FEEDBACK-BASED HANDWRITING RECOGNITION FROM INERTIAL SENSOR DATA FOR WEARABLE DEVICES

Yujia Li[1*], *Kaisheng Yao*[2], *Geoffrey Zweig*[2]

[1]University of Toronto, Toronto, ON, Canada [2]Microsoft Research, Redomond, WA, USA

yujiali@cs.toronto.edu {kaisheny, gzweig}@microsoft.com

ABSTRACT

This paper presents a novel interactive method for recognizing handwritten words, using the inertial sensor data available on smart watches. The goal is to allow the user to write with a finger, and use the smart watch sensor signals to infer what the user has written. Past work has exploited the similarity of handwriting recognition to speech recognition in order to deploy HMM based methods. In contrast to speech recognition, however, in our scenario, the user can see the individual letters that are recognized on a sequential basis, and provide feedback or corrections after each letter. In this paper, we exploit this key difference to improve the input mechanism over a classical source-channel model. For a small increase in the amount of time required to input a word, we improve recognition accuracy from 59.6% to 91.4% with an implicit feedback mechanism, and to 100% with an explicit feedback mechanism.

Index Terms— Handwriting recognition; inertial sensor; wearable device; user feedback; neural network; language model.

1. INTRODUCTION

Smart watches are emerging as an important new form of consumer technology. While they can provide a new level of convenience, they also pose challenges, as the data entry methods developed for devices with larger screens are no longer applicable. In particular, the commonly used touch based text entry and image based handwriting recognition [1–5] are difficult as the watch screens are much smaller than phones and tablets. Speech recognition is an alternative way to do text entry, but it is not applicable when there is high ambient acoustic noise or when people don't want to speak in public.

In this paper, we study handwriting based text entry methods that allow a user to write with their finger while wearing a smart watch. The data we use comes from inertial sensors, namely an accelerometer and a gyroscope, which can be found in most recent mobile devices. There are several challenges with using inertial sensor data for handwriting recognition. Firstly, the hand trajectory is not available or difficult to recover reliably. Secondly, stroke information is not available - unlike touch screens where finger up and finger down are very important indicators for strokes, inertial sensors cannot discriminate actual writing sequence from other hand movements. Thirdly, people write letters and words very differently.

Fortunately, in the context of finger-writing while wearing a smart-watch, we have a key advantage - *the user can see and react to the intermediate results*. This allows us to exploit schemes where, based on what the system recognizes, the user can either implicitly or explicitly make corrections as recognition proceeds. This is not the case in the related problem of speech recognition, where a person cannot adapt their speech mid-word as the recognizer operates.

[*]Most of the work done during an internship at Microsoft Research.

Our baseline recognition system consists of two main components: a letter-recognition module, which recognizes isolated letters, and a word recognition module that infers the likeliest word given the errorful letter sequence. Our baseline for the word recognition module is a standard source-channel model, described in Section 4.1. This architecture is illustrated in Figure 1.

We enhance this basic system by leveraging the fact that we can easily display the last letter that was recognized to the user. This enables powerful feedback mechanisms. In the first, which we term *implicit feedback*, the user simply repeats the letter if it was incorrectly recognized. The system, however, does not know if this new input is because the last letter was incorrect, or if the user is simply providing the next letter. It must be inferred. In the second type of feedback, the user first taps, and then repeats the letter. Since we can reliably recognize sharp taps, the system knows the last letter was incorrect, and we call this *explicit feedback*. The feedback architecture is illustrated in Figure 2.

We implement the handwriting recognition framework on a smart watch device, and conduct experiments on real users. Both methods produce dramatic accuracy improvements over a classical source-channel decoder.

2. RELATED WORK

The letter classification part of our work is related to other inertial sensor based gesture recognition work done in the past. There are three major approaches. The first approach uses time sequence modeling such as HMMs or RNNs [6–9]. The second approach uses template matching [10–12]. The third approach is to treat the problem as a classification problem [13–18]. Some more related work on this can be found in survey papers like [19, 20].

