# Comparisons of two approaches of pattern recognition for text detection

Ewa Lis[a], Roman Kluger[a]

[a] *Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44-100 Gliwice*

Abstract

The paper describes two methods of pattern recognition, one based on soft sets, another based on neural networks. Soft set theory is quite recently developed method of AI. That approach has rather simple mathematical background, but can perform satisfactory. Widespread neural network approach demands much more calculating power to perform at the same level of accuracy. Some hints to make neural network perform better have been written. Authors explain necessary theoretical terms and definitions then describe their idea of two AI systems. At the end some results of first system are presented.

**Keywords**

Text detection, Pattern recognition, Soft reasoning, Simple soft classifier, Weighted soft classifier, Weighted mean soft classifier

## 1. Introduction

Pattern recognition is a widespread problem in modern applications. It has many solutions which require usage of different kinds of neural networks, some image preprocessing [1] and plenty of another techniques. We stated our problem in such way: given photo or scan of some text (handwritten or printed) and photo of some word or letter (pattern) find all occurences of pattern. Our first solution was to use soft sets technique. That gave rather satisfactory results. Another approach was to use neural network classifier, but it failed due to problems highlighted in this article. Our solution needs some optimization, but it is a good starting point for further research.

Our first goal was to recognize names in historical handwritten documents. But we had difficulties in finding scans of these with sufficient quality of handwriting. There were also problems with binarization of such documents. The problem of historical document binarization is discussed in works: [2, 3, 4, 5, 6, 7]. Due to aforementioned problems we decided to simplify our task and use scans of books or our notes. These scans proved to be much easier to binarize and helped to concentrate on algorithmical part of our project.

### 1.1. Related works

There are many possible variations of intelligent systems which are trained to detect patterns. In [8] was presented a system developed to detect patterns of human voice in spectrograms. Some patterns are also detected in medical informatics, where we can search for disorders of patients [9]. In [10] was discussed how to develop a rule based system for detection of nodules in rtg lung images. Recently research aspects are also oriented on using soft sets approach. In [11] was proposed to mix neural networks with soft sets as detectors of patterns in images from various places.

Pattern detection in writing style is a complex problem. There are several approaches. In [12] was implemented transverse sequence detection. In [13] was given a discussion how instance segmentation of images can influence the efficiency of correct recognition. In [14] was proposed to use rectified attention method for text recognition.

In our model we have developed simplified, however efficient mechanism. The idea is developed on soft sets approach. We have defined a relation table which is used to compare symbols and therefore decide which of them match the pattern. Our experiments show that such idea is efficient both for handwritten and printed texts.

## 2. Mathematical part of the algorithm

### 2.1. Soft sets - introduction

Soft set theory is one of recently developed ideas. That is quite surprising due to its mathematical simplicity

*SYSTEM 2020: Symposium for Young Scientists in Technology, Engineering and Mathematics, Online, May 20 2020*

ewalis343@student.polsl.pl (E. Lis);
romaklu253@student.polsl.pl (R. Kluger)

and, as we shall see, powerful performance. First results have been published by Russian scholar Dmitri Molodotsov in his paper [15] in 1999. His idea was to simulate the uncertainty of membership in given set.

**Definition 1 (Soft set).** *Let $\mathbb{U}$ be the universe and $\mathbb{E}$ be the set of parameters describing elements of $\mathbb{U}$. A soft set is an ordered pair $(\mathcal{F}, \mathcal{A})$, where $\mathcal{A} \subseteq \mathbb{U}$ and $\mathcal{F} : \mathcal{A} \longrightarrow \mathcal{P}(\mathbb{U})$. By $\mathcal{P}(\mathbb{U})$ we denote the power set of $\mathbb{U}$. We shall reference to $\mathcal{F}$ as membership function.*

As we can see from the definition (1) the name *soft* originates from parametric description of membership. It can be clearly seen, that classical set considered in the set theory are special cases of soft sets. Its membership function is its indicator function.

