Grapheme Approach to Recognizing Letters based on Medial Representation

Anna Lipkina and Leonid Mestetskiy
Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Leninskiye Gory 1-52, Moscow, Russia

Keywords: Digital Text Image, Digital Font, Grapheme, Medial Representation, Aggregated Skeleton Graph.

Abstract: In this paper we propose a new concept of mathematical model of characters' grapheme which nowadays is not strictly formalized and a method of constructing graphemes based on the continuous medial representation of letters in digital images. We also suggest the recognition method of the printed text image on the basis of mathematical model of the grapheme used at generation of features and for classifier construction. The results of experiments confirming the efficiency of the grapheme approach, high quality of text recognition in different font variants and in different qualities of the text image are presented.

1 MOTIVATION

The concept of grapheme (Osetrova, 2006) is fundamental in writing and in reading. A literate person recognizes letters written or printed in different fonts, on paper or stone, on walls and on clothing. The basis of recognition is the schematic images of letters, which are called graphemes. The grapheme is the most general scheme of the alphabet symbol, and any literate person, even a child, can draw it. The school teaches reading and writing based on graphemes. However, text recognition software does not use this concept explicitly. Graphemes are used by philologists in their theoretical constructions, as well as designers when creating computer fonts. Both those and others do without the strict definition of the concept of grapheme. If you try to create algorithms for recognizing the characters of the alphabet based on graphemes, then you need to more strictly define this concept and the ways of its description and construction.

In this article we make such an attempt. We want to define the schematic descriptions of the characters of the alphabet so that they can be obtained from any font, and so that the letters can be recognized in all other fonts. To solve this problem, we propose a method of obtaining graphemes in the form of graphs from digital images of letters of a font and a method of recognizing characters of other fonts based on a comparison with graphemes. The main hypothesis is that to build a universal set of graphemes a single type font is enough, and the remaining fonts can be recognized by this set. Thus, the purpose of the study is to implement and test the grapheme approach for recognizing letters.

2 INTRODUCTION

When a literate person reads the text, he can immediately determine by the form of the symbol what letter this symbol depicts. He can do it regardless of the different variants of the artistic style of the symbol (with serif, italic, straight, decorative, etc. (ParaType, 2008)). That is, there is exists an "image" of the letter, which can be easily recognized by a human and easily distinguished from such "images" of other letters. This "image" is called *grapheme* (Osetrova, 2006).

Definition 2.1. *Letter* — a single character of the alphabet.

In the process of development of writing and cursive writing (Solomonik, 2017)(Zaliznyak, 2002) there are appeared multiple font styles: lowercase and capital writing, and later — different spellings of the same letter. Often these spellings can be quite different, although they denote the pronunciation of the same sound, for example: **A** and **a**. To describe these differences, the concept of grapheme is introduced:

Definition 2.2. *Grapheme* — writing unit, some graphical primitive that has the form of a geometric graph and depicts the canonical notation of a letter.

Lipkina, A. and Mestetskiy, L.
Grapheme Approach to Recognizing Letters based on Medial Representation.
DOI: 10.5220/0007366603510358
In *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2019)*, pages 351-358
ISBN: 978-989-758-354-4
Copyright © 2019 by SCITEPRESS – Science and Technology Publications, Lda. All rights reserved

Grapheme can be represented as images of letters in a thin font, for example, as in Fig. 1.

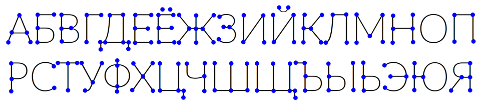


Figure 1: Images of Cyrillic letters in Lato font and vertices of geometric graphs.

Graphemes must have the following properties:

1. Any two graphemes are well distinguishable.
2. Let images I_1 and I_2 represent the same grapheme. Then the difference between I_1 and I_2 is *insignificant*. Thus, similarity is determined by some *similarity measure* between I_1 and I_2.

The concept of graphemes is introduced by designers and type designers in the verbal form, using the "general" design of the shapes of letters. That is, there is no formal definition of a grapheme. This paper proposes a mathematical description of this "general" construction (grapheme) and tests the hypothesis that such a description is sufficient to recognize letters in many different fonts.

