Gesture Keyboard
USING MACHINE LEARNING

Jonas Stendahl
stendah@kth.se

Johan Arnör
jarnor@kth.se

Examensarbete inom Datalogi, grundnivå DD143X
KTH – Computer Science and Communication
Handledare: Anders Askenfelt
# Table of Contents

Abstract ............................................................................................................................................. 2
Sammanfattning .................................................................................................................................. 2
A. Introduction .................................................................................................................................... 3
B. Problem Statement .......................................................................................................................... 3
C. Background ..................................................................................................................................... 3
  C.1. History of Mobile Input ............................................................................................................. 3
  C.2. Gesture Keyboards .................................................................................................................... 4
  C.3. Technical Approach .................................................................................................................. 5
D. Method ............................................................................................................................................ 6
  D.1. Implementation of Keyboard ..................................................................................................... 6
  D.2. Multilayer Perceptron ............................................................................................................... 7
  D.3. Input Data .................................................................................................................................. 8
    Sequence ......................................................................................................................................... 8
    Id .................................................................................................................................................... 9
    Data Sets ....................................................................................................................................... 10
  D.4. Dictionary ............................................................................................................................... 10
E. Results ............................................................................................................................................ 11
  E.1. Word Categories ....................................................................................................................... 11
  E.2. Dictionary Size .......................................................................................................................... 13
F. Discussion ....................................................................................................................................... 17
G. Conclusion ....................................................................................................................................... 19
H. Bibliography ................................................................................................................................... 20
Abstract
The market for mobile devices is expanding rapidly. Input of text is a large part of using a mobile device and an input method that is convenient and fast is therefore very interesting. Gesture keyboards allow the user to input text by dragging a finger over the letters in the desired word. This study investigates if enhancements of gesture keyboards can be accomplished using machine learning. A gesture keyboard was developed based on an algorithm which used a Multilayer Perceptron with backpropagation and evaluated. The results indicate that the evaluated implementation is not an optimal solution to the problem of recognizing swiped words.

Sammanfattning
A. Introduction
Gesture keyboard is a method of inputting text to a device without taps or clicks on each letter. The general idea is that a user would just press the mouse button and hold it down while dragging the pointer across the letters in the desired word before releasing the mouse button to complete the word. This can be implemented on touch devices as well where the mouse is simply replaced by the touch tracer. The keyboard should be able to determine which word was swiped\(^1\) by the path of the input and possibly other variables such as speed and change of direction.

With the rapidly growing market of mobile devices and devices that rely on touch input this input method is gathering attention from all around. For some users such devices can benefit substantially from gesture input. It makes one-handed use of a phone much easier and is implemented in many modern mobile devices. Swype is a well-known company which created a third party gesture keyboard and Google introduced this type of input as an option on all android devices as late as last year. This was a response to the rapidly growing interest in third party applications that offered the gesture experience. Mobile text input is intensely researched and almost every human-computer interaction conference since the 1990s contain research on the topic. (Zhai & Kristensson, 2012).

B. Problem Statement
This study examines how machine learning methods can be used to create a gesture keyboard with the purpose of finding approaches that could enhance future gesture keyboards.

The goal was to develop and implement an algorithm based on machine learning methods and validate if it results in a viable gesture input keyboard.

In order to test whether or not this goal has been reached the algorithm was evaluated from three perspectives:

- Examining the performance with dictionaries of different sizes.
- Examining the performance with dictionaries of different categories of words.
- Examining the usability regarding speed and accuracy.

In order to effectively analyze the potential of machine learning in the context of gesture keyboards it was necessary to limit the scope of the study. Different machine learning approaches will been considered but only one will be selected for implementation. Moreover; a limited dictionary is chosen with words testing different aspects of input in order to analyze performance on multiple levels.

C. Background
C.1. History of Mobile Input
Text input on mobile devices has taken many shapes since mobile devices became important. Companies have tried physical keyboards and the best known company in this area is

\(^1\)The word *swipe* will be used to describe the action of making a gesture on the keyboard.
Blackberry (former RIM). Their mobile phones are iconic for their typical miniaturized physical keyboards which made them popular among business users. However the physical keyboard layout takes much of the area of the handset which could otherwise be used by the screen. This resulted in different companies making physical keyboards that slides beneath the phone when not used. Both these designs appeal to a certain group of people but they both suffer from the fact that the keyboard takes up space which require the device to be bigger or the other features to be smaller.