**Definition 2 (Soft subset).** *Let $\mathbb{U}$ be the universe and $(\mathcal{F}, \mathcal{A})$, $(\mathcal{G}, \mathcal{B})$ be soft sets specified in $\mathbb{U}$. $(\mathcal{F}, \mathcal{A})$ is a soft subset of $(\mathcal{G}, \mathcal{B})$ if*

- *$\mathcal{A} \subseteq \mathcal{B}$ in classical sense,*
- *$\forall e \in \mathcal{A} \ : \ \mathcal{F}(e) = \mathcal{G}(e)$.*

We can also define soft set in a relation form. That representation is very useful in further applications.

**Definition 3 (Soft set in relation form).** *Let $(\mathcal{F}, \mathcal{A})$ be soft set in given universe $\mathbb{U}$. Let*

$$\mathcal{R}_{\mathcal{A}} = \{(u, e) \ : \ e \in \mathcal{A}, \ u = \mathcal{F}(e)\}. \tag{1}$$

*$\mathcal{R}_{\mathcal{A}}$ is a soft set $(\mathcal{F}, \mathcal{A})$ in relation form.*

Now we can consider Cartesian product of $\mathcal{A}$ and image of $\mathcal{F}$ (denote as $\mathcal{IM}(\mathcal{F})$). It can be seen that:

$$\mathcal{R}_{\mathcal{A}} \subseteq \mathcal{A} \times \mathcal{IM}(\mathcal{F}). \tag{2}$$

Using relation (2) we can express the soft set $(\mathcal{F}, \mathcal{A})$ in language of classical set theory. Especially, we can define an indicator function of that soft set.

**Definition 4 (Indicator function of a soft set).** *Let $\mathcal{R}_{\mathcal{A}}$ be a soft set in relation form corresponding to soft set $(\mathcal{F}, \mathcal{A})$ and $(u, e) \in \mathcal{A} \times \mathcal{IM}(\mathcal{F})$. The indicator function of the soft set $(\mathcal{F}, \mathcal{A})$ is given by following formula:*

$$\chi_{\mathcal{R}_{\mathcal{A}}}((u, e)) = \begin{cases} 1, \text{ when } (u, e) \in \mathcal{R}_{\mathcal{A}}, \\ 0, \text{ otherwise.} \end{cases} \tag{3}$$

Using function $\chi_{\mathcal{R}_{\mathcal{A}}}((u, e))$ a special matrix called binary relation table can be constructed. Its dimensions are $m \times n$, where $m$ denotes cardinality of the set $\mathcal{A}$ and $n$ denotes cardinality of $\mathcal{IM}(\mathcal{F})$. That matrix $\mathcal{M}$ is given by a formula:

$$\mathcal{M} = [a_{i,j}]_{m \times n} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}, \tag{4}$$

where $a_{i,j} = \chi_{\mathcal{R}_{\mathcal{A}}}((u_i, e_j))$. More definitions and theoretical introduction can be found in [16]. Our system will depend only on aforementioned terms.

## 2.2. Soft reasoning

The process of making decisions using soft sets will be called soft reasoning. The classical approach to soft reasoning can be divided into several steps:

1. choose your universe $\mathbb{U}$ of objects and set $\mathcal{A} \subseteq \mathbb{U}$,
2. choose parameters which describe your objects and construct set $\mathbb{E}$,
3. construct membership function of every $u \in \mathbb{U}$,
4. choose set $\mathbb{D}$ which will be called set of demands,
5. for every object $u \in \mathcal{A}$ calculate the value of classifier,
6. choose the best options using results of classifier.

First of all we shall discuss step 4. A demand $d$ will be a vector of length the same as the cardinality of $\mathbb{E}$. Elements of $d$ will be real numbers from interval $[0, 1]$. Each of the elements will correspond to priority of each of parameters from set $\mathbb{E}$.

Now, let us consider step 5. A soft classifier is a function which takes vector of parameters of an object and vector of demands and returns a numeric value [17]. We shall discuss several kinds of soft classifiers in next section.

Lastly, we shall present a few notes about step 6. The determination of number of the best options can depend on the specific problem. For example, sometimes we can wish to choose the best option, at some other time we wish to find all options better than given threshold value.

## 2.3. Soft classifiers

We shall discuss three kinds of soft classifiers:

- simple soft classifier (SSC),
- weighted soft classifier (WSC),
- weighted mean soft classifier (WMSC).