3 THE STRUCTURE OF THE ALGORITHM OF LETTERS' CLASSIFICATION

The algorithm actually consists of two parts:

1. Construction of a mathematical model of a grapheme.
2. Development of an algorithm based on the constructed model that recognizes the letter in the image.

The idea of constructing a mathematical model of a grapheme is to construct a skeleton graph of a binary image of a letter and remove some edges from it.

The conceptual approach to the recognition algorithm is as follows: a skeleton of a binary image of a letter is built, in this graph a subgraph is searched in some way, equivalent to the standard mathematical description of the grapheme as it is similar.

We define several basic concepts.

Definition 3.1. *Figure* — set of points on the plane.

Definition 3.2. *Empty circle of the figure* — circle lying entirely in the figure.

Definition 3.3. *Inscribed empty circle of the figure* — empty circle that are not contained in no other empty circle of the figure.

Definition 3.4. *Skeletal representation of the figure* — set of centers of all inscribed empty circles of the figure (see Fig. 2).

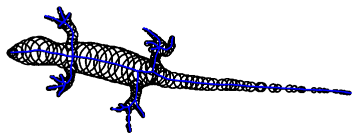


Figure 2: Skeletal representation of the figure.

In fact, the skeletal representation of a figure is a graph $\mathcal{S}$, called *skeleton (skeleton graph)* of the figure. The vertices of the graph are the centers of the inscribed empty circles having either one or three common points with the boundary of the figure, and the edges are the lines from the centers of the inscribed empty circles touching the boundary at exactly 2 points. The skeletal representation of a figure is discussed in more detail in (Mestetskiy, 2009).

Definition 3.5. *Silhouette of a skeleton graph* — a figure consisting of the union of all inscribed empty circles whose centers lie in the skeleton graph $\mathcal{S}$. Designation: $\mathcal{V}_{\mathcal{S}}$.

Definition 3.6. *Clipping of a skeleton graph (with a parameter* α*)* — the process of regularization of a skeleton graph $\mathcal{S}$ based on the removal of nonessential edges from the skeleton graph (see Fig. 3). In the process of such removal, a minimal subgraph $\mathcal{S}'$ of the original skeleton graph arises, for which $H(\mathcal{V}_{\mathcal{S}}, \mathcal{V}_{\mathcal{S}'}) \leqslant \alpha$ is executed, where $H(\mathcal{V}_{\mathcal{S}}, \mathcal{V}_{\mathcal{S}'})$ — Hausdorff distance (Hausdorff, 1965) between the silhouette of the skeleton graph $\mathcal{S}$ and the silhouette of the skeleton graph $\mathcal{S}'$.

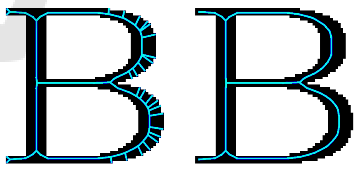


Figure 3: Example of a skeleton without a clipping (left) and with a clipping (right).

4 BUILDING A MATHEMATICAL MODEL OF A GRAPHEME

To build a mathematical model of the grapheme it is proposed to make two steps:

1. Segmentation of text images into images of individual characters (graphemes).
2. Selection of the structural description (mathematical model) of the image of each grapheme.

The second step is divided into the following steps: obtaining a skeletal graph of image letter; aggregation of a skeleton graph and processing of a skeletal graph, namely the removal of noise edges.

4.1 Obtaining a Skeleton Graph

The construction of a skeleton figure graph is described in detail in (Mestetskiy, 2009) and contains the following basic steps:

1. Approximation of the original figure F by a polygon of minimal perimeter M.
2. The construction of the Voronoi diagram of (Mestetskiy, 2009) for the vertices and the sides of the polygon M.
3. The removal of some of the segments of the Voronoi diagram.
4. The rectilinear approximation of the parabolic edges of the Voronoi diagram.

After constructing the skeleton graph, its subsequent clipping with the parameter α is performed. It is made in order to highlight the main elements of the skeleton graph, independent of minor changes in the boundaries of the symbol image.

4.2 The Aggregation of the Skeleton Graph

The resulting skeleton graph contains only the following types of vertices: vertices of degree 1 (leaves), vertices of degree 2, vertices of degree 3 (forks).