Onscreen keyboards first appeared on devices that were accompanied by a stylus which was used to tap on the keyboard. This solved the space problem but it required extra hardware. The first onscreen keyboard that could be used conveniently with fingers was introduced together with the iPhone (Zhai & Kristensson, 2012). This input method is now standard on any smartphone. The soft keyboard in its original form aimed to be as fast as possible and in the same way convenient to user, however, one handed use has proved difficult as screen size increased. Gesture keyboards started to evolve with the most notable solution being Swype which released a beta of their keyboard in 2009 (Swype, 2014).

C.2. Gesture Keyboards

At the core of gesture keyboards is the word recognition. The idea is that a user swipes a finger starting in the position of the first letter in a word and then proceeds to swipe across all letters that should be included (see Fig. 1). Upon reaching the last letter the gesture is completed by lifting the finger. The gesture has to be traced and analyzed in order to come up with the intended word. A common approach has been to identify the most probable word by finding the probability for words associated with a certain gesture. The total probability of a word is ideally a combination of two probabilities.

![Swype keyboard illustration](http://eurodroid.com/edpics3/swype-google-play-1.png)

**Fig. 1. Illustration of a gesture performed on Swype’s keyboard. The suggestions are the result of probabilities calculated for the words in their dictionary.**

The first, \( P(G|W) \), is the *conditional probability that the gesture matches a particular word*. A probability is calculated from the gesture using one or more of a variety of methods. (Zhai & Kristensson, 2012) (Bi, Chelba, Ouyang, Partridge, & Zhai, 2012) These methods can in turn be

---

2 Source: http://eurodroid.com/edpics3/swype-google-play-1.png
based on a number of different parameters, for example an analysis of the letters traversed can be used or analysis of the actual shape of the gesture.

The second, \( P(W) \), is the probability given earlier predictions made by the keyboard. It can also be based on other statistics regarding typing available to the keyboard.

The product of these results is a probability which can be used to give the best prediction what word the gesture belongs to (Zhai & Kristensson, 2012) (Bi, Chelba, Ouyang, Partridge, & Zhai, 2012).

\[
W' = P(G|W)P(W)
\]

C.3. Technical Approach

Machine learning is basically making computers learn from a set of input data; *Machine learning is about making computers modify or adapt their actions so that these actions get more accurate, where accuracy is measured by how well the chosen actions reflect the correct ones* (Marsland, 2009).

With the definition in place, one needs to examine if machine learning is a feasible approach to the gesture keyboard problem. To apply machine learning to a problem, three criteria must be fulfilled; a pattern in the data must exist, the problem cannot be expressed mathematically, and input data exists (Abu-Mostafa, 2012). With the gesture keyboard problem, a pattern clearly exists since a specific word will be swiped with highly correlated gestures. It is also hard to express it mathematically; one cannot simply write a formula that maps an arbitrary gesture to a word. The last criteria can be fulfilled as well by simply swiping a sequence and associating it with a word. This approach is feasible in a small study but with larger dictionaries this process should be automated.

How can machine learning be applied to make a computer recognize a keyboard gesture? Since the field of machine learning is very broad, this question has many answers. Three possible approaches are to use a Self-Organizing Map (SOM), a Multilayer Perceptron (MLP) or a Support Vector Machine (SVM). The first two are based on a sub-concept of machine learning, neural networks. Neural networks try to mimic the brain's neurons by translating biological features to mathematical concepts (Marsland, 2009, ss. 11-15). The SOM uses unsupervised learning to adjust to the data (Marsland, 2009, ss. 207-215). This means that the network is learning without a "correct" answer. This would be a feasible approach since unsupervised learning has previously been successfully used to classify gestures (Perez Utrero, Martinez Cobo, & Aguilar, 2000). Even though unsupervised learning is possible, the gesture keyboard problem suits a supervised learning model better (MLP or SVM). The advantage of MLP is that it supports "online learning" (Hermann, 2014). This means that a word can be added without the need to retrain the network with all the data, making it more efficient on very large dictionaries. This, however, will not have any impact on this project since the datasets will be relatively small. A SVM has previously been used successfully to classify gestures. In that case it was used to classify letters from the American Sign Language (Mapari & Kharat, 2012).
Although a SVM would be able to solve the problem, a MLP will be used since it might have an advantage when working with large datasets. Furthermore, MLPs have not been used in the same extent when classifying gestures, which makes it an interesting area to study.

