On-Pen handwritten Word Recognition Using Long Short-Term Memory Model

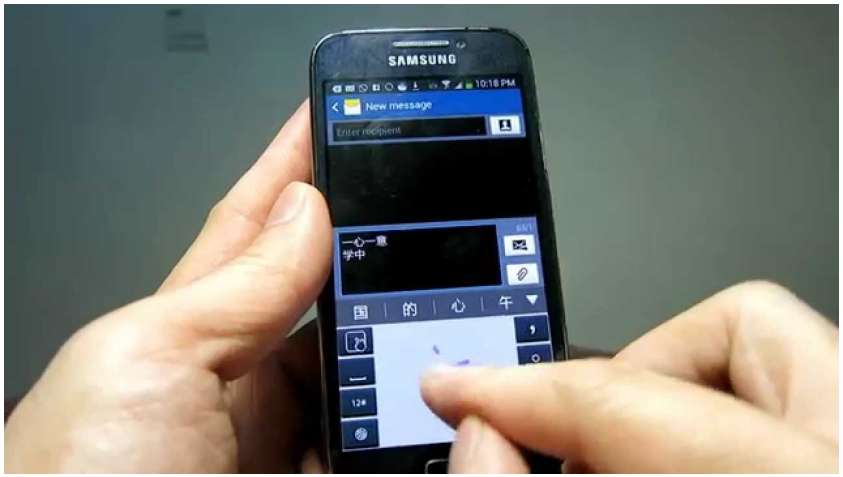
National Tsing Hua University
Embedded Platform Lab
Shun-Fa Zhou
Directed by Prof. Pai H. Chou

Outline

- Introduction
- Related Work
- System Overview
- Data Preprocessing
- Recognition Method
- Evaluation
- Conclusions and Future Work

Motivation

- Limitation of handwriting applications



Objectives

- Develop a convenient text-input interface with pen

Approach

- Use BLE as the wireless communication between pen-side and host-side
- Design a method to segment continuous motion of handwriting
- Calibrate the output of classifier by lexicon support

Contribution

- New hardware model allows writer use any pen to input text
- Improve the character-level on-pen handwritten text recognition to word-level

Outline

- Introduction
- Related Work
  - Isolated vs. continuous
- System Overview
- Data Preprocessing
- Recognition Method
- Evaluation
- Conclusions and Future Work

Isolated handwriting Identification

- For vision-based recognition
  - Feature fetching is the primary element
- Gao et al. [1] use LDA-based feature fetching to identify Chinese word——computer vision
- Ebrahimzadeh et al. [2] use HOG-based feature fetching to recognize isolated digit——computer vision
- Katiyar et al. [3] fetch total 144 hyper features to parse isolated English letter——mathematic and geometry approaches

Isolated handwriting Identification

- For motion-based recognition
  - Two strategy: reconstructing trajectory or using raw data directly
- Sepavand et al. [5] reconstruct the inertial pen's trajectory and use GP-based learning to recognize Arabic characters
- Oh et al. [6] project the raw 6-axis IMU signals to the class space by LDA-based method to identify digits and other three gesture

Continuous handwriting recognition

- Much more harder than isolated handwriting recognition
  - Segmentation pre-processing or post-processing is needed
- Also can be divided into two field:
  - vision-based and motion-based

Continuous handwriting recognition

- For vision-based continuous handwriting recognition
  - Multi-level structure is popular
- Bluche et al. [7] combine CNN feature fetching and HMM decoder
- Kozielski et al. [8] combine LSTM feature fetching, PCA dimensionality reduction, and Tendon GHMM
- Shkarupa et al. [9] propose a sequence-to-sequence model, which consists of LSTM encoder and LSTM decoder.