The main information about the skeletal graph are leaves and forks, as well as types of connections between them. To select these connections, the *aggregation operation of the skeleton graph is performed:* ”gluing” into one chain of all such consecutive edges whose incident vertices have degree either 1 or 2. After such ”glue” only leaves and forks are left (see Fig. 4).

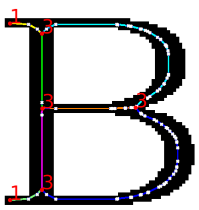


Figure 4: An example of an aggregated skeleton graph. White color marks the vertices of degree 2 of the original graph.

After aggregation, the skeleton graph become a hypergraph $\mathcal{S}_{agg,1}$ whose vertices are leaves and forks, and whose edges are selected chains.

4.3 Designations and Concepts

1. The input binary image of the symbol is considered. $\mathcal{B}$ — minimum square rectangular frame with horizontal and vertical sides, limiting the given symbol. $\mathcal{B}_H$ and $\mathcal{B}_W$ — frame height and width $\mathcal{B}$ respectively.
2. Let e — non-aggregated edge of skeletal graph $\mathcal{S}$. $v_1(e), v_2(e)$ — end vertices of this edge without taking into account any order.
3. $l(e)$ — length of edge e. It is calculated through the Euclidean distance between two points $v_1(e)$ and $v_2(e)$:
$$l(e) = \sqrt{\left(v_1(e)_x - v_2(e)_x\right)^2 + \left(v_1(e)_y - v_2(e)_y\right)^2}.$$
4. For an edge (chain) e_{agg} hypergraph $\mathcal{S}_{agg}$ through $v_1(e), v_2(e)$ the end vertices of this chain are denoted.
5. The edge e_{agg} of the designated hypergraph $\mathcal{S}_{agg}$ consists of n consecutive edges of the original graph $\mathcal{S}$ that connected into this chain e_{agg}: $\{e^1_{agg}, e^2_{agg}, \dots, e^n_{agg}\}$.
6. $l(e_{agg})$ — length of chain e_{agg}. It is calculated as the sum of the lengths of all edges included in this chain:
$$l(e_{agg}) = \sum_{i=1}^{n} l(e^i_{agg}).$$
7. $\deg v$ — degree of vertex v.

Definition 4.1. Let $d = [v_1(e_{ag}), v_2(e_{agg})]$ be given. Among all vertices of the chain e_{agg} exists the vertex v_h that the most distant from the segment d. On three points $v_1(e_g), v_2(e_g), v_h$ a circle can be formed. Then *approximating arc* is the arc of the smallest length bounded by points $v_1(e_g), v_2(e_g)$ (see Fig. 5).

Definition 4.2. *Angle of the curvature of the chain* — the central angle of its approximating arc.

Comment. In the case where three points $v_1(e_{agg}), v_2(e_{agg}), v_h$ lie on the same line or when there are no vertices in the e_{agg} chain, the curvature angle of the chain is assumed to be 0.

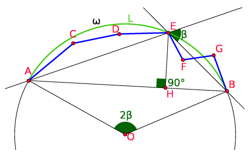
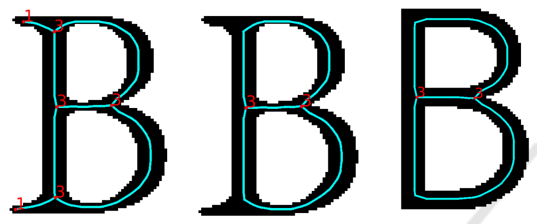


Figure 5: Example of approximating chain $[a, C, D, E, F, G, B]$ arc L and central angle BOA

4.4 Removal of Noisy Edges

After regularization and aggregation of the skeleton graph, in $\mathcal{S}_{agg,1}$ noise edges may still be contained. This is evident in the letters depicted in serif fonts (ParaType, 2008).

Serif is a kind of decoration for the letter, and their presence or absence does not prevent a person to recognize which letter is depicted. Thus, in the grapheme model, the serif letters should not be included. Therefore the next step in constructing a mathematical model of a grapheme is removing edges that are serifs from $\mathcal{S}_{agg,1}$ (see Fig. 6). Let $\mathcal{E}_S$ be the set of edges of a hypergraph $\mathcal{S}_{agg,1}$ that are serifs.



