engineering.asu.edu

FMHash: Deep Hashing of In-Air-Handwriting for User Identification

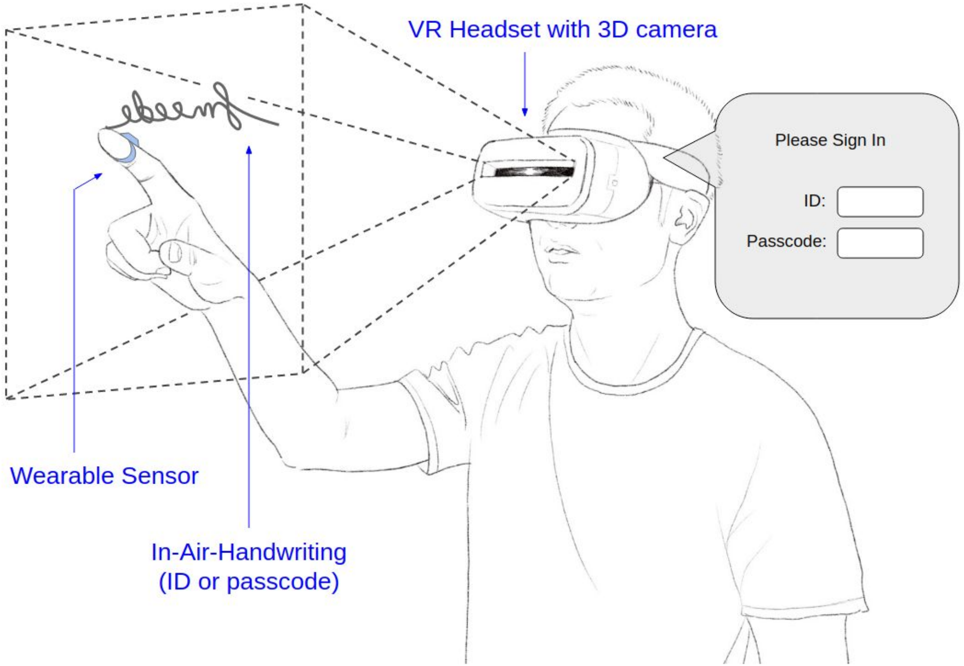
Duo Lu, Dijiang Huang, Anshul Rai

School of Computing, Informatics, and Decision Systems, Arizona State University

{duolu, dijiang.huang, anshulrai}@asu.edu

presented at IEEE International Conference on Communications (**ICC 2019**)

How to input an account ID using a piece of handwriting?



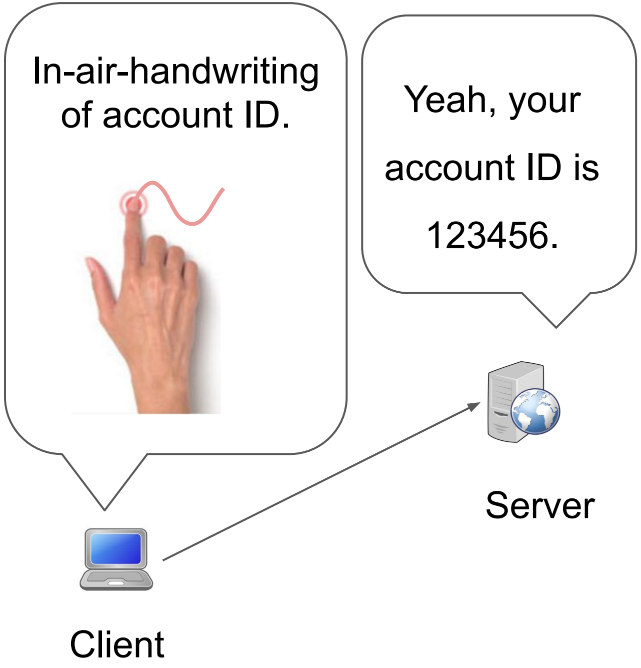
Deep Hashing of In-Air-Handwriting for User Identification

Design Goals:

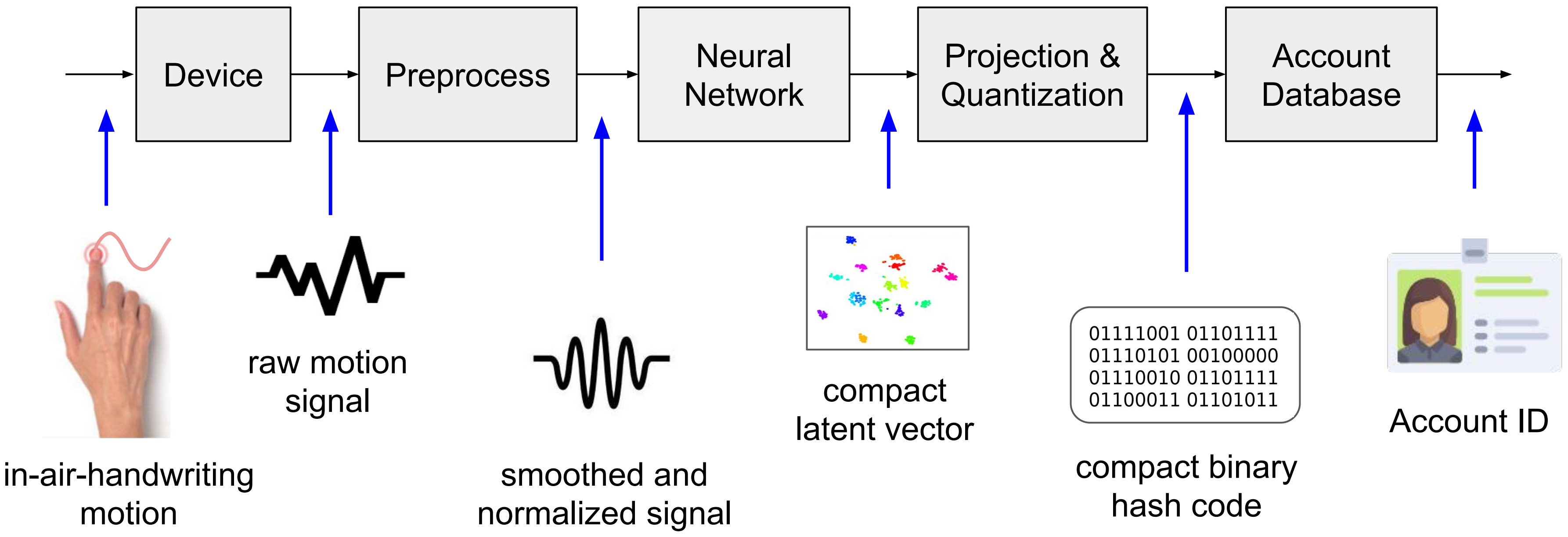
- Same hash code for the same user writing the same string.
- Hash code for different writing should differ at least 2 bits.
- Fast identification with large account database.
- Fast training.

Technical Challenges:

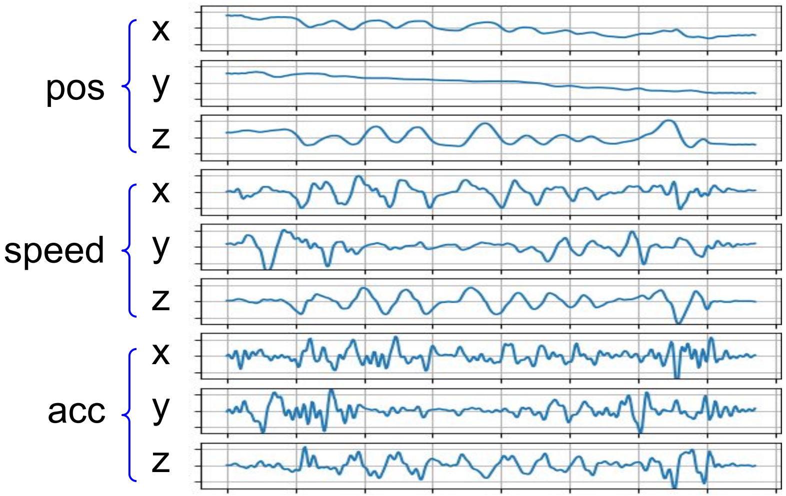
- Lack of understanding of the features.
- Difficult to train deep model with very a few data samples;