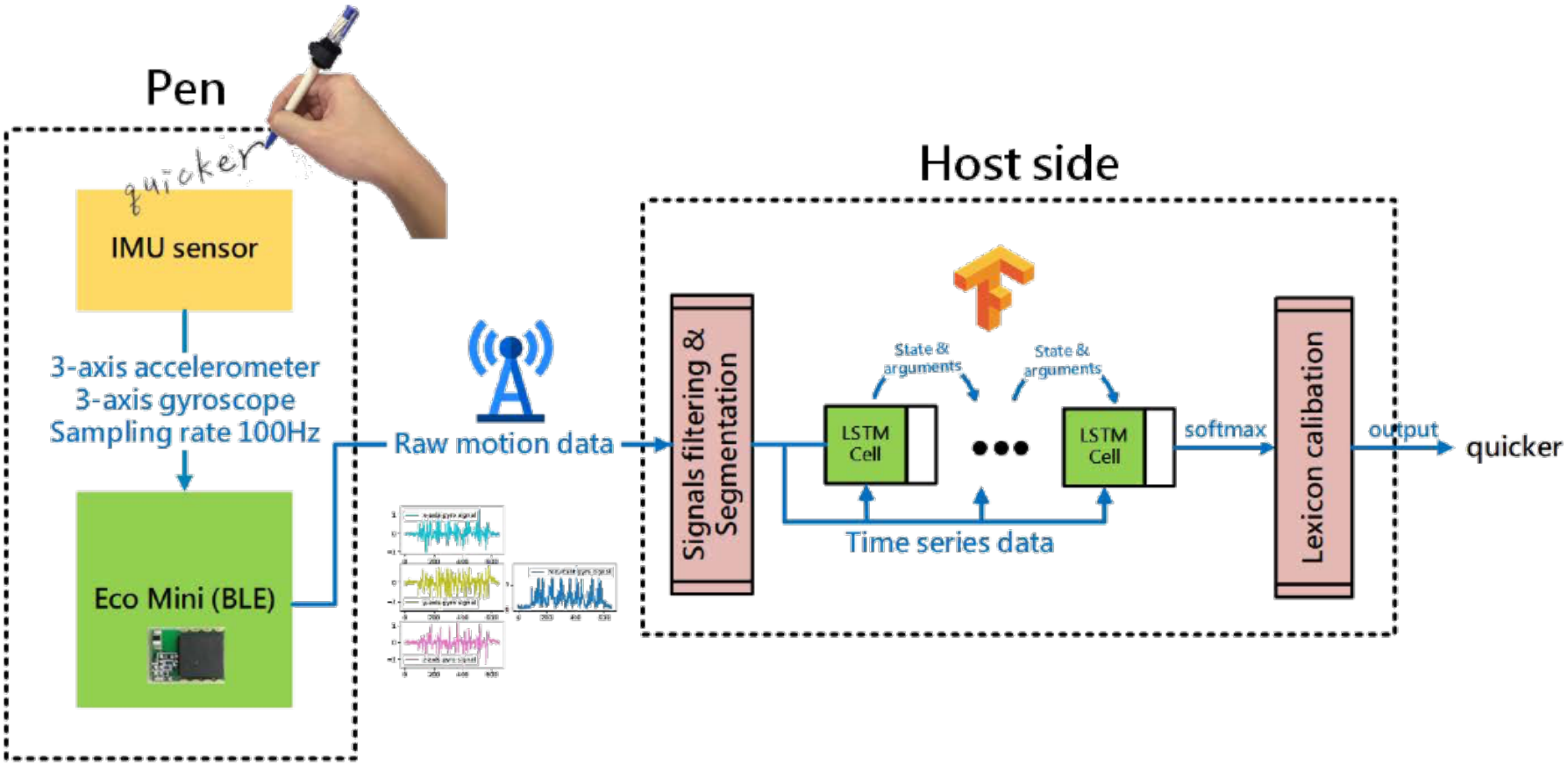
Continuous handwriting recognition

- For motion-based continuous handwriting recognition
  - Rare research about it
- Chen et al [11][12] reconstruct the trajectory of airwriting and classify the path by a HMM model
- Amma et al. [10] propose statistic-based NLP to recognize airwriting words

Outline

- Introduction
- Related Work
- System Overview
- Data Preprocessing
- Recognition Method
- Evaluation
- Conclusions and Future Work

System overview

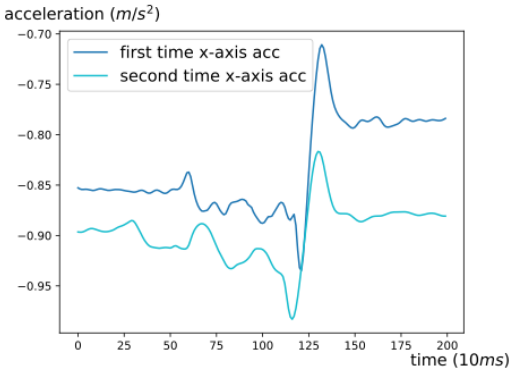


Outline

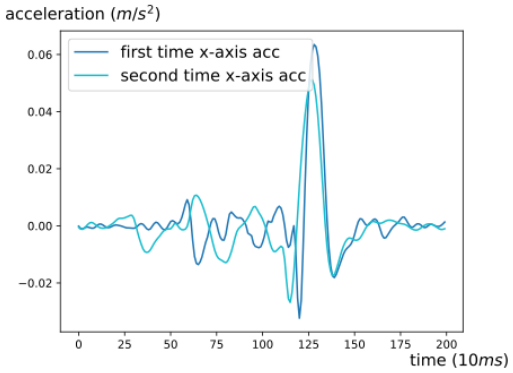
- Introduction
- Related Work
- System Overview
- Data Preprocessing
- Recognition Method
- Evaluation
- Conclusions and Future Work

Gravity elimination

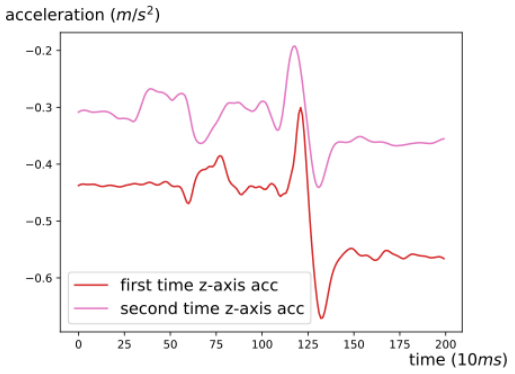
- If we writing letter ‘q’ twice



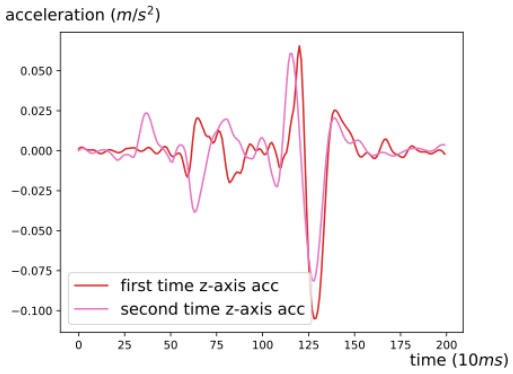
No eliminate gravity x-axis acc signals



Eliminate gravity x-axis acc signals



No eliminate gravity z-axis acc signals



Eliminate gravity z-axis acc signals

Gravity elimination

- Data difference caused by the sensor orientation
- How to eliminate gravity's contribution
  - Low-pass filter or high-pass filter