(a) (b) (c)

Figure 6: 6a: the skeleton of the letter in a serif font; 6b: the same skeleton with the removed edges from $\mathcal{E}_S$; 6c: the skeleton of letter in sans serif.

For the set $\mathcal{E}_S$ the following features can be distinguished:

1. $|\mathcal{E}_S| \geqslant 2$, that is, if the notches in the skeletal graph are present, their amount is not less than two.
2. $\forall\, e_{agg} \in \mathcal{E}_S$ the following features are typical:
   — exactly one of the vertices $\{v_1(e_{agg}), v_2(e_{agg})\}$ is leaf, and exactly one of them is a fork;
   — the length of edge $l(e_{agg})$ does not exceed some threshold $\mathcal{L}(\mathcal{B})$;
   — the central angle 2 *beta* of approximating e_{agg} arc is not less than some threshold $\mathcal{A}$.

The algorithm for removing noisy edges from $\mathcal{S}_{agg,1}$:

1. Determination of the set $\mathcal{E}_S$ based on its features.
2. Removing all edges from $\mathcal{E}_S$ from the aggregated skeleton graph $\mathcal{S}_{agg,1}$.

Since vertices of degree 2 may occur after removal, the aggregation of the skeleton is necessary to be made again.

The hypergraph obtained after edge removal and re-aggregation is denoted by $\mathcal{S}_{agg,2}$. It is the proposed mathematical model of the grapheme.

5 GRAPHEME RECOGNITION

At this stage from $S_{age,2}$ features will be allocated for the subsequent construction of the classifier graphemes.

5.1 Feature Generation

In this method it is proposed to allocate 2 types of descriptions: *top-level features* $\mathcal{F}_a$ and *bottom-level features* $\mathcal{F}_d$. They have the following properties:

— If from hypergraphs $\mathcal{S}'_{agg,2}, \mathcal{S}''_{agg,2}$ identical top-level features $\mathcal{F}'_a = \mathcal{F}''_a$ are allocated, the bottom-level features $\mathcal{F}'_d$ and $\mathcal{F}''_d$ lie in one feature space.

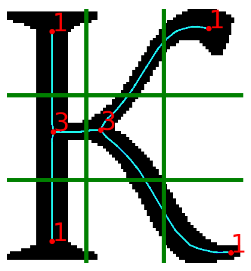
— If from hypergraphs $\mathcal{S}'_{agg,2}, \mathcal{S}''_{agg,2}$ different top-level features $\mathcal{F}'_a \neq \mathcal{F}''_a$ are allocated, then the bottom-level features $\mathcal{F}'_d$ and $\mathcal{F}''_d$ lie in different feature spaces.

5.2 Top-level Features

The idea of constructing top-level features is based on the analysis of the vertex position in the hypergraph $\mathcal{S}_{agg,2}$.

The frame $\mathcal{B}$ in which the grapheme is enclosed is divided into n equal parts by horizontal lines and m equal parts by vertical lines.

In each of the resulting $n \cdot m$ rectangles the number of leaves and the number of forks are counted, and these numbers are added to the top-level feature description. In addition as a part of top-level feature the number of connected components of the grapheme is considered (see Fig. 7).



(a)

1	**0**	**1**
0	**0**	**0**
1	**0**	**1**

(b)

0	**0**	**0**
1	**1**	**0**
0	**0**	**0**

(c)

Figure 7: 7a: skeleton $\mathcal{S}_{agg,2}$ of letter "K" and splitting the frame into 9 rectangles ($n = m = 3$); 7b: number of leaves in each rectangle; 7c: number of forks in each rectangle.

5.3 Bottom-level Features

We consider that the feature description of the top-level $\mathcal{F}_a$ is fixed. It means that the structure of the hypergraph $\mathcal{S}_{agg,2}$ is actually fixed: for each of the $n \cdot m$ partition rectangles, the number of leaves and forks that fall into it is known, and the number of edges of the hypergraph associated with each partition rectangle is also known. Thus, it is now possible to generate a fixed number of features for each of the $n \cdot m$ rectangles. The rectangles themselves are ordered from left to right and from top to bottom for certainty of the feature space.

