Softcomputing
Translation system based on a Multilayer Perceptron: pictures of characters and digits into a Braille alphabet signs - Report

Krystian Horecki 181079
Antoni Buszta 181013
Wroclaw University of Technology

15.11.2013 r.

1. Project description

1.1. Project goals

The goal of the project was to create automatic translation mechanism from pictures of characters to Brail alphabet signs. Pictures of the characters were supposed to be a source of input and training data for translation mechanism. Translation engine was supposed to be created in form of Multilayer Perceptron which is kind of artificial neural network. Important challenge was to create engine which allows to recognize and translate as many characters as it's possible and with many different distortions of different levels. There was no requirement to create user interface in order to focus on engine properties and results instead of good project appearance.

1.2. Input data format

Characters which were used as a source of input data were presented in form of images which were converted to tables containing boolean values. This conversion allowed us to make input data more friendly for further processing. At the same time data still contained exactly the same data as picture and could be converted to this form in any time. Example table containing character of capital **A** was presented below.

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 1, 1, 1, 1, 1, 1, 1, 0,
0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
0, 1, 0, 0, 0, 1, 0, 0, 1, 0,
0, 0, 1, 1, 1, 1, 1, 1, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

1.3. Translation mechanism

As a translation mechanism Multilayer Perceptron was used. To create MP we used Python programming language and pyBrain library which allows to easily create neural network with required nodes configuration. Preparing project we were experimenting with following properties of network:

- number of hidden layers
- number of neurons in hidden layer
- types of hidden layer
- types of output layer
- output data presentation methods

Results of those experiments were presented later in this paper.

1.4. Output data format

During experiments we decided to check 2 different approaches of output data computing:

1. **Translation:** Encoded Brail alphabet characters in form of table, one output neuron refers to one position in table

2. **Clasification:** One output neuron per one class, which means that each output neuron represents different Brail symbol

Because those two methods mean that different number of output neurons are used they can give different translation results. Results of testing those two approaches were presented later in this paper.

2. Used preprocessing and post processing methods

2.1. Preprocessing

To make network more sensitive to black and white pixels which are presented in input data as 0 and 1 we made function converting all zeros to -1. It allowed to improve learning and testing results irrespective of output data computation method(classification or direct translation).

2.2. Post processing

In case of post processing, used method depended on output data computation method, which means that two different post processing methods were used:

2.2.1. Translation post processing method

Post processing was based on assumption that neural network output was in form float numbers table. To create final result we rounded those numbers to 1 or 0 which gave us proper result.

2.2.2. Classification post processing method

During classification neural network as a result gives us table of float numbers with values referring to different classes of output data. To decide which class is a proper one we used method choosing the biggest value from the table. The index of the biggest value was used to determine final class of object recognized by neural network.

3. Training sets and methods

As a training set the pool of capital letters was used. Each letter was represented in set 300 times to make it easier to learn network. During preparations few tests for optimal learning were done with following properties as a variable:

- Learning epochs number
- Number of hidden neurons
- Learning rate
- Momentum - value for parameters of network are adjusted

On figure 1 it can be seen how average number of maximum noisy pixels changes along with number of training epoch. Network behavior was tested with both translation and classification methods, which also can be seen on chart. After tests we decided to use 55 training epochs to provide sufficient training level for network. It was also tested how number of hidden neurons affects results of characters recognition. As it can be seen on figure 2 that relatively good results we get with 55 hidden neurons so this number was later used for tests.

On figure 3 we can see that learning rate has no significant influence on later results, learning rate with value 0.016 was used for later tests.

Analysing chart on figure 4 it can be seen that best value of momentum is oscillating around 0.95 and that value was chosen for later tests.

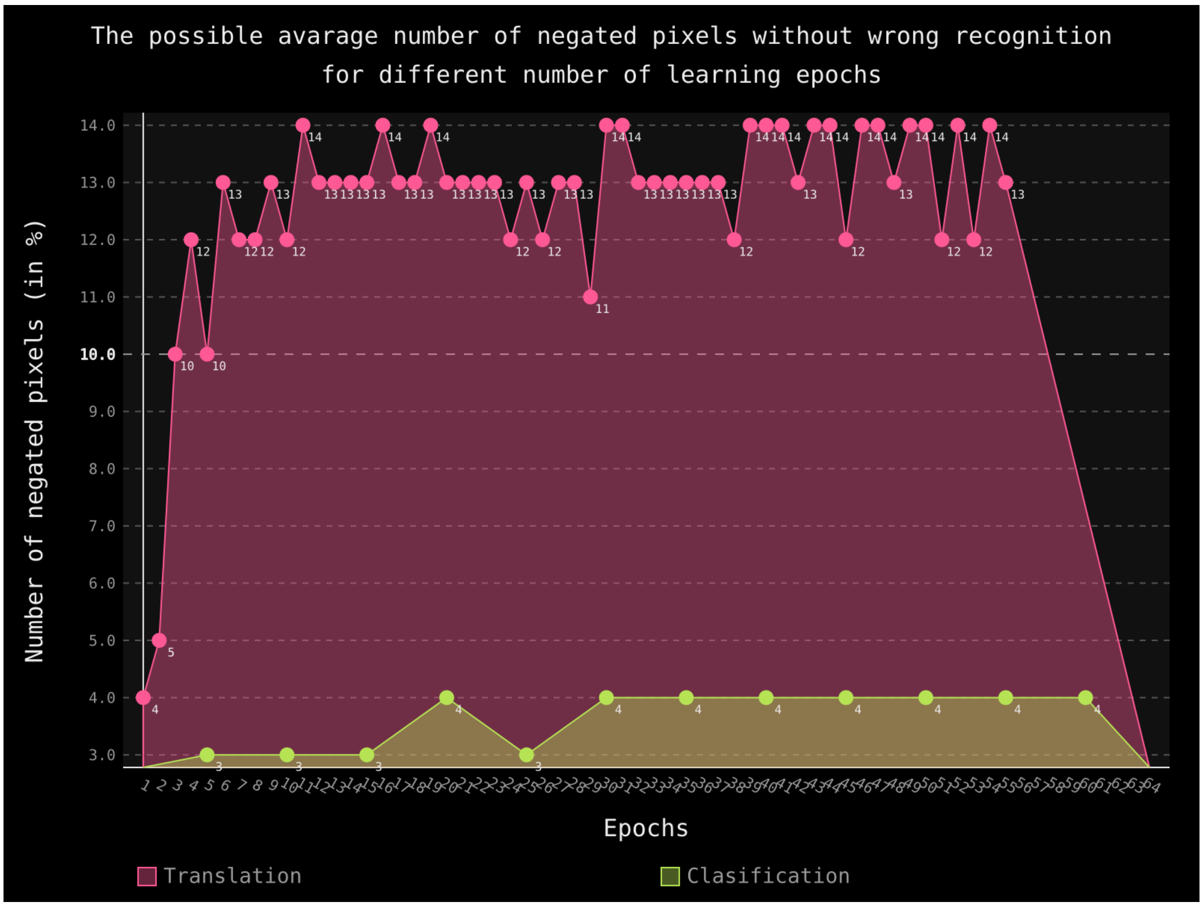


Figure 1:

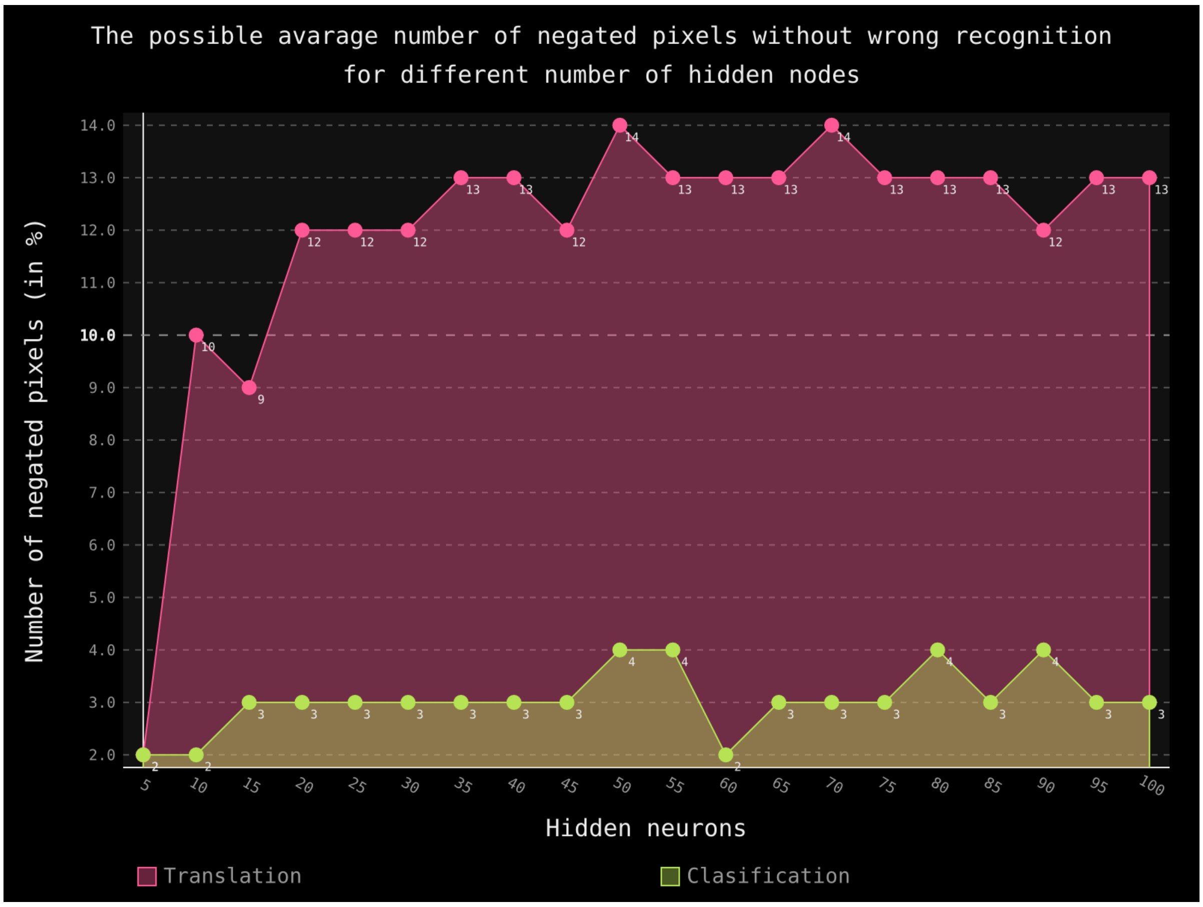


Figure 2:

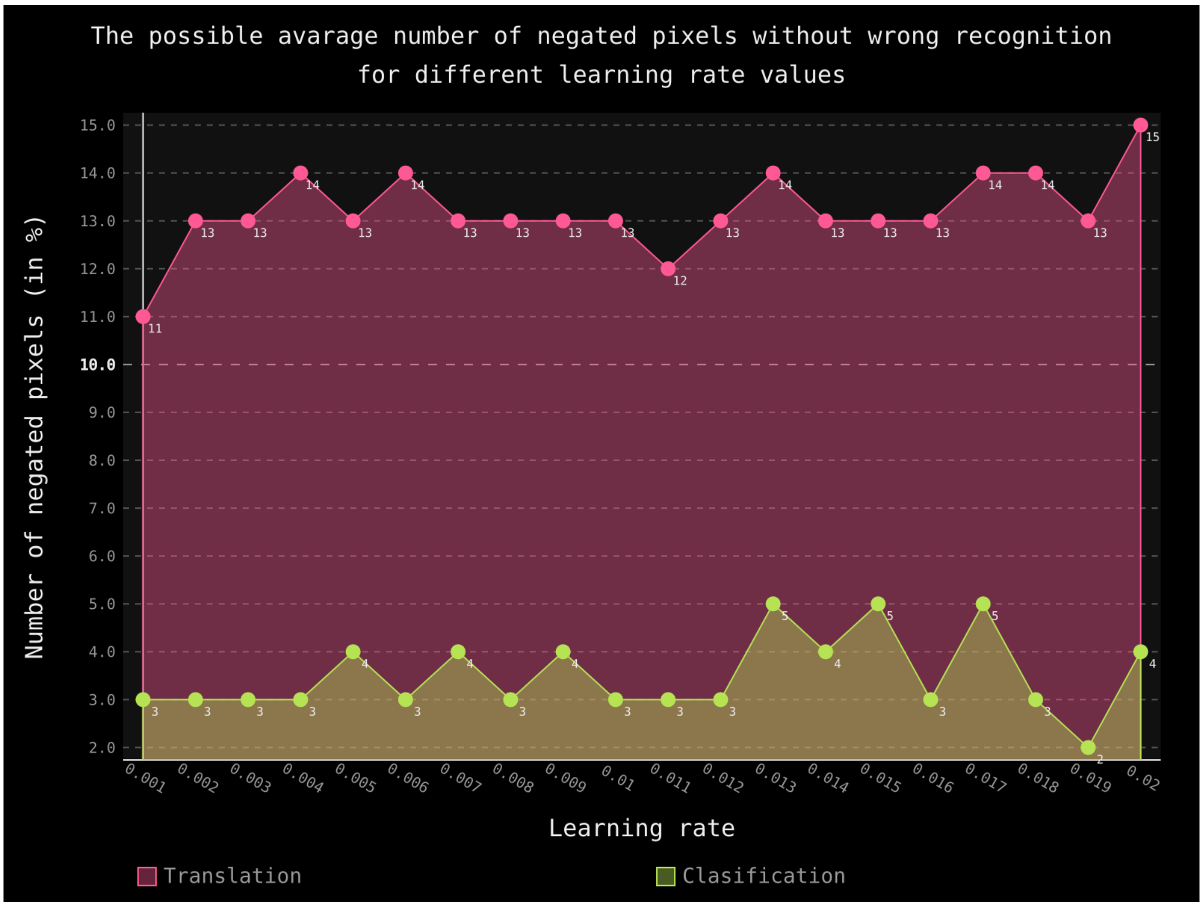


Figure 3:

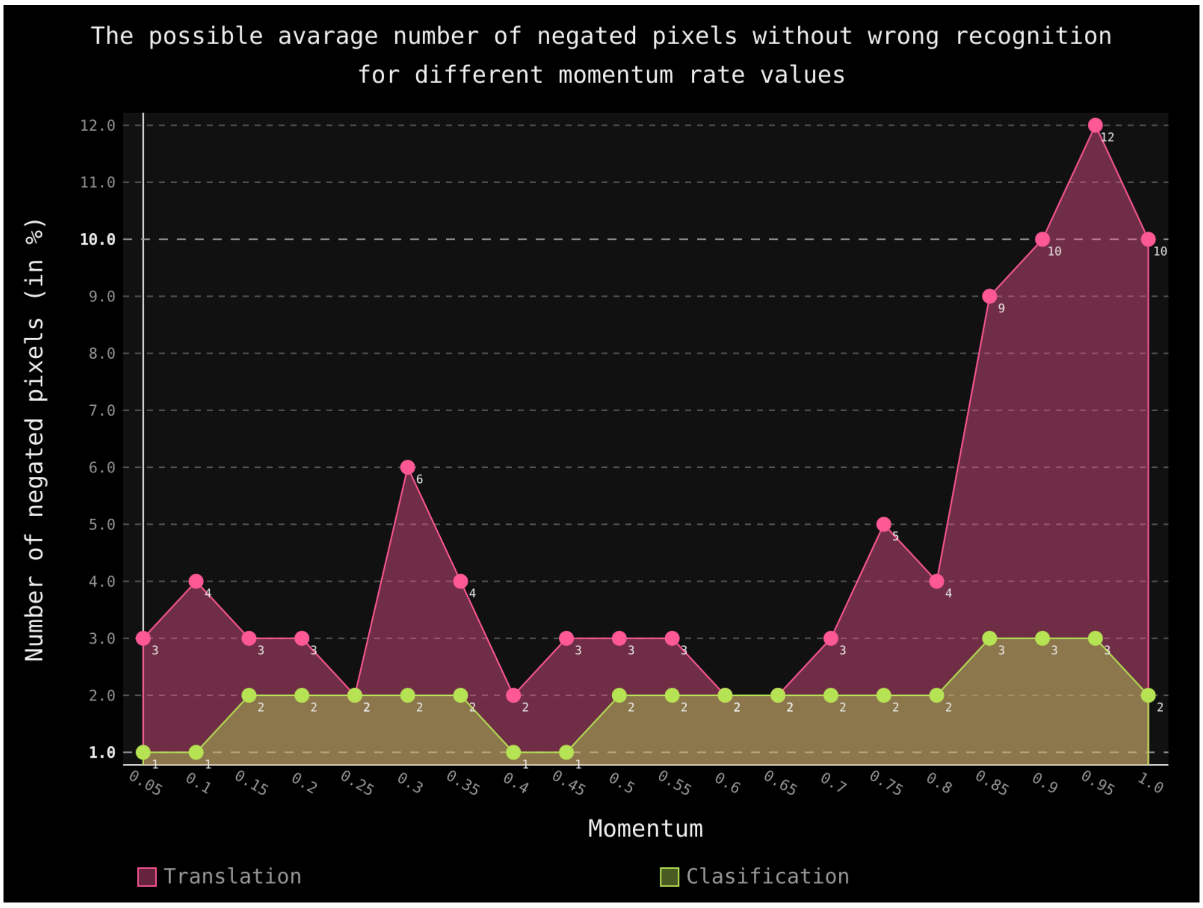


Figure 4:

Final training was done with following parameters:

- Training epochs: 55
- Number of hidden neurons: 55
- Learning rate: 0.016
- Momentum: 0.95

4. Engine description

After training and tests we decided to create Multilayer Perceptron with 1 hidden layer and 55 hidden neurons. During experiments it occurred that the best results in case of classification we get using Softmax hidden and output layer type. For translation the best was Sigmoid layer type so it was used.

5. Gained results

5.1. Test description

Below are presented all gained tests results for prepared tests with short description. We have tested our engine for both:

- Translation
- Classification

Engine was tested with following type of tests:

- Number of random pixels negation
- One vertical line removal success rate
- One horizontal line removal success rate
- Number of random vertical lines removal
- Number of random horizontal lines removal
- Square with random position and particular size removal
- Number of adjacent to character pixels

More accurate description was presented in subsection which concerns each test type.

5.2. Number of random pixels negation

Test was performed for increasing number of random pixels which were negated. Number of negated pixels was increased as long as wrong recognition occured. This was repeated 100 times and avarage number of negated pixels for wrong recognition was returned for each letter as a result. Figures 5 and 6 show results of tests for clasification and translation.

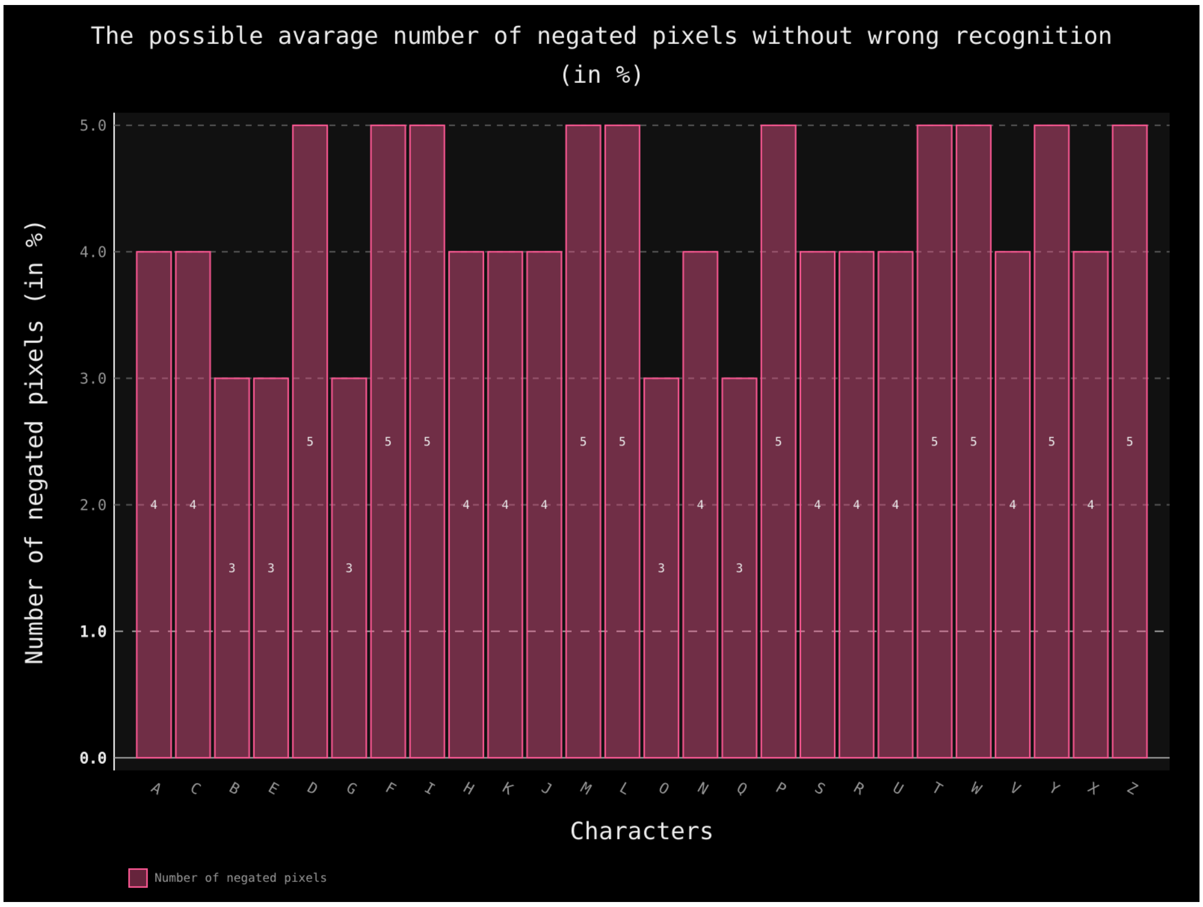


Figure 5:

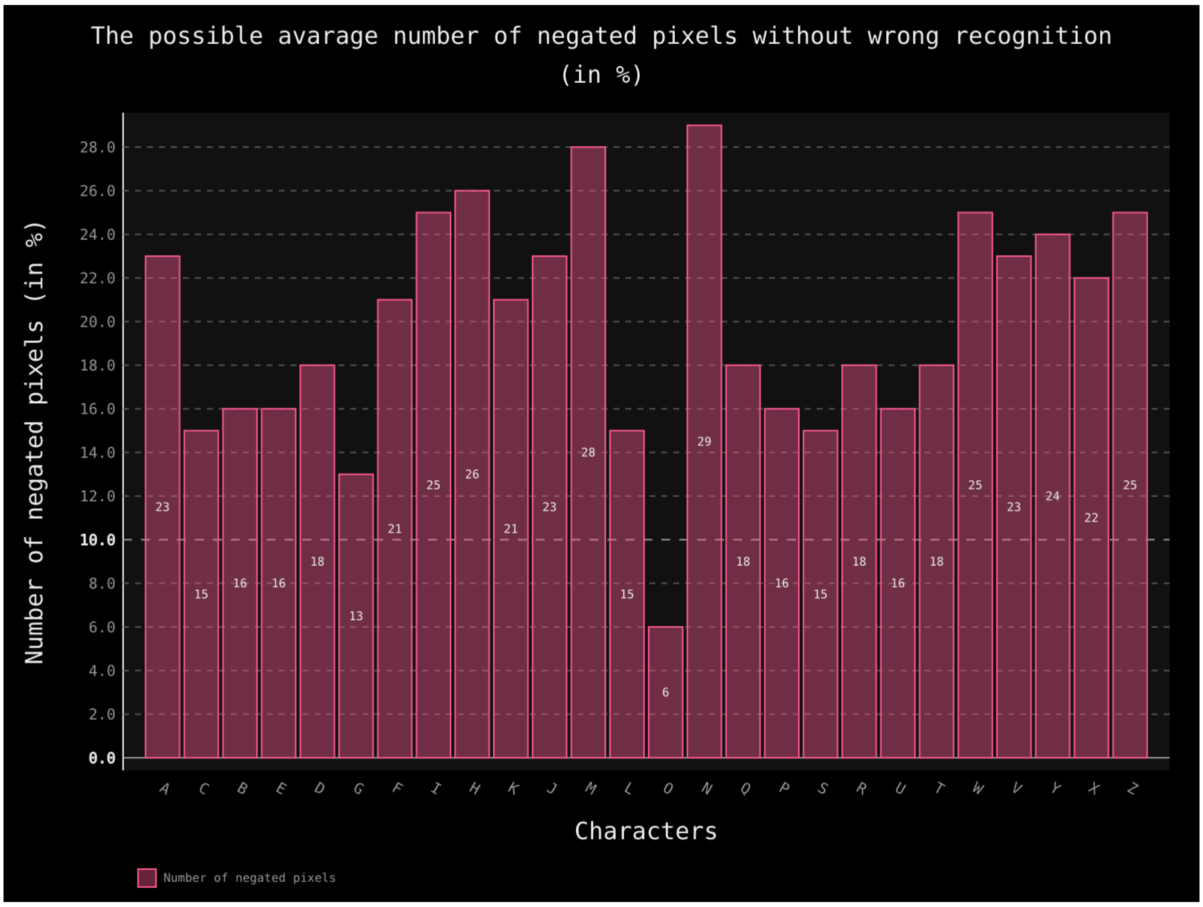

The possible avarage number of negated pixels without wrong recognition (in %)
Number of negated pixels (in %)
Characters
Number of negated pixels
A 23
C 15
B 16
E 16
D 18
G 13
F 21
I 25
H 26
K 21
J 23
M 28
L 15
O 6
N 29
Q 18
P 16
S 15
R 18
U 16
T 18
W 25
V 23
Y 24
X 22
Z 25

Figure 6:

5.3. One vertical line removal success rate

In test each one vertical line of character was removed. As a result test returned succes rate of this removal, which for both methods was presented on figures 7 and 8.

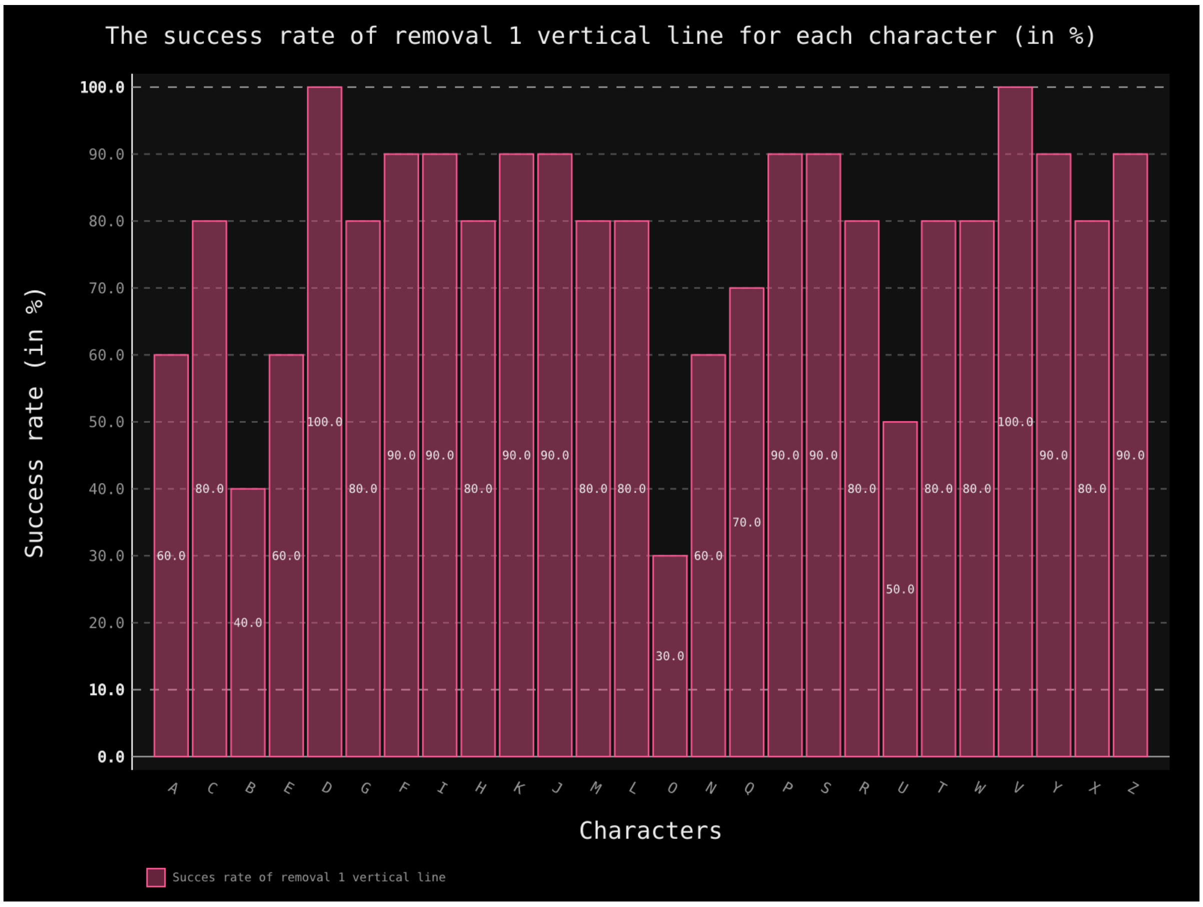


Figure 7:

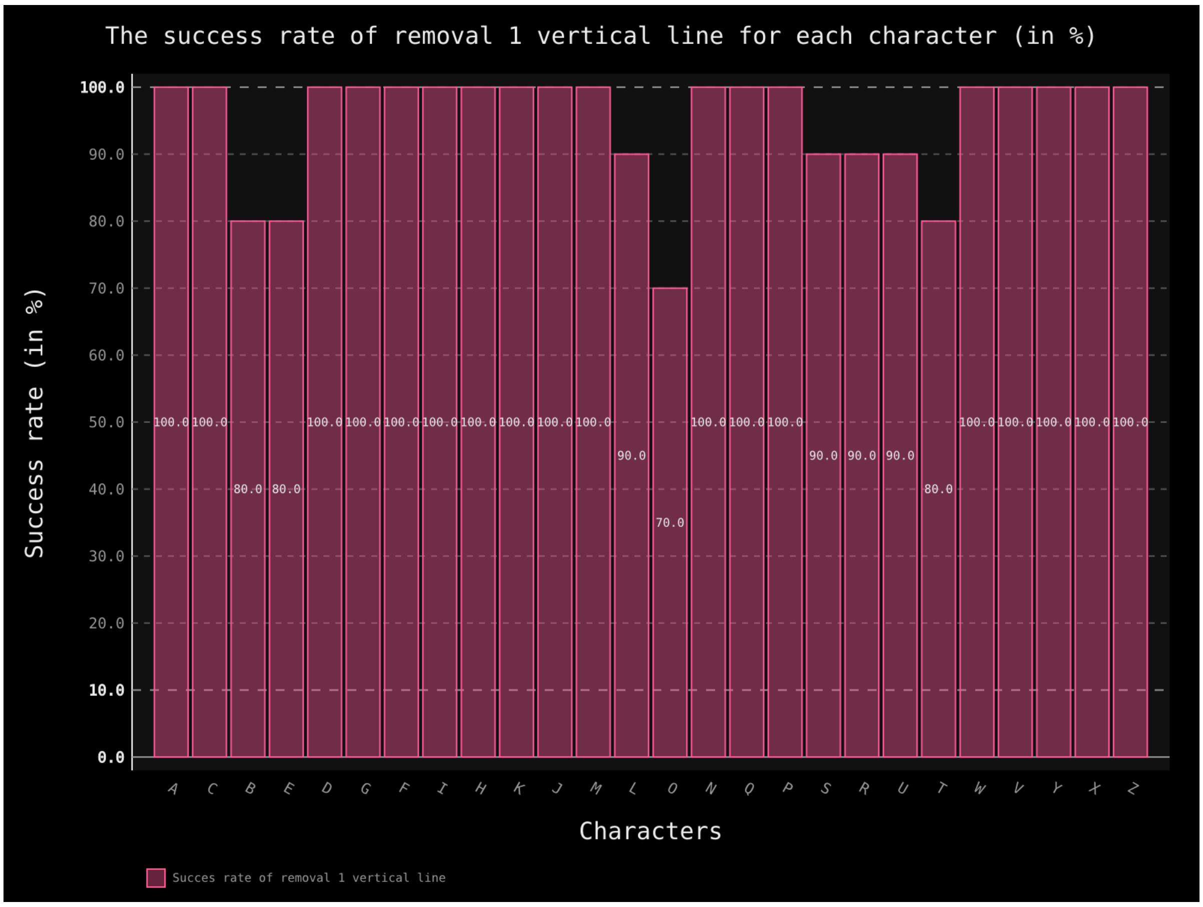

The success rate of removal 1 vertical line for each character (in %)
Success rate (in %)
Characters
A C B E D G F I H K J M L O N Q P S R U T W V Y X Z
100.0 100.0 80.0 80.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 90.0 70.0 100.0 100.0 100.0 90.0 90.0 90.0 80.0 100.0 100.0 100.0 100.0 100.0
Succes rate of removal 1 vertical line

Figure 8:

5.4. One horizontal line removal success rate

In test each one horizontal line of character was removed. As a result test returned succes rate of this removal, which for both methods was presented on figures 9 and 10.