Signal filtering

- Encountering problem:
  - IMU signals contain noises result from the sensor error
  - When writing, the frequency of pen's movement is in the range of 1 to 10 Hz
- Solution:
  - Butterworth low-pass filter
  - Use both raw and filtered as features

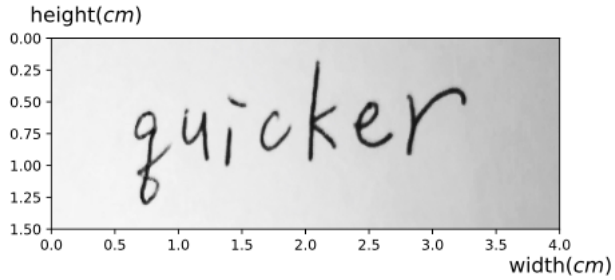
$$G_n(\omega) = 1/\sqrt{1+(\omega/\omega_c)^{2n}}, \begin{cases} n = 1 \\ \omega = 100Hz \\ \omega_c = 2.2Hz \end{cases}$$

Data segmentation

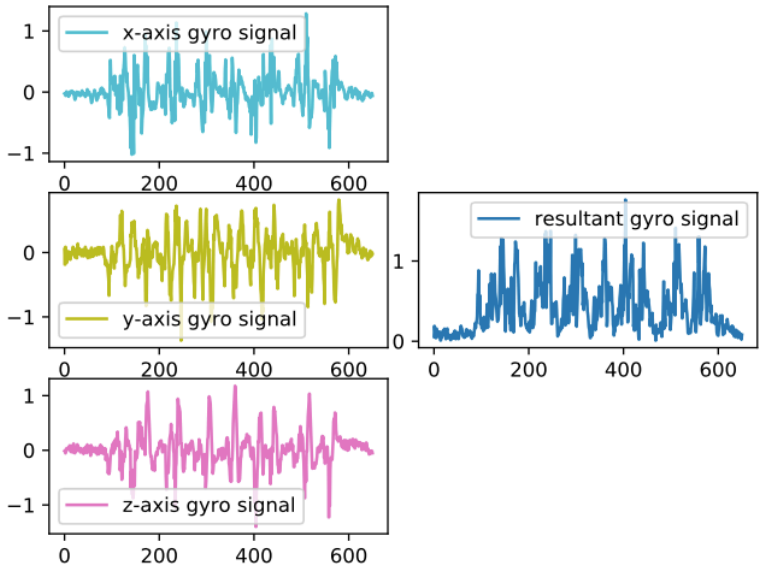
- Traditional ways to segment data:
  - Threshold
  - HMM (Hidden Markov Model)
  - DTW (dynamic time warping)
- Apply HMM on our system:
  - Bad results, only detect the start and end of the event
- Apply DTW on our system:
  - High computation cost:
    O(mn)*26 kinds of characters/judge

Data segmentation

- Solution:
  - Find valid windows by resultant gyroscope signal
- Step 1: Resultant the 3-axis gyroscope signals



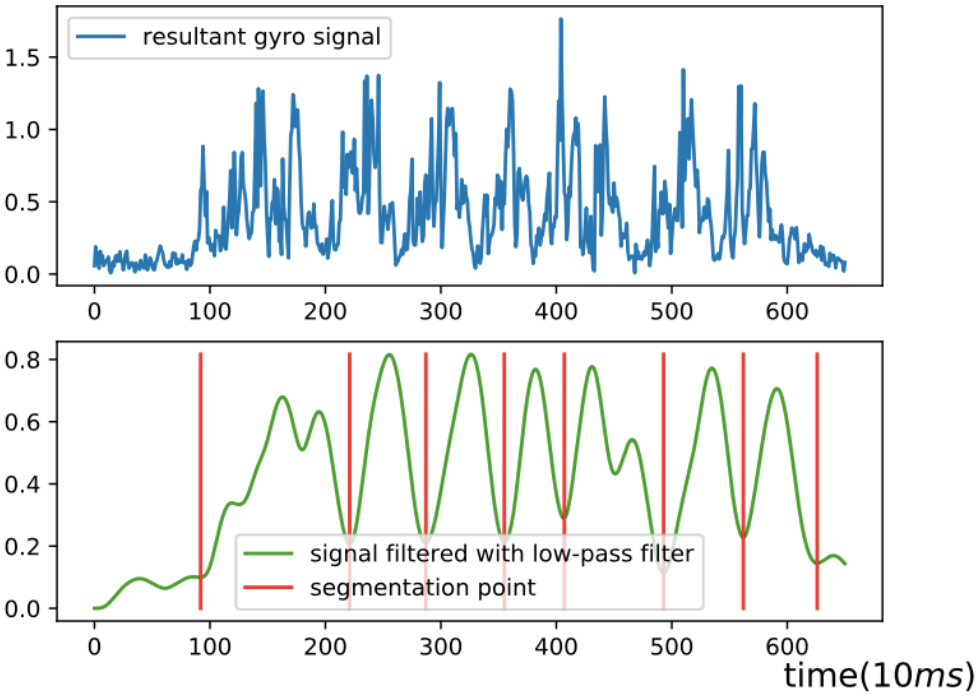
(a) Handwriting of 'quicker'



(b) Transform 3-axis gyroscope signal to resultant gyroscope signal by cosine formula

Data segmentation

- Step 2: low-pass filter and find local minimum
- Step 3: find valid windows by threshold-based method



