Mobile Application for Learning Japanese Syllabaries

Mobilní výuková aplikace japonských slabičných písem
Bachelor Thesis Assignment

Lukáš Pastuszek

Study Programme: B2647 Information and Communication Technology
Study Branch: 2612R025 Computer Science and Technology

Title: Mobilní výuková aplikace japonských slabičných písem
       Mobile Application for Learning Japanese Syllabaries

Description:
Create a mobile application for learning Japanese syllabaries - katakana and hiragana. The application is intended for devices with Windows Phone 8 operating system. The user should learn Japanese alphabets by trying to write the hiragana or katakana letters (presented in romaji) on the touch screen of the mobile phone. The application should recognize the characterized and evaluate the precision of the writings.

1. Prepare development tools for Windows Phone 8.
2. Create an application model and User Interface design.
4. Prepare the samples and let the algorithm learn them.
5. Test the results and evaluate the contributions.

The work should include the description of used technologies, user and programmers guide and UML diagrams.

References:
Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: Mgr. Ing. Michal Krumnikl, Ph.D.

Date of issue: 01.09.2014
Date of submission: 07.05.2015
doc. Dr. Ing. Eduard Sojka
Head of Department

prof. RNDr. Václav Snášel, CSc.
Dean of Faculty
I hereby agree to the publishing of the bachelor’s thesis as per the Study and Examination Regulations for Bachelor’s Degree Programmes at VSB-Technical University of Ostrava, Art. 26, Par. 9.

In Karvina 7th May 2015

I hereby declare that this bachelor’s thesis was written by myself. I have quoted all the references I have drawn upon.

In Karvina 7th May 2015
I would like to express my deepest thanks to the supervisor of this Bachelor Thesis, Mgr. Ing. Michal Krumnikl, Ph.D., who always gave me a good advice, when I needed it.
Abstract

This bachelor thesis summarizes the process of Optical Character Recognition (OCR) and divide it to its particular parts together with the overview and descriptions of the methods used to solve the particular problem of the whole process. Then the Windows Phone 8 platform is discussed and summarized, mainly the options and necessary tools a developer need to create a Windows Phone Application. The theoretical information is used in the second part of this thesis, where the algorithm for character recognition using artificial neural networks is implemented in the application for learning Japanese Syllabaries running on the platform with Windows Phone 8 operating system. The application process user-written character, one at a time on the screen and recognize, what character matches the given input. The thesis includes also description of the application together with its analysis, design and testing results.

Keywords: Character recognition, Classification, Windows Phone, Artificial neural networks, Japanese syllable scripts, Hiragana, Katakana

Abstrakt

Tato bakalářská práce shrnuje proces Optického rozpoznávání znaků (OCR) a popisuje jeho jednotlivé části společně s přehledem řešení dané problematiky. Dalé jsou zjištěny a shrnuty možnosti a nástroje, které jsou pro vývojáře potřeba při vytváření aplikace pro Windows Phone. Teoretické informace jsou využity ve druhé části práce, kde je algoritmus pro rozpoznávání používající umělé neuronové sítě implementován do aplikace pro učení japonských slabičných znaků pracující na platformě s operačním systémem Windows Phone 8. Aplikace zpracovává uživatelem napsaný znak na obrazovce a rozpoznává, který znak se shoduje se vstupem uživatele. Práce obsahuje také popis aplikace společně s analýzou, návrhem a výsledky testování.

Klíčová slova: Rozpoznávání znaků, klasifikace, Windows Phone, umělé neuronové sítě, japonské slabičné znaky, Hiragana, Katakana
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCR</td>
<td>Optical Character Recognition</td>
</tr>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>WP</td>
<td>Windows Phone</td>
</tr>
<tr>
<td>CSV</td>
<td>Comma Separate Value</td>
</tr>
<tr>
<td>MVVM</td>
<td>Model View ViewModel</td>
</tr>
<tr>
<td>XAML</td>
<td>Extensible Application Markup Language</td>
</tr>
<tr>
<td>XML</td>
<td>Extensible Markup Language</td>
</tr>
<tr>
<td>VS</td>
<td>Visual Studio</td>
</tr>
<tr>
<td>USB</td>
<td>Universal Serial Bus</td>
</tr>
<tr>
<td>JS</td>
<td>JavaScript</td>
</tr>
<tr>
<td>MSDN</td>
<td>Microsoft Developer Network</td>
</tr>
<tr>
<td>PNG</td>
<td>Portable Network Graphics</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machines</td>
</tr>
</tbody>
</table>
# Table of contents

## 1 Optical Character Recognition Process

1.1 Pre-Processing .................................................. 9  
1.1.1 Binarization ............................................. 10  
1.1.2 Image Cropping ......................................... 10  
1.1.3 Re-Scaling ............................................... 10  
1.1.4 Nearest Neighbour Algorithm ......................... 10  
1.1.5 Interpolation ............................................ 11  
1.2 Feature Extraction ........................................... 11  
1.2.1 Zoning Method ......................................... 12  
1.2.2 Projection Method .................................... 12  
1.3 Character Recognition ..................................... 13  

## 2 Approaches to Classification

2.1 Template Matching ............................................. 14  
2.2 Statistical Decision Theory ................................. 15  
2.2.1 Bayes Decision Rule .................................. 15  
2.2.2 Bayes Classifier ....................................... 15  
2.3 K-nearest Neighbours Classification ....................... 17  
2.4 Structural Approach ....................................... 17  
2.5 Support Vector Machines .................................. 17  
2.5.1 Two-class SVM ......................................... 17  
2.5.2 Multiclass SVM ........................................ 18  
2.5.3 Non-linear SVM ....................................... 19  
2.6 Comparison of the Classifiers .............................. 19  

## 3 Artificial Neural Networks

3.1 Biological and Artificial Neural Networks .................... 20  
3.2 Artificial Neuron ........................................... 20  
3.3 Forming artificial Neural Network ........................... 22  
3.3.1 Feed-forward Neural Network ......................... 22  
3.3.2 Recurrent Neural Networks ............................ 23  
3.4 Learning Process ........................................... 23  
3.4.1 Back-Propagation Algorithm ......................... 24  
3.5 Overfitting .................................................. 24  

## 4 Windows Phone Development

4.1 Development Preparation .................................... 26  
4.1.1 XAML Applications .................................. 26  
4.1.2 Native Development .................................. 26  
4.1.3 JavaScript Application ................................. 27
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Windows Phone Projects</td>
<td>27</td>
</tr>
<tr>
<td>4.3</td>
<td>Debugging</td>
<td>28</td>
</tr>
<tr>
<td>4.4</td>
<td>Deploying and Publishing</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>Practical Part</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Design of Supporting Applications</td>
<td>29</td>
</tr>
<tr>
<td>5.2</td>
<td>Feature extraction of learning data</td>
<td>30</td>
</tr>
<tr>
<td>5.3</td>
<td>Designing the Learning Application</td>
<td>30</td>
</tr>
<tr>
<td>5.4</td>
<td>Designing the Artificial Neural Network</td>
<td>31</td>
</tr>
<tr>
<td>5.5</td>
<td>Software Implementation</td>
<td>34</td>
</tr>
<tr>
<td>5.6</td>
<td>Training Neural Network</td>
<td>35</td>
</tr>
<tr>
<td>5.7</td>
<td>Programming the Neural Network</td>
<td>35</td>
</tr>
<tr>
<td>5.8</td>
<td>Description of the application</td>
<td>38</td>
</tr>
<tr>
<td>6</td>
<td>Testing</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Katakana Testing Analysis</td>
<td>40</td>
</tr>
<tr>
<td>6.2</td>
<td>Hiragana Testing Analysis</td>
<td>41</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>43</td>
</tr>
<tr>
<td>Appendix</td>
<td></td>
<td>44</td>
</tr>
<tr>
<td>A</td>
<td>Neural Network Accuracy Graphs</td>
<td>45</td>
</tr>
<tr>
<td>B</td>
<td>Tables of Japanese Syllabic Scripts</td>
<td>47</td>
</tr>
</tbody>
</table>
List of tables

