Online appendix for:
Air-Writing Recognition Using Smart-Bands

Tomer Yanay\textsuperscript{a}, Erez Shmueli\textsuperscript{a,}\textsuperscript{*}

\textsuperscript{*}Deptartment of Industrial Engineering, Tel-Aviv University
Ramat Aviv, 69978 Tel Aviv, Israel

Abstract

This is an online appendix for the paper: “Air-Writing Recognition Using Smart-Bands”. It reports additional experiments that we conducted to answer the following research questions:

1. What is the contribution of different system components (e.g., different motion signals) to the method’s performance?
2. How sensitive is our method to different writing sizes?
3. How sensitive is our method to the letter case used?
4. How robust is the user-independent method with respect to the number of train subjects used?
5. How does the user-independent method compare to other state-of-the-art methods?

Keywords: Air-writing recognition, Gesture recognition, Wearables, Dynamic-time-warping, Convolutional-neural-networks

*Corresponding author

Email addresses: yanaytomer@gmail.com (Tomer Yanay), shmueli@tau.ac.il (Erez Shmueli)
Appendix A. Contribution Analysis

In this part of the evaluation we sought to understand the contribution that different components had to the performance of our method by comparing our method to the following alternatives:

- The same method without the preprocessing phase, which enables us to understand the contribution of this phase.
- The same method using different subsets of signals, which enables us to identify the contribution of different motion signals.

First, we compared our method to a variation that did not apply the preprocessing phase. A comparison between the performance of the two methods is presented in figure A.1(left). As can be seen from the figure, applying the preprocessing phase improves mean accuracy from 85.5% to 89.2%. This difference was found to be significant (t-test, \( p < 0.001 \)). This improvement can be attributed to reducing the noise associated with writing speed, writing size and unbalanced sampling intervals (caused for example by delays in Bluetooth transmission).

![Figure A.1: Contribution of the preprocessing phase for the user-dependent method. The subfigure on the left shows recognition accuracy with and without the preprocessing phase (each point represents a single subject). The subfigure on the right presents the distribution of writing times of all samples (across all subjects and letters), where the orange dotted line marks the chosen writing time \( l \).](image)

It is important to note that throughout this paper we used a fix writing time of \( l = 2.5 \) seconds for the linear interpolation preprocessing stage. The rational behind choosing this number is given in Figure A.1(right), where it is shown that 95% of the samples had a writing time lower than 2.5.
In order to better understand the classification power of the different signals, we repeated the same evaluation process that was described above, each time using a different subset of signals. More specifically, we used the following six subsets of features: (1) the two X axis signals, (2) the two Y axis signals, (3) the two Z axis signals, (4) the three Accelerometer signals, (5) the three gyroscope signals, and (6) all six signals. Figure A.2 reports the obtained results for these six subsets of features. One can see that signals of the Y and Z axes perform better than the X axis signal. This make sense, as most of the movement in writing involves two main axes only. Moreover, the two sensor-based subsets of signals (i.e., ACC and GYR) perform similar to each other and both obtain a relatively high accuracy. This implies that for this specific task of writing letters, using just a single sensor might be sufficient. Finally, as one would expect, combining all six signals together further improves the results, yielding the best performance.

Figure A.2: Recognition accuracy for the user-dependant method when using different subsets of signals
Appendix B. Sensitivity to Writing Size

One may argue that the proposed method is extremely sensitive to the size of the writing movements. To better understand the effect of the writing size, we recruited 10 new subjects, and performed a dedicated data collection experiment. In this experiment, subjects were asked to provide 6 sets of all 26 English uppercase letters. In contrast to the main data collection experiment, we controlled the writing size of subjects by asking them to air-write within a hollow wooden frame, and in a manner that “fills the entire frame” (see Figure B.3(left)). Since the use of the frame also imposes limitations on the position of writing (and not only on its size), we allowed subjects to adjust the height of the frame to their convenience, by using a height-adjustable tripod. The above procedure was repeated 3 times with different frame sizes: large, medium and small, corresponding to the standard paper sizes A5, A6 and A7 respectively (see Figure B.3(right)), resulting in 4,680 (10 subjects * 3 sizes * 6 sets * 26 letters) samples.

Due to the relatively small number of sets collected for each subject, we included only a single set in the user’s reference set (i.e., $k = 1$), and the remaining five sets were used as the test set.

A comparison of the method’s performance for the three sizes is presented in Figure B.4. The left subfigure presents the results where the same writing size is used for the reference set and the test set. As can be seen from the subfigure as well as from a statistical significance test that we performed, we did not find a significant difference between the means of the three tested writing sizes (One Way ANOVA, $p > 0.7$). This result was surprising to us, as we hypothesized that
increasing the letter size would allow the smart-band to capture minor differences between writing movements, thereby increasing classification performance.

