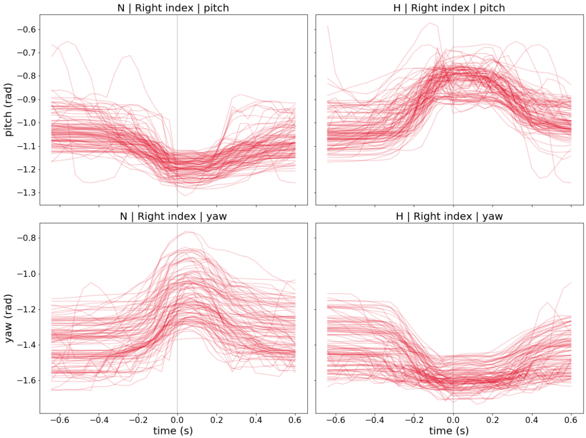


Figure 14: Multiple repetitions of N (*left*) and H (*right*) key strokes, overlayed at the moment of pressing the key (center line); value plotted is extracted relative pitch (*top*)/yaw (*bottom*) angles of right index finger.

Convolutional Neural Networks

Convolution and Pooling

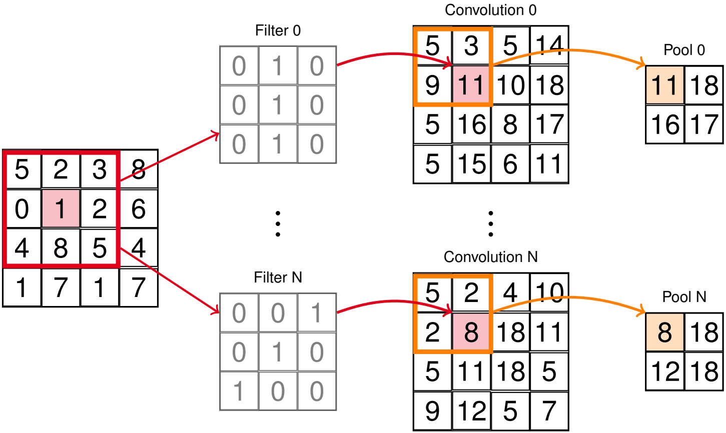
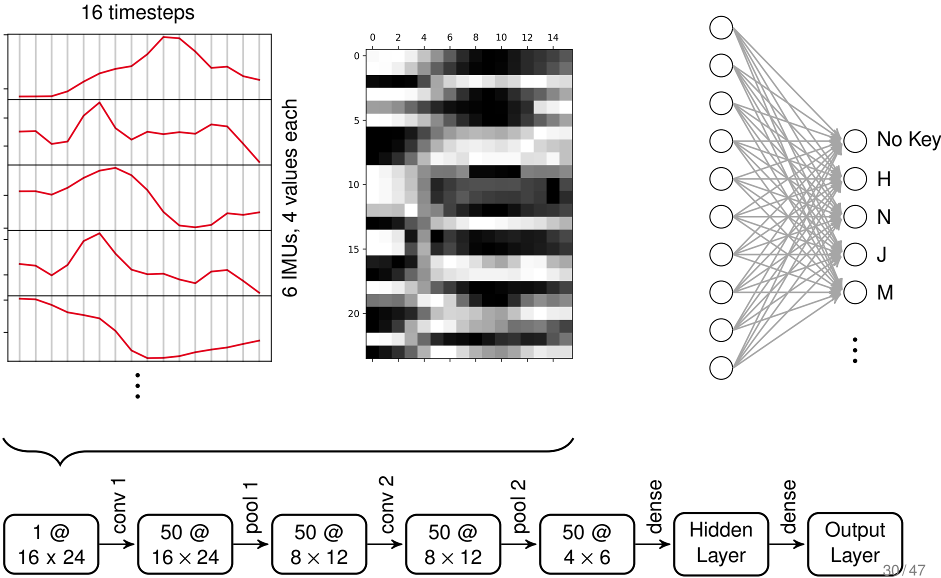


Figure 15: Feature Extraction with CNN

Convolutional Neural Networks

Our Implementation



Experiments

Phase 0 – Pipeline Setup

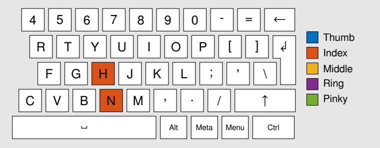
```
imu_ids: [0, 1, 2, 3, 4, 5]
key_codes: [21, 22, 23, 34, ...]
sequence_length: 16

epochs: 0 //infinite
learning_rate: 0.002
batch_size: 100
sampling_rate: 25

network_type: cnn2d
cost_function: mse

convolution_n_filters: 50
convolution_filter_size: [3, 3]
convolution_n_pairs: 2
convolution_arr_dense: [10]
```