Relatively fewer people have explored the task of recognizing gesture sequences. The work most related to ours is the 'Air Writing' system by Amma *et.al.* [21, 22]. In this system, the authors used HMM based systems to directly recognize words and use a language model to correct errors in sentences. Our work differs from theirs in the following aspects. Firstly, we treat handwriting as an interactive process with the device, and propose to use feedback from user to improve recognition. With feedback, we are able to significantly reduce word recognition error and improve writing speed, which they did not discuss. To our best knowledge, our work is the first to study using user feedback in the writing process. Secondly, we use letters

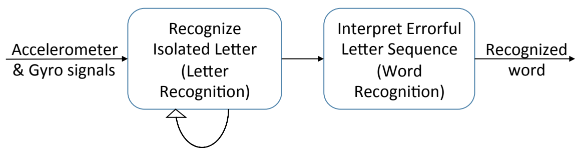
Accelerometer & Gyro signals
Recognize Isolated Letter (Letter Recognition)
Interpret Errorful Letter Sequence (Word Recognition)
Recognized word

Fig. 1. Baseline System Architecture.

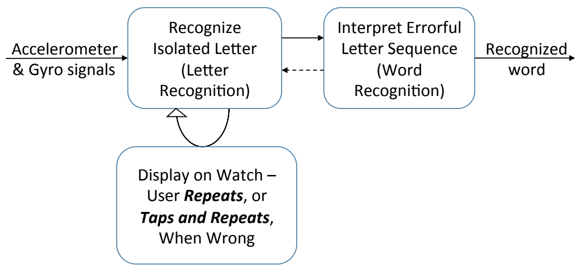
Accelerometer & Gyro signals
Recognize Isolated Letter (Letter Recognition)
Interpret Errorful Letter Sequence (Word Recognition)
Recognized word
Display on Watch – User Repeats, or Taps and Repeats, When Wrong

Fig. 2. Feedback Architecture. The dotted line indicates information that is used to adjust the letter priors.

as the basic recognition unit and use letter (not word) level language models when recognizing words, and therefore our system is not restricted to any particular dictionary.

3. LETTER RECOGNITION

The input to the letter recognition module comes from the accelerometer and gyroscope sensors mounted on a smart watch. These sensors are now standard on most mainstream wearable devices. The sensor signals are sampled at 500Hz, and each sensor sample is a 3D vector, so each frame of data is a 6 dimensional vector. The accelerometer provides Cartesian acceleration measurements, while the gyroscope provides angular velocity.

The user is asked to wear a smart watch on their writing hand, and write capital letters using their index finger on a desk surface. After each letter, the user is required to hold the hand still for a short period of time, which is used to segment writing sequences from non-writing sequences. We set a threshold on the moving variance of the signal in a window of 100 frames (0.2s) to discriminate writing sequence against non-writing sequence.

Our letter recognizer takes as input a writing sequence segment and predicts one of the 26 letter classes. As different people write letters at different speed, we first normalize all the writing sequences to a fixed length of 200 frames using linear interpolation. Then the mean of each dimension is subtracted and we integrate the signals over time once with initial velocity/angle set to 0. The 200×6 transformed sequence is concatenated into a 1200 dimensional feature vector, which is then normalized across all training sequences to have zero mean and standard deviation 1. The feature vector is fed into a deep neural network for classification.

We collected capital letter writing data from 10 users. Each user was asked to write each letter at least 20 times. For each letter and each user, part of the samples are used as validation and test sets, and the rest are used for training.

On this dataset, our DNN classifier achieves a test accuracy of 80.4%. As a comparison, a carefully tuned SVM with RBF kernel achieves a test accuracy of 77.1%. We also tried time sequence models like RNNs and LSTMs [23], which take a window of 50 frames of the transformed signals as input at each time step, and output a letter class label for each time. Test performance on this dataset for RNN and LSTM are 77.3% and 79.8% respectively. We therefore use the best DNN model for letter classification.

4. WORD RECOGNITION

Once the user stops writing, the word recognition module is used to determine which word has been written. As letter prediction errors are unavoidable, we study how to resolve the errors at word level. In this section, we first present a standard noisy-channel model, and then describe our methods for integrating user feedback.

4.1. Noisy-Channel Baseline

Consider an input sequence of $\mathbf{x} = \{x_1, ..., x_T\}$, where each x_t corresponds to the writing data of one letter, and is associated with one output y_t which is one of the 26 letters. Our task is to find the correct $\mathbf{y} = \{y_1, ..., y_T\}$ sequence that corresponds to the word the user intended to write.