Feature description of the lower level $\mathcal{F}_d$ is proposed to generate from the edge structure $\mathcal{S}_{agg,2}$.

5.4 Generating Features from an Edge

Let $[A,B]$ be an edge of hypergraph $\mathcal{S}_{agg,2}$. The mask of splitting this edge into k parts is fixed:

$$\mathcal{Z}_k = [z_1, z_2, \dots, z_k], \;\; z_j \in (0,1) \;\; \forall j = \overline{1,k}.$$

The starting vertex is fixed (without limiting the generality we assume that it is A). We apply the partition $\mathcal{Z}_k$ to the edge $[A,B]$ starting from the vertex A as follows: the edge $[A,B]$ is divided by k points, counting from the point A, by $k+1$ segments s_i so that:

$$\sum_{i=1}^{j} l(s_i) = z_j l([A,B]) \;\; \forall j = \overline{1,k}.$$

Let the ends of segments s_i have coordinates C_{i-1}, C_i:

$$s_i = [C_{i-1}, C_i] \;\; \forall i = \overline{1,k+1}.$$

Note that $C_0 = A$ and $C_{k+1} = B$. Also mark that $\overrightarrow{b} = \overrightarrow{AC_1}$.

It is proposed to highlight the following bottom-level features:

1. Vectors $\overrightarrow{AC_i}$, $i = \overline{1,k+1}$ are considered. Let $m_i = ||\overrightarrow{AC_i}||_2$, $i = \overline{1,k+1}$. These vectors are normalized to their lengths:

   $$\overrightarrow{c_i} = \frac{\overrightarrow{AC_i}}{m_i}, \; i = \overline{1,k+1}.$$

   As features, the coordinates of the resulting vectors $\overrightarrow{c_i}$, $i = \overline{1,k+1}$ are taken sequentially (by i).
2. Let $\overrightarrow{g} = (1,0)$. The following oriented angles are added sequentially (by i) as features:

   $$\angle(\overrightarrow{g}, \overrightarrow{AC_i}), \; i = \overline{1,k+1}.$$
3. The following oriented angles:

   $$\angle(\overrightarrow{C_iC_{i-1}}, \overrightarrow{C_iC_{i+1}}), \; i = \overline{1,k}.$$
4. The ratio of the lengths of adjacent vectors:

   $$\frac{m_i}{m_{i-1}}, \; i = \overline{2,k+1}.$$

Thus, for each edge e its bottom-level features description f_e consists of $5k+3$ elements.

5.5 Generating Features for a Single Rectangle

Let $\mathcal{R}$ be the current considered rectangular area in partition of box $\mathcal{B}$. For reasons of ordering the features, the hypergraph vertex $\mathcal{S}_{agg,2}$, caught in $\mathcal{R}$, are sorted by polar angle (in the case of equality of polar angles — in length relative to the lower left corner $\mathcal{R}$). We denote the characteristic description of the domain $\mathcal{R}$ by $f_{\mathcal{R}}$.

First, consider all the leaves, then all the forks. In all cases, the starting vertex will be the current vertex in question

1. v is a leaf. Then features f_e are generated for the corresponding edge e, and they are added to the final feature description $f_{\mathcal{R}}$.
2. v is a fork. Consider the corresponding three outgoing edges of the vector $\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_3}$. The outgoing edges are sorted in ascending order of the oriented angles $\angle(\overrightarrow{b_i}, \overrightarrow{g})$, $i = 1,2,3$, then for them features f_e are generated. The resulting features are added in the sorting order of the edges to the final feature description $f_{\mathcal{R}}$.

5.6 Feature Generation for Grapheme

Bottom-level features $\mathcal{F}_d$ for a grapheme are obtained by combining the features $f_{\mathcal{R}}$ in the order of ordering rectangular areas $\mathcal{R}$.

5.7 Classifier Training

Now, within each attribute of the top-level feature $\mathcal{F}_a$, it is possible to train its classifier — each on its own feature space corresponding to its feature space of the bottom-level $\mathcal{F}_d$.

At the training stage, a labeled training dataset $(\mathbb{X}_{tr}, \mathbb{Y}_{tr})$ is taken, where $x \in \mathbb{X}_{tr}$ — binarized symbol image, $y \in \mathbb{Y}_{tr}$ — corresponding image class.