Figure 16: Example of a configuration file (truncated)

- easily adjustable
- repeatable experiments

Phase 1 – Slow Single Finger

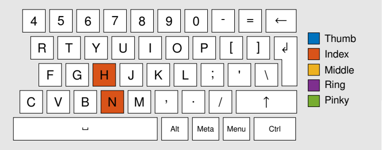
Overview



1 finger, 2 keys

Phase 1 – Slow Single Finger

Overview



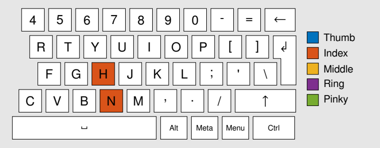
1 finger, 2 keys

Goals:

- detect keystrokes, ignore idle pose
- distinguish between close keys
- evaluate the configuration of the CNN

Phase 1 – Slow Single Finger

Accuracy and Cost Function



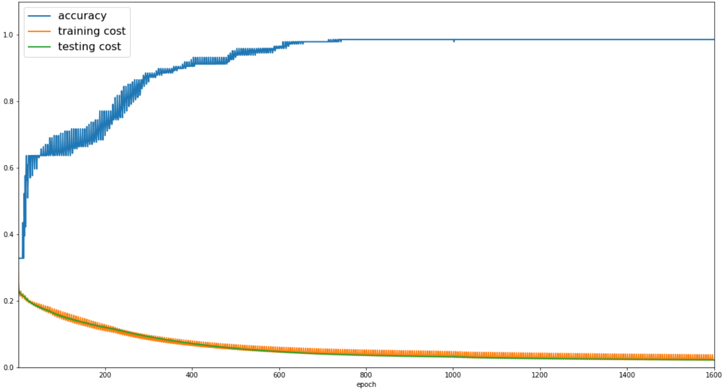
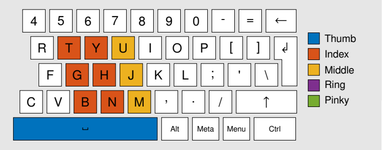


Figure 17: Test results in the first 1600 epochs of learning phase 2

Phase 2 – Slow Multiple Finger

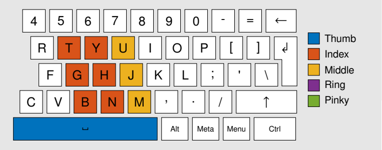
Overview



3 fingers, 10 keys

Phase 2 – Slow Multiple Finger

Overview



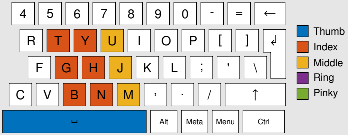
3 fingers, 10 keys

Goals:

- distinguish between fingers
- handle hand movement

Phase 2 – Slow Multiple Fingers

Accuracy and Cost Function



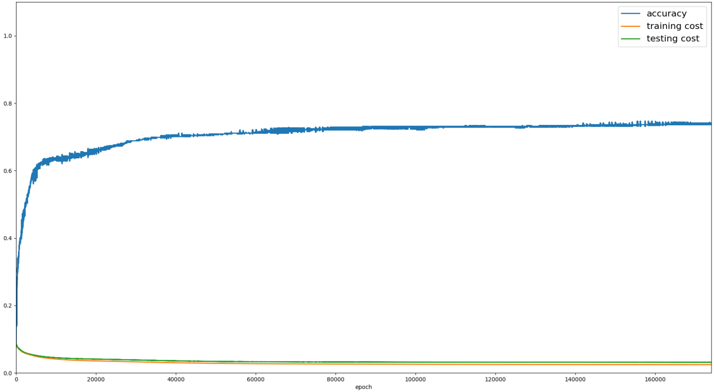
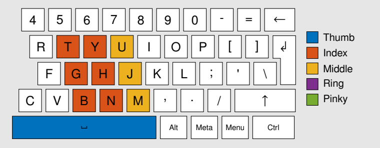