The noisy-channel model [24–26] is widely used in speech recognition, which finds the optimal $\mathbf{y}^*$ that maximizes the posterior. In our case, letters are clearly segmented, and the only type of errors that need to be considered are substitutions. The probability of a particular letter sequence $\mathbf{y}$ is given by:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &\propto p(\mathbf{x}|\mathbf{y})p(\mathbf{y}) = p(\mathbf{y})\prod_{t=1}^{T} p(x_t|y_t) \\ &\propto p(\mathbf{y})\prod_{t=1}^{T} p(y_t|x_t)/p(y_t) \end{aligned} \tag{1}$$

where $p(y_t|x_t)$ is given by the DNN, $p(y_t)$ is estimated using letter frequency, and $p(\mathbf{y})$ can be computed using a letter level language model or a dictionary with word frequency. As the state space for $\mathbf{y}$ is on the order of K^T where $K = 26$, exhaustive search is prohibitive for even a small T, we use beam search to do decoding.

4.2. Interactive Word Recognition with User Feedback

As our DNN letter classifier can make predictions in real time, it is possible to show the letter predictions that have been made so far to the user during writing. The user can then use this information to correct errors, which can improve the recognition performance or reduce writing time. We study two types of feedback in this paper:

- **Explicit feedback**, where whenever the user sees an incorrect letter prediction, the user does an extra ‘tap’ action to remove the incorrect prediction from prediction history, and writes the letter again.
- **Implicit feedback**, where the user keeps writing the same letter until the letter predictor gets it right.

The explicit feedback requires the recognition of the extra ‘tap’ actions (which is chosen to be easily detectable), and the extra time for the ‘tap’s during writing. The implicit feedback does not require the ‘tap’s, but the error signal is more indirect, as the user never tells the system which letter is correct, and which is not. An example of an interaction sequence with implicit feedback is shown in Table 1.

Intended	a	a	a	i	m	m
Recognized & Displayed	d	d	**a**	**i**	n	**m**

Table 1. An input interaction with implicit feedback, where the user inputs the word "aim". The top line indicates what the user intended after seeing the last recognized letter. The bottom line shows what the system recognized and displayed.

4.2.1. *Improving Letter Recognition using Word Level Information*

As the letter-input process proceeds, it is possible to infer which letters are most likely to be input next, and this can be used to dynamically improve the performance of the letter recognition module. For example, when using explicit feedback, we know the correct letter prefix, and can use a language model to bias towards likely extensions. To combine the information from the DNN letter recognizer with the information from a language model, we propose a log-linear model for predicting true label z_t for the tth letter given x_t and the prediction history $y_{1:t-1} = \{y_1, ..., y_{t-1}\}$

$$p(z_t|x_t, y_{1:t-1}) \propto \exp\left(\log p_{\mathrm{NN}}(z_t|x_t) + \lambda \log q(z_t|y_{1:t-1})\right) \tag{2}$$

Here, p_{NN} is the DNN prediction distribution, q is a prior for z_t based on history up to the $t-1$'th letter, and λ trades off the two. We emphasize the distinction between z_t and y_t where y_t is the prediction and z_t is the intended letter which is not visible to the system.

There are many possible choices for q. For explicit feedback, we consider two alternatives: a letter unigram and a letter level language model. As recognition errors are explicitly removed from prediction history, $y_{1:t-1}$ should always be the correct context for a language model. The errors are also not predicted in the next time, which we found to be a very useful rule.

For implicit feedback, as $y_{1:t-1}$ may contain errors, a straight application of language model can be problematic. However, because of the restrictions in the writing process, the true context must be a substring of $y_{1:t-1}$. Ideally, we should have a distribution over all possible contexts and aggregate the language model prediction distribution over all of them. We observe that each correct context has a one-to-one mapping to a binary sequence $m_{1:t-1}$, and $m_i = 1$ corresponds to y_i being correct and $m_i = 0$ otherwise. We can therefore formulate q as

$$q(z_t|y_{1:t-1}) = \sum_{m_{1:t-1}} p(z_t|m_{1:t-1}, y_{1:t-1}) p(m_{1:t-1}|y_{1:t-1}) \tag{3}$$

where $p(z_t|m_{1:t-1}, y_{1:t-1})$ can be computed using language model, as $m_{1:t-1}$ and $y_{1:t-1}$ jointly determines the correct context, the second term $p(m_{1:t-1}|y_{1:t-1})$ assigns a weight to every $m_{1:t-1}$. The distribution $p(m_{1:t-1}|y_{1:t-1})$ can be updated recursively over time,

$$\begin{aligned} p(m_{1:t-1}|y_{1:t-1}) &\propto p(m_{1:t-1}, y_{t-1}|y_{1:t-2}) \\ &= p(m_{t-1}, y_{t-1}|m_{1:t-2}, y_{1:t-2}) p(m_{1:t-2}|y_{1:t-2}) \end{aligned} \tag{4}$$