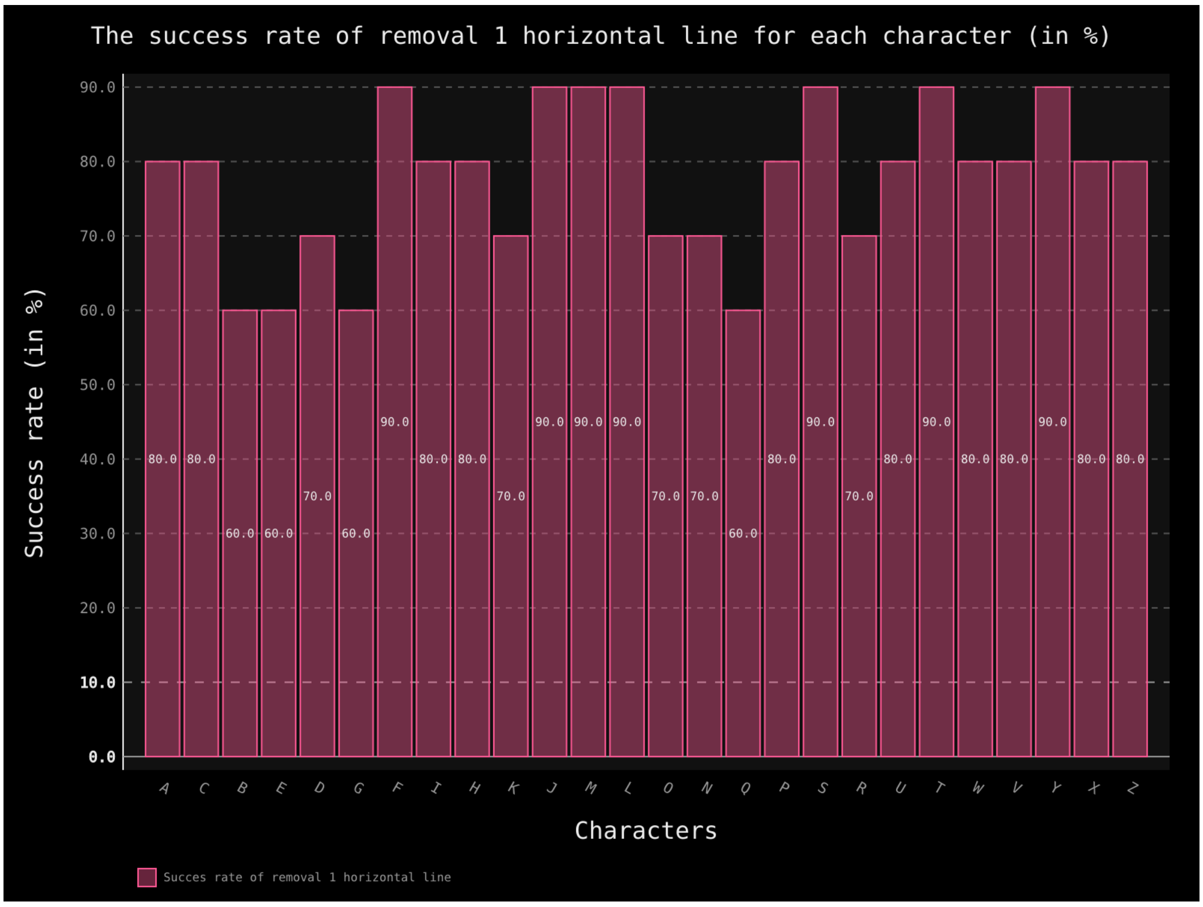


Figure 9:

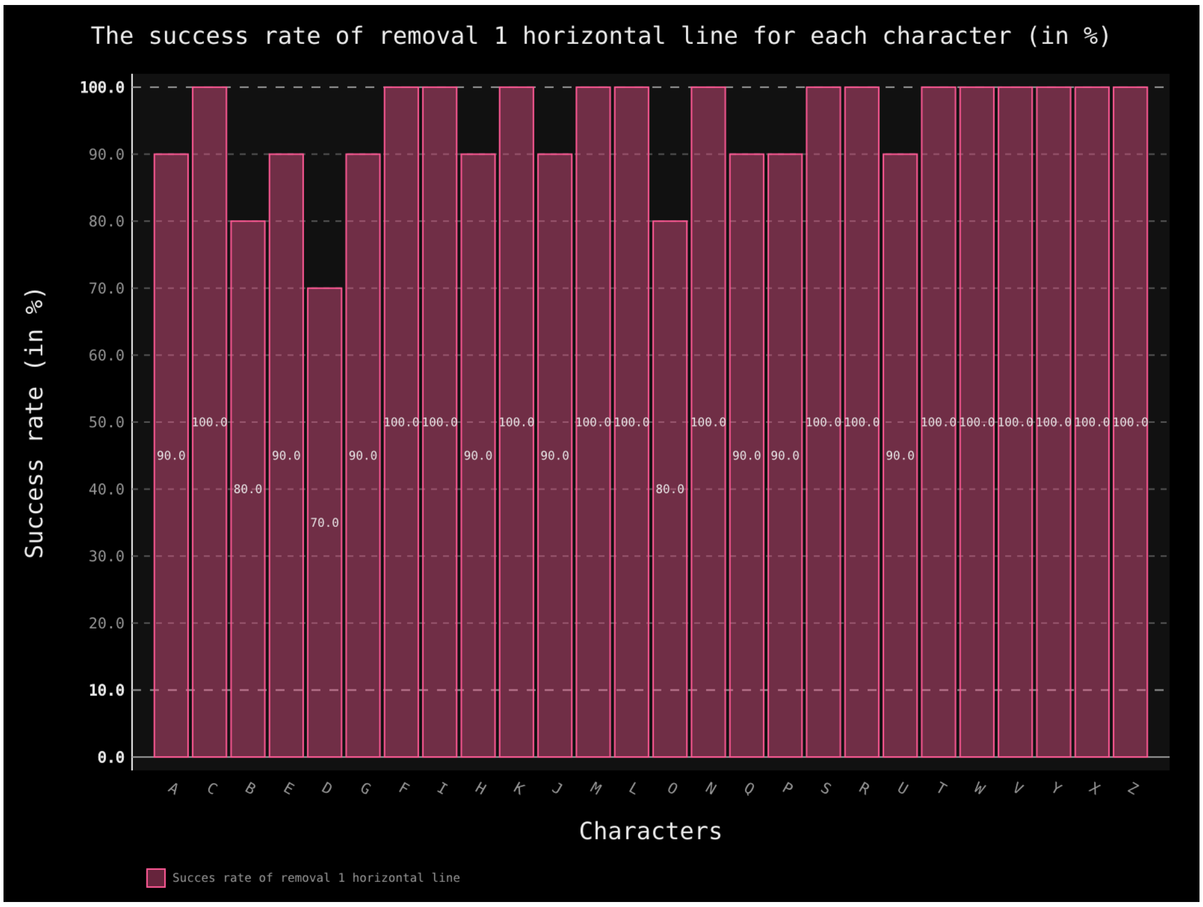

The success rate of removal 1 horizontal line for each character (in %)
Success rate (in %)
Characters
Succes rate of removal 1 horizontal line
A 90.0
C 100.0
B 80.0
E 90.0
D 70.0
G 90.0
F 100.0
I 100.0
H 90.0
K 100.0
J 90.0
M 100.0
L 100.0
O 80.0
N 100.0
Q 90.0
P 90.0
S 100.0
R 100.0
U 90.0
T 100.0
W 100.0
V 100.0
Y 100.0
X 100.0
Z 100.0

Figure 10:

5.5. Number of random vertical lines removal

Test was performed for increasing number of random vertical lines which were removed. Number of vertical lines was increased as long as wrong recognition occured. This was repeated 100 times and avarage number of vertical lines for wrong recognition was returned for each letter as a result. Figures 11 and 12 show results of tests for clasification and translation.

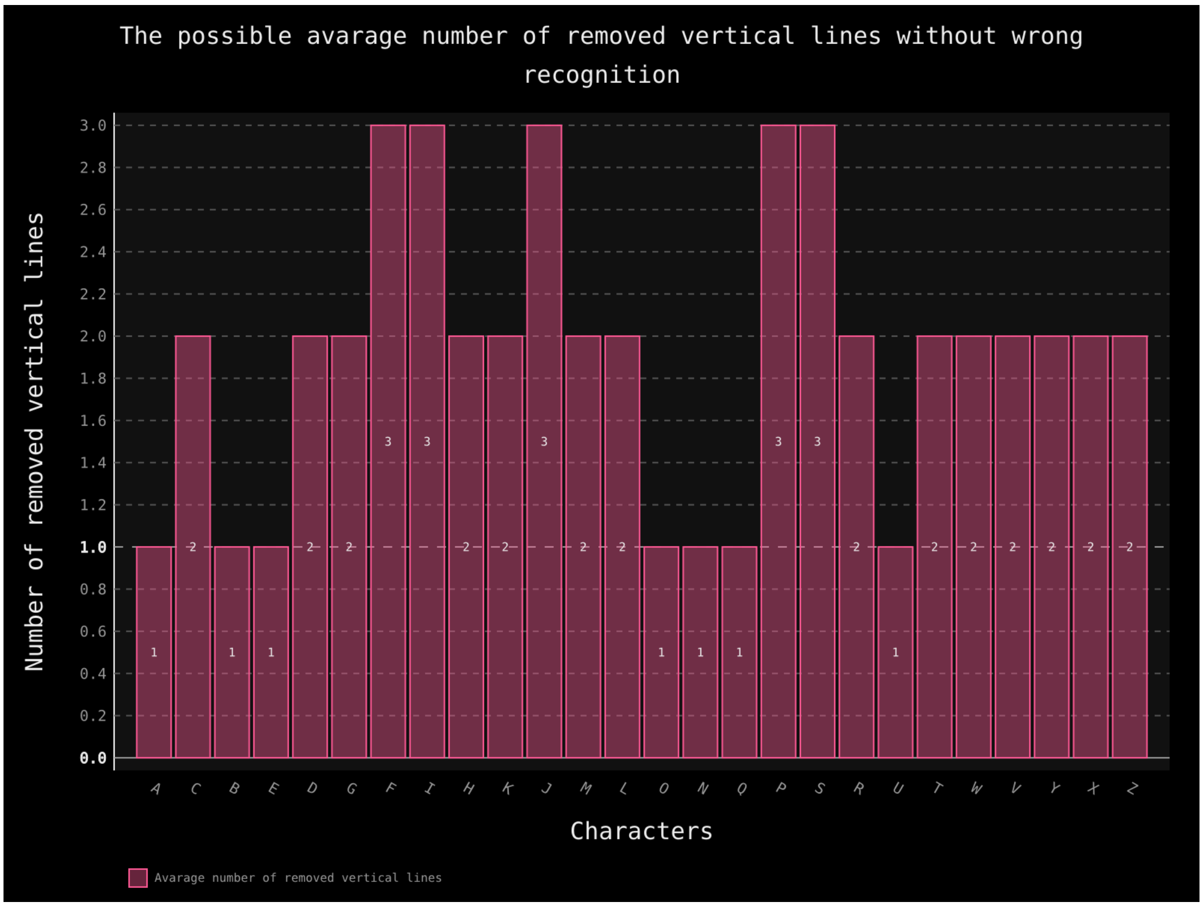


Figure 11:

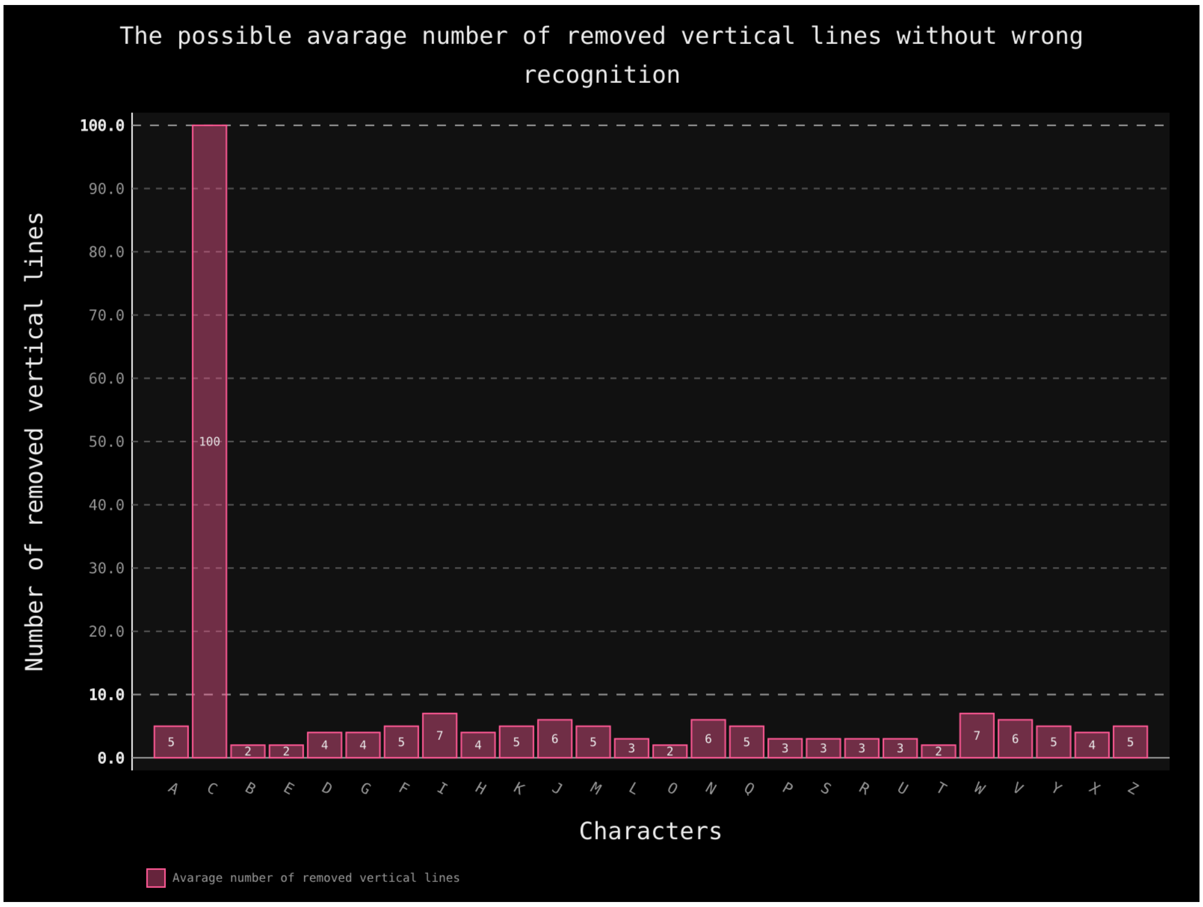

The possible avarage number of removed vertical lines without wrong recognition
Number of removed vertical lines
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
A C B E D G F I H K J M L O N Q P S R U T W V Y X Z
5 100 2 2 4 4 5 7 4 5 6 5 3 2 6 5 3 3 3 3 2 7 6 5 4 5
Characters
Avarage number of removed vertical lines

Figure 12:

5.6. Number of random horizontal lines removal

Test was performed for increasing number of random horizontal lines which were removed. Number of horizontal lines was increased as long as wrong recognition occured. This was repeated 100 times and avarage number of horizontal lines for wrong recognition was returned for each letter as a result. Figures 13 and 14 show results of tests for clasification and translation.

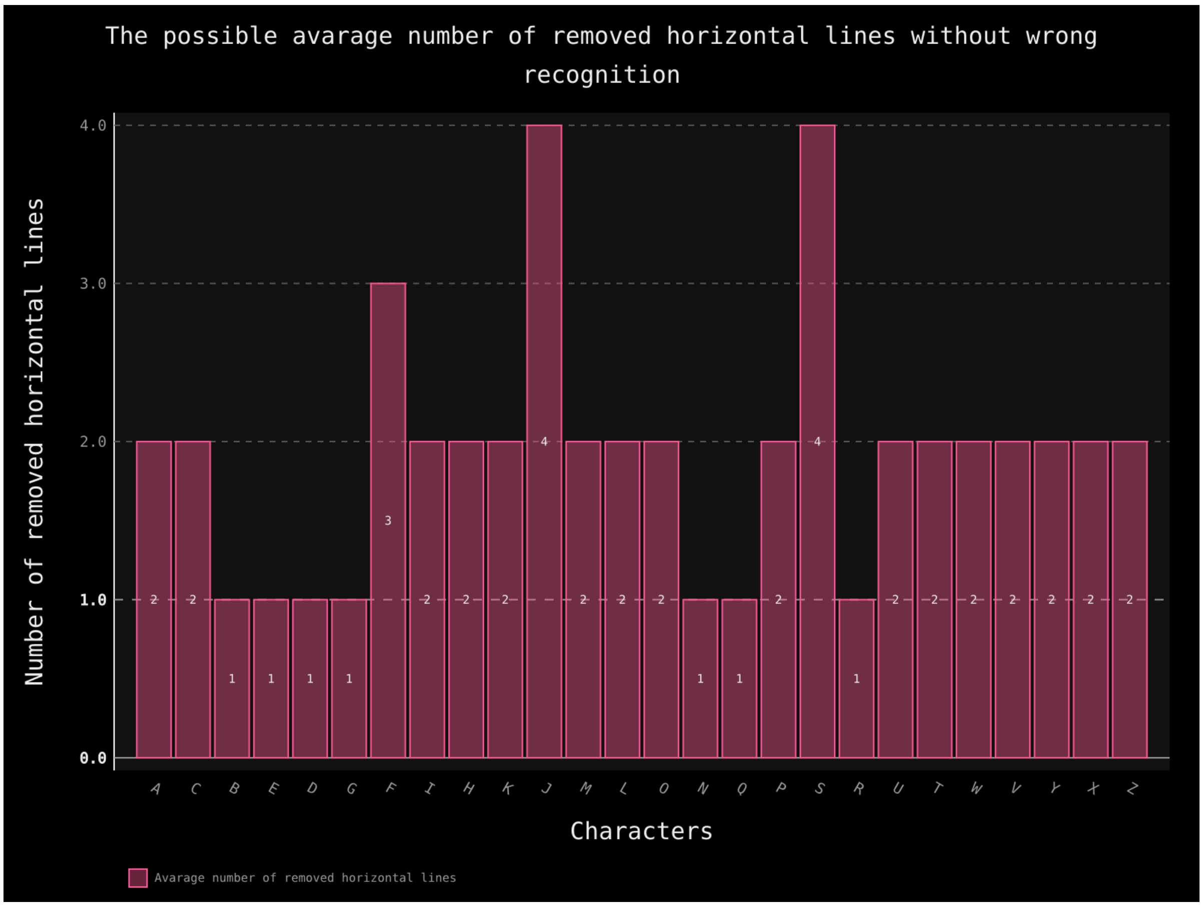


Figure 13:

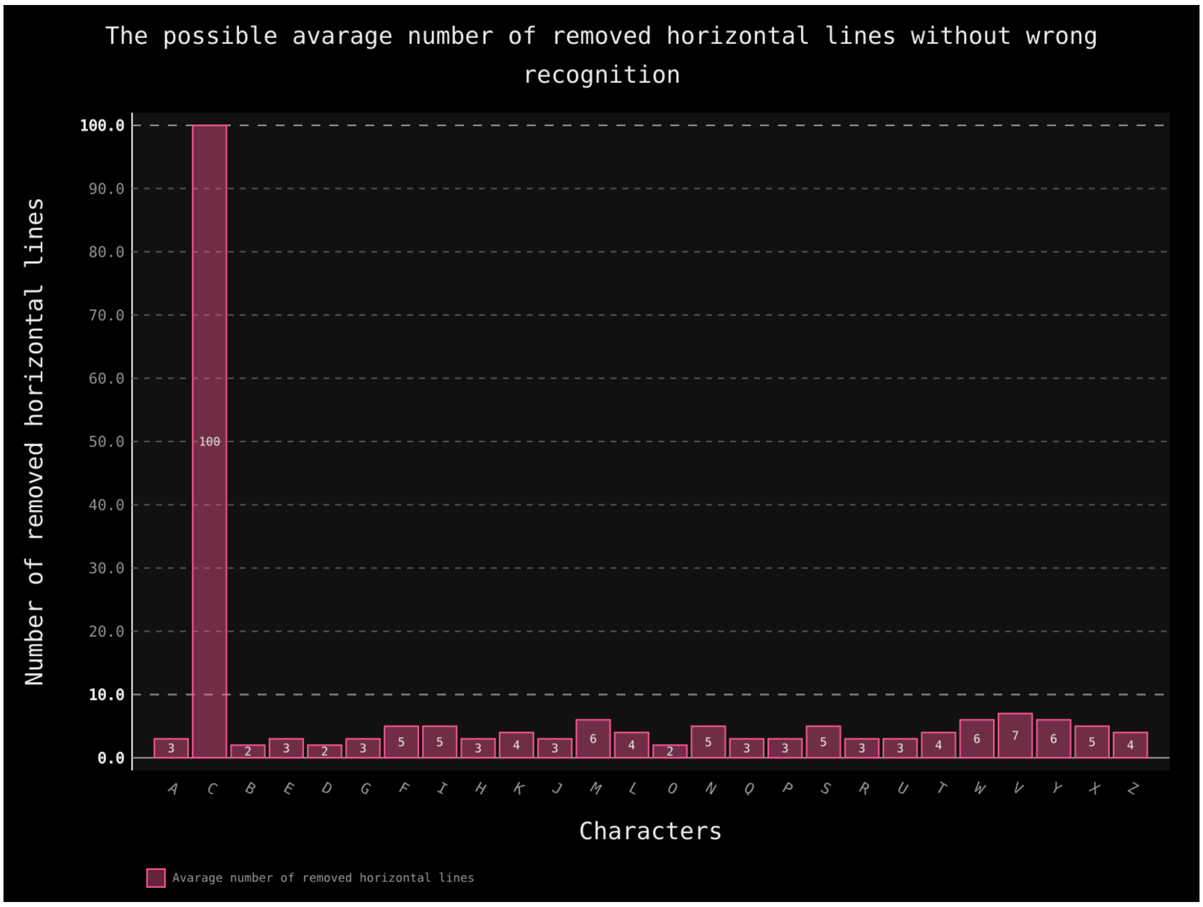

The possible avarage number of removed horizontal lines without wrong recognition
Number of removed horizontal lines
Characters
Avarage number of removed horizontal lines
0.0
10.0
20.0
30.0
40.0
50.0
60.0
70.0
80.0
90.0
100.0
A 3
C 100
B 2
E 3
D 2
G 3
F 5
I 5
H 3
K 4
J 3
M 6
L 4
O 2
N 5
Q 3
P 3
S 5
R 3
U 3
T 4
W 6
V 7
Y 6
X 5
Z 4

Figure 14:

5.7. Number of random horizontal and vertical lines removal comprasion

Test for removal of number of random vertical and horizontal lines were accumulated in one chart. Figures 15 and 16 show results of tests for clasification and translation.

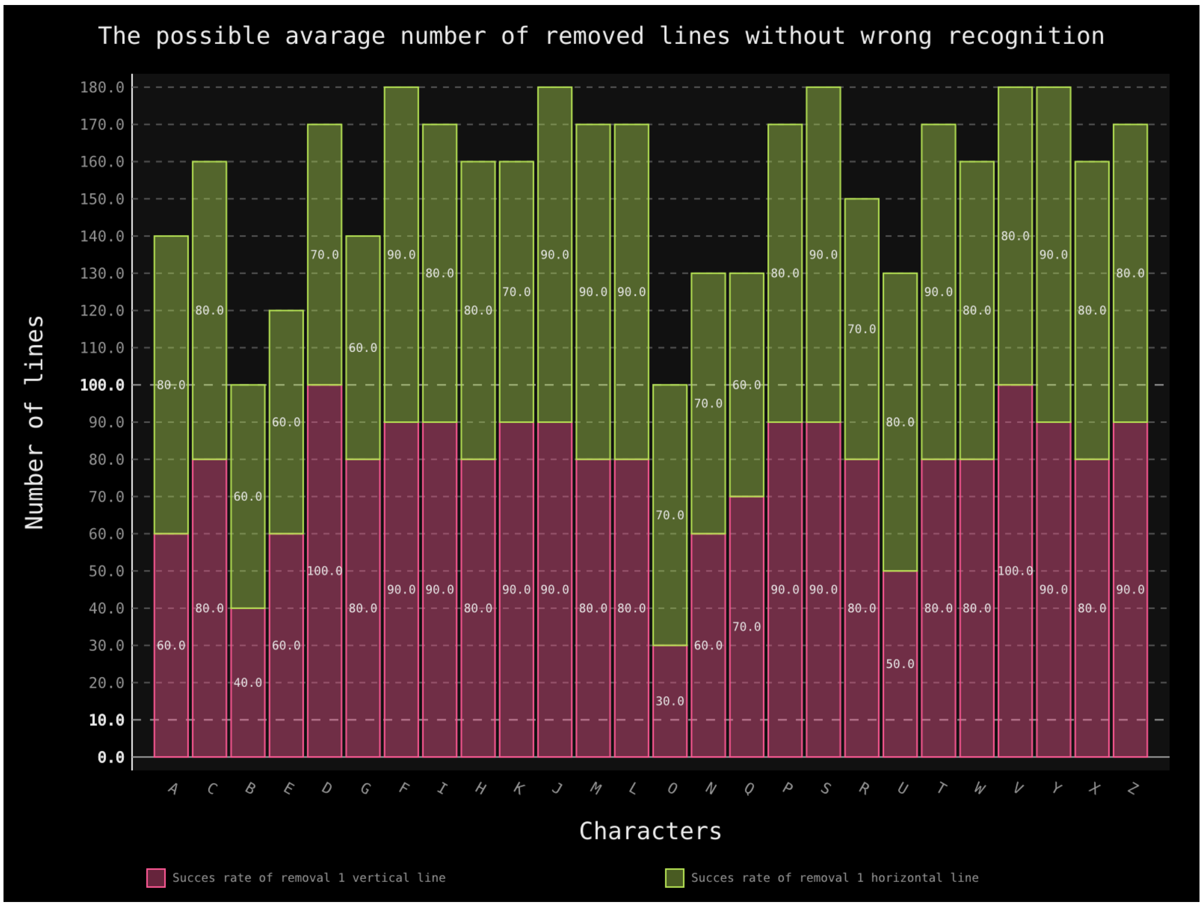


Figure 15:

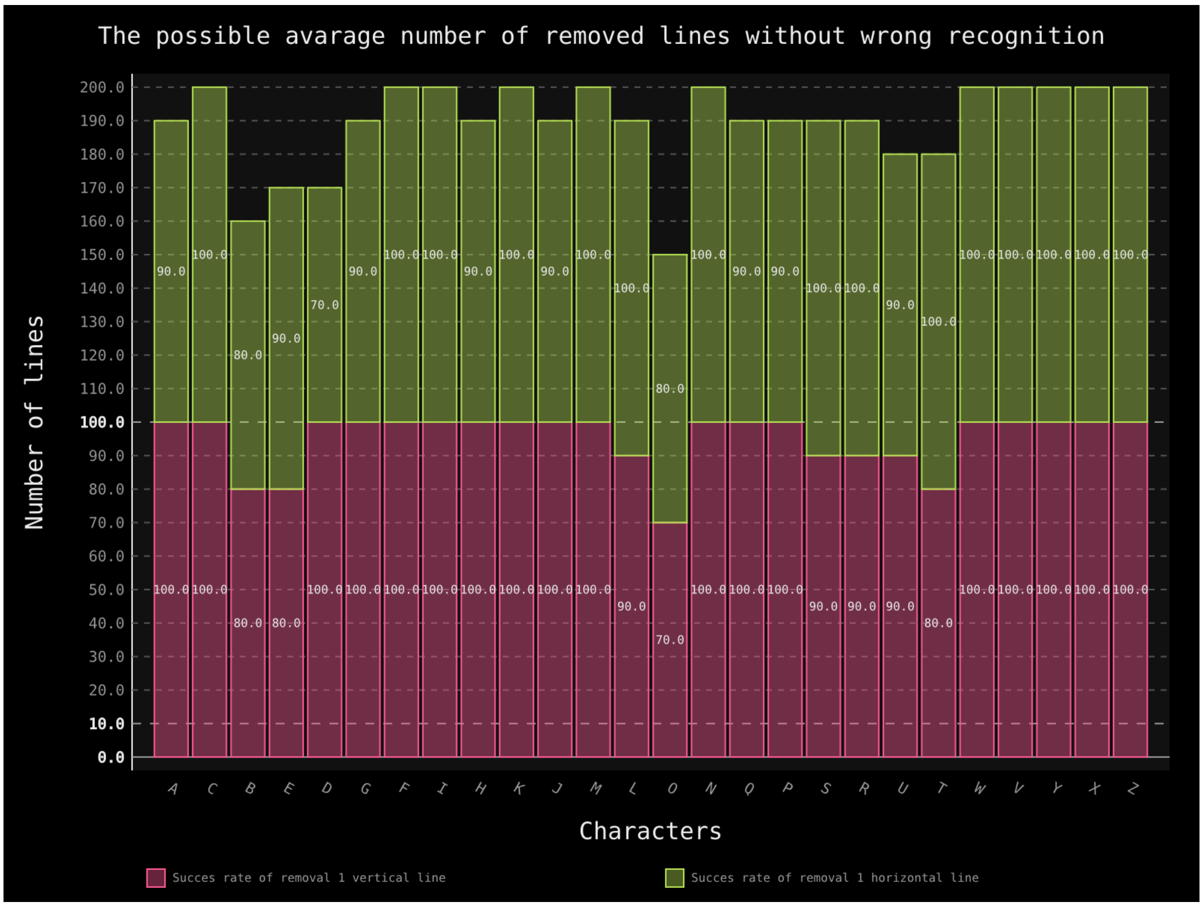

The possible avarage number of removed lines without wrong recognition
Number of lines
Characters
A C B E D G F I H K J M L O N Q P S R U T W V Y X Z
Succes rate of removal 1 vertical line
Succes rate of removal 1 horizontal line

Figure 16:

5.8. Square with random position and particular size removal

Test was performed for increasing size of randomly located square which was removed from character picture. Size of square was increased as long as wrong recognition occured. This was repeated 100 times and avarage size of square for wrong recognition was returned for each letter as a result. Figures 17 and 18 show results of tests for clasification and translation.