(c) Filter the resultant gyroscope signal and find segmentation point

Signal normalization

- Feature scaling (4.3)
- Linear interpolation (4.4)

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{4.3}$$

$$f(x) = \frac{f(x_{prev})(x_{next} - x) + f(x_{next})(x - x_{prev})}{x_{next} - x_{prev}} \tag{4.4}$$

- After data preprocessing, raw data of each segment is encoded to a N*100 matrix, N denotes the number of features

Outline

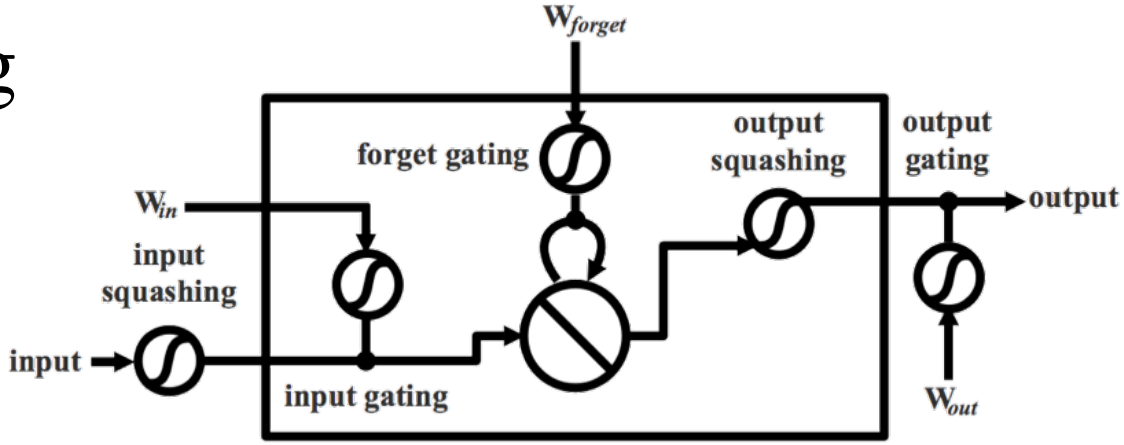
- Introduction
- Related Work
- System Overview
- Data Preprocessing
- Recognition Method
- Evaluation
- Conclusions and Future Work

Background

- RNN (Recurrent neural network)
  - STM (Short-Term Memory) mechanism
  - Suitable for sequence data
- Weak point:
  - STM is implemented by delay training
    - The longer the delay, the poor the performance
  - Gradient vanishing or exploding problem

Long Short-Term Memory

- Solve the gradient problem of RNN:
  - CEC (Constant Error Coursel)
  - Input and output gating
- The memory are kept too long
  - Forget gating

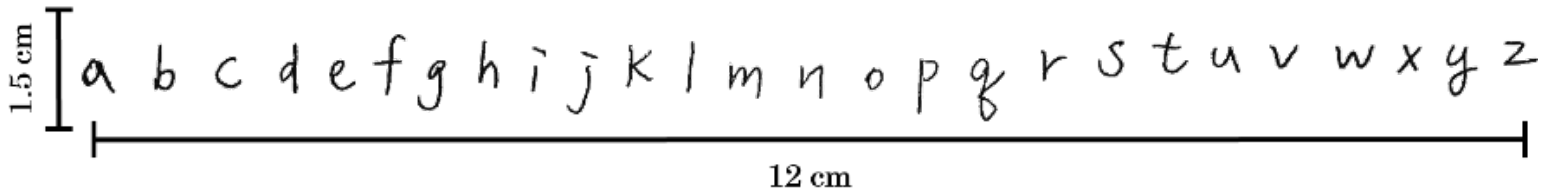


Lexicon Support

- Bayes' corrector
  - We can assume that

$$P(W) = \phi P(c_0)P(c_1)\cdots P(c_n), \quad W = c_0c_1\cdots c_n \tag{5.10}$$

- Step 0: define the probability of misspelling
  - {b,c,f,h,u,v} are easily mistaken as {p,e,l,n,a,r}



Bayes' corrector

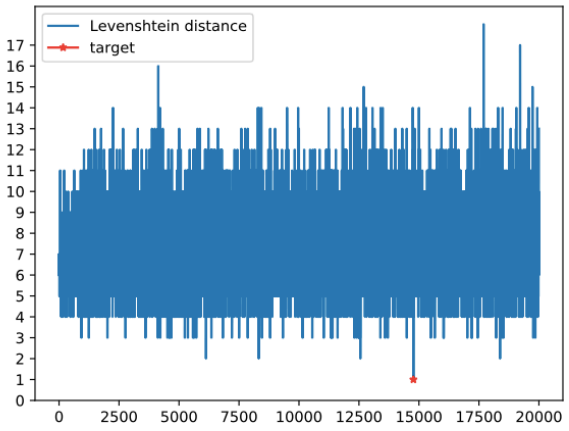
- Step 1: generate all possible string that has 1 or 2 edit distance to the origin string
  - For example: LSTM output string 'qaicker'
  - Generate 'quicker', 'quickev', 'qaickev', etc.
- Step 2: select strings that exist in the dictionary as candidates
- Step 3: calculate the probability of each candidate according to Bayes' theorem