5.1 Table of misclassification error in % of the NN using different number of zones of the image .......................... 33
## List of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>Mobile OS unit shipment share</td>
<td>7</td>
</tr>
<tr>
<td>1.1</td>
<td>Image size normalization</td>
<td>11</td>
</tr>
<tr>
<td>1.2</td>
<td>Results of re-scaling image from 50x50 pixels to 200x200 pixels using different algorithms</td>
<td>12</td>
</tr>
<tr>
<td>2.1</td>
<td>An example of the probability density functions for two classes in two dimensional feature space</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Linearly separable problem of two classes solved by SVM</td>
<td>18</td>
</tr>
<tr>
<td>3.1</td>
<td>Model of brain neuron</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Model of artificial neuron</td>
<td>21</td>
</tr>
<tr>
<td>3.3</td>
<td>Activation functions</td>
<td>21</td>
</tr>
<tr>
<td>3.4</td>
<td>Feed-forward neural network</td>
<td>22</td>
</tr>
<tr>
<td>5.1</td>
<td>Wireframes of the main application</td>
<td>32</td>
</tr>
<tr>
<td>5.2</td>
<td>Dependence of the misclassification error on number of hidden neurons</td>
<td>34</td>
</tr>
<tr>
<td>5.3</td>
<td>Class diagram of the ANN</td>
<td>36</td>
</tr>
<tr>
<td>5.4</td>
<td>Sequence diagram of computing the result by the ANN</td>
<td>37</td>
</tr>
<tr>
<td>5.5</td>
<td>Screenshots of the main application</td>
<td>39</td>
</tr>
<tr>
<td>6.1</td>
<td>Often misclassified pairs of Katakana characters</td>
<td>40</td>
</tr>
<tr>
<td>6.2</td>
<td>Image of Tsu/Shi and So/N characters</td>
<td>41</td>
</tr>
<tr>
<td>6.3</td>
<td>Katakana N character misclassification</td>
<td>41</td>
</tr>
<tr>
<td>6.4</td>
<td>Often misclassified pairs of Hiragana characters</td>
<td>42</td>
</tr>
<tr>
<td>A.1</td>
<td>Accuracy of the neural network in classifying Hiragana script</td>
<td>45</td>
</tr>
<tr>
<td>A.2</td>
<td>Accuracy of the neural network in classifying Katakana script</td>
<td>46</td>
</tr>
<tr>
<td>B.1</td>
<td>Table of Hiragana characters</td>
<td>47</td>
</tr>
<tr>
<td>B.2</td>
<td>Table of Katakana characters</td>
<td>48</td>
</tr>
</tbody>
</table>
List of source code

1    Pseudocode for the back-propagation algorithm . . . . . . . . . . . . . . . . . . 24
Introduction

The goal of the thesis is to develop an application for learning Japanese syllables Hiragana and Katakana running on the platform Windows Phone 8. The main feature of the application and the most important part of the thesis is using Optical character recognition (OCR) process with artificial neural networks (ANN) to identify, if the user’s hand-written character matches the expected one. The phonetic transcription of the Japanese character is given and the user writes down the Japanese character for the given transcription. The application then evaluates with the recognition system, if the entered character is correct and displays the result to the user.

Japanese Writing System

Writing system in Japan differs greatly from the one used throughout the Indo-European language family. It consists of two syllabic scripts called Hiragana, Katakana and one logographic script called Kanji, each having its own specific function in the Japanese language.

Kanji is the largest of all the three scripts. It developed from original Chinese characters in 8th Century. It is used to write nouns, adjectives and verbs. Total number of characters is at least five thousand, from which around two thousands are considered to be basic and are learned by Japanese children during first level of education.

Due to the existence of great differences between Japanese and Chinese languages, only adapting Chinese characters was not sufficient. For that reason, with the introduction of the Chinese characters in Japan, a completely new script of Hiragana was developed to be used originally by women for writing poetry, diaries, letters and some other genres of literature. Today the script is mainly used for writing word suffixes, particles, and in some cases even whole words.

Katakana was developed later on in 10th century by Buddhist monks to ease the understanding of the Buddhist Sutras. Today it is mainly used for writing loanwords, transcription of foreign names, brands, colloquialisms and to emphasize words in a text. Both syllabic scripts are known together as Kana. A miniature version of the two scripts called Furigana is used to explain the pronunciation of Kanji characters as a hint in various textbook or fairy-tale books for children. [14]

Market analysis

Windows Phone 8 is relatively new operating system released at the end of 2012 to succeed the first version of Windows Phone (i.e. Windows Phone 7). Over next years, Windows Phone has become the third most popular mobile operating system by share in unit shipments (Fig. 0.1) but it is still not able to increase its market share to catch up the two dominant platforms iOS and Android. For the reason of smaller market share, not so many applications are developed for the Windows Phone operating system if we compare it to those two major OS.
Introduction

There are already several applications concerning learning Japanese characters available on the Windows Phone Store, but only very little of those teach by writing the character on the screen. Typical examples of already developed learning applications are Obenkyo\(^1\) and Japanese4Beginners\(^2\). These applications only list japanese syllable characters or show them in detail one-by-one with their phonetic transcription. To test his knowledge, the user can take a quiz. During the quiz, a Japanese character and several possible transcriptions are shown and the user chooses the correct transcription. Another application with much better user experience is called Write Japanese\(^3\). This application is available for 50 CZK, but its limited free version (without Kanji, but with Katakana and Hiragana scripts) is suitable for the analysis of the application for this thesis. User learns how to write characters with or without guidelines. The recognition system is run after each stroke is written on the screen. The application analyses if there is enough similarities between the expected and the actual stroke written by the user. Even though this is a user friendly approach, the recognition of the complete character written by the user is not done and only the shape of the stroke, not its position is compared. This allows the user to write the stroke anywhere on the screen and the application still says it is correct.

Thesis Structure

At the beginning, theoretical knowledge of Optical Character Recognition is described (Chapter 1), mainly the parts of the OCR, which are essential for this thesis - pre-

\(^1\)Available on http://www.windowsphone.com/cs-cz/store/app/obenkyo/a6c50add-65cc-4d52-9384-59ecdb4871b9


\(^3\)Available on http://www.windowsphone.com/cs-cz/store/app/write-japanese/fce465f2-72b3-406ab83d-1f70db635ac
processing, feature extraction and character recognition. Approaches to character recognition is a topic of bigger importance so it is described in the standalone Chapter 2. Later a modern approach for character recognition - using artificial neural networks is elaborated more deeply in Chapter 3. After that, we take a look into application development for Windows Phone (Chapter 4). We describe developer’s options for Windows Phone application development. All the knowledge is used in the practical part of the thesis (Chapter 5) where the application is designed and created. After that, precision of the application is tested and the results are processed and evaluated. (Chapter 6).
1 Optical Character Recognition Process

In a modern society, there is a need to rely more on the computers than on people to handle great amount of data, thus there is a need of converting non-digital information into machine readable form. The complete process regarding conversion of characters into the digital form is called Optical Character Recognition Process (OCR). It is a process, which uses an image of handwritten or printed characters as an input and produces an output in a form of digital information which is further processable by computers.

The OCR process is usually divided into several consecutive parts which can be considered as independent. The output of the previous process is an input for the next one. Those processes are:

- **Optical scanning** is a process of capturing written text as a digital image by a scanner or a camera.
- **Pre-processing** performs operations with the digital image to improve the probability of successful segmentation and recognition.
- **Location, segmentation** is a process of dividing the image into smaller parts. At first into lines and later into characters.
- **Feature extraction** is a process of acquiring characteristics of individual characters.
- **Character recognition** is the part, where the input characteristics are transformed into the recognized characters.
- **Post-processing** finalizes the process by composing the individual characters into words, lines and pages. It may also include an error correction of the recognition.

All the steps listed above form the complete OCR process, but not all of them are important for the purpose of this thesis. The goal of the thesis is to create an application where the input image consists of only one character at a time and it is drawn on the touch screen of a mobile device. The fact that image is already in the digital form means there is no need to perform any optical scanning of the image. It also simplifies the pre-processing process, because any noise or faults of the image are expected to occur. Location, segmentation and post-processing are also excluded when the characters are processed one-by-one. Parts which have left are pre-processing, feature extraction and especially character recognition and those will be discussed.

1.1 Pre-Processing

The main task of the image pre-processing is to prepare an image for successful and more precise recognition. The processes used for pre-processing are:

- **Background removal** - removes background i.e. non-character elements from the image. This needs very context specific solution e.g. removal of the background watermark which is lighter than the characters.
• **Noise removal** - is a substitution of pixels the color of which does not have a relation to its surrounding pixels.