For SSC we simply need binary vector of demands $d$ - its values are only 0 or 1. For readability purposes we will write that kind of vector by naming those parameters which have value 1. Let $n$ be the length of vector of demands $d$. This classifier $d$ and object $u$ is given by a formula:

$$SSC(d, u) = \sum_{i=1}^{n} d_i u_i. \tag{5}$$

WSC does not demand binarity of vector of demand. We will use weighted sum in our calculations:

$$WSC(d, u) = \sum_{i=1}^{n} d_i u_i. \tag{6}$$

WMSC enables us to set convenient threshold value for our classification. It is given by a formula:

$$WMSC(d, u) = \frac{\sum_{i=1}^{n} d_i u_i}{\sum_{i=1}^{n} d_i}. \tag{7}$$

## 2.4. SSC - example

Let $\mathbb{U}$ be the set of goods in grocery store and $\mathcal{A} \subseteq \mathbb{U}$ such that:

$$\mathcal{A} = \{apple, orange, pepper, spinach, tomato, potato\}. \tag{8}$$

Now we can choose set of interesting parameters:

$$\mathbb{E} = \{fresh, frozen, hot, sweet, green, red, local, tropical, leafy, tuber\}.$$

Now we can construct a binary relation table using standard $\chi$ function. That table is presented in table (3).

Now consider three binary vectors of demands:

- $A = \{fresh, hot, red\}$,
- $B = \{frozen, green, sweet, leafy\}$,
- $C = \{fresh, green, red, sweet\}$.

We calculate the values of the SSC which are presented in table (1). Now we can choose goods with highest value of SSC for every demand:

- A - pepper,
- B - spinach,
- C - apple.

**Table 1**
SSC - results

| | A | B | C |
|---|---|---|---|
| apple | 1 | 2 | **3** |
| orange | 1 | 1 | 2 |
| pepper | 3 | 0 | 2 |
| spinach | 0 | 3 | 1 |
| tomato | 2 | 0 | 2 |
| potato | 1 | 0 | 1 |

**Table 2**
WSC - results

| | A | B |
|---|---|---|
| apple | 1.1 | 0.4 |
| orange | 0.6 | 0.4 |
| pepper | 0.9 | 0.6 |
| spinach | 2.1 | 0.8 |
| tomato | 0.7 | 0 |
| potato | 0.4 | 0.7 |

## 2.5. WSC - example

Let $\mathbb{U}$, $\mathcal{A}$ be defined as in previous example. Now, we shall construct non-binary vectors of demands (we shall also omit zero-valued elements):

- $\{green - 0.7, red - 0.3, frozen - 1, local - 0.4, tropical - 0.6\}$
- $\{hot - 0.6, sweet - 0.4, frozen - 0.5, leafy - 0.3, tuber - 0.7\}$

Values of WSC are presented in table (2). As we can see the best option for each of demands is spinach. It can be clearly seen that SSC is special case of WSC, but WSC gives much more precise results.

## 2.6. Neural networks - model of neuron

To analyze neural network architecture we have to start with the definition of a McCulloch-Pitts's neuron model. It is an attempt to simulation of real neurons known from neurobiology. Three main parts can be separated in that model:

- input signals,
- activation function,
- output signal.

Input signals come to neuron through synapses which are connections with other neurons. Let us suppose, that we have $n$ synapses in given neuron. Signal of $i$-th synapse shall be referenced to as $s_i$ Each of synapses has special number called weight. We shall denote $w_i$ as weight assigned to $i$-th synapse. A weighted sum of values of input signals is calculated:

$$\mathcal{S} = \sum_{i=1}^{n} s_i w_i. \tag{9}$$

Every neuron can also have a special number $b$ called bias. That means a fixed treshold value of each neuron. It is added to weighted sum $\mathcal{S}$:

$$\hat{\mathcal{S}} = \mathcal{S} + b. \tag{10}$$

Now, special function $f(\hat{\mathcal{S}})$ called activation function is calculated. Result of that calculation is assigned to output signal $s$ of the neuron. There exist plenty of possible activation functions. One of the simplest is threshold function:

$$f_1(\hat{\mathcal{S}}) = \begin{cases} 1 \text{ when } \hat{\mathcal{S}} > a, \\ 0 \text{ otherwise} \end{cases}. \tag{11}$$