![Box plots showing recognition accuracy for different frame sizes.](image)

Figure B.4: Recognition accuracy for the user-dependent method with $k = 1$ for the different writing sizes - when the reference set writing size is identical to that of the test set writing size (left), and when it does not (right).

The result above led us to a new hypothesis that our method is potentially robust to different writing sizes. To test this hypothesis we performed a new comparison, where the reference set of a subject was selected from a single writing size, and all other sets of the subject (from all three writing sizes) were used as a test set, Figure B.4(right) presents the results of this analysis for the three writing sizes. Again, as can be seen from the subfigure, as well as from a statistical significance test that we performed, we did not find a significant difference between the means of the three tested writing sizes (One Way ANOVA, $p > 0.9$). Moreover, the performance of the three writing sizes in this comparison is very similar to those presented in the first comparison. This implies that a user can provide a single reference set in whatever writing size he prefers and later use the same system to recognize letter at different writing sizes. This is encouraging as even the same person might need to write in different sizes, depending on the situation at hand.

**Note:** The performance reported in this subsection is slightly higher than the one described in the main paper for $k = 1$. While this difference can be attributed to the different population tested and the number of test sets examined, we believe that this improvement was a result of using the wooden frame. The wooden frame imposes clear borders to the hand movement while writing, what we think gave better spatial orientation to subjects, and might have prevented the disorientation phenomenon described in the paper.
Appendix C. Sensitivity to Letter Case

In order to understand how the proposed method performs with respect to letter case (uppercase vs. lowercase), we performed an additional experiment, to which we recruited again 10 new subjects. In this experiment, subjects were asked to provide 6 sets of all 26 English uppercase letter, as well as 6 sets of all 26 English lowercase letter, resulting in 3,120 (10 subject * 2 letter case * 6 sets * 26 letters) samples.

Similarly to the previous section, in order to test the sensitivity of our method to letter case, we performed a complementary data collection experiment to which we recruited again 10 new subjects. In this experiment, subjects were asked to provide 6 sets of all 26 English uppercase letter, as well as 6 sets of all 26 English lowercase letter, resulting in 3,120 (10 subject * 2 letter case * 6 sets * 26 letters) samples.

Again, due to the relatively small number of sets collected for each subject, we included only a single set in the subject’s reference set (i.e., \(k = 1\)), and the remaining five sets were used as part of the test set.

A comparison of the method’s performance for lowercase and uppercase letters is presented in Figure C.5. Surprisingly, we find that lowercase letters are somewhat easier to recognize (mean accuracy = 80%) than uppercase letters (mean accuracy = 75.6%), and the variance in the uppercase results is also much higher. Performing a t-test between the two distributions yielded a p-value slightly lower than 0.1, which we may consider significant taking into account the relatively small number of subjects. We believe that this difference can be attributed to two factors: (1) lowercase letters are much more common in everyday writing, so people are more accustomed to it, and (2) uppercase writing involves a relatively large number of pen lifts, thereby adding a layer of noise.
To conclude, Table C.1 presents the five pairs of lowercase letters with the highest misclassification error. For example, over 12% the total error comes from misclassifying the letter h as n and vice versa. Clearly, the pairs of lowercase letters being misclassified are entirely different than the pairs of uppercase letters being misclassified.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Letters pair</th>
<th>% of total error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>h, n</td>
<td>12.6%</td>
</tr>
<tr>
<td>2</td>
<td>b, p</td>
<td>6.1%</td>
</tr>
<tr>
<td>3</td>
<td>g, y</td>
<td>5.0%</td>
</tr>
<tr>
<td>4</td>
<td>a, d</td>
<td>3.8%</td>
</tr>
<tr>
<td>5</td>
<td>a, q</td>
<td>3.8%</td>
</tr>
</tbody>
</table>

Table C.1: Recognition accuracy for the user-dependent method with $k = 1$ on lowercase letters - the 5 most confusing letters pairs
Appendix D. Number of Training Subjects

In this subsection, we sought to test the robustness of the suggested system with respect to the number of training subjects. Recall that each training subject was associated with 390 training samples (26 letters * 15 sets). To this end, we trained our model using a varying quantity of subjects for training, ranging between 10 to 54. For each of these quantities, we repeated the analysis 55 times using a leave-one-person-out procedure, as described in the main paper, with the only exception that the relevant quantity of subjects was randomly selected from the set of 54 training subjects.