Weighted Levenshtein distance

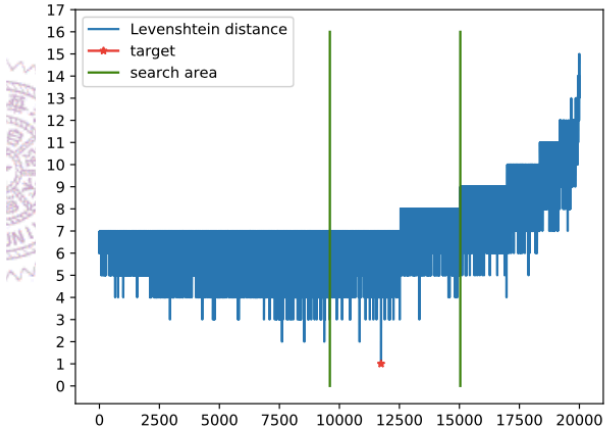
- Encounter problem:
  - Basic Bayes' corrector may fail
- Solution:
  - Search the most similar word in dictionary by Levenshtein distance, also called 'edit distance'
- However, the substitution, insertion, and deletion weight not always equal to 1 in each case
- For example:

Weighted Levenshtein distance

- Lower easily-mistaken substitution weights
  - {b,c,f,h,u,v} to {p,e,l,n,a,r}
- The insertion and deletion weights are also tuned
- Search the most similar word in length-ordered dictionary



(a) search in a lexicographical ordered dictionary with 20k vocabulary size

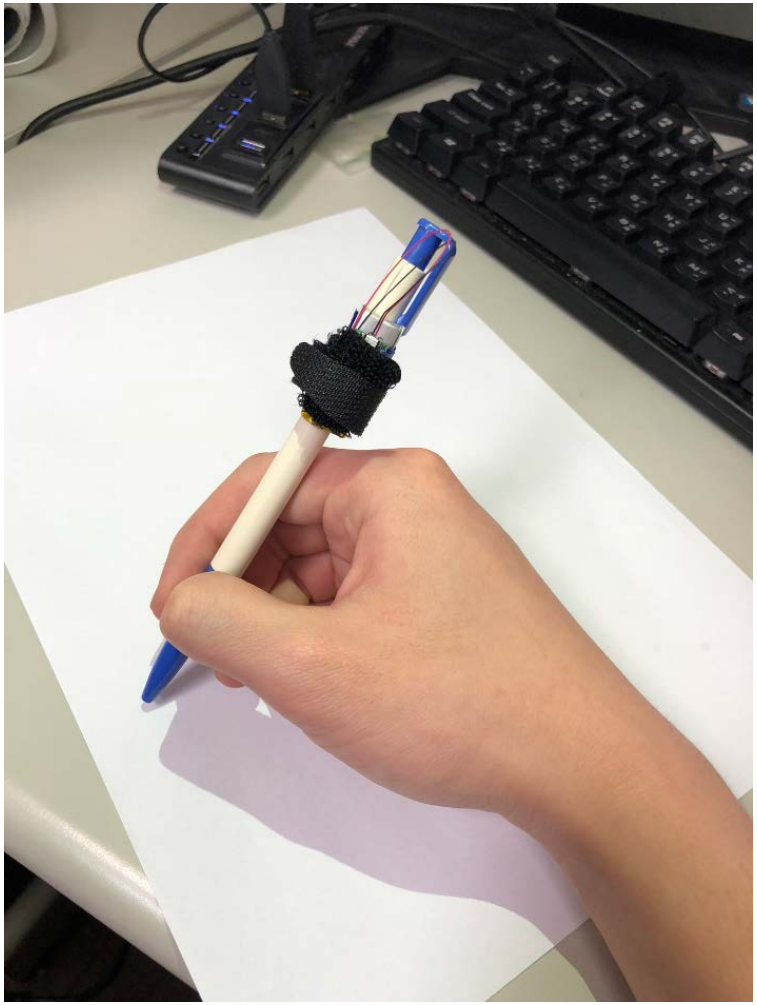


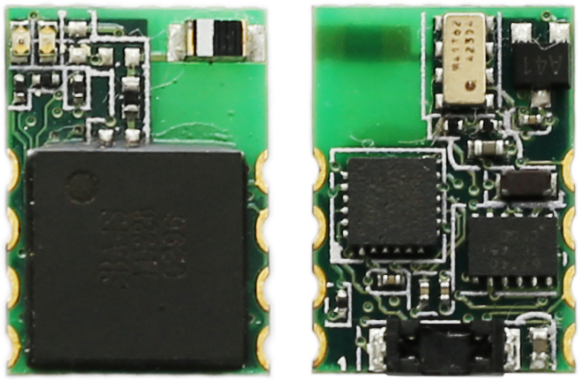
(b) search in a length ordered dictionary with 20k vocabulary size

Outline

- Introduction
- Related Work
- System Overview
- Data Preprocessing
- Recognition Method
- Evaluation
- Conclusions and Future Work

Hardware setup





EcoMini

Data Acquisition

- Single writer and fixed sensor orientation (because without transformation to global coordinate)
- All words we write are generated from a 20k vocabulary size dictionary
- The dataset contains 749 words, total 3761 seconds data
- 538 training data, 86 validation data, and 125 test data