• **Skew and slant detection and correction** - uses horizontal and vertical lines or histograms to detect that characters are not upright or the line on which they are written is not horizontal. For correction, the image is rotated or skewed back.

• **Thinning** normalizes the width of a stroke of a character.

• **Character size normalization** normalizes the size of the image to the given resolution to make all the images of the same width and height in pixels.

Most of the pre-processing steps are performed on the whole image before the segmentation. Only thinning and size normalization is done afterwards.

### 1.1.1 Binarization

When having an RGB or a grayscale image, every pixel or subpixel can have numerous values, usually between 0 and 255. Binarization of an image is a process, where all the pixels of an original image are replaced by only one of two values 1 or 0. These values are then much easier and faster to process.

### 1.1.2 Image Cropping

Image cropping is the first step of character size normalization. Automatic image cropping can be done very easily. As shown in Fig. 1.1a, the character itself does not precisely cover the whole area of the image. A cropping from top, right, bottom and left is done to create an image in which the character touches the borders. To do this, only top-most, right-most, bottom-most and left-most non-white pixel of the image has to be found. A new border is created where these pixels are found and the resulting image is created within this area (Fig. 1.1b).

### 1.1.3 Re-Scaling

The image can have various dimensions after cropping and that would make the recognition itself much more difficult. Because of this a re-scaling is used for changing the number of pixels of the image. The dimensions of individual character image can either be lowered or increased to be normalized. A lot of algorithms exists in image processing with various complexity, efficiency, smoothness and sharpness.

### 1.1.4 Nearest Neighbour Algorithm

Is the easiest method for re-scaling an image. The algorithm, when up-scaling the number of pixels in the image takes the value of the closest adjacent pixel and fill the new pixel with the same value. During down-scaling more pixels are removed from the image and they are replaced by the pixel with one of the value removed. The best advantage is very low complexity. This method preserves the hard edges of the image.
1.1.5 Interpolation

An interpolation is another method for re-scaling an image. It estimates new unknown value in a given space by already known values. The simplest one is a Linear Interpolation which estimates the value of 2 closest adjacent pixels. When considering two known points \( A[x_1,y_1] \) and \( B[x_2,y_2] \) and unknown point \( C[x_0,y_0] \) the computation is the following:

\[
y_0 = y_1 + (y_2 - y_1) \frac{x_0 - x_1}{x_2 - x_1}
\]

Computed value of the unknown point is given by the color of other known points in the space and their distances from the unknown point. Similarly, when downscaling an image, pixels are removed from the image and they are replaced by the new color pixel taking distances to these point into consideration.

More complex interpolations are taking more adjacent pixels into computation. Bilinear interpolation takes 4 known values (pixels), Bicubic interpolation 16 values. The more pixels the interpolation takes into computation, the smoother the image is for the price of increasing computational time. The difference between results of interpolation algorithms can be seen on Fig. 1.2.

1.2 Feature Extraction

Feature extraction goal is to obtain from a signal, image or any other type of information a set of values (usually a vector). Those values are acquired by some kind of measurements on the input pattern. When the extration is made, a vector, which is created is called a feature vector. The division of feature extraction methods can be made:

- **Structural features** is an approach, which counts number of occurrences of a feature in a raw input. Those are for characters for example: intersections of strokes, end points of strokes etc.
Optical Character Recognition Process

- **Statistical features** is a approach where raw input data are transformed into a vector using some kind of statistical estimation. This estimation is necessary, when input data are too large to be used directly by character recognition algorithm e.g. having image in size 200 x 200 pixels contains 40 000 values of pixels, which is usually too much to be processed directly.

Different types of feature extraction methods can be combined for more complex recognition tasks. If multiple methods are used, they can be combined to form either one larger feature vector or multiple feature vectors.

### 1.2.1 Zoning Method

Zoning of an image is a statistical feature extraction process. The original image of dimensions \( n \times m \) is divided into smaller parts called zones. Less zones of the original image are created, bigger is the approximation and the loss of data. Every zone is then represented as one real or integer value in the feature vector. The value representing each zone can be computed in various ways, but the most common calculations would be the average value of all pixels within the zone or a ratio between number of black pixels \( n_b \) and total number of pixels in the zone \( n_t \) within the zone.

\[
X_i = \frac{n_b}{n_t}
\]

### 1.2.2 Projection Method

Vertical and horizontal projection of a binary image can be done by the sum of black pixels in the given row/column. If we do such a sum operation for all rows of an image a vector with size \( n \) will be created, where \( n \) will be a number of rows of the image. Values of an image will be in range \( <0; m> \), where \( m \) is a number of columns of the image. The projection can be done on the whole image or the image can be divided to several parts and projection can be done on those parts resulting in more feature vectors.
1.3 Character Recognition

Character recognition is a problem of pattern recognition. Pattern recognition in general can be divided to 3 main parts

- **Classification** - is a categorization of input data into classes, which are known a priori. Output is a discrete value.

- **Regression** - is a prediction of output value from input data. Output is a continuous value.

- **Clustering** - is a process of categorization of input data into classes based on similarities without a priori knowing the classes.

We furthermore consider character recognition as a problem of classification which goal is to assign an input vector (image) an appropriate class (character label).

Character recognition is still part of the OCR process, but its approaches as a classification problem are discussed in the following standalone chapter.
2 Approaches to Classification

Classification is a scientific discipline whose goal is to assign a class label to input object [1]. The input object is a $d$-dimensional data (feature) vector $x = (x_1, \ldots, x_p)$ in the $d$-dimensional feature space, whose components $x_i$ are measurements on the input object. There exist $C$ groups or classes, denoted $\omega_1, \ldots, \omega_C$. With each input $x$, there is associated a categorical variable $z$ that denotes the class or group membership; that is if $z = i$ then the input vector is classified to belong to class $\omega_i$ [4]. In some of the problems, there is a possibility for another class to be, which means there are $C + 1$ classes. This additional class is called undefined class or reject option. Input vector which is not suitable to any of the $C$ classes is assigned to this class [4].

The classification is a problem using supervised learning - labeled data are used to construct the algorithm used for classification. On the other hand unsupervised learning used with clustering categorizes input data to $C$ classes without previous knowledge about these classes.

We can divide pattern recognition approaches and techniques into several categories discussed in the following subsections.

2.1 Template Matching

Template matching is the simplest technique for the classification. Its mathematical representation if we consider a template to be an image (two dimensional vector) is described as follows:

$$D_k = \sum_{i=1}^{M} \sum_{j=1}^{N} Z(x_i, y_j) - T_k(x_i, y_j)$$

where $Z$ is an input character image and $T_k$ is a template $k \in [1..K]$ with known class $c \in [1..C]$, both input and template data with dimensions $N \times M$. $D_k$ is computed distance of a given input $Z$ from the given template $T_k$. $D_k$ value is computed for every template $T_k$. Class of a template for which the distance is the smallest $\min(D)$ is most likely the class of the input data. Usually a threshold value of lowest allowed dissimilarity $D_T$ is set. If $\min(D) > D_T$, the input data is not classified. [12]

For the simplest type of template matching classification no feature extraction is necessary, because the input image is used directly as input data and it is compared to the pattern images. In this case an input is a two-dimensional vector (the image itself). For this type of template matching using the binarization of an image is desirable to increase the speed of the algorithm.

Template matching biggest disadvantage is a big computational costs $O(M \times N \times K)$ which increase with number of templates $K >= C$ i.e. one class can have multiple patterns.
2.2 Statistical Decision Theory

In the statistical approach, each input is represented in terms of $d$ features or measurements and is viewed as a point in a $d$-dimensional space.

During the learning process, the complete $d$-dimensional space is divided into $C$ regions $\Omega_i$, where $i = 1,...,C$. The input vector is classified as class $\omega_i$, if the vector is placed inside the region $\Omega_i$.

The goal during constructing the classifier is to divide the feature space in the way that the classifier have the best performance. Very important factor is choosing a correct feature extraction method in such way that input patterns which belongs to the same class occupy compact and disjoint region in the $d$-dimensional feature space [3].