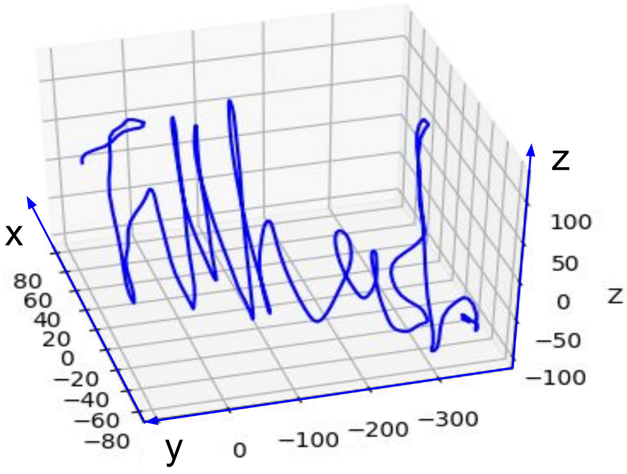
FMHash System Architecture



An Example of the Hand Motion Signal



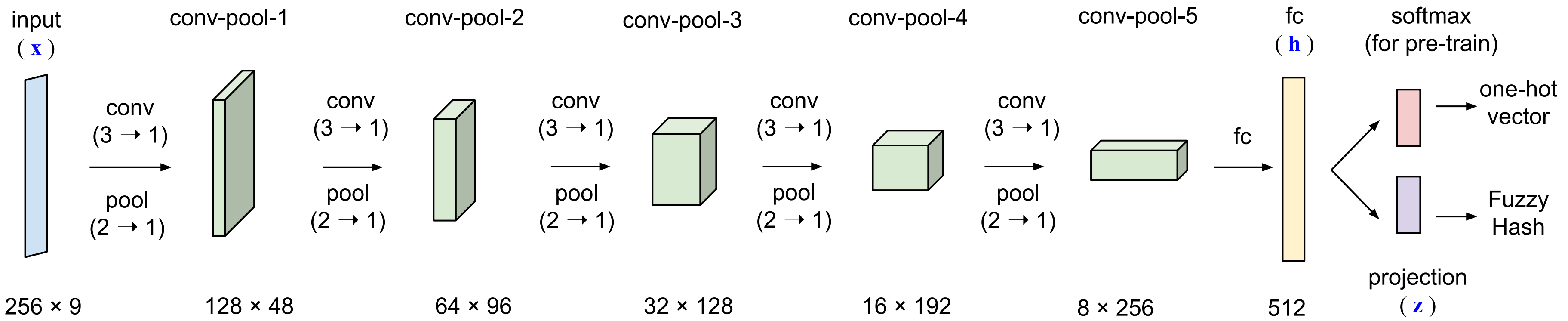
motion signal



in-air-handwriting trajectory

The hand motion signal is obtained using the Leap Motion Controller.

Neural Network Architecture



This network takes input : A piece of in-air-handwriting signal $\mathbf{x}$ corresponding to an account ID;

This network output: (1) one-hot vector of account ID, only for pre-train;

(2) latent vector $\mathbf{h}$ and projected vector $\mathbf{z}$ of the hash code size (K bits);

(3) hashcode $b_i = \mathrm{sign}(z_i)$.

Progressive Training

Step 1: Pre-train the network with cross entropy loss, in 1,000 iterations

Step 2: Train the network with minibatch of pairs $(\mathbf{z_1^{(i)}}, \mathbf{z_2^{(i)}})$, and label $y^{(i)}$ = 0 (same) or 1 (diff).

projected vectors of different classes should separate at least m in L_2 distance

same class, same projected vector

- loss function:

$$L^{(i)} = (1 - y^{(i)})||\mathbf{z}_1^{(i)} - \mathbf{z}_2^{(i)}|| + y^{(i)}\max(m - ||\mathbf{z}_1^{(i)} - \mathbf{z}_2^{(i)}||, 0)$$
$$+\alpha(P(\mathbf{z}_1^{(i)}) + P(\mathbf{z}_2^{(i)})) + \beta(Q(\mathbf{z}_1^{(i)}) + Q(\mathbf{z}_2^{(i)})).$$

regularizer

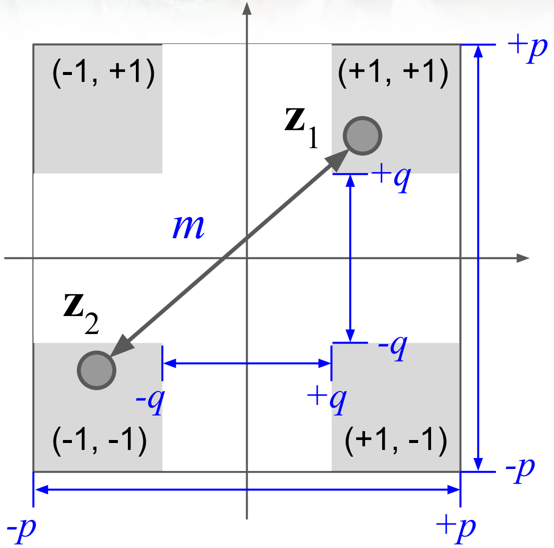
- 8,000 iterations, where α = 1, β = 0

Step 3: Fine tune the network with the same loss, 2,000 * 3 iterations, where α = 0.1, β = 0.001 to 0.1