System Environment

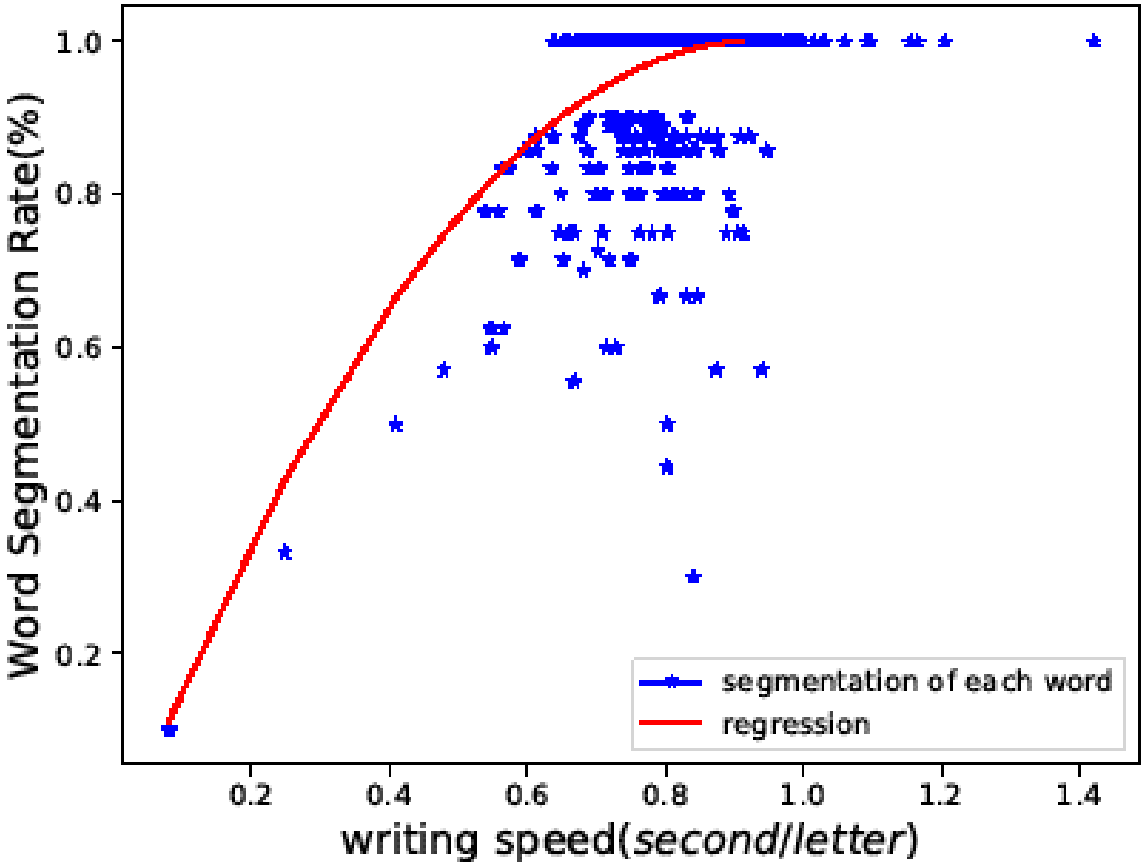
- Training environment:
  - TensorFlow Library
  - Nvidia GTX1080 Ti GPU with 11G device memory
  - 0.001 learning rate, 64 neurons in LSTM
  - Evaluate each configuration after 2500 epoch, roughly 20-30 minutes
- Test environment:
  - Ubuntu 16.04 virtual machine environment
  - 2 cores and 2G memory

Experiment results

- Segmentation robustness
- Accuracy of different feature selection
- Improvement by lexicon support

Experiment results

- Segmentation robustness



Experiment results

- Accuracy of different feature selection

feature	symbol
x-axis accelerometer	a_x
y-axis accelerometer	a_y
z-axis accelerometer	a_z
x-axis gyroscope (angular velocity)	g_x
y-axis gyroscope (angular velocity)	g_y
z-axis gyroscope (angular velocity)	g_z
filterd x-axis accelerometer	a_x^f
filterd y-axis accelerometer	a_y^f
filterd z-axis accelerometer	a_z^f
filterd x-axis gyroscope (angular velocity)	g_x^f
filterd y-axis gyroscope (angular velocity)	g_y^f
filterd z-axis gyroscope (angular velocity)	g_z^f
resultant gyroscope	g_r
filterd resultant gyroscope	g_r^f

(a) 14 features we generate and their symbols

feature selection	accuracy
full configuration	0.688
$a_x, a_y, a_x^f, a_y^f, G, G^f, g_r, g_r^f$	**0.763**
$a_x, a_z, a_x^f, a_z^f, G, G^f, g_r, g_r^f$	0.708
$a_y, a_z, a_y^f, a_z^f, G, G^f, g_r, g_r^f$	0.692
$a_x, a_x^f, G, G^f, g_r, g_r^f$	0.609
$A, A^f, g_x, g_y, g_x^f, g_y^f, g_r, g_r^f$	0.720
$A, A^f, g_y, g_z, g_y^f, g_z^f, g_r, g_r^f$	0.615
$A, A^f, g_x, g_z, g_x^f, g_z^f, g_r, g_r^f$	0.663
$A, A^f, g_x, g_x^f, g_r, g_r^f$	0.471