2.2.1 Bayes Decision Rule

Statistical classification methods are based on the Bayes decision theory. [9] It considers prior probability $p(y)$ of occurring of class $y$ without any further knowledge. Since we have additional information in form of input vector $x$, we can introduce the likelihood (sometimes probability density) function in form $p(x|y)$ as a probability of the input vector $x$ belonging to class $y$ and total probability of $x$ by computing the marginal. The Rule is the following:

$$p(y|x) = \frac{p(x|y) \times p(y)}{p(x)} = \frac{\int p(x|y) \times p(y) dy}{p(x)}$$

The computed value $p(y|x)$ is posterior probability which is a probability of observing class $y$ when taking input vector $x$ into consideration.

2.2.2 Bayes Classifier

The Bayes classifier minimizes the risk $R_y(x)$ using loss matrix $c$ containing loss values $c_{y,z}$ as misclassification costs of class $y$ to class $z$. If we consider the loss value to be 0 for correct classification and 1 for incorrect classification, the risk of misclassification input vector $x$ to class $y$ is computed as

$$R_y(x) = 1 - P(y|x)$$

It can be seen the risk is the smallest, where the probability is the highest. Bayes classifier classify the input vector $x$ to belong to class with the highest posterior probability i.e. smallest risk.

The Bayer classifier requires the knowledge of the a priori probabilities of classes and likelihood functions. These has to be acquired from the training data.

As a visualising example, the division of 2 dimension feature space can be seen on Fig. 2.1.
Fig. 2.1: An example of the probability density functions for two classes in two dimensional feature space [13]
2.3 **K - nearest Neighbours Classification**

Is a simple classifier that takes an input data vector $X$ and classify it to the $k$-nearest neighbour vector in the feature space. The $k$ can be a number greater than 0. With the simplest example when $k = 1$, an algorithm finds the closest vector $Z$ with class $c$ (determined by some algorithm e.g. Euclidean distance, Manhattan distance, etc.) and assign the class $c$ to the input vector $X$. If $k > 1$, $k$ closest vectors are found and the most often class $c$ within these $k$ closest numbers is assigned to the input vector.

The nearest neighbour algorithm is very easy to implement, its computational time is $O(N)$, and it is quite complex for big learning data sets. The computational time can be decreased if better algorithms are used to search the feature space e.g. approximate nearest neighbour search.

2.4 **Structural Approach**

This approach uses a decision rules to assign characters into classes. In its simplest version it make uses of structural features like number of end points, number of loops in the character, number of strokes, number of vertical lines. An algorithm goes through the decision tree, which is constructed from the learning set. The class for which the input data is classified is found, when an algorithm reaches the leave of the decision tree.

2.5 **Support Vector Machines**

Support Vector Machines (SVM) is a type of classifier which divides the feature space by hyperplanes (discriminant function). The division of the feature space can be either linear or non-linear.

2.5.1 **Two-class SVM**

If considering 2-dimensional feature space with 2 linearly separable classes in disjoint regions, there is an infinite number of hyperplanes $g(x)$ which can divide the feature space in the way that

$$g(x) \geq 1, \forall x \in \omega_1$$

$$g(x) \leq -1, \forall x \in \omega_2$$

i.e. all vectors above the line will be classified as class $\omega_1$ and vectors below the line will be classified as class $\omega_2$.

SVM is looking for the hyperplane to divide these two regions in the way that the margins between the hyperplane and closest points of each class is the maximum possible and the same to both sides (see Fig. 2.2) [13] using Karush–Kuhn–Tucker conditions$^4$.

When the learning sets are not linearly separable, Karush-Kuhn-Tucker conditions$^4$ are used similarly, but some of the training features are not taken into consideration when constructing the hyperplane. The hyperplane is constructed in the way that the margins

$^4$The mathematical representation can be found in ref. [13], section 3.7
are as large as possible, but number of points not taken into consideration is as small as possible.

### 2.5.2 Multiclass SVM

So far, we have discussed only binary classification task. The problem usually gets more complex and there is a need to solve $M - \text{class}$ problem. There are number of options how to implement multiclass SVM.

- **One-against-all** - straightforward extension to two-class SVM would be to construct $C$ standalone hyperplanes. This solution fails, when more than one $g_i, i \in [1..C]$ is positive.

- **One-against-one** - this would create $\frac{M(M-1)}{2}$ binary classifiers, each separates two classes. The input is assigned to the class based on a majority vote. The disadvantage is the need of training a lot of classifiers.

- **L-classifiers** - for $C$ class, there are $L$ classifiers used, where $L$ is a number designed by the classifier designer. Each class is represented by a binary code word of length $L$ with bit values +1 or -1. [13]. One classifier is trained to return one value for multiple classes, but the word is unique for the class. If the class word does not corresponding to none of the expected, input is classified to the word with smallest Hamming distance (number of places were the code words differs). [13]
2.5.3 Non-linear SVM

When the classes in the feature space are not linearly separable, the SVM allow to map the feature space into more dimensional space, where the classes are separable by a hyperplane. The complexity is not increased dramatically, because the vectors do not have to be mapped into the more-dimensional space, only their inner product is computed with using of kernel function.

2.6 Comparison of the Classifiers

For the completeness we have to note here that neural network is another approach which is discussed in the following stand-alone chapter.

Every classifier has its advantages and disadvantages. Statistical classifiers are easy to implement, its output is not just the class label, but we can acquire the probability of that decision, it is fast, but it requires bigger data set. It is a good choice to use, when the probability of the occurrence of each class is of some importance.

If one is interested in memory usage then k-nearest neighbour would be a bad option, compared to e.g. neural networks. Training time is another issue which has to be considered. Neural networks takes much longer time to train than other approaches like template matching or k-nearest neighbour where the learning time is theoretically 0. Another property is a classification time which is very fast for neural networks so as for decision trees, but which is very high for the template matching. Very important property is how the classifier can generalize the problem, where SVM and neural networks tends to have a good capability of generalization compared to other techniques.

There are of course more algorithms to choose from when creating a classifier and one has to always evaluate what properties are important for the problem and what will be the best choice for him. Modern classifiers and character recognition systems often uses more approaches combined together.
3 Artificial Neural Networks

Neural network is a standalone approach to pattern recognition based on neuroscience. The basic idea is based on the model, how the human brain works.

3.1 Biological and Artificial Neural Networks

The brain, as known from biology and neuroscience, is able to work thanks to its electrochemical activity. This activity is performed by its cells called neurons. Millions of neurons in the brain attached together form the neural network. (a neuron’s structure is shown at Fig. 3.1) The neuron differs from other cells in the human’s body in the presence of input projections called dendrites and output projection called axon. Dendrites carry the electric signal from other neurons or from parts of the sensory system into the cell’s body. There are thousands of dendrites and just one axon with lots of terminals at its end. The cell itself maintains the voltage gradient taken out through the axon according to to the electric signals from dendrites.

The artificial neural network (ANN) is a computational model very much inspired in its neural paradigm. It is also composed of neurons, but the original neuron is replaced by the mathematical model (Fig. 3.2). Each artificial neuron consists of input links (replacing the function of dendrites), body and output links (replacing the function of axon). Artificial neurons are interconnected with others to form the ANN.

3.2 Artificial Neuron

As shown in Fig. 3.2, the body of the neuron is a mathematical model performing two functions - input function \(in_j\) and activation function \(g\).

The input function is given as a sum of its inputs:

\[
in_j = \sum_{i=0}^{n} w_{i,j} a_i
\]

where \(a_i\) is the output value of the neuron \(i\) or input value of a network and \(w_{i,j}\) is a bias weight (strength) associated with the given input \(a_i\) link into the neuron \(j\). The weight is usually a value between 0 and 1. It is also important to notice here that one special dummy input link \(a_0\) with bias weight \(w_{0,j}\) which is not connected to output of any previous neuron.