D. Method

D.1. Implementation of Keyboard

To be able to conduct relevant empirical research a gesture keyboard is needed. Since the primary goal is to measure how well the keyboard recognizes gestures for single words the implementation only needs to handle input of one word at a time. More importantly, however, is the ability to collect data needed when training the neural network. These requirements suit an implementation with two modes; one for providing data and one for typing.

The fundamentals are the same for both modes. The keyboard starts to trace the coordinates of the mouse pointer when the mouse is pressed and stops the trace when it is released. The coordinates are then translated into numbers ranging from 1 to 29 (one number for each letter in the Swedish alphabet). Each letter occupies a certain coordinate space on the keyboard (see Fig. 2). A sequence is captured when dragging the mouse. Every time the mouse cross over a new letter that letter is recorded and translated into the corresponding number. This results in a sequence representing the order in which the letters were traversed. The sequence is a part of the input data to the neural network.

![Fig. 2. The keyboard layout implemented.](image)

In order for the learning mode to work the sequence needs to be accompanied by the word that the user intended to swipe, as well as an identifier for that word. The capture of this information is implemented by prompting the user to write the intended word as soon as the mouse is released when the keyboard is in learning mode. The word typed is then associated with an identifier which is a binary number. Each unique word has an individual identifier and if the same word is trained twice or more times the identifier will be the same for all trained instances of the word. More details about the data can be found is Section D.3.

In typing mode no extra data is required. However, to be able to test the accuracy of the network, the predicted word is displayed when the gesture is completed.
D.2. Multilayer Perceptron

The MLP neural network consists of three layers; an input layer, a hidden layer and an output layer. Between each layers there are weights that connect one node from the previous layer to all the nodes in the next layers. During the learning phase data is fed into the input layer, which is then traversed through the network to generate an output vector. The output is then compared to the target vector and the weights are updated according to this difference.

Since the task is to identify distinct words, the problem is regarded as a classification problem where each word corresponds to a class. This requires the output layer to be the same size as the number of words the network is trained on. Since one node in the output layer corresponds to a specific word, the numbers can be interpreted as a probability that the inputted data would match the corresponding word. A value close to 1 means *high probability* and a value close to 0 means *low probability*. To force the output to match a target, the highest value is set to 1 and the rest to 0.

The input layer must match the input data and 40 input nodes (see Section D.3) are therefore needed. The number of hidden nodes is determined by testing since too few or too many will affect the performance of the network. A pilot test showed that 10 hidden nodes resulted in a satisfying balance between accuracy and training time.

Before training is done, it is important to shuffle the input data to avoid spurious results. Since the training is performed several times in the same order, the last word recorded will be favored if all input data are placed in recording sequence (Hermann, 2014).

To implement this network MATLAB’s Neural Network Toolbox was used since it is a quick and robust way to implement several types of neural networks. Before the training the available data was split up randomly into 3 sets; (1) a *training set* which is used to train the network, (2) a *validation set* which is used to stop training when the performance is at its best, and (3) a *test set* to measure the accuracy of the network. The division was 70/15/15% for training/validation/test. Fig. 3 shows how the mean squared error decreases when the network adjusts to input data during training. As mentioned, training stops when the error of the validation set is minimized.
The mean squared error of the different data sets. Training is stopped when the error of the validation set is minimized (circle). During one epoch each data sample is presented to the algorithm once.

The training is performed using the Levenberg-Marquardt backpropagation algorithm. This is often the fastest backpropagation algorithm in MATLAB’s neural network toolbox, and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms (MathWorks, 2014).