Regularizer

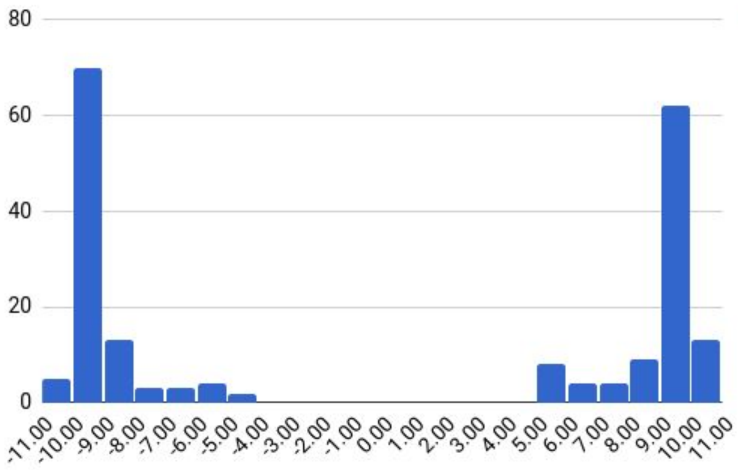
$$L^{(i)} = (1 - y^{(i)})||\mathbf{z}_1^{(i)} - \mathbf{z}_2^{(i)}|| + y^{(i)}\max(m - ||\mathbf{z}_1^{(i)} - \mathbf{z}_2^{(i)}||, 0)$$
$$+\alpha(P(\mathbf{z}_1^{(i)}) + P(\mathbf{z}_2^{(i)})) + \beta(Q(\mathbf{z}_1^{(i)}) + Q(\mathbf{z}_2^{(i)})).$$

$$P(\mathbf{z}^{(i)}) = \sum_{j=1}^{K} max(|z_j^{(i)}| - p, 0) \quad Q(\mathbf{z}^{(i)}) = \sum_{j=1}^{K} max(q - |z_j^{(i)}|, 0)$$

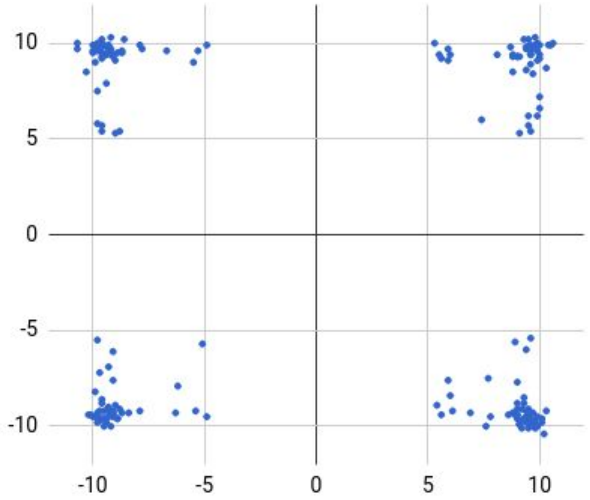


- The regularizer forces each element of the $\mathbf{z}$ vector to be in the region $[-p, -q]$ or $[+q, +p]$.
- After quantization, $[-p, -q]$ is -1, $[+q, +p]$ is +1.
- Pair $(\mathbf{z}_1, \mathbf{z}_2)$ of different classes is forced to separate at least m in L_2 distance.
- Elements of $(\mathbf{z}_1, \mathbf{z}_2)$ will reside in different regions if m is carefully chosen (e.g., $p\sqrt{K}$).