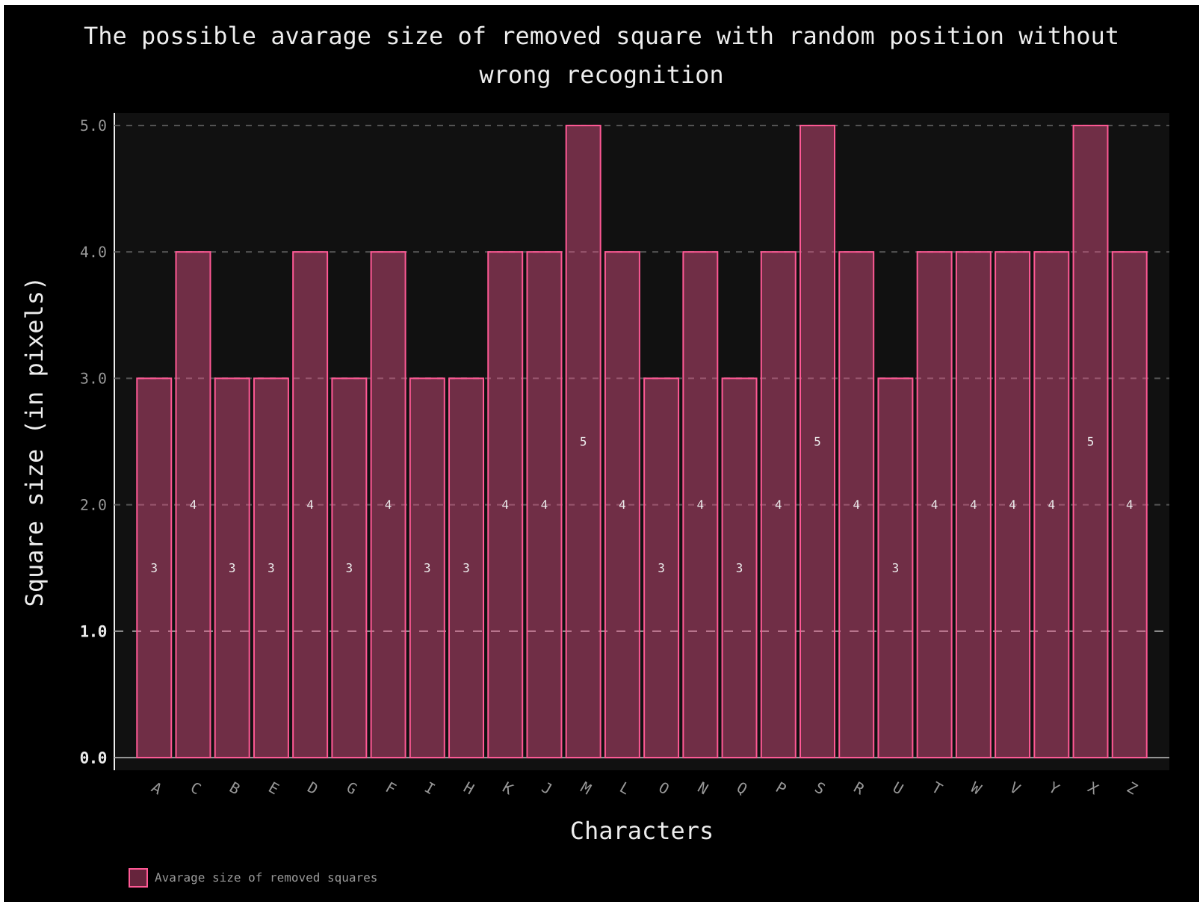


Figure 17:

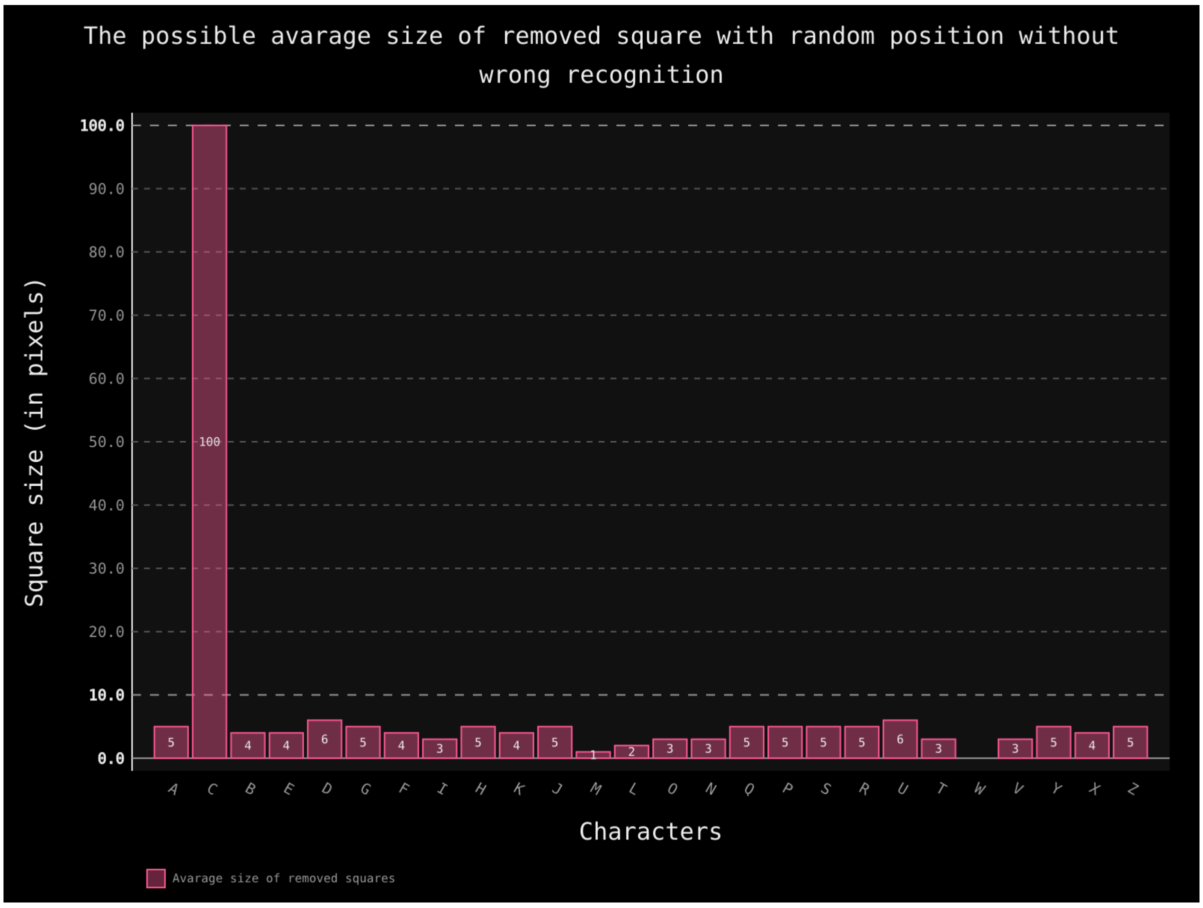

The possible avarage size of removed square with random position without wrong recognition
Square size (in pixels)
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
A C B E D G F I H K J M L O N Q P S R U T W V Y X Z
5 100 4 4 6 5 4 3 5 4 5 1 2 3 3 5 5 5 5 6 3 3 5 4 5
Characters
Avarage size of removed squares

Figure 18:

5.9. Number of adjacent to character pixels

Test was performed for increasing number of randomly located adjacent pixels which were coloured on characters picture. Number of pixels was increased as long as wrong recognition occured. This was repeated 100 times and avarage number of adjacent pixels for wrong recognition was returned for each letter as a result. Figures 19 and 20 show results of tests for clasification and translation.

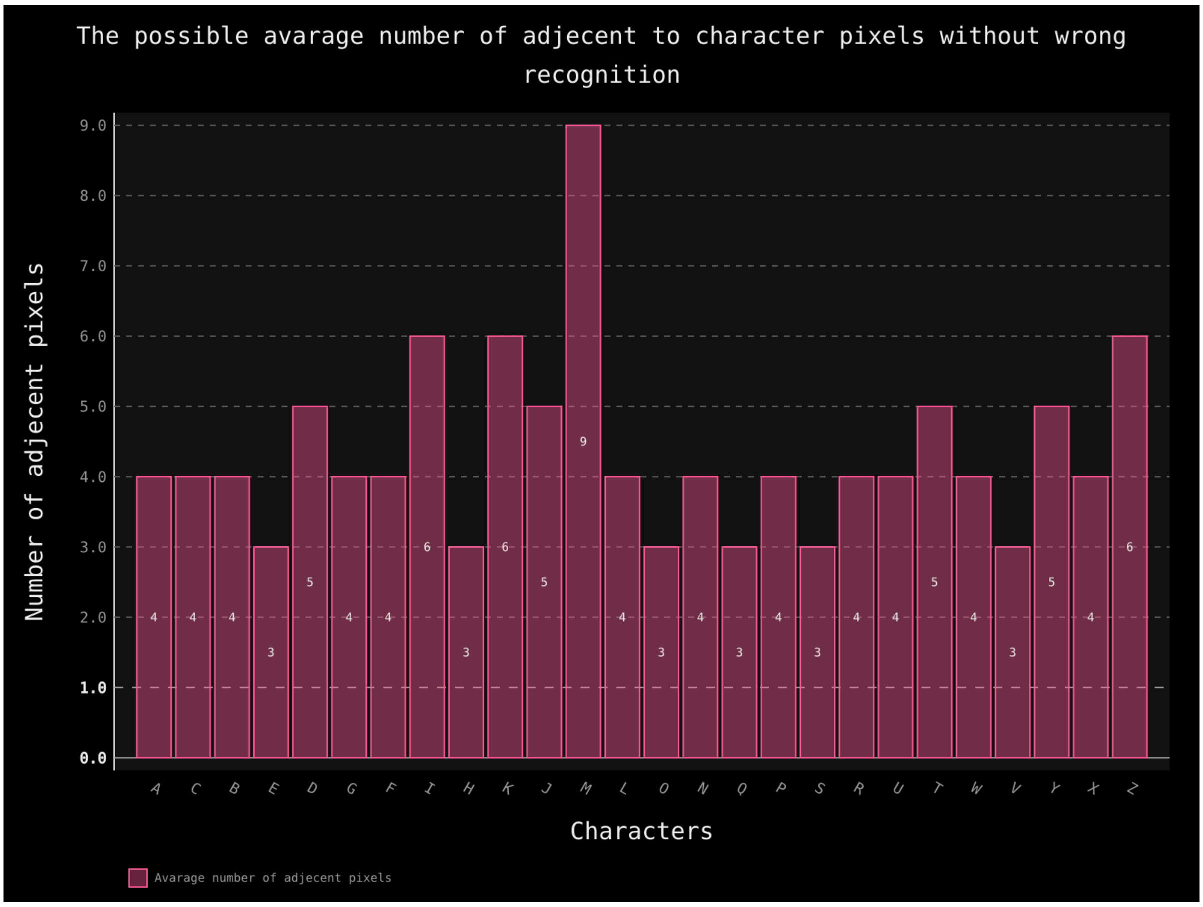


Figure 19:

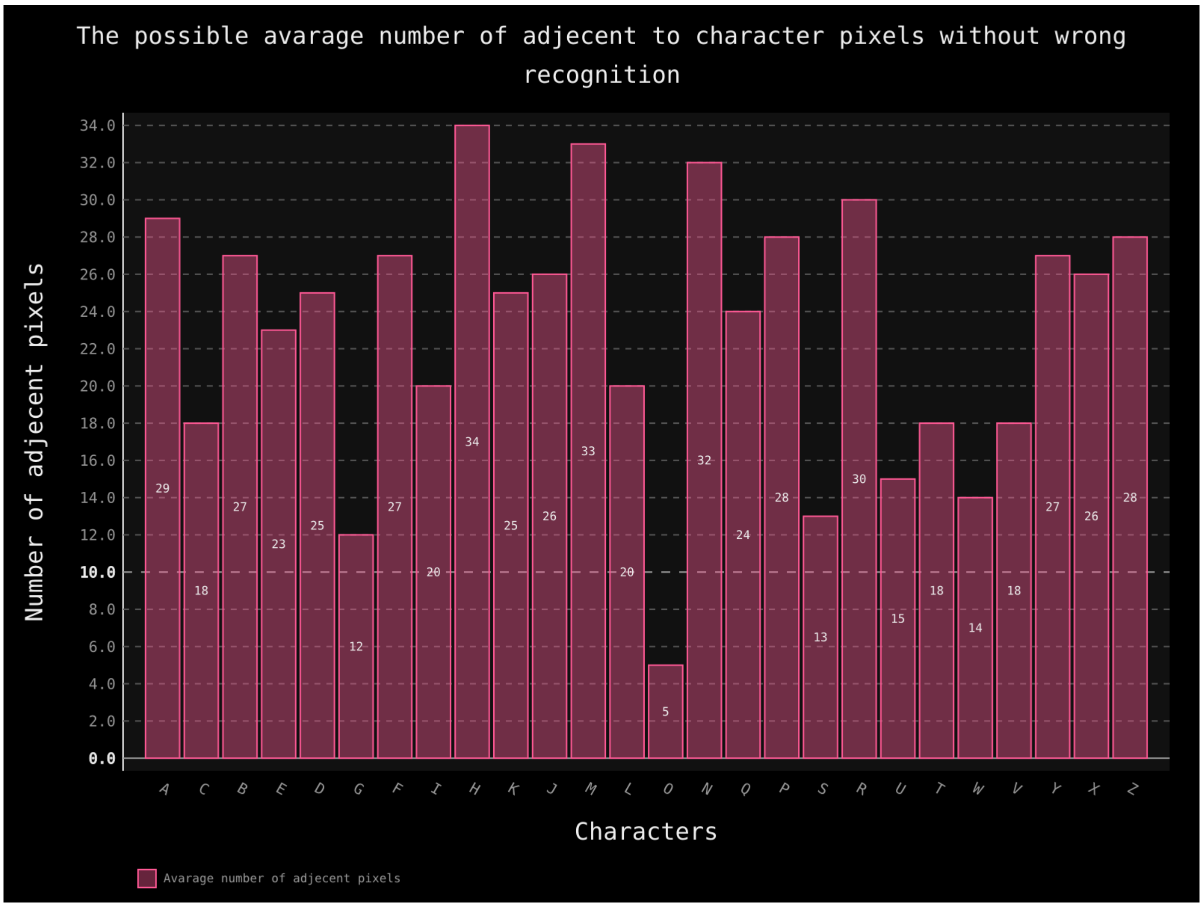

The possible avarage number of adjecent to character pixels without wrong recognition
Number of adjecent pixels
34.0
32.0
30.0
28.0
26.0
24.0
22.0
20.0
18.0
16.0
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
29
18
27
23
25
12
27
20
34
25
26
33
20
5
32
24
28
13
30
15
18
14
18
27
26
28
A
C
B
E
D
G
F
I
H
K
J
M
L
O
N
Q
P
S
R
U
T
W
V
Y
X
Z
Characters
Avarage number of adjecent pixels

Figure 20:

5.10. Summarization of all tests

All tests which were presented earlier in this paper were accumulated on one chart for both translation and classification. Figures 21 and 22 show results of accumulated tests for classification and translation for all characters.

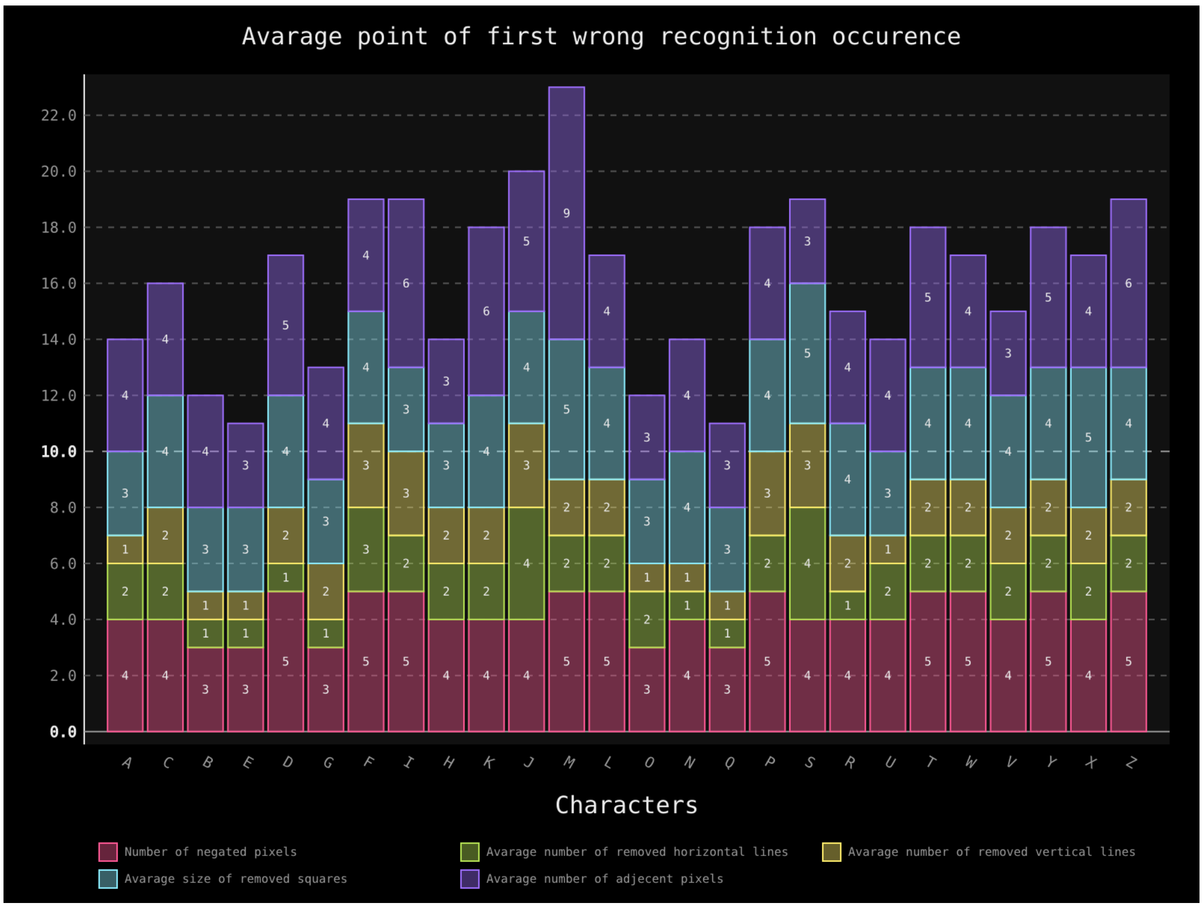
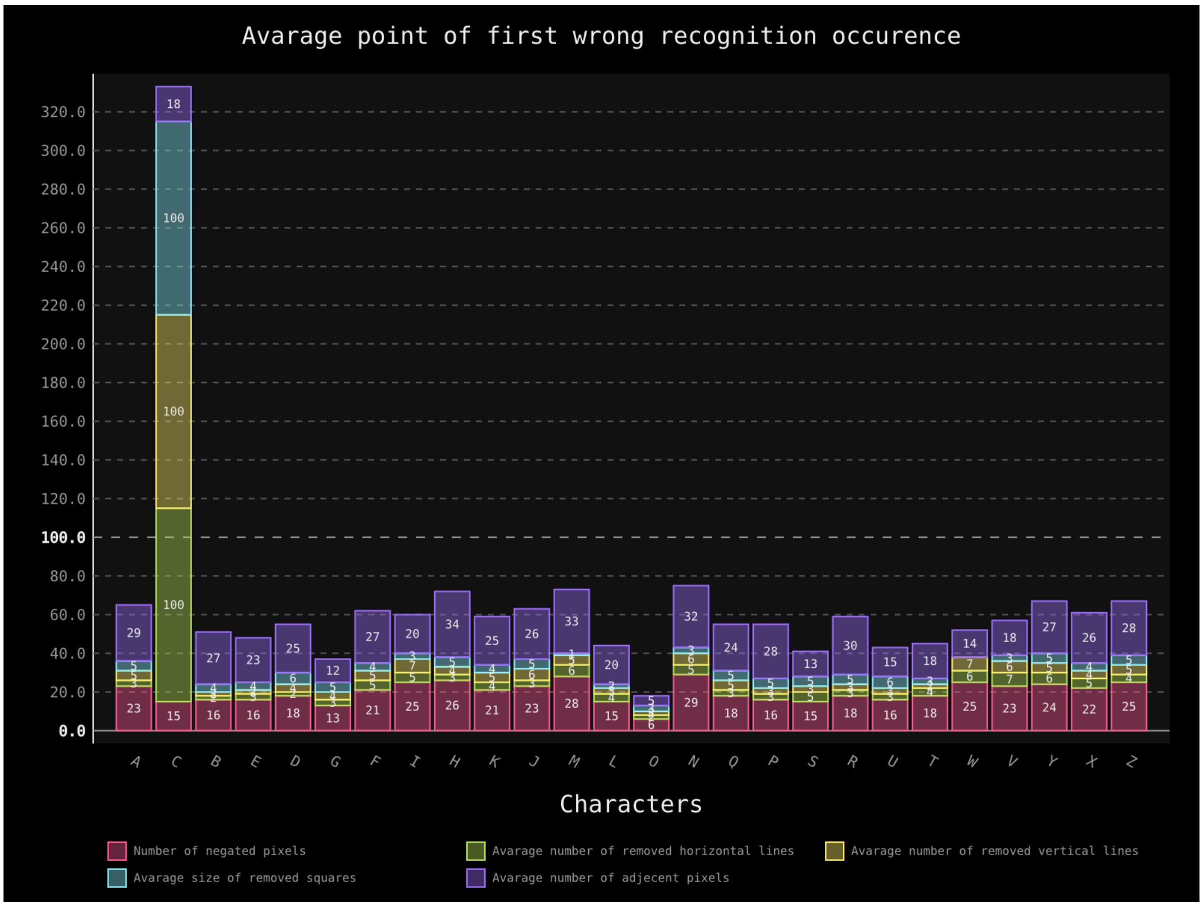


Figure 21:



Avarage point of first wrong recognition occurence
Characters
A C B E D G F I H K J M L O N Q P S R U T W V Y X Z
Number of negated pixels
Avarage number of removed horizontal lines
Avarage number of removed vertical lines
Avarage size of removed squares
Avarage number of adjecent pixels

Figure 22:

6. Final remarks

It can be noticed that some characters are easier to be recognized by engine. On figure 21 and 22 it is visible that letters like O, U, G are much sensible for different distortions. In the other hand letters like K, M, H, with many lines are resistant to them. Because it is much easier for network to give proper output for one class per one output neuron it is visible that automatic classification is much better than translation. There is no reson to use automatic translation because it gives much worse results trying to recognize characters with different noises and distortions. Summarizing we have to admit that results exceeded our expectations. Correct recognition of letter with almost 30% of additional pixels which make character more bold or 30% of random noise is a really big number and could make hard to recognize character even for human.