When the training is done a confusion value for the test set is calculated. This value measures the fraction of misclassified sequences. To get a good sense of the general performance of the network, the training was carried out around 30-40 times.

Matlabcontrol, a third party Java API was used to connect the Java keyboard to MATLAB.

D.3. Input Data

The input data that is generated by the keyboard and passed to the network consists of three main parts, (1) a sequence, (2) an id and (3) a word corresponding to the sequence in the dictionary (see Section D.4).

Sequence

The sequence is the string of numbers that represent the gesture in the data sent to the network. The length is chosen based on the length of the words in the dictionary. The sequence is captured by the keyboard and extended with zeroes to a total length of 40 characters which is sufficient for the words used in this study. To support longer words the length of the entire
sequence would have to be extended (See Section D.4 for more information about chosen words). The following sequences are examples of how a translation to a sequence is completed.

**Word to write:** Hej  
**Recorded letter sequence:** h g t r e r t g h j  
**Resulting data sequence:** 17 16 5 4 3 4 5 16 17 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**Word to write:** Hej  
**Recorded letter sequence:** h g f r e r t y j  
**Resulting data sequence:** 17 16 15 4 3 4 5 6 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

**Word to write:** Abelsk  
**Recorded letter sequence:** a s d f g c v b v g f d r e r t h j k l k j h g f d s d f g h j k  
**Resulting data sequence:** 12 13 14 15 16 25 26 27 26 17 16 15 4 3 4 5 17 18 19 20 19 18 17 16 15 14 13 14 15 16 17 18 19 0 0 0 0 0 0

**Id**

The id is individual for each unique word but not for each sequence. The id associate different sequences to the correct word. The length of the id is the same as the number of words in the dictionary with a 1 at any of the positions and 0 on the remaining positions. The id for the words in a dictionary of 75 words would be:

- Hej: 1 0 2 0 3 0 4... 0 75  
- Hej: 1 0 2 0 3 0 4... 0 75  
- Abelsk: 0 1 0 2 0 3 0 4... 0 75

**Example of input data sent to the network:**

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Id</th>
<th>Dictionary word</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 16 5 4 3 4 5 16 17 18 0... 0 40</td>
<td>1 0 2 0 3 0 4... 0 75</td>
<td>Hej</td>
</tr>
<tr>
<td>17 16 15 4 3 4 5 6 18 0 0 0... 0 40</td>
<td>1 0 2 0 3 0 4... 0 75</td>
<td>Hej</td>
</tr>
<tr>
<td>12 13 14 15 16 25 26 27 26 17 16 15 4 3 4 5 17 18 19 20 19 18 17 16 15 14 13 14 15 16 17 18 19 0 0 0 0 0 0 0 0 0 0</td>
<td>0 1 0 2 0 3 0 4... 0 75</td>
<td>Abelsk</td>
</tr>
</tbody>
</table>

*Table 1. Representation of the input data from the keyboard to the network. The leftmost column shows the captured sequence of the gesture. The middle column shows the id of the word. The rightmost column shows the dictionary word.*
Data Sets
The data for the network was generated by one person who manually swiped each word in the dictionary 5 times. This means that a dictionary of 8 words generates 8*5 sequences of data. The data was then split according to Table 2 and supplied to the learning algorithm in MATLAB.

<table>
<thead>
<tr>
<th>Dictionary size</th>
<th>No. of sequences</th>
<th>Training set size</th>
<th>Validation Set Size</th>
<th>Test Set Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>8 words</td>
<td>40</td>
<td>28</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>12 words</td>
<td>60</td>
<td>42</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>20 words</td>
<td>100</td>
<td>70</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>40 words</td>
<td>200</td>
<td>140</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>75 words</td>
<td>375</td>
<td>263</td>
<td>56</td>
<td>56</td>
</tr>
</tbody>
</table>

Table 2. Illustration of how the data is split according to the method described in section D.2.

D.4. Dictionary
The keyboard used a Swedish layout with å, ä & ö and the dictionary contains Swedish and English words.