The activation function \(g\) is a non-linear function performed on the output of the input function

\[
a_j = g(in_j)
\]

The most simple activation function is a threshold function (Fig. 3.3a) with only two possible output values \(y_1, y_2\) and a threshold value \(x_T\). Output value \(y = y_1\) for \(x < x_T\) and \(y = y_2\) for \(x >= x_T\). Another possibility is to use a logistic function, which is defined as \(\frac{1}{1+e^{-x}}\). Other functions used are e.g. Softmax function and Gaussian function. Depending on the activation function used, the output value can be discrete or continuous point in N-dimension space [11].
Artificial Neural Networks

Fig. 3.1: Model of brain neuron. Source: http://en.wikipedia.org/wiki/Neuron #mediaviewer/-File:Neuron_Hand-tuned.svg [13]

Fig. 3.2: Model of artificial neuron [10] [13]

Fig. 3.3: Activation functions
3.3 Forming artificial Neural Network

A neuron is considered to be a stand-alone processing unit performing mathematical operation on the input values to compute the output value. Complete ANN consists of inputs, output neurons and neurons between them. Those neurons are interconnected into some topology, most often into layers.

Neural network is capable of solving real world problems by approximating non-linear functions of the inputs. Non-linearity of the network is ensured by using non-linear activation functions to compute outputs of artificial neurons.

Computation within the neural network goes as follows. First input signal is stored in the input cells. The input values of each particular neuron in network is determined by the values taken from the input cell and/or other neuron’s output values together with weights assigned to them. This is done for all neurons in the network, until the output layer is populated with output data.

For the same given input vector \( a \) the output value of the neuron can always differ according to the weight vector \( w \). When ANN is created, input weights of all neurons are randomized and there is no real classification going on. To make the network work properly, all the weights of the network has to be adjusted.

3.3.1 Feed-forward Neural Network

Consists of neurons which has connections only in one direction. Every node receives input from “upstream”nodes and delivers output to “downstream”nodes[10]. The “upstream”can either be set of neurons or inputs and “downstream”can be either set of neurons or output.

The minimum number of layers is one - that means an input data goes directly to the output layer (Fig. 3.4a). This is called a single-layer ANN or perceptron network. Depending on the problem there can be one or more hidden layer between the input and output layers. When a hidden layer is also presented, the network is then called a multilayer network (Fig. 3.4b).
3.3.2 Recurrent Neural Networks

Recurrent networks is also structured into layers, similarly to its feed-forward sibling, but it feeds neuron’s outputs back to their inputs. This makes recurrent neural networks to reach stable state, oscillations or chaotic behaviour. They are much more complex and complicated to understand, but it allows them to create short-term memory, which can be desired for some problems.

3.4 Learning Process

The process of adjusting weights of the complete ANN is called learning and the mechanism used for the learning is called the learning algorithm. The learning process has to be done before any classification of real data is done. When running a basic learning process at least two data sets are used:

- **training data** - data used for adjusting weights of the network
- **testing data** - used for testing the accuracy of the neural network, when the learning is done

There is a need for great amount of learning data to make the network as precise as possible. The number of data is also dependent on the type of neural network used. For more complex ANN, more data has to be used. The distribution of the data for training, validation and testing can vary and it depends on one’s need and preference.

Data set used for the learning process consists of number of vectors of length $n$, where $n$ is number of cells in input layer. Together with each input vector, the output vector has to be provided with length $m$, which is also the number of cells in the output layer.

The whole process of learning is done in cycles (called epoches or iterations). At first the input vector is provided to the input layer of the network and (so far untrained) layer provides the output. This output is compared to the expected one provided for learning. When the output vector is not the same as expected, the network has to adjust its weights. That is done with the learning algorithm, which is capable of changing the weights in the way that when the same input vector is provided next time, the output will be more likely more accurate than before. For one epoch, all input vectors are provided to the network only once. The learning has to stop after a number of epochs.

At the end of the learning process, testing data are used to evaluate the actual performance and precision of the network on completely independent set of data which were not known and were not used for the learning process.

Testing data set can not be subset of the learning data or vice versa. If the same data were used for learning and testing, the precision of the recognition of testing data could be very close to 100%, but the network could not be able to deal with real world data.
3.4.1 Back-Propagation Algorithm

Back-Propagation is an algorithm used together with feed-forward neural networks. The error at the ANN output layer has to be able to adjust all the previous weights in the network. The back-propagation pseudo-code is shown in source code no. 1.

In the output layer, the error $\Delta[j]$ is computed for each output neuron $j$ by using the derivative of the activation function on the input function of the neuron.

In all hidden layers, the derivative of the activation function is used in the same manner, but the error rate $\Delta[j]$ is computed from the weight of the links to the cells in the next (following) layer together with the error previously computed in the next layer. This way, the total error is back propagated throughout the whole network in the way that each neuron is responsible for some fraction of the output error in the output layer of the network.

At the end, all weights in the network are adjusted. To adjust a single weight $w_{i,j}$ we use the error of the neuron $\Delta[j]$, output value of the previous neurons (or input value in the first hidden layer) $a_i$ and a parameter $\alpha$ called a learning rate, which is a value smaller than 1. It can be either constant or it can change during the training process.

```plaintext
foreach node j in output layer L do
    $\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$
end
for $l \leftarrow L - 1$ to 1 do
    foreach node i in layer $l$ do
        $\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j} \Delta[j]$
    end
end
foreach weight $w_{i,j}$ in network do
    $w_{i,j} \leftarrow w_{i,j} + \alpha \times \alpha_i \times \Delta[j]$
end
```

Source code 1: Pseudocode for the back-propagation algorithm [10]

3.5 Overfitting

Overfitting is one on the great problems associated with neural networks. When a neural network is overfitted, it tends to memorize input data for learning rather than finding the underling pattern to be able to generalize the problem. Overfitting can occur when the neural network is designed in too complex way than needed to solve the problem. One can avoid overfitting by designing the neural network correctly.

As stated in the previous chapter, learning is done in epochs and learning data set for each epoch is the same. When too many learning iterations are done and data are still repeating to be used the error rate on learning data is decreasing with each iteration, because the network adapts to the learning data with every iteration. At some point while learning, the actual performance on real data will get worse, even though it will still getting better for the learning data. At this breaking point the network tends to memorize
learning data and if the network would be trained more further, it would get overfitted and thus have worse performance in the real world.

To solve overfitting from this reason there exists an approach called Early stopping. This algorithms tells us, when to stop the learning of the neural network. To use it, the data set has to be divided into three subsets instead of two. The additional set is called a validation data set. After one or more epochs during learning, the error of the network is evaluated on the validation set, which represents the real world data. The error tends to get smaller with each iteration in the same manner as the error on learning data, but at the point, where the network starts to get overfitted, the error rate on learning data is still going to get smaller, but the error rate on the validation set starts to increase. The learning is stopped at this point to prevent from overfitting. The learning is not stopped after the error rate is increased just once, but for several successive evaluations to avoid stopping for random deviation.

As the error rate, the value called Mean Squared Error is used, which is computed as follows:

$$\frac{1}{N} \sum_{i=1}^{N} (e_i)^2$$

where $e_i$ is a difference between a single expected and real output.
4 Windows Phone Development

In this chapter I describe what are the development options, tools and technologies necessary for creating a Windows Phone application. I did not try to describe any specific functions and features the WP allows like living tiles, notifications etc. The complete guide for Windows Phone development including specific functions and features are described in details in Windows Phone 8 Development Internals [5].

4.1 Development Preparation

Since the Windows Phone is a Microsoft platform, the most common way of developing an application is using Visual Studio. Microsoft offers its Express version for free on its websites 5. The download package of Visual Studio Express 2013 for Windows includes complete set of necessary tools for developing including Windows Phone SDK, device simulator, IDE.

There exist more options, usually cross-platform for creating application for more mobile operating systems, one of the well-known is Apache Cordova. These alternatives are suitable for some of the applications, since its API does not have to support 6 every piece of hardware of the device or it is also not possible to use any or at least some of the platform specific advantages. For that reason more focus is aimed on solution provided by Microsoft and MSDN (Microsoft Developer Network).

4.1.1 XAML Applications

Windows Phone XAML applications use MVVM architectural pattern for the separation of GUI (view), Domain login (model) and a ViewModel layer responsible for connection of GUI and Domain login. The View components are bounded to ViewModel and the specific function is called, when a user interaction happens. The connection works both-ways since ViewModel is able to change components of GUI or the displayed data. XAML applications are divided into pages, each page is represented with 2 files.

- **XAML file** (Extensible Application Markup Language) file is Mircosoft standard for defining User Interface. It is a file including XML-based markup language. Every page element is created with solo or pair tag with defined attributes and actions.

- **ViewModel file** stands behind the XAML file. It is written either in C# or Visual Basic programming language. It is responsible for handling user interactions with the GUI.