For the first term $p(m_{t-1}, y_{t-1}|m_{1:t-2}, y_{1:t-2})$, we introduce the intended letter z_{t-1} and have

$$\begin{aligned} & p(m_{t-1}, y_{t-1}|m_{1:t-2}, y_{1:t-2}) \\ = & \sum_{z_{t-1}} p(m_{t-1}, y_{t-1}, z_{t-1}|m_{1:t-2}, y_{1:t-2}) \\ = & \sum_{z_{t-1}} p(m_{t-1}|y_{t-1}, z_{t-1}) p(y_{t-1}|z_{t-1}) p(z_{t-1}|m_{1:t-2}, y_{1:t-2}) \end{aligned} \tag{5}$$

where $p(m_{t-1} = 1|y_{t-1}, z_{t-1}) = 1$ when $y_{t-1} = z_{t-1}$ and 0 otherwise, $p(z_{t-1}|m_{1:t-2}, y_{1:t-2})$ is given by the language model. We estimate $p(y_{t-1}|z_{t-1})$ as $1-\epsilon$ when $y_{t-1} = z_{t-1}$ and $\frac{\epsilon}{K-1}$ when $y_{t-1} \neq z_{t-1}$, where ϵ is the class-independent error rate.

The proposed model will be referred as AggregatedLM in the experiments. In practice, we keep at most top N different $m_{1:t-1}$ candidates at each t ordered by the probability $p(m_{1:t-1}|y_{1:t-1})$ and then renormalize the distribution. We use $N = 64$ throughout all the experiments.

The AggregatedLM model can be made even better based on a simple observation. As the user is assumed to keep writing the same letter when a letter is predicted incorrectly, the model should avoid predicting the incorrect predictions made so far. We can use this observation when computing $p(z_t|m_{1:t-1}, y_{1:t-1})$. More specifically, when $m_{t-s:t-1}$ are all 0 for some s, $y_{t-s:t-1}$ are then all incorrect. We therefore set the probability of z_t taking any of these predictions to be 0 and renormalize the distribution. This improved method will be referred as AggregatedLM$^+$.

4.2.2. *Post-Writing Error Correction for Implicit Feedback*

For implicit feedback, after the user finishes writing a word, we still need to decode the correct word from the sequence $y_{1:T}$ as it may contain errors.

One baseline method is to use a word dictionary and find the closest word to $y_{1:T}$ in dictionary, in terms of edit distance. Note that there are special constraints on the writing process, which restricts the only error to be insertion error and y_T should always be correct as the user will stop after writing the last letter in a word. Ties are handled by choosing the word with the highest frequency in the corpus used to build the dictionary.

We further improve the baseline method using probabilistic models. Using w for the correct word, we find the optimal w^* that maximizes

$$p(w|y_{1:T}) \propto p(y_{1:T}|w) p(w) \tag{6}$$

where we model $p(y_{1:T}|w)$ as $\left(\frac{\epsilon}{K-1}\right)^{T-|w|}(1-\epsilon)^{|w|}$. The prior $p(w)$ can be modeled either using word frequency and a dictionary or using a letter level language model.

When using a letter language model, we only need to search through a space of 2^{T-1} candidates as w must be a substring of $y_{1:T}$ and y_T is always correct. This is a much more manageable space than K^T as for the noisy-channel model proposed in Section 4.1.

5. EXPERIMENTS

In this section we present experiments to study the effect of the different feedback types: no feedback (noisy-channel model), implicit feedback and explicit feedback. For the two feedback methods, we further study the proposed letter prediction improvements. For implicit feedback, we also study the proposed post-writing error correction methods, evaluating word prediction accuracy.

We used a 6-gram letter level backoff language model trained on the Gutenberg corpus[1]. The word dictionary and letter frequency used in the experiments are also computed using this corpus. The 'tap' recognition model is trained on a small set of 'tap' data from 3 users, which new users can easily adapt and learn to use during writing.