In further applications differentiability of function $f(\hat{\mathcal{S}})$ is demanded. Very popular activation function is called sigmoid function:

$$f_2(\hat{\mathcal{S}}) = \frac{e^x}{e^x + 1}. \tag{12}$$

Another widespread activation function is hyperbolic tangent:

$$f_3(\hat{\mathcal{S}}) = \tanh\left(\alpha\hat{\mathcal{S}}\right) = \frac{e^{\alpha x} - e^{-\alpha x}}{e^{\alpha x} + e^{-\alpha x}}. \tag{13}$$

We can observe analogies and similarities between single neuron and soft classifiers. The neuron can be considered as a little more sophisticated soft classifier.

## 2.7. Neural networks - architecture

A neural network is set of interconnected neurons. Neurons are organized into layers, every neuron of next layer is connected to every neuron in previous layer. We define three categories of layers:

- input layer,
- hidden layers,
- output layer.

Dimension of input layer has to be equal to number of parameters describing given category of objects. It is first layer in the network, so neurons are only connected to the next layer. For example if we want to classify pictures of dimension $28 \times 28$ pixels we have to use $28 \times 28 = 784$ input neurons. There are no set rules about optimal number and dimension of hidden layer, but generally more hidden layers cause better performance of neural network.

Those numbers have to be tailored for each task separately. There will be $k$ neurons on output layer where $k$ corresponds to number of classes in our system. For binary classification 1 neuron will be sufficient, but for 10 classes we will need 10 neurons on output layer.

## 2.8. Neural reasoning

To make a decision using neural network we have to feed forward input signal. We calculate output of each of layer and the output of the last layer is decision of the network. Formula for output of $i$-th neuron on $j$-th layer is as follows:

$$f(s_{i,j}) = f\left(\sum_{k=1}^{K}(w_{k,i}^{j} s_{k,j-1}) + b_{i,j}\right), \tag{14}$$

where:

- $K$ is number of neurons on $j-1$-st layer.
- $w_{k,i}^{j}$ is weight of connection between $k$-th neuron on $(j-1)$-th layer and $i$-th neuron on $j$-th layer,
- $b_{i,j}$ is bias of $i$-th neuron on $j$-th layer,
- $f$ is activation function.

Formula (14) has to be applied on every layer starting with first hidden layer. After several steps we get the output of the network. That process, however, requires well suited weights. Due to complexity of architecture there can be thousands of weights to adjust which task is unbearable to do manually.

A special algorithm called back-propagation of error was developed to improve performance of neural network. Process of adjusting the weights is called training or learning of the network.

## 2.9. Back-propagation algorithm

Differentiability of activation function shall be used in that algorithm. We shall start with randomly chosen weights and biases. We shall compare desired output

**Table 3**
SSC - binary relation table

| | fresh | frozen | hot | sweet | green | red | local | tropical | leafy | tuber |
|---|---|---|---|---|---|---|---|---|---|---|
| apple | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| orange | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| pepper | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| spinach | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| tomato | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| potato | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

$d$ of the network with actual output $a$ of the network. We have to define an mean square error function:

$$MSE(a, d) = \sum_{i=1}^{N} \frac{1}{2}(a_i - d_i)^2 \tag{15}$$

Let us consider how much $MSE(a, d)$ is being influenced by each of weights and biases. That influence can be expressed as partial derivative with respect to given parameter $p$:

$$\frac{\partial MSE(a, d)}{\partial p}. \tag{16}$$

We have to apply chain rule to calculate the derivative (16). That is the reason of demanding differentiability of activation function. Now we have to define the learning rate $\eta \in (0, 1)$. That coefficient will tell us how fast we want to correct parameters in the network. We can formulate correction equations:

$$w = w - \eta \frac{\partial MSE(a, d)}{\partial w}, \tag{17}$$

$$b = b - \eta \frac{\partial MSE(a, d)}{\partial b}, \tag{18}$$

where $w$ and $b$ are given weights and biases respectively. After several steps we can significantly minimize error of the network. Neural classifier can be applied only after training to give acceptable results. As we can see, large networks demand a lot of calculations to be trained, because gradients (16) has to be calculated for each of multiple parameters.