Word length, number of words and type of words are key points considered when creating the network. The length of each data input is important since a longer word requires more data for the training to reach maximum potential. The amount of input stands in relation to the number of weights of the network. This means that an increase in the number of weights requires a greater input data volume (Hermann, 2014). The amount of words is an important factor as well since more words will make it harder to identify differences between words. Changing this parameter can test the resilience of the network. The category of words is important in order to test if the network is able to distinguish words that have parts in common. Three categories of words were defined and tested: varied words, similar words and joined words.

The category of varied words is defined as words where the corresponding gestures have little in common, similar words are defined as words that require a similar swipe to produce, and joined words are words that contain other words. Examples of each category are given in Table 3.

<table>
<thead>
<tr>
<th>Varied</th>
<th>Similar</th>
<th>Joined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abelsk</td>
<td>Helium</td>
<td>Blad</td>
</tr>
<tr>
<td>Bilist</td>
<td>Helmet</td>
<td>Ekblad</td>
</tr>
<tr>
<td>Egoist</td>
<td>Helped</td>
<td>Ek</td>
</tr>
<tr>
<td>Fabrik</td>
<td>Helper</td>
<td>Hus</td>
</tr>
<tr>
<td>Helgon</td>
<td>Health</td>
<td>Husfru</td>
</tr>
<tr>
<td>Indien</td>
<td>Hereby</td>
<td>Bladlöss</td>
</tr>
<tr>
<td>Kobolt</td>
<td>Height</td>
<td>Löss</td>
</tr>
<tr>
<td>Jazz</td>
<td>Hectic</td>
<td>Bokblad</td>
</tr>
</tbody>
</table>

Table 3. The dictionaries used to test different categories of words. Each dictionary contains 8 words.
E. Results
Two aspects of the neural network were tested. The size of the dictionary and the category of words in the dictionary.

E.1. Word Categories
Figs. 4 - 6 show the percentage of misclassified words for dictionaries of the same size but with different categories of words. Ten hidden nodes were used. For more details about categories see section D.4.

For varied words the network classified all words correctly in 32 of 38 test sessions (see Fig. 4). For all except one test no more than one word was misclassified.
For joined words the network classified all words correctly in 36 of 38 test sessions (see Fig. 5). No test resulted in more than one misclassified word.

**Fig. 5.** Dictionary of 8 joined words with a test set of 6 swiped sequences. The training was completed 38 times. The result for each training session is displayed along the x-axis.

For similar words the network classified all words correctly in 8 of 38 test sessions (see Fig. 6). In 3 of the tests, 3 out of 6 words were misclassified.

**Fig. 6.** Dictionary of 8 similar words with a test set of 6 swiped sequences. The training was completed 38 times. The result for each training session is displayed along the x-axis.
Fig. 7 is a compilation of the average percentage of misclassified words for the three categories. The average percentages of misclassified words for similar words, varied words, and joined words were 21%, 3%, and 1%, respectively.

![Average Percent Misclassified](image)

Fig. 7. Average percentage of misclassified words for each of the categories.

E.2. Dictionary Size
Training was performed 30 times and after each training a test set of sequences was fed into the network. The percentage in the diagrams below show the fraction of the sequences misclassified by the network.

Figs. 8 - 11 show the percentage of misclassified words with different dictionary sizes.
For the dictionary containing 12 words the network classified all words correctly in 12 of 30 test sessions (see Fig. 8). For all except three tests no more than one word was misclassified.

**12 Varied Words**

![Bar chart showing fraction of misclassified words for 12 varied words](image)

*Fig. 8. Dictionary of 12 varied words with a test set of 9 patterns. The training was completed 30 times. The result for each training session is displayed along the x-axis.*

For the dictionary containing 20 words the network classified all words correctly in only 1 of 30 test sessions (see Fig. 9). In all except four tests two or more words were misclassified.

**20 Varied Words**

![Bar chart showing fraction of misclassified words for 20 varied words](image)

*Fig. 9. Dictionary of 20 varied words with a test set of 15 patterns. The training was completed 30 times. The result for each training session is displayed along the x-axis.*
For the dictionary containing 40 words the network did not classify all words correctly in any test (see Fig. 10). All tests classified 8 or more words incorrectly.

**40 Varied Words**

![Graph showing the fraction of misclassified words for 40 varied words.]