The learning algorithm consists of the following steps:

1. For the entire training dataset $(\mathbb{X}_{tr}, \mathbb{Y}_{tr})$ select the top-level features and use them to construct a classification dictionary $\mathcal{D}$.

2. For each unique top-level feature $\mathcal{F}_a$, select the objects that have this feature $\mathcal{F}_a$. For each of these objects construct a bottom-level feature $\mathcal{F}_d$. As the result a new subsample of objects from the $\mathcal{F}_d$ feature space is selected, with which the classifier is trained (see Fig. 8).

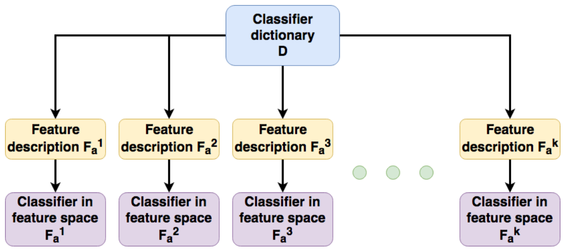

Classifier dictionary D
Feature description Fa1
Feature description Fa2
Feature description Fa3
Feature description Fak
Classifier in feature space Fa1
Classifier in feature space Fa2
Classifier in feature space Fa3
Classifier in feature space Fak

Figure 8: Classification dictionary $\mathcal{D}$ structure.

5.8 Classification Algorithm

Let we have a new object x (binary image of a single character), and it must be classified. To classify it, the following steps are needed:

1. The selection of a mathematical model of grapheme $\mathcal{S}_{agg,2}$ from x.
2. Building top-level features $\mathcal{F}_a$ from $\mathcal{S}_{agg,2}$.
3. Check if $\mathcal{F}_a$ in the classification dictionary $\mathcal{D}$, that is obtained at the training stage. If the feature is not present, the operation of postprocessing of the skeleton graph is performed. If it is present then skip to the next step.
4. The construction of bottom-level feature $\mathcal{F}_d$, the application to it of the corresponding trained classifier and receiving a response.

The idea of post-processing is as follows: continuation of the search of the subgraph, which may be classified according to the trained classification dictionary $\mathcal{D}$. If such a graph was not found, the classification rejection will be returned

The final algorithm can be seen in Fig. 9.

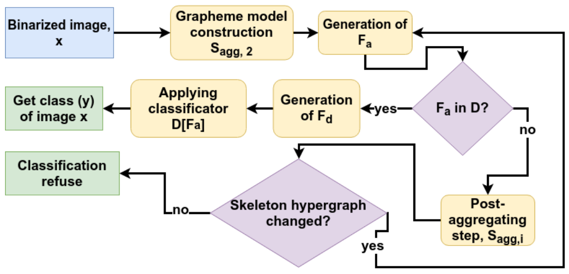

Binarized image, x
Grapheme model construction Sagg, 2
Generation of Fa
Fa in D?
yes
no
Generation of Fd
Applying classificator D[Fa]
Get class (y) of image x
Classification refuse
no
Skeleton hypergraph changed?
yes
Post-aggregating step, Sagg,i

Figure 9: Binarized image classification algorithm.

5.9 Quality Metric

Classification accuracy is used as a quality metric.

Let a be a classification algorithm, $(\mathbb{X}_{te}, \mathbb{Y}_{te})$ — test sample, $|\mathbb{X}_{te}| = n_{te}$, $\mathbb{X}^i_{te}$ — i-th test sample object, $\mathbb{Y}^i_{te}$ — its true class. Then the classification accuracy is calculated from the test sample according to the following formula:

$$Q(a, (\mathbb{X}_{te}, \mathbb{Y}_{te})) = \frac{1}{n_{te}} \sum_{i=1}^{n_{te}} \mathbb{I}[\mathbb{Y}^i_{te} = a(\mathbb{X}^i_{te})].$$

6 COMPUTATIONAL EXPERIMENTS

6.1 Training Dataset

For constructing a training dataset 88 different fonts were selected, 33 letters of the Russian alphabet in lowercase and uppercase versions (that is, only 66 graphemes) in three font sizes were generated from each: 30, 50, 100 pixels. Image generation was performed without smoothing, that is, immediately in binary format. The training sample size $(\mathbb{X}_{tr}, \mathbb{Y}_{tr})$ is $n_{tr} = 17424$ binarized letter images. As a true class $\mathbb{Y}^i_{tr}$ for the object $\mathbb{X}^i_{tr}$ of the training dataset the letter *in lowercase* was taken.