The average accuracy of the user-independent method as a function of the number of training subjects is shown in Figure D.6(left). As expected, the general trend is an increase in performance with more training subjects. Interestingly, our model provides reasonable performance for 20 training subjects (mean accuracy of 76.8%) and continues to improve with a diminishing effect after.

Figure D.6(right) presents the training runtime as a function of the number of training subjects. As can be seen from the figure, the model’s training time increases linearly with the number of training subjects. However, even for 54 subjects, the model’s training time does not exceed 7 minutes. It is also important to note that the time required for classifying a single sample during the operation phase is several orders of magnitudes lower than that required for the personalized model (0.001 second vs. 1 second respectively).

Figure D.6: Recognition accuracy (left) and training time (right) for the user-independent method for different amounts of training subjects
Appendix E. Comparison to Existing Methods

In this section, we compare the proposed CNN-based method to four other benchmark methods, which were described in the literature, and can be adjusted relatively easily to the task of air-writing recognition using smart bands:

- A features-based user-independent method, which was described in [6, 10], and was used in the context of writing on a surface using smart bands.

- An HMM-based user-independent method, which was described in [1, 11, 2], and was used in the contexts of air-writing using a holdable device (i.e., a Wii-remote) or a dedicated wearable device (i.e., a wearable glove equipped with motion sensors).

- An LSTM-based (combined with Fisher criterion) user-independent method, which was described in [5]. This method was used to distinguish between 12 different air-written characters (six upper case letters and six digits) captured by a mobile phone held by the user.

- A DTW-based user-dependent method, which was described in [7]. This method was used to recognize a gesture from a predefined set of eight gestures, captured by a Wii remote held by the user. In contrast to the DTW-based user-dependent method proposed in this paper, their method applied a different preprocessing procedure which was based on moving average and signal discretization.

Appendix E.1. A Features-Based User-Independent Method

The features-based method was implemented by us in Python 3.6. We used the same set of features as reported in [6, 10] (a detailed description of these features is provided in [8]). For the filtering task, we used the “SciPy” library [4]. As the classification algorithm we chose a Neural Network, which was reported in [6] to be the best performing method, and used the implementation of “scikit-learn” [9] with the parameters reported in [6].

The two methods (the benchmark features-based method and our CNN-based method) were evaluated using a leave-one-person-out procedure and without autocorrect phase. The classification performance obtained for the two methods is reported in Figure E.7. As can be seen, the CNN-based method outperform the benchmark method with a remarkable gap (83.2% vs.12.1% respectively).
One explanation for the inferiority of the features-based method can be the transition from the setting of writing on a surface to that of air-writing. We conjecture that in the case of air-writing, the used features are unable to capture the variance in the projection of the motion data on the smart-band axes. Therefore, additional effort should be invested in extracting a better set of features.

To complete the picture, Figure E.8 presents the training and classification runtimes for the two methods. Figure E.8(left) presents the runtime in seconds for training over 21,060 samples (all samples of 54 subjects). As can be seen from the figure, the CNN-based method is trained significantly slower than the feature-based method (461 seconds vs. 4 seconds respectively). However, it is important to mention that the training phase is executed only once, and therefore it does not affect the operation phase of the system. Figure E.8(right) presents the runtime in milliseconds for classifying a single sample. (In practice, the runtime was calculated for classifying 1000 samples and was then divided by 1000.) Again, as can be seen from the figure, the CNN-based method runs than the feature-based method (12 vs. 0.01 milliseconds respectively). Nevertheless, a classification runtime of 12 milliseconds can be considered reasonable given the gained improvement in classification accuracy.
Figure E.8: A comparison to the features-based method - training runtime (left) and classification runtime (right)

Appendix E.2. An HMM-Based User-Independent Method

The HMM-based method was implemented by us in Python 3.6. We built an HMM model for each letter and fitted the model to our data using the python library “hmmlearn” [3]. As described in [11], before training, we performed Z-score normalization and moving average smoothing. For the model architecture, we used a Gaussian Mixture Model (GMM) with 5 gaussians as the emission function for each state. A different number of states was used for each letter, where we first split the letters into five groups according to their average writing length, and then applied a grid search to find the best number of states for each group (see Table E.2).