4.1.2 Native Development

Another option is creating a Direct3D applications, which are pure native ones suitable for writing games in C/C++ with aim on the performance. Combining XAML and Direct3D


6https://cordova.apache.org/docs/en/4.0.0/guide_support_index.md.html
is also possible, creating Mixed applications combines these two approaches - uses XAML 
application structure, but supports Windows Runtime component to be included in the 
application which uses native code in C/C++ [5].

4.1.3 JavaScript Application

The third option for creating a WP application is with using web technologies HTML5, CSS3, and JS (JavaScript). It is possible to access all the mobile hardware with Windows 
Runtime and for Windows Phone 8.1 also modern mobile controls with using of WinJS
library.[6]. JavaScript Application is structured similarly to pages as XAML Application, 
although at least 3 pages are created for one page - HTML with content and structure, CSS file with stylings and JS file for handling events and for domain logic.

4.2 Windows Phone Projects

At the moment of writing the thesis, Windows Phone 8.1 is the newest version of mobile 
operating system created by Microsoft. It is a major release update provided for all 
devices which can run Windows Phone 8. Microsoft offers number of different projects 
for targeting single or multiple devices. The whole category of development projects is 
called Store Apps including

- **Windows Apps** - this options will create a single project targeting desktop Windows 
  operating system. With this type of project the it is possible to offer the application in 
  Windows Store as a full-screen Metro application in Windows 8/8.1. In Windows 10 
  developer preview this type of applications are not forced to be full-screen anymore 
  and can be opened in its own windows on the desktop.

- **Windows Phone Apps** - this group of projects targets Windows Phone 8 and 8.1 
  operating system. These application therefore can be offered on the Windows Phone 
  Store.

- **Universal Apps** - with this type of applications, Microsoft are merging development 
  for both desktop and mobile together. This type of project will generate three projects 
  for targeting 8/8.1 versions of the operating systems. One project for desktop OS, 
  one for mobile OS and one shared project used by both previous projects. When 
  compiling, two applications are created - each targeting one type of OS.

Windows Phone applications have two more options to choose from. A Windows 
Phone project or Windows Phone Silverlight project. To understand, why there are two 
types we have to look into the past a little bit. Windows 7 and Windows Phone 7 at 
time of its creation had different API, not compatible with each other and development 
of Windows mobile application was independent of development of Windows desktop 
application. When Microsoft introducing Windows 8 and Windows Phone 8, they wan-
ted to unify the development for both platforms introducing WinRT (used in Windows 
applications) into Windows Phone application. Later in Windows 8.1, Microsoft crated 
two types of applications, a Windows Phone 8.1 application which does not used the API
of the Silverlight so more code can be shared across platforms and the second Windows Phone Silverlight 8.1 application which preserves backward compatibility with Windows Phone 8, but still can use most of the new Windows Phone 8.1 features.

4.3 Debugging

A programmer has two options how to debug the application in the developing process. First one and the most common is to use one of the Windows Phone Emulators variants provided by Microsoft, which differs in its resolution and amount of RAM. Emulator is built on Microsoft Hyper-V. That means the computer have to be equipped with 64-bit CPU that support SLAT (Second Level Address Translation) technology. [5].

The second option for debugging and running the application is using Windows Phone device. For this option, the phone has to be registered using Windows Phone Developer Registration Tool. Developers has to own a Microsoft Developer account for which he has to pay, if he is not a student, for whom the account is free of charge. A software can then be debugged on the physical device, when it is connected through USB port. Microsoft provides also a tool named Project My Screen App which allows to display and to control the physical device through the computer.

Every time a code of the application is changed and the user select it to debug the application through VS, it is removed from the device or emulator, built, installed again and run.

4.4 Deploying and Publishing

Deploying the application can be done through VS in the same way as debugging, only release option has to be selected instead of debug. The application can be deployed to both emulator or device. Standalone application for deploying an application without VS is delivered by Microsoft called Windows Phone Application Deployment. With this tool a user choose a XAP file in the Release folder of the built project. The WP Application Deployment then install the application into USB connected device, which has to be registered for development.

When the application is ready, it is possible to publish it to the Windows Phone Store through web application interface. When the application is submitted and all necessary information is filled including screenshots, application title, logo, description, version and much more, the application is submitted. It then continues to the application certification process to be approved by Microsoft. If it suffice all requirements 7, it is available in the Windows Phone Store which can take up to 7 business days, otherwise it is returned with the list of failures, which caused certification to be stopped.

---

7Windows Phone Store Policies can be found on https://msdn.microsoft.com/en-us/library/windows/apps/dn764944
5 Practical Part

For the purpose of the thesis, an Application for WP has been developed to implement the recognition algorithm in the application for learning Japanese syllables. As a classifier I chose to use the artificial neural network for its low computational and memory usage so it is suitable for mobile devices and its great ability of generalization, which is the reason I expect to have a good testing results. The design and the implementation of both, the application and the neural network, is described in this chapter.

5.1 Design of Supporting Applications

Not only the application for learning of the syllables, but several other supporting applications had to be developed for the purpose of this thesis. Names and purpose of those are listed here.

- **Pattern creator Application** - this application was never meant to be publicly available. Its goal was to draw a pattern image on the WP device, which was caught as a sequence of points and those were then uploaded to the on-line database for further using as animated image. As a data format XML was used to store the strokes and its separate points.

- **Data collector Application** - this application was developed to be installed on the WP device and thus it was made publicly available on the Windows Phone Store. Its goal was to collect data from users. User was suppose to draw a Japanese character as similar to the pattern image as possible. Data were then also exported as an image in Base64 text format to the database. As a pattern image the animated image acquired from the Pattern Creator Application was used. This data were meant to be used in the learning process for the ANN afterwards.

- **Feature Extractor Application** - for both types of data acquired from the mobile devices (patterns and learning data), a third application was developed, which is able to process images gathered by Data Collector Application. The processing was composed of necessary preprocessing and feature extraction.

- **Network Testing Application** - a console application designed for testing the precision and accuracy of the neural network.

There was one more application developed in JAVA called Image Downloader. This application allowed to download images from external on-line database, convert images from Base64 to PNG format and store them in storage on the local computer.

---

8This application is not a part of the thesis, because the database will be deleted and therefore won’t be possible to connect to it and use the program anymore.
5.2 Feature extraction of learning data

This is mainly a task of the Feature Extractor Application. The data to be process are at the beginning image with given resolution 390x280. At first step, the image is cropped to cut out blank white spaces around the character. This makes the character fit the image, but the resolution after this step varies for every image. Therefore another normalization step rescaling is performed on the cropped image. This makes the resolution unified at the value 100x100. For the rescaling, I used a Bilinear interpolation algorithm. More complex algorithm is not needed, because another loose of data will happen during zoning later on in the process. Depend on the initial size of the character, image can either increase or decrease its resolution during the interpolation. The algorithm can also deform an image (scale it on x-axis more than on y-axis, because not all characters are squared), but that is not a problem, because the learning data characters are deformed in the same way as the real data will be, when working with the application. Because there would be big amount of input data for ANN, also zoning to 100 zones is done for every image. The zone number is computed for every zone as a ration $P_B / P_T$, where $P_B$ is number of black pixels in the zone and $P_T$ total number of pixels in the zone - in this situation always 100. The very same techniques are also used for every input data, when by the learning application before delivering the input data into the ANN.

For learning, the data has to be in the CSV format to be fed into MATLAB. For this reason Feature Extractor Application takes as one of the arguments file path to the input images. The program crates two CSV files in desirable format containing one row for every image.

- `data.csv` contains all the zone data. For one row there are 100 real data values in interval $< 0; 1 >$ from left-top zone to the right-bottom zone.
- `result.csv` contains $n$ data values for each row, where $n$ is a number of output layer neurons. On every row, only single value is equal to 1, the rest is equal to 0 indicating the input was classified as the character on the specific position. Each value corresponds to one character. Characters are ordered alphabetically.

5.3 Designing the Learning Application

As a main application for this thesis is named Japanese Syllable. Its purpose is to learn the user two of Japanese script - Hiragana and Katakana. This application was divided into two parts.