[1]Available at `http://www.nltk.org/nltk_data/`

	Letter Predictor	Acc.
NFB	NN	58.7
IFB	+Unigram	59.0
	+LM	62.9
	+AggregatedLM	70.2
	+AggregatedLM$^+$	**71.1**
EFB	NN	63.6
	+Unigram	68.5
	+LM	**80.6**

Table 2. Letter prediction results. 'NFB' for no feedback, 'IFB' for implicit feedback, 'EFB' for explicit feedback, and 'Acc.' for per letter accuracy (in %).

Method	Unary	EditDist	ProbEditDist	LM
Acc. (%)	78.6	88.0	88.0	**91.4**

Table 3. Comparison of error correction methods.

5.1. Analysis of Language Model Effectiveness

In this section, we perform an initial set of experiments to compare the performance of the different language modeling methods proposed in Section 4.2.1.

To achieve a fast turnaround, these experiments are done with a user simulator. The simulator is based on letters collected from 8 previously unseen users. The simulator interacts with the recognition algorithm and provides a random occurrence of the segmented letters whenever a new input is required. While this misses the nuances of live interactions, it allows for rapidly testing many feedback mechanisms, and we present the results of end-to-end human interactions in the next sections. Table 2 shows the results of the different methods.

5.2. Post-writing Error Correction Effectiveness

In this experiment we compare the different post-writing error correction methods presented in Section 4.2.2 for implicit feedback. Users were asked to write words while interacting with the system. The user simulator was not employed. For the error correction methods, we have as input the $y_{1:T}$ sequence obtained using IFB+AggregatedLM$^+$, which may contain errors, and output the estimated correct word w.

We compare the model that does not use any error correction ('Unary'), the deterministic minimum edit distance method ('EditDist'), the probabilistic model that uses a word dictionary+frequency for $p(w)$ ('ProbEditDist') and the probabilistic model that uses a letter language model for $p(w)$ ('LM').

Experiment results are reported in Table 3. On this task, we noticed that both EditDist and ProbEditDist perform equally well. They all improve word accuracy significantly from the baseline. Using a letter language model further improves word accuracy to 91.4%.

5.3. Full User Study Results

In this section, we summarize the results of both implicit and explicit feedback methods. In addition to the data we had in the previous section, we have data[2] from the same 8 users writing the same list of words twice. Once with the baseline noisy-channel model without using feedback (NFB+NN) and once with explicit feedback together with language model for letter recognition (EFB+LM).

	NFB+NN	IFB+AggregatedLM$^+$	EFB+LM
Acc.	59.6	78.6	100.0
		91.4 (with correction)	
Time/letter	2.39	2.80	2.76

Table 4. Comparison for the three feedback types on word recognition accuracy (in %) and average writing time per each correct letter (in seconds; pause, tap and error time included).

Table 4 shows the experiment results. The standard noisy-channel model achieved a word prediction accuracy of 59.6% on this task. The implicit feedback approach boosts the accuracy up to 78.6%. The post-writing error correction further improved the recognition rate to 91.4% (see Section 5.2). For explicit feedback, as all letter errors are explicitly corrected by the user, a 100% accuracy is guaranteed. We also note that both implicit and explicit feedback methods achieved very significant accuracy boost without increasing the writing time too much.

While in these experiments the explicit feedback method outperformed the implicit method, our test subjects were careful in tapping, so we had no tap recognition errors. On larger populations, this may not be the case, and implicit feedback may be preferable.

6. CONCLUSIONS

In this paper, we studied handwriting recognition using the inertial sensor data from a smart watch. We presented a recognition framework composed of DNN based letter classifier and interactive word recognition using user feedback. Experiments on real users show that our novel feedback-based word recognition methods can achieve significantly better accuracy than a baseline that does not use feedback, with small increase in writing time. A prototype system is built that can handle end-to-end processing in real time.

7. REFERENCES

[1] B. Huang, Y. Zhang, and M Kechadi, "Preprocessing techniques for online handwriting recognition. intelligent text categorization and clustering," pp. 25–45, 2009.

[2] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 8, pp. 787–808, 1990.

[3] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for improved unconstrained handwriting recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 855–868, 2009.