<table>
<thead>
<tr>
<th>Group #</th>
<th>Letters in group</th>
<th># of states examined</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C, L, U, V</td>
<td>4, 5, 6</td>
</tr>
<tr>
<td>2</td>
<td>N, O, S, X, Z</td>
<td>5, 6, 7</td>
</tr>
<tr>
<td>3</td>
<td>J, M, P, T, W, Y</td>
<td>6, 7, 8</td>
</tr>
<tr>
<td>4</td>
<td>B, D, G, H, I, K, Q</td>
<td>7, 8, 9</td>
</tr>
<tr>
<td>5</td>
<td>A, E, F, R</td>
<td>8, 9, 10</td>
</tr>
</tbody>
</table>

Table E.2: Groups of letters according to their average writing length

The two methods (the benchmark HMM-based method and our CNN-based method) were evaluated using a leave-one-person-out procedure and without autocorrect phase. The classification performance obtained for the two methods is reported in Figure E.9. As can be seen, the CNN-based
method outperform the benchmark method with a remarkable gap (83.2% vs. 15.1% respectively).

![Box plot comparing CNN and HMM methods](image)

**Figure E.9:** A comparison to the HMM-based method - recognition accuracy

Interestingly, a previous study that used an HMM-based method and investigated air-writing using a wearable glove, reported a much higher classification performance (81.9% accuracy). Our setting, however, differs in three major aspects: (1) in their setting, subjects were asked to write letters in a specific way, by indicating the order of movements, (2) the glove is worn closer to the body part that does the actual writing (the palm vs. the wrist in the case of a smart-band), and (3) the glove offers a significantly faster sampling frequency of 820 Hz in comparison to the 62 Hz offered by the Microsoft Band 2 used in our study.

To complete the picture, Figure E.10 presents the training and classification runtimes for the two methods. Figure E.10(left) presents the runtime in seconds for training over 21,060 samples (all samples of 54 subjects). As can be seen from the figure, the CNN-based method is trained significantly faster than the HMM-based method (461 vs. 882 seconds respectively). Figure E.10(right) presents the runtime in milliseconds for classifying a single sample. (In practice, the runtime was calculated for classifying 1000 samples and was then divided by 1000.) Again, as can be seen from the figure, the CNN-based method runs significantly faster than the HMM-based method (12 vs. 41 milliseconds respectively).
Appendix E.3. An LSTM-Based User-Independent Method

For the LSTM-based method, we used the original code provided by the authors of [5] for the F-BiLSTM variant. The neural network was trained over 100 epochs with batch size of 200 samples.

In this subsection, we used a slightly different evaluation setting than the one described above, due to the extremely slow training time of the LSTM-based method (demonstrated below). Specifically, we randomly selected 30% of the subjects as a test set (6,240 samples) and the remaining 70% of the subjects were used for training (15,210 samples).

As can be seen in Figure E.11 (left) our method performed slightly better than the LSTM-based method (87% vs 82.9%) in terms of recognition accuracy. However, as can be seen in Figure E.11 (right) the CNN method’s training time was significantly faster than that of the LSTM-based method (458 vs. 21867 seconds respectively). In particular, while the running times of a single epoch of the two methods were not dramatically different (78 vs. 218 seconds on average respectively), the CNN-based method converged after 6 epochs whereas the LSTM-based method converged after 100 epochs.
Figure E.11: A comparison to the LSTM-based method - recognition accuracy (left) and training runtime (right)

Appendix E.4. A DTW-Based User-Independent Method

The DTW-based method described in [7] was implemented by us in Python 3.6, paying special attention to the details provided in that paper. In particular, their method used a single reference set of gestures (i.e. \( k = 1 \)). Moreover, the discretization preprocessing method used by them was tailored to the Accelerometer sensor, and therefore only features based on the Accelerometer signal were used\(^1\).

The DTW-based method of [7] was compared both to our CNN-based method and to our DTW-based method (due to their resemblance). Both of the DTW-based methods (that of [7] and ours), were evaluated by selecting a single set of letters as the reference set for each user (i.e., \( k = 1 \)). The remaining sets of letters were used as the test set for this user. Finally recognition accuracy was calculated for each user and averaged over all users. As for the CNN-based method, we performed a leave-one-person-out procedure as done in the previous subsections\(^2\).

As can be seen in Figure E.12 our CNN-based method significantly outperformed the DTW-based method of [7] (83.2% vs. 19.7% average accuracy respectively). Interestingly, the DTW-based method of [7] also performed poorly compared to our DTW-based method (19.7% vs. 80.6% average accuracy respectively). One explanation for the inferiority of the the DTW-based method of [7] is the discretization preprocessing method originally designed for the setting of holdable devices and

---

\(^1\) Adjusting the discretization method to other sensors (e.g. Gyroscope), was not trivial and was therefore avoided.

\(^2\) The test set for each user in the case of the CNN-based method contained one additional reference set of letters - the one used as a reference set in the DTW-based methods.
a different set of gestures in mind.

Figure E.12: A comparison to uWave - recognition accuracy
References