First one was designed for learning the user, how to draw characters correctly. It provides both, the phonetic transcription and graphical representation of the character to the user. To the outlined space on the screen, the user is intended to write down the character to match the pattern of the character. The application then uses the recognition system to evaluate the precision of the character and shows to the user, if his representation is similar enough compared to the pattern.

The second part of the application tests user’s ability of remembering characters and his ability to draw them correctly. Test consists of several questions, were only the
phonetic transcription of the character is provided to the user and application using the recognition system again evaluate the input provided from the user and decide, if the user is correct or not. Second type of quiz was also designed to test user’s ability, but to recognize characters and matches them to the correct transcription.

Before the application was created, I created wireframes for expected pages of the software. For the design of wireframes, an on-line tool called iPlotz was used. The design can be seen on Fig. 5.1.

5.4 Designing the Artificial Neural Network

To decide, what the neural network will look like, one has to analyse the problem he has to solve. In this thesis’s problem, there are two independent alphabets (Katakana and Hiragana) which have no reason to form one ANN together, so the problem can be divided into 2 networks, which can be made and trained independently. If we analyse even further every alphabet independently, I came up with two main options how to form the network.

- To form one neural network per script
- Since there is finite number of steps at which every character has to be written and those varies from 1 to 4. We can even further divide every script to 4 ANN to decrease the complexity of the network.

Since there is no “correct” solution in designing ANN, there must be chosen the solution as good as possible, therefore I did not choose any of those two noted above, but the actual design is a compromise of those two. Two ANN per alphabet were created - first for characters written by 1 or 2 strokes, the second for characters written by 3 or 4 strokes. This solution balances two negative properties of the original designs. In the first design, the networks would be imbalanced in their complexity - there are much less characters written by 1 and 4 strokes compare to number of characters with 2 and 3 strokes e.g. only 4 characters written by 1 stroke and 24 characters written by 2 strokes in Katakana. The second design would get quite complex, because the script has relatively many characters which would make the network learning harder.

To be able to design the network there are several properties to choose - what kind of neural network we want to use to solve the problem, number of layers ANN will consist of, learning algorithm used, number of neurons in each layer and activation functions of the neurons.

When choosing properties for the neural network, only one network - the one for Hiragana with 1 and 2 strokes was tested and the properties of this particular network were therefore applied to all the networks created. All properties were discovered with Matlab software using 25% of the set as a testing data (default ratio of learning : validation : testing data is 70:15:15) so the actual training set is bigger and more accurate for the price of possible decreased precision by using less learning data.

As the activation function in the hidden layer a logistic function was used. In the output layer, a maximum function was used to determine the most-likely class, the character
Practical Part

(a) List of characters  (b) Character detail  (c) Test of writings  

(d) Usual settings page  (e) Test with options

Fig. 5.1: Wireframes of the main application
Practical Part

matches. After that, a threshold function is applied. The output values of neurons in the output layer were observe and were noticed to be between -10 and +18. Based on these observations, threshold value was used to be +9. If the output value is under below this value, the particular input is not classified as a character and the neural network returns an EMPTY character.

A feed-forward neural network was used with the learning by the back-propagation algorithm. It contains one hidden layer, therefore there is an input layer, one hidden and output layer. The input layer size is the number of zones on which every image is divided. The best results were most likely to be achieved with 100 zones in square image, meaning 10 vertical and 10 horizontal division lines. There were also other possibilities tested from 8x8 to 12x12 as can be seen in Tab. 5.1 There are 10 values meaning misclassification of the network in percentage and their average value for each possible solution and dimensions of the image division - the reason for that is that Matlab always generates new random initial weights for the network, when it is trained so the precision of the network is slightly varying every time the network is trained so only one learning attempt is not enough to discover the best precision.

<table>
<thead>
<tr>
<th>Training attempt no.</th>
<th>8x8</th>
<th>9x9</th>
<th>10x10</th>
<th>11x11</th>
<th>12x12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.47</td>
<td>5.97</td>
<td>3.3</td>
<td>5.82</td>
<td>4.47</td>
</tr>
<tr>
<td>2</td>
<td>4.62</td>
<td>3.89</td>
<td>3.43</td>
<td>3.89</td>
<td>5.37</td>
</tr>
<tr>
<td>3</td>
<td>5.52</td>
<td>4.77</td>
<td>4.77</td>
<td>4.32</td>
<td>6.56</td>
</tr>
<tr>
<td>4</td>
<td>5.52</td>
<td>3.58</td>
<td>5.07</td>
<td>4.48</td>
<td>5.22</td>
</tr>
<tr>
<td>5</td>
<td>4.92</td>
<td>5.07</td>
<td>4.03</td>
<td>4.32</td>
<td>4.47</td>
</tr>
<tr>
<td>6</td>
<td>6.26</td>
<td>4.02</td>
<td>3.6</td>
<td>4.77</td>
<td>5.67</td>
</tr>
<tr>
<td>7</td>
<td>4.33</td>
<td>4.48</td>
<td>4.5</td>
<td>4.78</td>
<td>5.33</td>
</tr>
<tr>
<td>8</td>
<td>8.2</td>
<td>3.58</td>
<td>3.43</td>
<td>5.37</td>
<td>5.37</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>3.89</td>
<td>5.67</td>
<td>2.5</td>
<td>6.26</td>
</tr>
<tr>
<td>10</td>
<td>4.47</td>
<td>4.32</td>
<td>5.22</td>
<td>4.77</td>
<td>5.67</td>
</tr>
<tr>
<td>Average</td>
<td>5.53</td>
<td>4.36</td>
<td>4.3</td>
<td>4.95</td>
<td>5.44</td>
</tr>
</tbody>
</table>

Table 5.1: Table of misclassification error in % of the NN using different number of zones of the image

Number of output neurons is given by the number of characters the network is responsible to recognize - in the example, there are 29 characters which number of strokes equal to 1 or 2.

To summarize it, number of input layers for the example network is 100, number of output neurons is 29. There exist a general rule that number of neurons in the hidden layer is between those two number. It was tested, what is the best size of the hidden layer and the results can be seen on figure no. 5.2. We can see the error rate is not changing dramatically when no. of neurons is between 50 and 70 (although these values are not perfectly precise, because the misclassification is an average of just 10 values). After this knowledge I have chosen to use a geometrical average of the number of input and output
Practical Part

Fig. 5.2: Dependence of the misclassification error on number of hidden neurons

\[ h = \text{round}(\sqrt{i + o}) \]

where \( h \) is number of hidden neurons, \( i \) is number of input neurons and \( o \) is number of output neurons. For our example network the computation is the following: \( h = \text{round}(\sqrt{100 + 29}) = 54 \).

5.5 Software Implementation

For implementation of the application, I chose C# programming language and created XAML Windows Phone 8.1 Application. Visual Studio allows to create one solution with independent Projects, which can be re-used by more other projects. To use this feature, I divided all code into 8 projects in the following way:

- **DataCollector** - Windows Phone project corresponding to the view of Data Collector application
- **DataLayer** - Portable project containing images and data for all applications
- **DomainObjects** - Windows Phone project containing classes to share some common domain functionality across applications
- **Neural network** - Portable project containing the neural networks
- **PatternCreatorApplication** - Windows Phone project corresponding to the view layer of the Data Collector application
- **WP Application** - Windows Phone project corresponding to the view layer of the main learning application
- **Feature Extractor** - Console Application for processing images
• Network testing - Console Application for testing ANN

Console application runs on the desktop, Windows Phone application on the mobile device and portable projects can be used by both.

5.6 Training Neural Network

For training the network, MATLAB software was used, more precisely the environment named Neural Network Toolbox. This allows a user to train the ANN if input and output data are provided together with the design of ANN (number of hidden layers, neurons in particular layers, activation function, distribution of learning data etc.). There are number of output values after the training process is done. All of these are available as CSV with semicolon or space as a delimiter.

- **normalization** - $n$ values, where $n$ is a number of input neurons. MATLAB expects the input values to be in range $<-1;1>$, but the values after zoning are in range $<0;1>$ so they have to be adjusted. MATLAB analysis the input data before learning and choose the normalization value for each neuron from the lowest and highest value in the input data. Generally we can say that values in range $<0;1>$ can be transformed to the range $<-1;1>$ by the function $y = 2x - 1$

- **biases** - values of the dummy Bias of each neuron which is not connected to any previous neuron

- **weights** - values of all weights of all neurons in the network.