Figure 18: Test results in the first 160000 epochs of learning phase 2

Phase 2 – Slow Multiple Fingers

Evaluation of Predictions

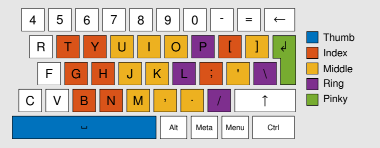


		Predicted									
		no key	U	I	G	H	J	B	N	M	SPC
Actual	no key	0	13	5	9	5	121	2	6	3	0
	U	0	175	0	0	0	0	0	0	0	0
	I	0	1	167	0	0	0	0	0	0	0
	G	0	0	0	181	0	0	0	0	0	0
	H	0	1	5	0	1	169	0	0	0	0
	J	0	0	1	0	1	215	0	3	0	0
	B	0	0	0	0	0	0	167	0	0	0
	N	0	0	0	0	0	0	0	180	0	0
	M	0	0	0	0	0	0	0	0	190	0
	SPC	0	0	0	0	4	178	1	0	0	0

Figure 19: Confusion matrix[15]. From this we calculate the accuracy and per key recall & precision [6]

Phase 3 – Fast Typing

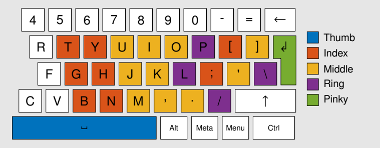
Overview



5 fingers, 27 keys

Phase 3 – Fast Typing

Overview



5 fingers, 27 keys

Goals:

- recognizing every righthand key stroke
- achieve high accuracy
- learn a robust model
- fluent typing

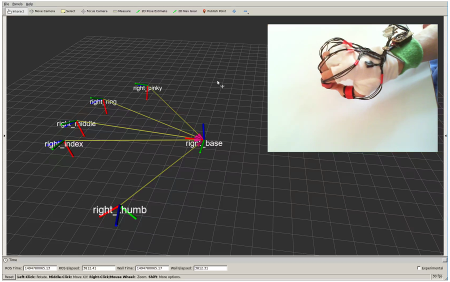
Demo

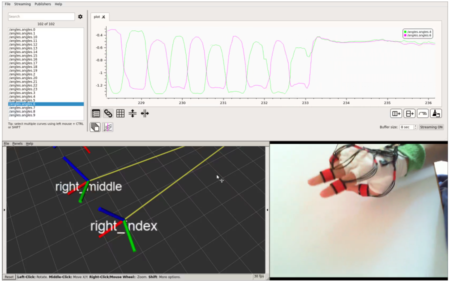
Demo

The Glove

Demo

Backup Videos





Conclusion

Results System Design

Goal (Reminder)

Design a system for recording characteristic hand movements of typing and the corresponding input.

Results System Design

Goal (Reminder)

Design a system for recording characteristic hand movements of typing and the corresponding input.

Successes

- the architecture proved suitable
- the glove is non-obstructive
- performance is good enough for a prototype

Improvements

- gyro clipping
- single robust glove
- generalization to different hand types

Results Machine Learning

Goal (Reminder)

Define an approach for utilizing machine learning to map the recorded data back to the keyboard input.

Results Machine Learning

Goal (Reminder)

Define an approach for utilizing machine learning to map the recorded data back to the keyboard input.

Successes

- slow typing can be distinguished
- preprocessing helps the learning progress
- CNNs can distinguish between different keys

Improvements

- better accuracy
- reduce delay
- detect holding a key
- detect different modes →(non-)writing position

Outlook

Goal (Reminder)

Evaluate the quality of such mapping and discuss whether this principle could be turned into a working keyboard alternative.

Outlook

Goal (Reminder)

Evaluate the quality of such mapping and discuss whether this principle could be turned into a working keyboard alternative.

- reduce delay, remove lookaheads
- increase prediction quality
- two hands
- generalize glove & model
- implement online learning
- better hand pose reconstruction for more use cases

References I

[1] URL: http://dev-blog.mimugloves.com/data-gloves-overview/ (visited on 04/30/2017).

[2] Jason Brownlee. *8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset*. Aug. 2015. URL: http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/ (visited on 04/30/2017).

[3] F. Cavallo et al. "Preliminary evaluation of SensHand V1 in assessing motor skills performance in Parkinson disease". In: *Rehabilitation Robotics (ICORR), 2013 IEEE International Conference on*. June 2013, pp. 1–6. DOI: 10.1109/ICORR.2013.6650466.

[4] Myung-Chul Cho et al. "A pair of Braille-based chord gloves". In: *Proceedings. Sixth International Symposium on Wearable Computers,* 2002, pp. 154–155. DOI: 10.1109/ISWC.2002.1167238.