# 3. Description of proposed system

## 3.1. Soft classifier

Our system takes two pictures - one large which will be referenced as page and one small, which will be referenced as pattern. Then page is converted to gray-scale with value:

$$grayscale(x, y) = \frac{red(x, y) + green(x, y) + blue(x, y)}{3} \tag{19}$$

where $red(x, y)$, $green(x, y)$, $blue(x, y)$ correspond to RGB value of pixel with coordinates $(x, y)$. Let:

$$sum(x, y) = \sum_{i=-1}^{1} \left( \sum_{j=-1}^{1} grayscale(x + i, y + j) \right), \tag{20}$$

and:

$$surr(x, y) = sum(x, y) - grayscale(x, y). \tag{21}$$

After converting to grayscale image is binarized using the formula:

$$f(x, y) = \begin{cases} 1 \text{ when } grayscale(x, y) < \frac{surr(x,y)}{8}, \\ 0 \text{ otherwise.} \end{cases} \tag{22}$$

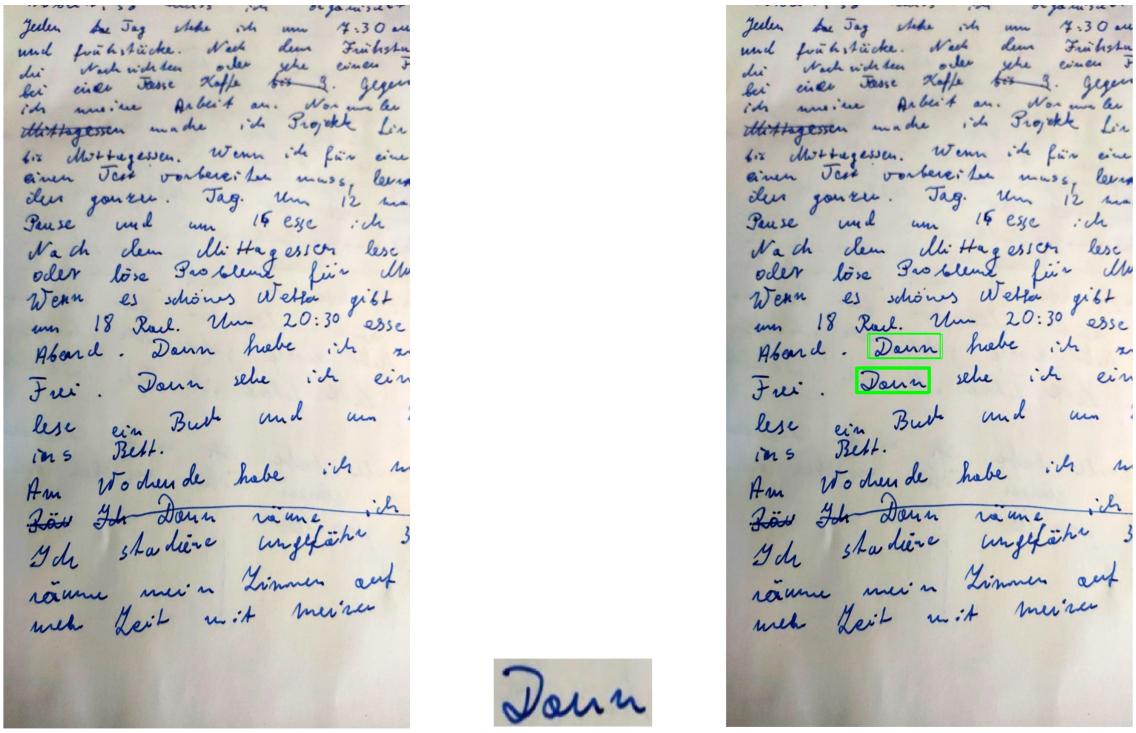
Then the pattern is converted to gray-scale using the same function and normalized into interval $[0, 1]$ using following formula:

$$g(x, y) = \frac{255 - grayscale(x, y)}{255}. \tag{23}$$

Gray-scale is reversed, that operation enables us to treat dark points as having higher values.