All these values are stored in XML files in the DataLayer project, folder Data/Network_data, each file contains values for one network. Data are parsed on the program runtime, when neural networks are created. These files include also ordered list of letters, which are recognized by the particular network and size of each layer of neurons (input, hidden, output). The total number of images used for learning was 5 883.

5.7 Programming the Neural Network

It is not in my opinion very important to discuss all the programming I has to make to create the application, but I would like to show my design of neural network. The class diagram can be seen on Fig. 5.3. How the class interacts can be seen on the sequence diagram on Fig. 5.4.

When the application gathers the input data for the network first the data are passed into NetworkRepository which passes the data further on to the correct instance of Network class. All the instances of InputNeuron class are populated with data. The Network gathers the output data from instances of OutputNeuron class and returns the appropriate character name.

To create a network first the class NetworkRepository is created and it loads all network models from external file and creates all 4 instances of Network class and populate it with the data gathered from external XML file.
Fig. 5.3: Class diagram of the ANN
Fig. 5.4: Sequence diagram of computing the result by the ANN
5.8 Description of the application

The application final opening screen can be seen on Fig. 5.5a. It has 3 main options - Test, Hiragana and Katakana. First two options displays a screen of all characters of a given script as shown in Fig. 5.5b with transcriptions. If a user clicks a character from a list, its detail is displayed on Fig. 5.5c with animated pattern, which stars to be drawn automatically when the page is loaded. The surface for writing the character by the pattern is located in the bottom part of the page. When users clicks the confirmation button, the neural network analyse the character and changes the border color of the drawing surface to green or to red accordingly. Here must be noted that notification alert is shown, when the character is not drawn with the correct number of strokes. User can navigate to other characters or he can return to the previous page with list of characters using hardware phone return button.

The test of knowledge is the third option in the opening screen. When selected, a user is asked to choose settings for the test - if only Hiragana or Katakana alphabet will be tested or both. And if the user wants to test his ability to find the transcription of the given character - fig 5.5d or the user wants to test his ability to draw the character from the transcription by himself. The number of quiz questions for the first option is fixed to 20 and user can not skip questions randomly. The correct solution is displayed, if the user is wrong.

The test of transcription writing allows to switch between questions, the user has unlimited number of options of drawn the given character. 10 random questions are given and the state of each question is indicated by the background color of button associated with the question. Red color states for incorrect answer, green for the correct answer, white color for an unvisited question and gray color for question which has not been answered, yet. The question can be submitted by the button, which will not occur before all questions have been visited.

After submitting the test, an overview is shown with number of correct/wrong answers and the percentage evaluation.
Practical Part

(a) Opening screen  
(b) List of characters  
(c) Character detail  
(d) Test transcription  
(e) Test of writing characters

Fig. 5.5: Screenshots of the main application
6 Testing

As noted in the previous chapter, there was a supportive piece of software created for testing purposes of the recognition software. Also separate data set had to be used. The Network Testing console application program takes a path to the set of images categorized in folders as an input and generated output data in form of csv file. These generated output data were afterwards processed with using of MS Excel 2013, a part of Microsoft Office. The total number of testing images was 2 091.

The accuracy of both scripts seems very good. There are several reason for that - learning and training image sets were separate, but images used for testing were drawn by the same people who drawn the images of training set. For that reason people who did not participate on the learning process may have slightly lower successful rate. Second reason is that any faults or noise occurs in the images since it is gathered in the digital form directly. One last reason is the setting of threshold value to relatively low value. The value was chosen by observation and it could be increased to expect the input characters to be written more accurately. I choose slightly lower threshold so the application accepts a little bit more variations in the input images.

6.1 Katakana Testing Analysis

The final accuracy achieved on the testing data for Katakana script was 94.33%. The classification accuracy of each particular character can be seen on Fig. A.2. Also the complete script can be seen on Fig. B.2. The most difficult ones to categorized were characters A, Ma, Nu, Su because of similarities to each other. The worst successful rate had a character Nu, because it was often mistaken with not just one, but two of these characters as is shown with other often misclassified characters on the Fig. 6.1

![Fig. 6.1: Often misclassified pairs of Katakana characters](image)

Surprisingly good result can be seen when recognizing Tsu/Shi and So/N (see Fig. 6.2). The notable difference between characters Tsu and Shi is the alignment of the two smaller strokes in the top-left corner. Tsu has the strokes aligned horizontally next to each
other, forming angle to the horizontal axis less than 20 degrees. Shi has these two strokes aligned vertically next to each forming an angle less than 20 degreeed to the vertical axis. Similarly the angle rule applies for So/N.

The network is able to distinguish between So and N very well only by the direction of the small stroke, if the stroke is long enough. To distinguish Tsu from Shi correctly these two small strokes have to be shifted to right when drawing Tsu as seen on Fig. 6.2. It would not be problem for the recognition system to evaluate only based on the stroke alignment and direction also, but if we take a look on the learning data, these strokes were shifted in most of the images so the neural network learns the character this way.

Characters N and So are also sometimes misclassified to the characters Ri or Fu. We can avoid this by writing the characters longer stroke with bigger radius (more like a straight line) as seen on Fig. 6.3. If the radius of the stroke is smaller.

![Fig. 6.2: Image of Tsu/Shi and So/N characters](image1)

![Fig. 6.3: Katakana N character misclassification. Incorrect classification on the left and correct classification on the right.](image2)

### 6.2 Hiragana Testing Analysis

The final successful rate for hiragana script is 97.19%. Complete script can be seen on the Fig. B.1 and the successful classification rate on Fig. A.1. The most often misclassified pairs are shown on the Fig. 6.4.
Fig. 6.4: Often misclassified pairs of Hiragana characters
7 Conclusion

The application was successfully created and published on the Windows Phone Store\(^9\). The accuracy on the testing sets were 97.19% for Hiragana and 94.33% for Katakana script.

The recognition system included in the application performs reasonably well and the testing accuracy exceeded my expectations. Even though the recognition system could be improved in several ways. One way to improve the accuracy could be by using on-line recognition techniques - analysing the order of points i.e. direction of the strokes written to the screen. This would require additional research and analysis of the recognition algorithms and more testing data. The advantage of using the offline recognition is in the possibility of future implementing of the recognition system with some preprocessing modifications to recognize characters in captured images.

The character recognition is still a subject of development and research and combining different feature extraction methods and more classification methods together could also lead to better generalization. This thesis can be seen as a good starting point using best practices for creating and implementing recognition system with ANN, but which could search for new innovative ways of recognition which allows it to be improved in the future.

References


A Neural Network Accuracy Graphs

Fig. A.1: Accuracy of the neural network in classifying Hiragana script
Fig. A.2: Accuracy of the neural network in classifying Katakana script
## B Tables of Japanese Syllabic Scripts

<table>
<thead>
<tr>
<th>a</th>
<th>i</th>
<th>u</th>
<th>e</th>
<th>o</th>
</tr>
</thead>
<tbody>
<tr>
<td>ka</td>
<td>ki</td>
<td>ku</td>
<td>ke</td>
<td>ko</td>
</tr>
<tr>
<td>sa</td>
<td>shi</td>
<td>su</td>
<td>se</td>
<td>so</td>
</tr>
<tr>
<td>ta</td>
<td>chi</td>
<td>tsu</td>
<td>te</td>
<td>to</td>
</tr>
<tr>
<td>na</td>
<td>ni</td>
<td>nu</td>
<td>ne</td>
<td>no</td>
</tr>
<tr>
<td>ha</td>
<td>hi</td>
<td>fu</td>
<td>he</td>
<td>ho</td>
</tr>
<tr>
<td>ma</td>
<td>mi</td>
<td>mu</td>
<td>me</td>
<td>mo</td>
</tr>
<tr>
<td>ya</td>
<td>ri</td>
<td>ru</td>
<td>re</td>
<td>ro</td>
</tr>
<tr>
<td>wa</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. B.1: Table of Hiragana characters
| a | i | u | e | o |
| ka | ki | ku | ke | ko |
| sa | shi | su | se | so |
| ta | chi | tsu | te | to |
| na | ni | nu | ne | no |
| ha | hi | fu | he | ho |
| ma | mi | mu | me | mo |
| ya | yu | |
| ra | ri | ru | re | ro |
| wa | |

Fig. B.2: Table of Katakana characters