Effects of the Regularizer



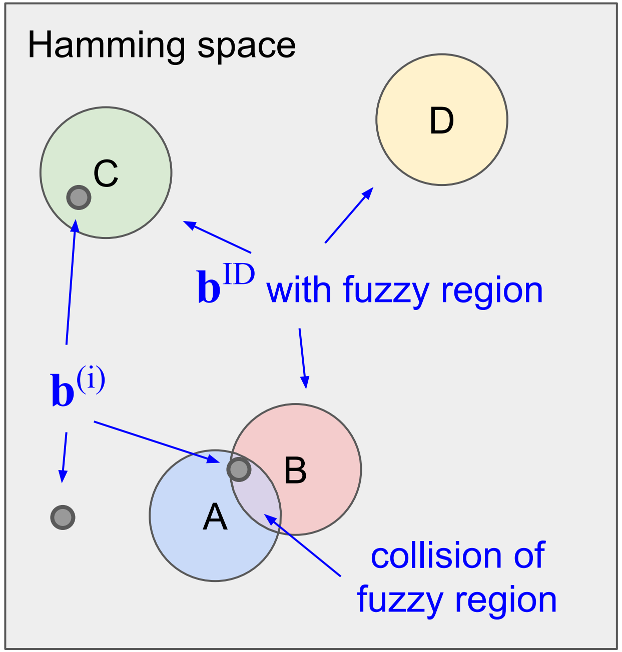
Histogram of **z**



Plot of the first two dimensions of **z**

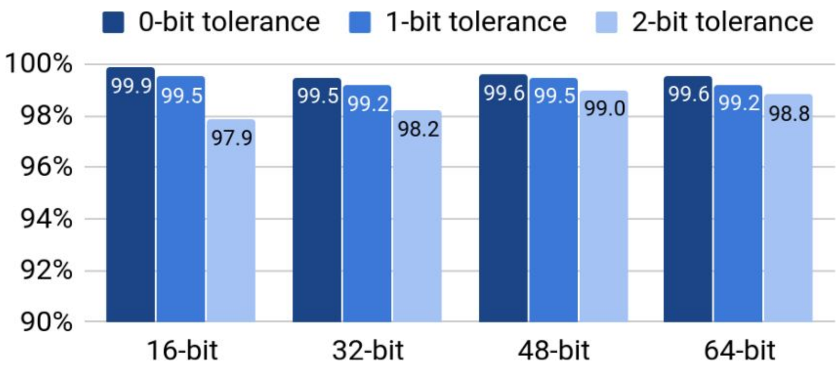
Identification Procedure and Verification

1. Each account has an ID, a hash code $\mathbf{b}^{\mathrm{ID}}$ and a vector $\mathbf{h}^{\mathrm{ID}}$.
2. A hash table is built with all accounts and a tolerance l.
   - $\mathbf{b}^{\mathrm{ID}}$ occupies only 1 bucket if l = 0;
   - $\mathbf{b}^{\mathrm{ID}}$ occupies K buckets if l = 1, with 1 bit fuzziness;
3. Given the hash table and an identification request $\mathbf{x}^{(\mathrm{i})}$,
   - a hash code $\mathbf{b}^{(\mathrm{i})}$ and a vector $\mathbf{h}^{(\mathrm{i})}$ is generated,
   - hash table search for candidate accounts $\{\mathrm{ID1}, \mathrm{ID2}, \ldots\}$
4. Obtain nearest neighbor of $\mathbf{h}^{(\mathrm{i})}$ in $\{\mathbf{h}^{\mathrm{ID1}}, \mathbf{h}^{\mathrm{ID2}}, \ldots\}$
5. The result is the final identified account ID of $\mathbf{x}^{(\mathrm{i})}$.