(b) feature selection performance

Table 6.1: Analysis of feature selection

Experiment results

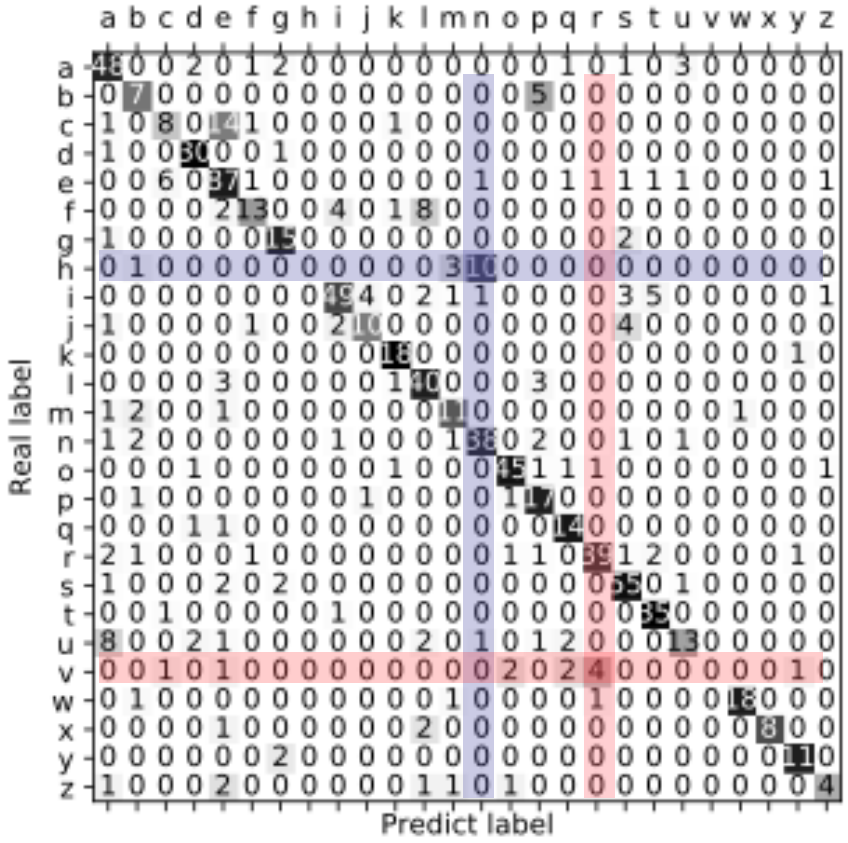
- Improvement by lexicon support

Corrector	WER	CER	Correction time (second/word)
without corrector	0.776	0.237	None
with basic Bayes' corrector	0.504	0.452	**0.032**
with only WLDS	0.328	0.144	0.484
with Bayes' corrector support by WLDS	**0.264**	**0.120**	0.232

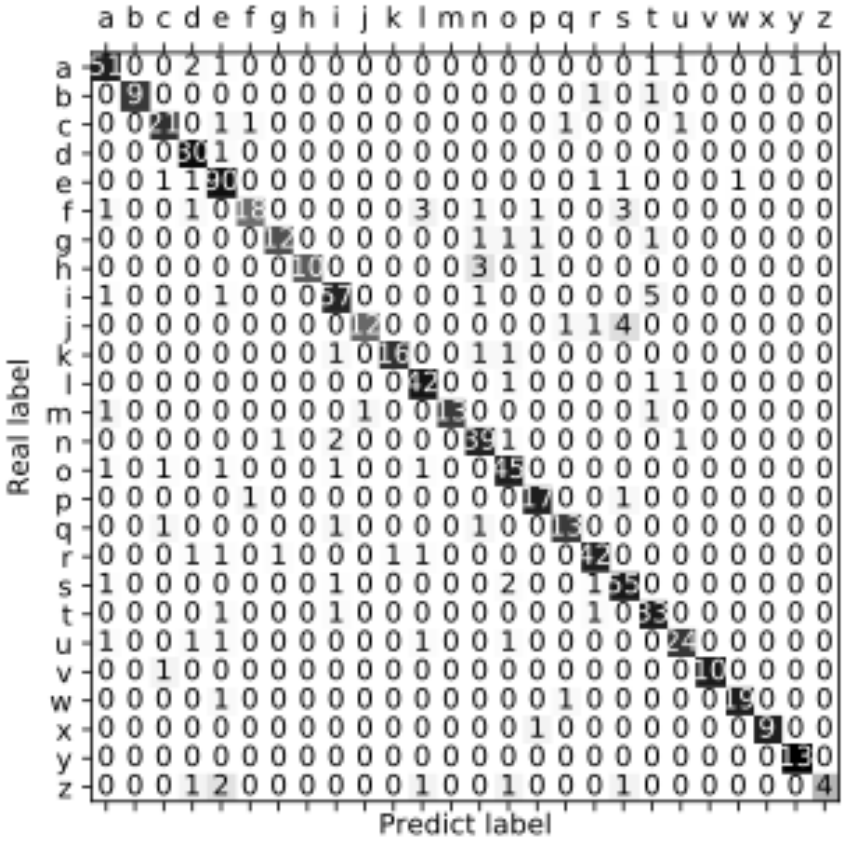
Table 6.2: Corrector performance comparison