Now binarized page is divided into segments of size of pattern. Then both, pattern and segment of page are converted into one dimensional vectors. After repeating whole process for every possible segment of page we have our universe $\mathbb{U}$ consisting of segments $u$. For every $u$ we have our membership function and we can build binary relation table for them. Then we take vector built from pattern and check which of the segments meets the demands expressed by number from interval $[0, 1]$.

Simple soft classifier or weighted soft classifier do not work well in our case. That happens because they



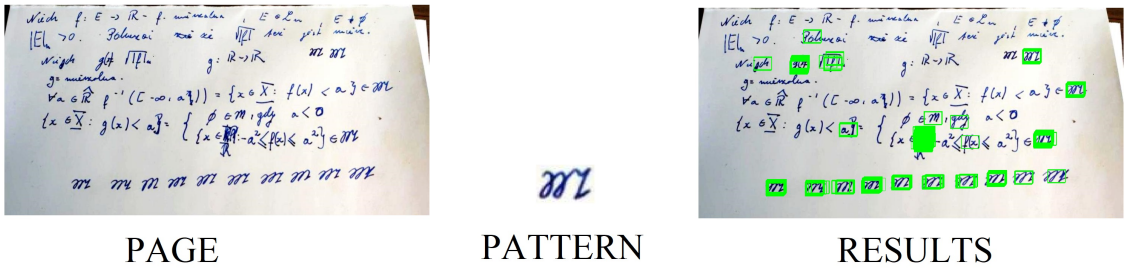




**Figure 1:** Handwriting recognition

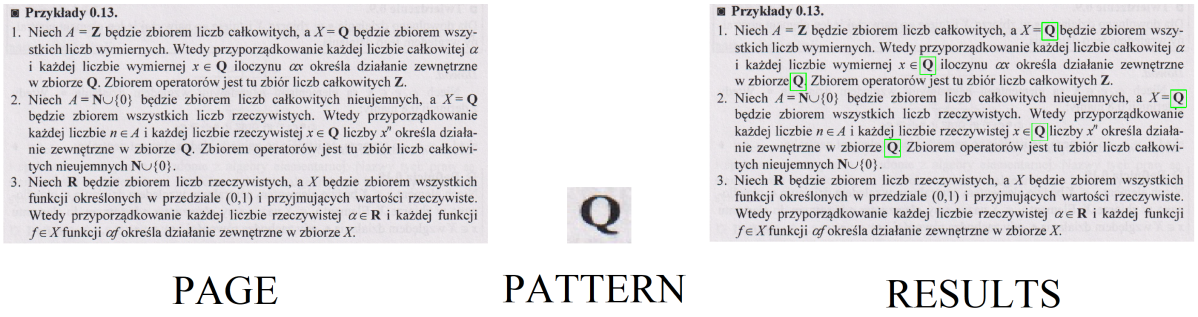



**Figure 2:** Printed text recognition

**Table 4**
Threshold values for WMSC

| | |
|---|---|
| single handwritten symbol | 0.45 |
| handwritten word | 0.4 |
| printed symbol | 0.6 |

demand knowledge about number of patterns existing in our page or calculating threshold values for every pattern. After taking these objectives into account we decided to use WMSC. After several trials we have determined sufficient values for aforementioned threshold. Results are presented in table (4).

After calculation every segment with WMSC value greater than specified treshold value is highlighted and final result is saved.

### 3.2. Neural classifier

We propose also a neural classifier for that task. We wanted to train our classifier on MNIST dataset of handwritten digits to recognize similar patterns and then apply that to our binarized page and pattern. Our idea was to build a neural network with $2 \times 28 \times 28 = 1568$ inputs and one binary output telling us whether images are similar or not. After training it had to go through whole segment of page and find similarities with pattern.

## 4. Experiments

### 4.1. Soft classifier

After some trials and errors we determined satisfying threshold values for WMSC. We prepared test set for our classifier. It consisted of pairs (page, pattern). We manually included type of documents and run the program. Then we also manually checked the behavior the classifier. Results are presented on following figures:

- searching for printed sign - fig. 2,
- searching for handwritten word or letter - fig. 1.

Threshold values determined experimentally are shown in table 4.