[5] *CyberGlove Website*. URL: http://www.cyberglovesystems.com/ (visited on 04/30/2017).

[6] Jesse Davis and Mark Goadrich. "The relationship between Precision-Recall and ROC curves". In: *ICML '06: Proceedings of the 23rd international conference on Machine learning*. Pittsburgh, Pennsylvania: ACM, 2006, pp. 233–240. ISBN: 1-59593-383-2. DOI: 10.1145/1143844.1143874.

References II

[7] Jonathan Ellithorpe and Pearl Tan. "The Learning Keyboard. Using the Xbox Kinect to Learn User Typing Behavior". In: 2012.

[8] Marcus Georgi, Christoph Amma, and Tanja Schultz. "Recognizing Hand and Finger Gestures with IMU based Motion and EMG based Muscle Activity Sensing.". In: *BIOSIGNALS*. Ed. by Harald Loose et al. SciTePress, 2015, pp. 99–108. ISBN: 978-989-758-069-7. URL: `http://dblp.uni-trier.de/db/conf/biostec/biosignals2015.html#GeorgiAS15`.

[9] *Gest*. May 2017. URL: `https://gest.co/` (visited on 04/30/2017).

[10] *Hi5 VR Glove*. 2017. URL: `http://hi5vrglove.com/` (visited on 05/01/2017).

[11] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions.". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6.2 (Mar. 2, 2004), pp. 107–116. URL: `http://dblp.uni-trier.de/db/journals/ijufks/ijufks6.html#Hochreiter98`.

[12] *Keyglove*. 2015. URL: `http://www.keyglove.net/` (visited on 05/01/2017).

[13] S. A. Mehdi and Y. N. Khan. "Sign language recognition using sensor gloves". In: *Neural Information Processing, 2002. ICONIP '02. Proceedings of the 9th International Conference on*. Vol. 5. Nov. 2002, 2204–2206 vol.5. DOI: `10.1109/ICONIP.2002.1201884`.

[14] *mi.mu gloves*. URL: `http://mimugloves.com/` (visited on 05/01/2017).

References III

[15] David Poole and Alan K. Mackworth. *Artificial Intelligence - Foundations of Computational Agents.*. Cambridge University Press, 2010, pp. I–XVII, 1–662. ISBN: 978-0-521-51900-7.

[16] *Project Virtual Keyboard*. Dec. 2013. URL: `http://www.senseboard.com/?p=174` (visited on 04/30/2017).

[17] Stuart Reid. *10 misconceptions about Neural Networks*. May 2014. URL: `http://www.turingfinance.com/misconceptions-about-neural-networks/#blackbox` (visited on 05/05/2017).

[18] Jack V Tu. "Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes". In: *Journal of clinical epidemiology* 49.11 (1996), pp. 1225–1231.

[19] Thomas G Zimmerman et al. "A hand gesture interface device". In: *ACM SIGCHI Bulletin*. Vol. 18. 4. ACM. 1987, pp. 189–192.

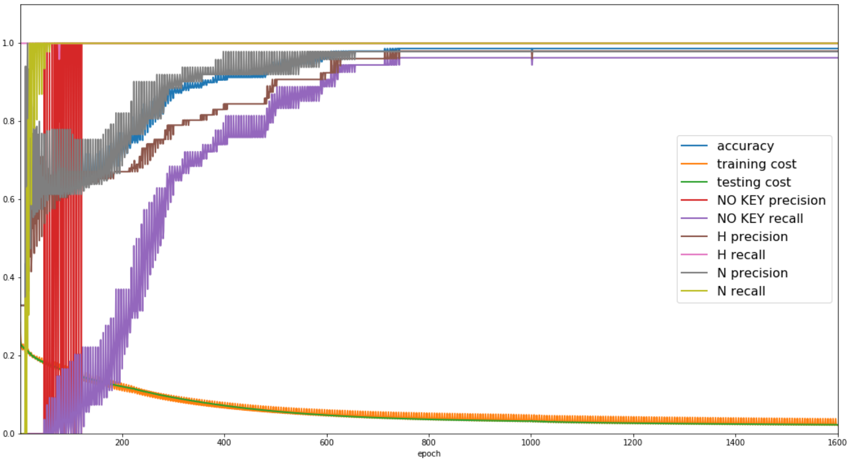


Figure 20: Performance metrics of phase 1, including per key precision and recall