[4] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *NIPS*, 2009, pp. 545–552.

[2]We collected writing data for each user in one single session, the ordering of the three feedback methods in data collection were randomized.

[5] R. Plamondon and S. N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 63–84, 2000.

[6] F. Hofmann, P. Heyer, and G. Hommel, "Velocity profile based recognition of dynamic gestures with discrete Hidden Markov Models," in *Gesture and Sign Language in Human-Computer Interaction*, 1998, vol. 1371.

[7] T. Pylvanainen, "Accelerometer based gesture recognition using continuous HMMs pattern recognition and image analysis," in *Lecture Notes in Computer Science*, 2005, vol. 3522, pp. 413–430.

[8] G. Lefebvre, S. Berlemont, F. Mamalet, and C. Garcia, "BLSTM-RNN based 3D gesture classification," in *International Conference on Artificial Neural Networks*, 2013, pp. 381–388.

[9] C. Zhu and W. Sheng, "Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 41, no. 3, pp. 569–573, 2011.

[10] J. Liu, Z. Wang, L. Zhong, J. Wickramasuriya, and V. Vasudevan, "uWave: Accelerometer-based personalized gesture recognition and its applications," in *International Conference on Pervasive Computing and Communications*, 2009, pp. 1–9.

[11] A. Akl and S. Valaee, "Accelerometer-based gesture recognition via dynamic-time warping, affinity propagation, & compressive sensing," in *ICASSP*, 2010.

[12] B. Hartmann and N. Link, "Gesture recognition with inertial sensors and optimized dtw prototypes," in *Systems Man and Cybernetics (SMC), 2010 IEEE International Conference on*. IEEE, 2010, pp. 2102–2109.

[13] S. Duffner, S. Berlemont, G. Lefebvre, and C. Garcia, "3D gesture classification with convolutional neural networks," in *ICASSP*, 2014.

[14] J. Wu, G. Pan, D. Zhang, G. Qi, and S. Li, "Gesture recognition with a 3-D accelerometer," in *Ubiquitous Intelligence and Computing*, 2009, pp. 25–38.

[15] M. Hoffman, P. Varcholik, and J. LaViola, "Breaking the status quo: improving 3D gesture recognition with spatially convenient input devices," in *Virtual Reality Conference (VR)*, 2010, pp. 59–66.

[16] S. Kallio, J. Kela, and J. Mantyjarvi, "Online gesture recognition system for mobile interaction," in *IEEE International Conference on Systems, Man and Cybernetics*, 2003, pp. 2070–2076.

[17] D. Kim, H. Choi, and J. Kim, "3D space handwriting recognition with ligature model," in *Ubiquitous Computing Systems*, 2006, vol. 4239, pp. 41–56.

[18] W.-C. Bang, W. Chang, K.-H. Kang, E.-S. Choi, A. Potanin, and D.-Y. Kim, "Self-contained spatial input device for wearable computers," in *ISWC*, 2003.

[19] A. Bulling, U. Blanke, and B. Schiele, "A tutorial on human activity recognition using body-worn inertial sensors," *ACM Computing Surveys (CSUR)*, vol. 46, no. 3, pp. 33, 2014.

[20] A. Akin, S. Bosch, M. Marin-Perianu, R. Marin-Perianu, and P. Havinga, "Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey," in *International Conference on Architecture of Computing Systems (ARCS)*. VDE, 2010, pp. 1–10.

[21] C. Amma, M. Georgi, and T. Schultz, "Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors," in *ISWC*, 2012.

[22] C. Amma, D. Gehrig, and T. Schultz, "Airwriting recognition using wearable motion sensors," in *Proceedings of the 1st Augmented Human International Conference*, New York, NY, USA, 2010, AH '10, pp. 10:1–10:8, ACM.

[23] A. Graves, "Generating sequences with recurrent neural networks," Tech. Rep. arxiv.org/pdf/1308.0850v2.pdf, 2013.

[24] E. Brill and R. C. Moore, "An improved error model for noisy channel spelling correction," in *Proceedings of ACL 2000*. January 2000, Association for Computational Linguistics.

[25] L. Rabiner, "A tutorial on hidden Markov models and selected applications," *Proceedings of the IEEE*, vol. 77, pp. 257–286, 1989.

[26] G. Hinton, L. Deng, D. Yu, G. Dahl, A. r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, November 2012.