### 4.2. Neural classifier

As it was mentioned earlier we constructed few architectures of neural network and tried to train them on MNIST dataset. All of the attempts unfortunately failed - network learned to return the same answers rather than recognize patterns. After some research we found main disadvantages of our system:

- too shallow architecture,
- too low computing power accessible,
- too large training batches.

Keeping that in mind we can construct classifiers with better performance in the future.

## 5. Conclusions

As we can see, soft classifiers are quite well performing in given task. We plan to develop method of determination of threshold values of soft classifier and use them to more demanding tasks. We want also to improve training of neural classifier. To achieve that goal we will have to optimize code for training algorithm to perform calculations faster. That will allow us to explore deeper architectures of neural networks and then find sufficient number of layers to our task.

## References


[1] G. Capizzi, S. Coco, G. Lo Sciuto, C. Napoli, A new iterative fir filter design approach using a gaussian approximation, IEEE Signal Processing Letters 25 (2018) 1615–1619.

[2] C. Mello, A. Oliveira, A. Sanchez, Historical document image binarization., volume 1, 2008, pp. 108–113.

[3] C. Napoli, G. Pappalardo, E. Tramontana, An agent-driven semantical identifier using radial basis neural networks and reinforcement learning, arXiv preprint arXiv:1409.8484 (2014).

[4] M. Almeida, R. Lins, R. Bernardino, D. Jesus, B. Lima, A new binarization algorithm for historical documents, Journal of Imaging 4 (2018) 27.

[5] A. Venckauskas, A. Karpavicius, R. Damaševičius, R. Marcinkevičius, J. Kapočiūte-Dzikiené, C. Napoli, Open class authorship attribution of lithuanian internet comments using one-class classifier, in: 2017 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2017, pp. 373–382.

[6] O. Boudraa, W. K. Hidouci, D. Michelucci, Degraded historical documents images binarization using a combination of enhanced techniques, 2019.

[7] C. Napoli, E. Tramontana, G. L. Sciuto, M. Woźniak, R. Damaevicius, G. Borowik, Authorship semantical identification using holomorphic chebyshev projectors, in: 2015 Asia-Pacific Conference on Computer Aided System Engineering, IEEE, 2015, pp. 232–237.

[8] D. Połap, M. Woźniak, R. Damaševičius, R. Maskeliūnas, Bio-inspired voice evaluation mechanism, Applied Soft Computing 80 (2019) 342–357.

[9] F. Beritelli, G. Capizzi, G. Lo Sciuto, C. Napoli, M. Woźniak, A novel training method to preserve generalization of rbpnn classifiers applied to ecg signals diagnosis, Neural Networks 108 (2018) 331–338.

[10] G. Capizzi, G. Lo Sciuto, C. Napoli, D. Polap, M. Woźniak, Small lung nodules detection based on fuzzy-logic and probabilistic neural network with bio-inspired reinforcement learning, IEEE Transactions on Fuzzy Systems 6 (2020).

[11] M. Woźniak, D. Połap, Soft trees with neural components as image-processing technique for archeological excavations, Personal and Ubiquitous Computing (2020) 1–13.

[12] Y. Liu, L. Jin, S. Zhang, C. Luo, S. Zhang, Curved scene text detection via transverse and longitudinal sequence connection, Pattern Recognition 90 (2019) 337–345.

[13] Y. Zhu, J. Du, Textmountain: Accurate scene text detection via instance segmentation, Pattern Recognition (2020) 107336.

[14] C. Luo, L. Jin, Z. Sun, Moran: A multi-object rectified attention network for scene text recognition, Pattern Recognition 90 (2019) 109–118.

[15] D. Molodtsov, Soft set theory—first results, Computers & Mathematics with Applications 37 (1999) 19–31.

[16] Onyeozili, T. M. Gwary, A study of the fundamentals of soft set theory, International Journal of Scientific & Technology Research 3 (2014) 132–143.

[17] G. Cardarilli, L. Di Nunzio, R. Fazzolari, A. Nannarelli, M. Re, S. Spano, N-dimensional approximation of euclidean distance, IEEE Transactions on Circuits and Systems II: Express Briefs 67 (2020) 565–569.