*Fig. 10. Dictionary of 40 varied words with a test set of 30 patterns. The training was completed 30 times. The result for each training session is displayed along the x-axis.*

For the dictionary containing 75 words the network did not classify all words correctly in any test (see Fig. 11). All tests classified 11 or more words incorrectly.

**75 Varied Words**

![Graph showing the fraction of misclassified words for 75 varied words.]

*Fig. 11. Dictionary of 75 varied words with a test set of 56 patterns. The training was completed 30 times. The result for each training session is displayed along the x-axis.*
Fig. 12 is a compilation of the average percentage of misclassified words for different dictionary sizes. The average percentages of misclassified words for dictionary size 12, 20, 40 and 75 words were 9%, 18%, 37% and 56%, respectively. The amount of misclassifications grew steadily as dictionary size increased.

Fig. 12. Average percentage misclassified compilation.

Fig. 13 shows the average time required to train the network with different dictionary sizes. The average time required to train the network for dictionary size 12, 20, 40 and 75 words was 1, 2.5, 6.5 and 54 s, respectively. The training time increased dramatically in the tests with the largest dictionary.

Fig. 13. Average time required for training. Values rounded to nearest half second.
F. Discussion

Gesture keyboards aim to simplify input where full size keyboards are not an option. To be able to do this a gesture keyboard needs to be easy to use, accurate and support lots of words.

The keyboard implemented in this study was not able to accurately distinguish between words when the dictionary grew in size. Fig. 12 shows that the average percentage of words that were misinterpreted grew from 9% to 18% for an increase in dictionary size of only 8 words (12 - 20). It continued to grow to 56% with a dictionary of 75 words. Since a lot of optimizations of the network still remains, the main problem is not the high numbers of misinterpretations as such, but the fact that the increase of misinterpreted words with increasing dictionary size is large. A high but relatively stable percentage of misinterpretations could possibly have been lowered by optimizations if it was not influenced by the size of the dictionary. However, this is not the case since the number of misclassified examples is correlated to the dictionary size.

An increased dictionary results in more intertwined sequences which in turn makes it harder for the network to distinguish between them. Kristensson and Zhai found that in their dictionary of 20 000 words, 1117 were identical in what they called normalized form (Zhai & Kristensson, 2012). They analysed the shape of the gesture rather than the actual letters traversed. Our situation is much the same but for different reasons. For example words like "helper" and "helped" result in very similar patterns with only one letter that distinguishes them from each other. This results in ambiguities between such types of words.

The results shown in Fig. 7 display the differences in misclassification between different categories of words. It is important that a gesture keyboard can distinguish between all types of words. In order to find which categories the MLP based keyboard can handle the results from the first test can be compared. The results from this test are displayed in Figs. 4 - 7. In Fig. 7 the average percentage of misclassified words for joined words is the lowest at 1%, varied words is at 3% and similar words is at 21%. Varied words would be expected to have resulted in the lowest percentage of misclassifications. The reason for this result is most likely the length of the words in each dictionary coupled with the pattern matching in the MLP. The dictionaries for this test can be found in Table 3. The length of the words in the varied dictionary is 6 letters for all except one word, while in the dictionary of joined words the length varies between 2 and 8 letters. Since the network matches complete patterns the length of the sequence is of great importance. E.g. a sequence with only 10 letters can easily be distinguished from a sequence with 20 letters, even if the first 10 letters are identical. This means that even though the joined words have more in common when it comes to actual letters in the sequence and less in common when it comes to sequence length, length is of greater importance in the pattern matching. That is the reason why joined words are more distinguishable.

A reason for the overall unsatisfying performance for larger dictionaries could lie in the relative naive way data is collected. As mentioned in Section D.1, a letter (represented by a number) is added to the sequence if it is touched by the gesture generated by the mouse pointer. This method can create very different sequences, even though the same word in swiped. In the

---

3 For further reading see Introduction to Shape Writing (Zhai & Kristensson, 2012).
example in Table 4 it can be noticed that the middle sequence share more letters with the sequence of the word “helper” than it does with the sequence of the actually swiped word (“helped”).