6.2 Parameters of the Proposed Algorithm

1. Parameter of clipping is $\alpha = 0.06 \cdot \mathcal{B}_H$.
2. Threshold for trimming by length:
$$\mathcal{L}(\mathcal{B}) = \frac{2}{7}\max(\mathcal{B}_H, \mathcal{B}_W).$$
3. The trimming threshold for length at the post-processing stage increases in 1.8 times:
$$\mathcal{L}'(\mathcal{B}) = 1.8 \cdot \mathcal{L}(\mathcal{B}) \quad (\kappa = 1.8).$$
4. The trimming threshold for angle:
$$\mathcal{A} = \frac{\pi}{5}.$$
5. At the stage of extraction of top-level features $n = m = 3$ is supposed.
6. The fixed grid is assumed to be equal to:
$$\mathcal{Z}_8 = \left[\frac{1}{50}, \frac{1}{5}, \frac{1}{3}, \frac{2}{5}, \frac{1}{2}, \frac{3}{5}, \frac{2}{3}, \frac{4}{5}\right].$$
7. As classifiers at bottom-level is considered *Random forest* (Ho, 1995).

6.3 Basic Algorithm

As the base algorithm (baseline) has been selected convolutional neural network (CNN) (LeCun et al., 1998)(Bishop, 2006), the architecture of which is shown in Fig. 10:

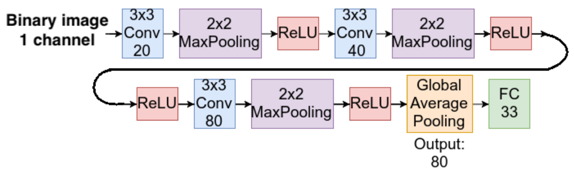


Figure 10: Neural network architecture.

Decoding of designations:

- $k \times k$ Conv f — convolution layer with kernel of size $k \times k$ and f output filters (channels);
- $k \times k$ MaxPooling — max-pooling layer with kernel of size $k \times k$;
- ReLU — ReLU layer (Glorot et al., 2011), (Jarrett et al., 2009);
- Global Average Pooling — global average-pooling layer (Lin et al., 2013);
- FC (Fully Connected) m — a fully connected layer with an output layer of m neurons (Bishop, 2006).

For reasons of solving the classification problem over the output layer $(x^1, x^2, \dots, x^{33})$, softmax-activation is performed from 33 neurons:

$$y^j = \text{softmax}(x^j) = \frac{e^{x^j}}{\sum\limits_{j=1}^{33} e^{x^j}}, \quad j = \overline{1,33}.$$

Let C be a number of classes in the classification problem. Cross-entropy is taken here as the optimized loss function.

$$\mathcal{L}(y_i, \hat{y}_i) = -\sum_{j=1}^{C} y_i^j \log \hat{y}_i^j$$

$$\mathcal{L}(\mathbb{Y}_{tr}, \hat{y}) = \frac{1}{n_{tr}} \sum_{i=1}^{n_{tr}} \mathcal{L}(y_i, \hat{y}_i),$$

where $\hat{y}_i \in [0,1]^C$ — prediction of the network on i-th object, $\hat{y}$ — prediction of the network on the entire training dataset, y_i — vector describing the observed value: $y_i \in [0,1]^C$, $\sum_{j=1}^{C} y_i^j = 1$ and if i-th object has class j (i.e. $\mathbb{Y}_{tr}^i = j$) then $y_i^j = 1$.
Remark. Since the input images can be of different sizes, at the training stage the data in the neural network was supplied by a batch consisting of 1 image.

6.4 Experiment 1

As a test dataset, the same 88 fonts that were used in the training were taken, but a different font size, which is 80 pixels. So $n_{te} = 5800$. Images of letters are generated using the program, that is, high-quality images, without noise and binarized. The results of two methods (structural analysis (SA) is the recognition method described in the article) are presented in the table 1:

Table 1: The results of the two methods.