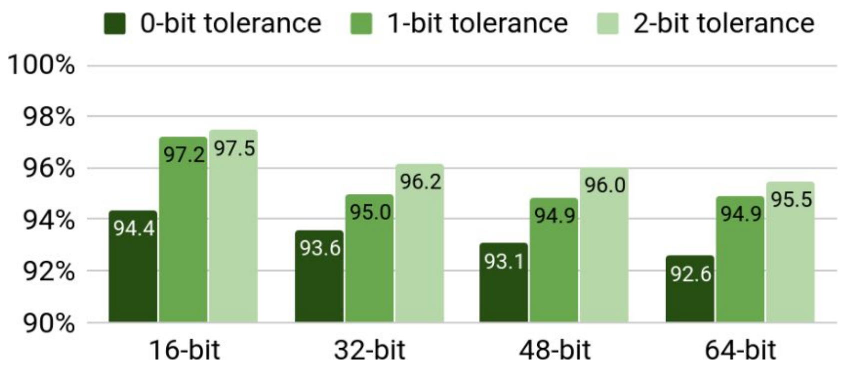


Empirical Results

precision



recall

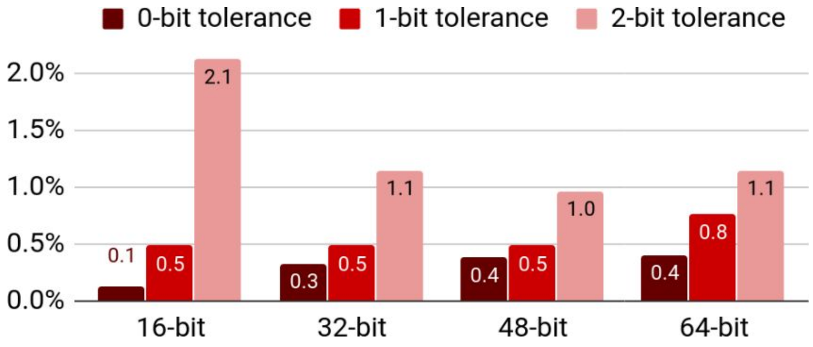


- Performance is robust with the number of hash bits.
- Larger searching range (i.e., more tolerance) helps recall but hurts precision.
- Misidentification is more harmful, and hence, we favor precision than recall.

Empirical Results

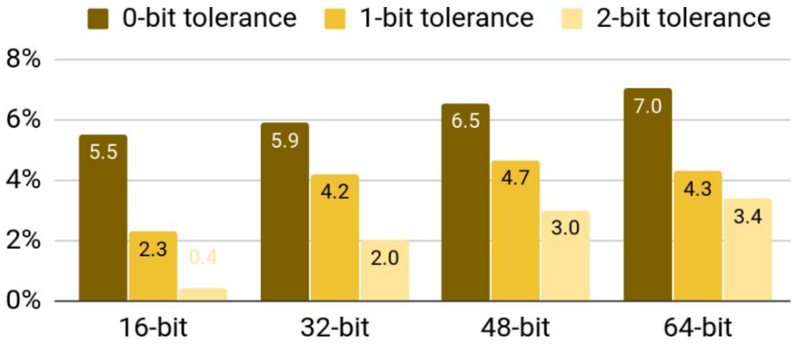
misidentification

(wrongly identified as some account)



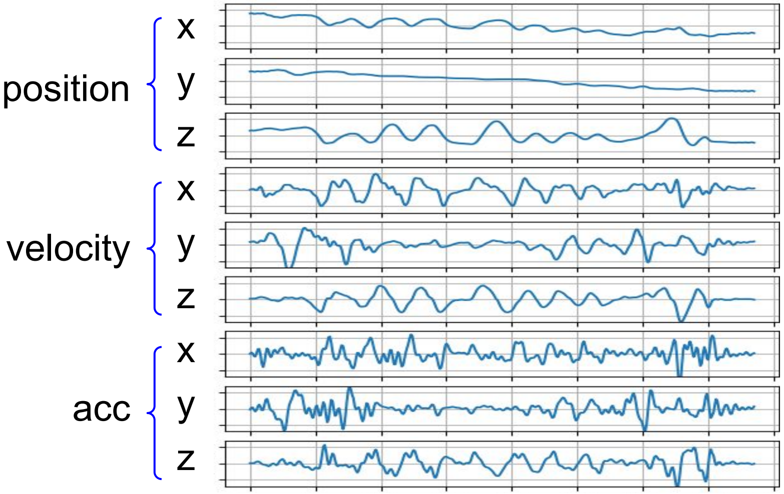
failure of identification

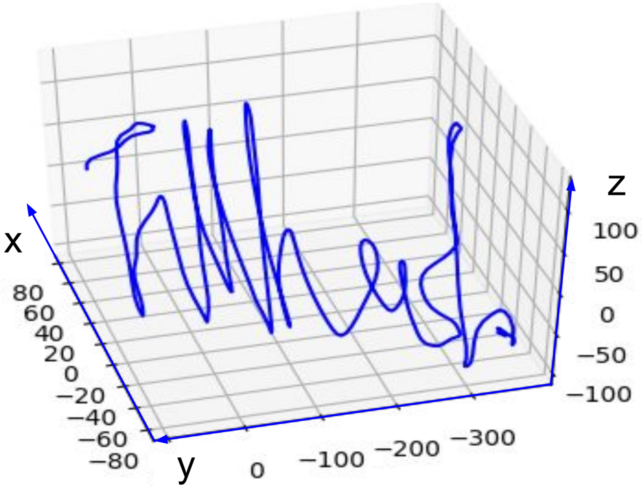
(failed to be identified as any account)



- More tolerance causes less failure of identification but more misidentification.
- Misidentification is more harmful, and hence, we favor precision than recall.