Experiment results

- Confusion matrix



Classification of LSTM

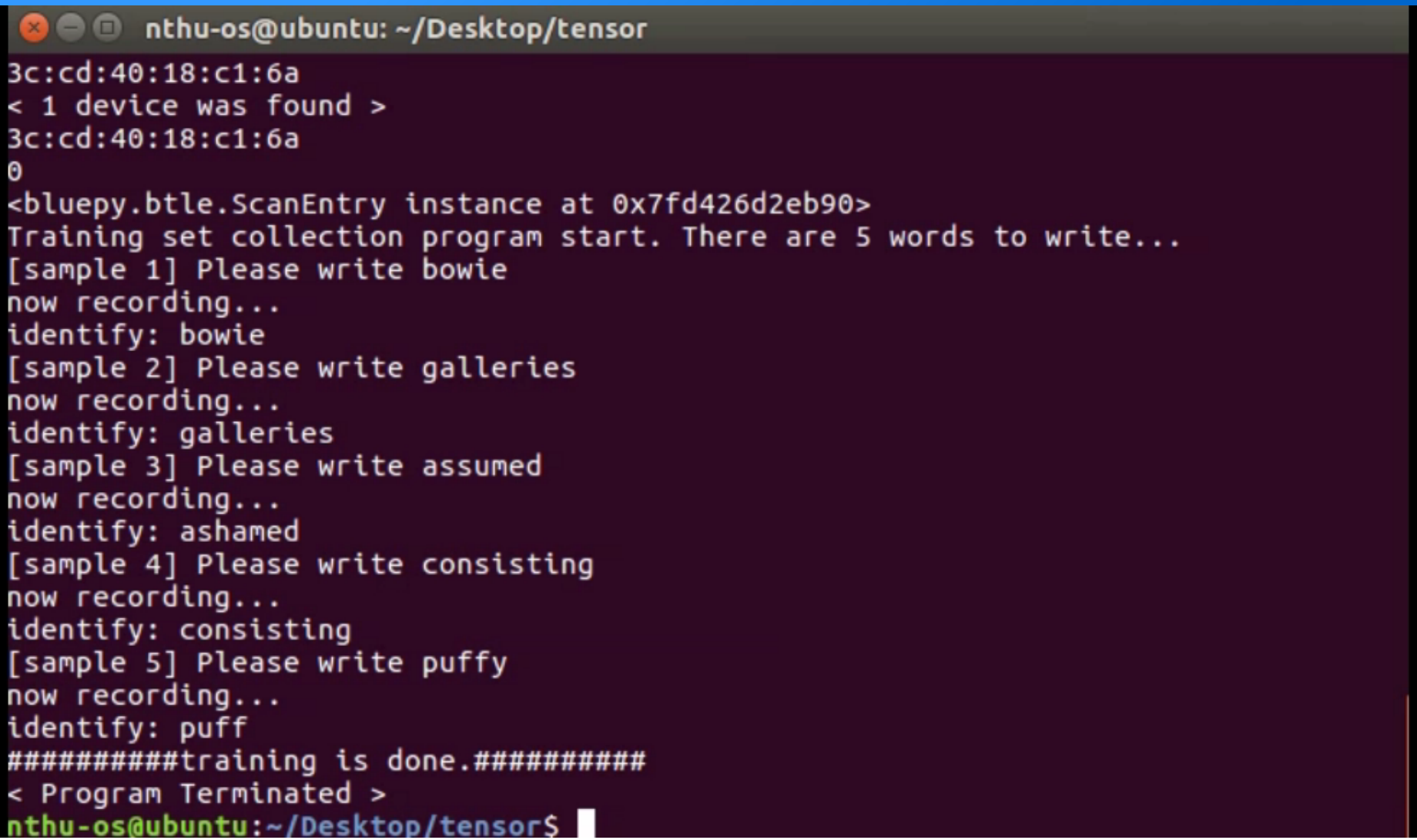


Calibrated by lexicon support

Discussion

- Convenience
  - Use any pen as a text-input interface without a writing pad
  - After transforming to global coordinate, it will be a user independent application
- Scalability
  - If the number of recognized label increasing, we may spend more time on training. However, the recognition time is roughly the same.
- Performance
  - Handle a writing speed slower than 0.7 second/letter
  - 26.4% WER and 12% CER

Demo


nthu-os@ubuntu: ~/Desktop/tensor
3c:cd:40:18:c1:6a
< 1 device was found >
3c:cd:40:18:c1:6a
0
<bluepy.btle.ScanEntry instance at 0x7fd426d2eb90>
Training set collection program start. There are 5 words to write...
[sample 1] Please write bowie
now recording...
identify: bowie
[sample 2] Please write galleries
now recording...
identify: galleries
[sample 3] Please write assumed
now recording...
identify: ashamed
[sample 4] Please write consisting
now recording...
identify: consisting
[sample 5] Please write puffy
now recording...
identify: puff
##########training is done.##########
< Program Terminated >
nthu-os@ubuntu:~/Desktop/tensor$

Outline

- Introduction
- Related Work
- System Overview
- Data Preprocessing
- Recognition Method
- Evaluation
- Conclusions and Future Work

Conclusions

- A new hardware model allows writer use any pen as a text-input interface
- By auto-segmenting and lexicon support, we improve on-pen handwritten text recognition from character-level to word-level
- The auto-segmentation can handle 0.7 sec/letter writing speed
- The lexicon support improve LSTM recognizer from 0.776 WER to 0.264, 0.237 CER to 0.12

Future Work

- More powerful classifier
  - Bidirectional LSTM, Multi-level LSTM
  - Attention mechanism of seq2seq Model
- Transforming from the sensor coordinate to the Earth coordinate

Acknowledgments

- Thanks to Professor **Pai H. Chou** for guidance.
- Thanks to Yu-Chieh Teng, Yi-Chang Huang, Yu-Hung Yeh, James Chan, and Zhe-Ting Liu for deep learning and Tensorflow Library guidance.
- Thanks to Embedded Platform Lab of National Tsing Hua University, Taiwan.

Thank You

Determine the number of neurons

- The implementation of LSTM
  - $i[t] = \sigma(W[x \to i]x[t] + W[h \to i]h[t-1] + b[1 \to i])$
  - $f[t] = \sigma(W[x \to f]x[t] + W[h \to f]h[t-1] + b[1 \to f])$
  - $z[t] = \tanh(W[x \to c]x[t] + W[h \to c]h[t-1] + b[1 \to c])$
  - $c[t] = f[t]c[t-1] + i[t]z[t]$
  - $i[t] = \sigma(W[x \to o]x[t] + W[h \to o]h[t-1] + b[1 \to o])$
  - $i[t] = o[t]\tanh(c[t])$
- For each gate: $(N_I \times N_H) + (N_H \times N_H) + (N_H)$
- Total weights of four gates: four times of the equation