	SA	**CNN**
Quality, Q	0.99689	**0.99862**
Refusal rate	0.00086	0

6.5 Experiment 2

The test dataset consists of 50 fonts that were not used when learning (FontsDatabase, 2018). The font size is 80 pixels, $n_{te} = 3300$. Images of letters are generated using the program. The results are presented in the table 2:

Table 2: The results of the two methods.

	SA	**CNN**
Quality, Q	**0.97**	0.96515
Refusal rate	0.01364	0

6.6 Experiment 3

The test dataset consists of the same 50 fonts as in the previous 6.5 experiment, and the same size. First, the document is generated (.doc) with all the letters from the test sample, then this document is converted into a .png image with a resolution of 300 dpi. The images from the RGB color representation were converted to gray tones Y by the formula:

$$Y = 0.299R + 0.587G + 0.114B.$$

The images were then binarized using the Otsu method (Otsu, 1979).
The results are presented in the table 3:

Table 3: The results of the two methods.

	SA	**CNN**
Quality, Q	**0.94818**	0.94454
Refusal rate	0.01485	0

6.7 Experiment 4

In this experiment, 18 sampled fonts from 50 fonts of the 6.5 experiment are taken as a test dataset. The font size is assumed to be 80 pixels, $n_{te} = 1188$. The document is generated (.doc) with all the letters from the test sample, then this document is printed. Then the obtained samples are scanned with a resolution of 300 dpi. That is, the images are of lower quality than in the previous case (see Fig. 11).

Figure 11: Example letter from the input image

The results are presented in the table 4:

Table 4: The results of the two methods.

	SA	**CNN**
Quality, Q	**0.95538**	0.94696
Refusal rate	0.01263	0

6.8 Analysis of Experiments

The experiments show that: the quality of the proposed method is not worse than the selected basic algorithm and it has a small proportion of refuses from the classification, which increases with the deterioration of image quality.

7 CONCLUSIONS

This paper proposes a formalization of the concept of ”grapheme”, namely a mathematical model of grapheme.

On the basis of this model, a method of generating features used for the subsequent construction of the algorithm of classification of images of letters is proposed (that is, the measure of similarity between mathematical models of graphs is determined). Also in this article the algorithm of recognition of the text on the image is proposed.

The advantages of the proposed letters recognition method: independence from the size, type of font and type of lettering; allocation of the general structure (mathematical model of grapheme) for letters, which is enough to recognize letters in new fonts; interpretability of features.

The disadvantages of the method: the presence of refuses of classification and the dependence of the recognition quality from the quality of the binarization of the image.

The experiments confirm that the proposed mathematical model of the grapheme has shown its efficiency.

The objectives of further research are:

1. Improvement of top-level and bottom-level features.
2. Solution to the problem of classification refuses.
3. Modification of the iterative part (postprocessing) of the classification algorithm.

ACKNOWLEDGEMENTS

The work was funded by Russian Foundation of Basic Research grant No. 17- 01-00917.

REFERENCES

Bishop, C. (2006). *Pattern recognition and machine learning*. Springer.

FontsDatabase (2018). *https://www.fontsquirrel.com/*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323.

Hausdorff, F. (1965). Grundzüge der mengenlehre (reprint; originally published in leipzig in 1914). *Chelsea, New York*.

Ho, T. K. (1995). Random decision forests. In *Document analysis and recognition, 1995., proceedings of the third international conference on*, volume 1, pages 278–282. IEEE.

Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al. (2009). What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

Mestetskiy, L. M. (2009). *Continuous morphology of binary images: figures, skeletons, circulars (In Russian)*. FIZMATLIT.

Osetrova, O. V. (2006). *Semiotics of the font (In Russian)*. Bulletin of Voronezh state University. Series:Philology. Journalism.

Otsu, N. (1979). A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66.

ParaType (2008). *Digital Fonts (In Russian)*. ParaType.

Solomonik, A. (2017). *About language and languages (In Russian)*. Publishing House ’Sputnik+’.

Zaliznyak, A. A. (2002). *Russian nominal inflection by application of selected works on modern Russian language and General linguistics (In Russian)*. languages of Slavic culture.