Example





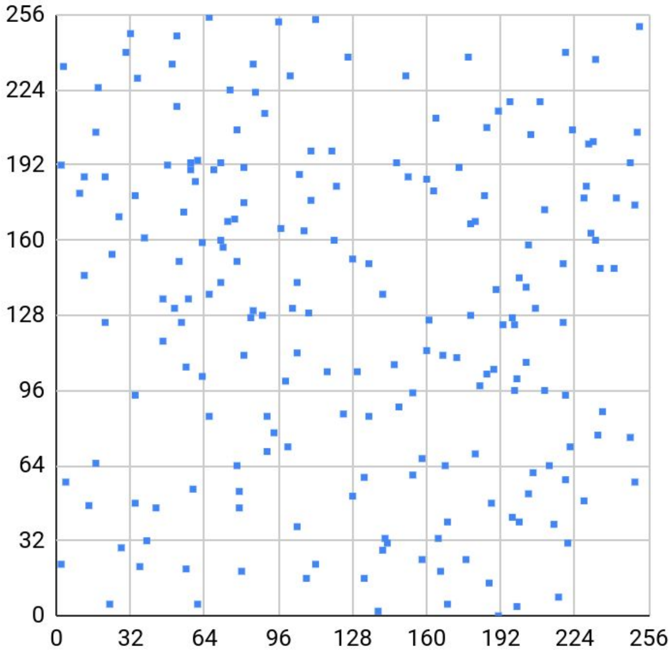
Hash code (64 bit): 11101101 11010110 00001001 11111000 11000010 00101000 01101010 00001101

Hash code (hex): ED D6 09 F8 C2 28 6A 0D

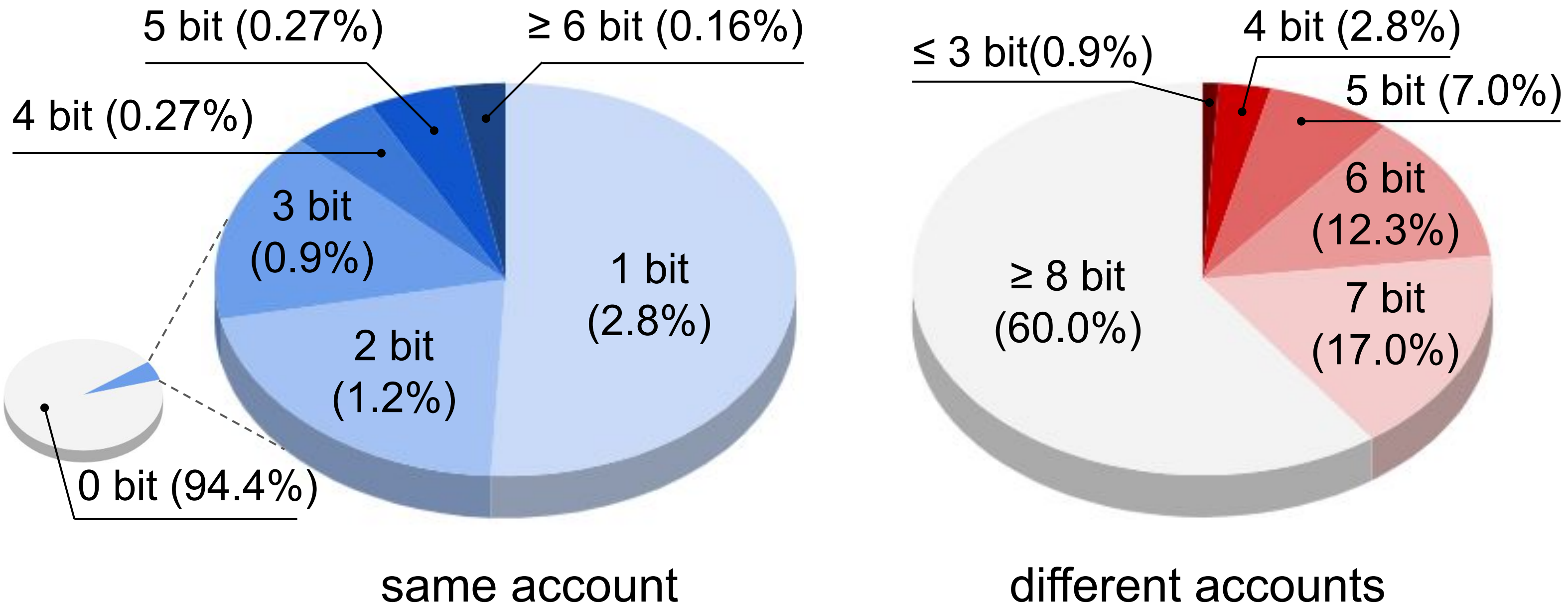
The hash code is not deterministic, i.e., it is different if the neural network is trained from scratch again.