<table>
<thead>
<tr>
<th>Sequence</th>
<th>Word</th>
</tr>
</thead>
<tbody>
<tr>
<td>17 16 15 4 3 4 5 16 17 18 19 20 9 10 9</td>
<td>helped</td>
</tr>
<tr>
<td>17 16 15 4 3 4 5 17 18 19 20 10 9 8 7 6 5 4 3 14</td>
<td>helped</td>
</tr>
<tr>
<td>17 16 4 3 4 5 17 18 19 20 9 10 9 8 7 6 5 4 3 4 0 0</td>
<td>helper</td>
</tr>
</tbody>
</table>

Table 4. The middle sequence share 8 letters with a different sequence of the same word, but 10 letters with a sequence of another word.

Even though the words in this example are very similar to each other, the case can be the same with fairly different words. In practice, two sequences of the same word can match by as little as one letter, given the starting letter is the same. This phenomenon can occur when a gesture includes an extra letter in the beginning of the swipe, thus displacing the rest of the letters in the sequence. For a MLP this will result in poor performance, since a MLP only compares the number on the given index. This is emphasized by the test of similar words, shown in Figs. 6 and 7.

When sequences with large variances are set to match the same target, problems will also arise when a MLP is used. It will probably make classification harder and prevent convergence of the network. One may argue that input data should be generated more strictly, by creating more similar sequences. However, this would not allow any fault tolerance, and a user would certainly question the performance of such a gesture keyboard.

A more sophisticated way to gather data would possibly be to examine the gesture from a broader perspective, e.g. to study the angles of swept lines and starting/ending letters. This might allow for more fault tolerance since the angle of a line would be more resistant to small changes in the gesture. A way to improve performance given the current network structure and input data could be to differentiate the input data for a given word. If each group of similar input data for a given word was assigned a unique target, the MLP would be able to categorize the data better. This would however increase the complexity of the network, which might lower the performance.

Another possible solution to increase accuracy could be to replace the MLP with another neural network structure. As mentioned above, a MLP does not handle displaced sequences well. This problem may be solved using an algorithm designed to look for particular sequences inside a larger series of numbers.

A further important aspect to consider is the time required to train a network. The time seems to increase exponentially when increasing the number of words, which makes the current approach practically infeasible (see Fig. 13). Online training might be used to work around this problem, but sometimes, e.g. before first use, complete training is necessary. Other backpropagation algorithms were tried, but these resulted in poorer performance.

Since the length of the output is correlated to how many words the dictionary contains, the complexity of the network increases with the size of the dictionary. A more complex network, i.e. more weights, results in additional calculations which increase training time. One way to work
around this problem could be to use binary encoding in the output layer instead of using each separate index as a distinct class. This would decrease the number of output nodes, making the network less complex. Instead of a linear correlation the result would be a logarithmic correlation.

G. Conclusion

The gesture keyboard produce a problem that in many regards fit the description of a problem that could be solved using machine learning. A pattern exists, the problem cannot be pinned down mathematically and a lot of data can be generated.

One requirement of a gesture keyboard is to support a large dictionary. The results in the study show that with a growing number of words the accuracy as well as the speed of the network decreased. Using 75 words the percentage of misclassified words was 56%, combined with a training time of 56 seconds, which suggests that the approach tried in this study fails to reach the desired performance for larger dictionaries.

A further requirement is that gesture keyboards have to be able to handle all categories of words. When testing joined words the network delivered satisfying performance with a misclassification level of about 1%. However, quite surprisingly, varying words resulted in lower accuracy with a misclassification of about 3%. This is due to the fact that the network handles sequences of different lengths well while sequences of similar lengths are harder to distinguish between.

The study found that the sequences collected for each word varied due to variations in the gestures, causing the rest of the sequence to be displaced. This problem was emphasized in tests with similar words where sequences of different words could actually, from the neural network’s point of view, be more similar than sequences of the same word.

Machine learning offers many different approaches to problems and some have been considered in this study. The results presented here indicate that our implementation of a MLP is not the optimal approach for this problem even though optimizations could be made to improve performance. However, other methods such as a SVM or a SOM would need to be tried before making any final conclusions regarding the performance of machine learning as a way of creating a gesture keyboard.
H. Bibliography