More Examples

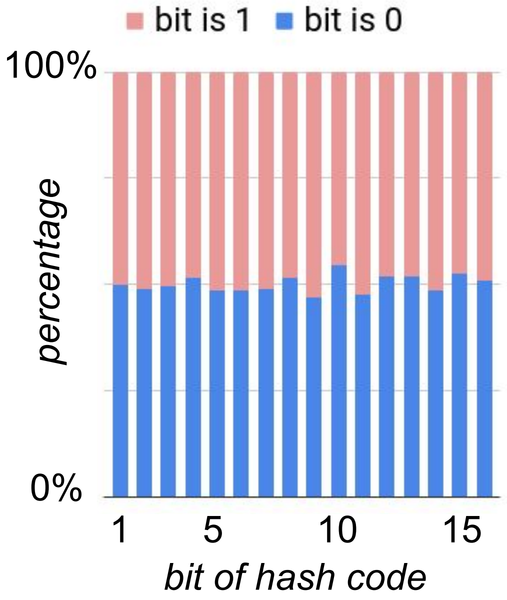
account	hash code
account A	11101101 11010110
account B	00010010 01001001
account C	11010100 11110100
account D	11010001 01001010
account E	00000110 10010111
account F	11101101 01000110
account G	00111011 11011111
account H	11111101 01001100
account I	11111101 11010110
account J	11111101 11010110
account K	11010100 10110000



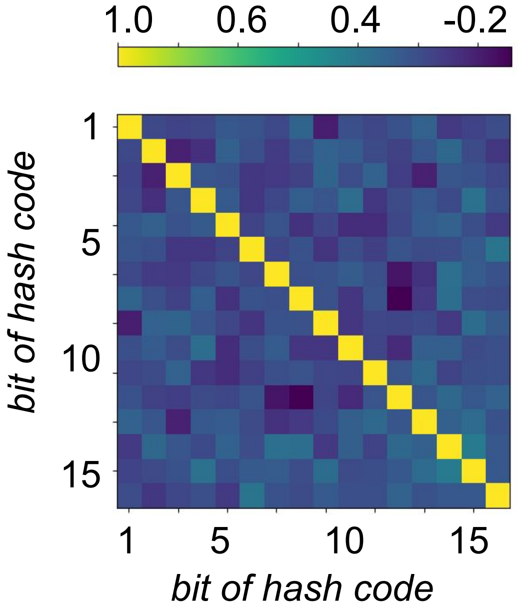
Distribution of Hash Code Fuzziness (16-bit hash)



Analysis of the Quality of Hash Code Bits (16-bit hash)



distribution of 0 and 1 in each bit



correlation of hash code bits

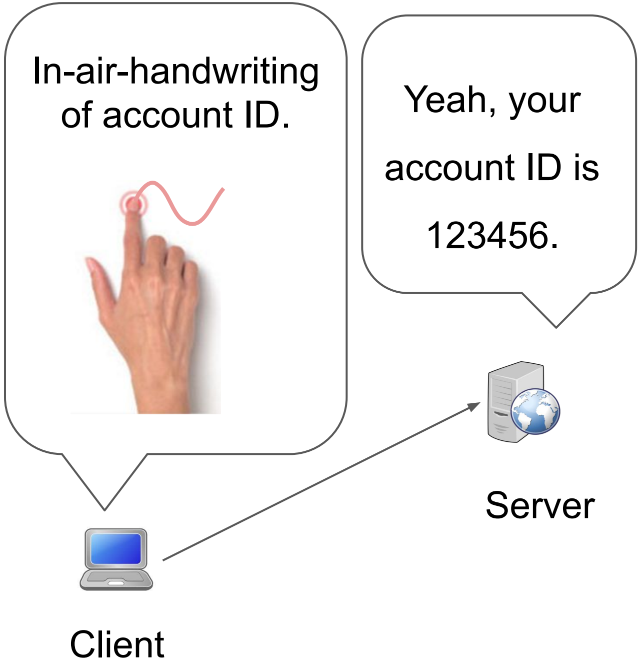
Conclusions, Limitations and Future Work

Conclusions:

- In-air-handwriting has a good amount of information to distinguish different users in a database by fuzzy hashing.
- Minor movement variation can be tolerated without much loss of discriminative capability by careful design.

Limitations and Future Work:

- Larger dataset is needed for more rigorous evaluation.
- Adding new accounts requires retraining. Can we design a deterministic feature encoding method?
- Can we generate cryptographic keys from this fuzzy hash?



Thank you!

Q & A