WROCLAW UNIVERSITY OF TECHNOLOGY
DEPARTMENT OF ELECTRONICS

FIELD:	Computer Science
SPECIALITY:	Internet Enginering

SOFTCOMPUTING

Translation system based on Multilayer Perceptron: pictures of characters into a Morse alphabet signs.

AUTHOR:
Jarosław Szumega 196018

COURSE SUPERVISOR:
dr inż. Jacek Mazurkiewicz

GRADE:

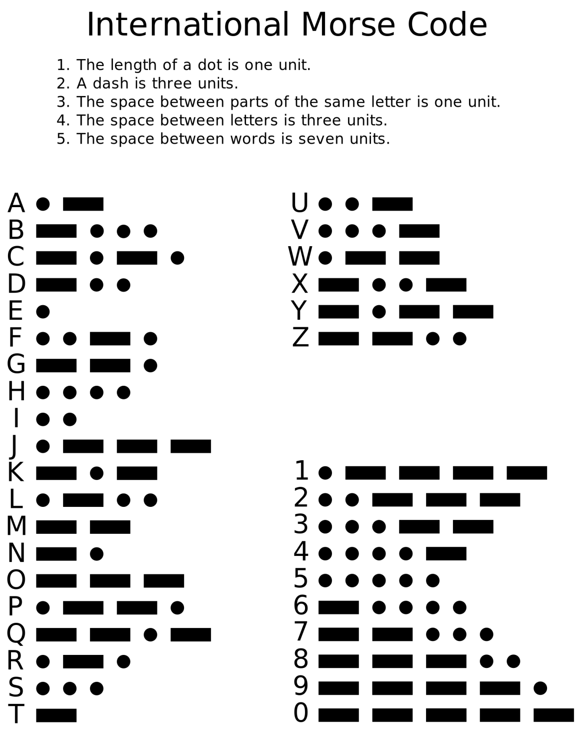
WROCŁAW 14.12.2015

Contents

Chapter 1

Problem description

Main task is to implement a system to recognize a pictures of characters (assumed that they will be computer's fonts) and then translate them into a Morse codes using some kind of dictionary.
Recognition will be performed by using Multilayer Perceptron - form of neural network that besides input and output layer has also one or more hidden layers. Their task is to improve the computations and make entire network more sensitive.


International Morse Code

1. The length of a dot is one unit.
2. A dash is three units.
3. The space between parts of the same letter is one unit.
4. The space between letters is three units.
5. The space between words is seven units.

A	● ▬	U	● ● ▬
B	▬ ● ● ●	V	● ● ● ▬
C	▬ ● ▬ ●	W	● ▬ ▬
D	▬ ● ●	X	▬ ● ● ▬
E	●	Y	▬ ● ▬ ▬
F	● ● ▬ ●	Z	▬ ▬ ● ●
G	▬ ▬ ●		
H	● ● ● ●		
I	● ●		
J	● ▬ ▬ ▬		
K	▬ ● ▬	1	● ▬ ▬ ▬ ▬
L	● ▬ ● ●	2	● ● ▬ ▬ ▬
M	▬ ▬	3	● ● ● ▬ ▬
N	▬ ●	4	● ● ● ● ▬
O	▬ ▬ ▬	5	● ● ● ● ●
P	● ▬ ▬ ●	6	▬ ● ● ● ●
Q	▬ ▬ ● ▬	7	▬ ▬ ● ● ●
R	● ▬ ●	8	▬ ▬ ▬ ● ●
S	● ● ●	9	▬ ▬ ▬ ▬ ●
T	▬	0	▬ ▬ ▬ ▬ ▬

Figure 1.1 Standard for coding characters in Morse code.

As the figure 1.1 presents, the Morse Code is an alphabet with non-fixed word length. Multilayer perceptron should have the fixed number of input neurons. Due to that fact only the characters coded with word of length four were chosen to perform experiments.

Chapter 2

Solution

2.1 Tools to be used

For purpose of implementation **Python** was chosen. It is scripting, high-level language with many scientific libraries. And as a plus it was primely designed to work with UNIX-like systems.
Following libraries were used to realize the project:

- PyBrain - it is a Machine Learning Library for Python. Its modules contain a number of algorithms for neural network and machine learning purposes. In following project it was used to build a neural network and perform supervised learning.
- Pillow - is still being developed fork of PIL (Python Image Library that is no longer supported). It is similar to Matlab Image Processing Toolbox.
  Used mainly for reading images and preprocessing purposes for input of neural network.

2.2 Network design

The neural network build to realise characters recognition has following structure:

- 900 neurons in input layer - the number of neurons is a results of using pictures of size 30x30 pixels. Each neuron corresponds to particular pixel.
- 60 neurons in hidden layer - that layer exists to improve computations and perform transformation inputs to outputs. Hidden layer uses also activation function. e.g. sigmoid or hyperbolic tangent. The number of neurons is fixed according to often used formula:
  $hidden_neurons = \sqrt{in\_neurons * out\_neurons}$.
- 4 neurons in output layer - four signs Morse code characters were chosen, so at the output there is a need to establish four elements.

2.3 Datasets and network training

For teaching purposes there was prepared eight sets of four-sign characters. Different fonts were chosen in order to keep variety of characters (picture 2.1). Presented above

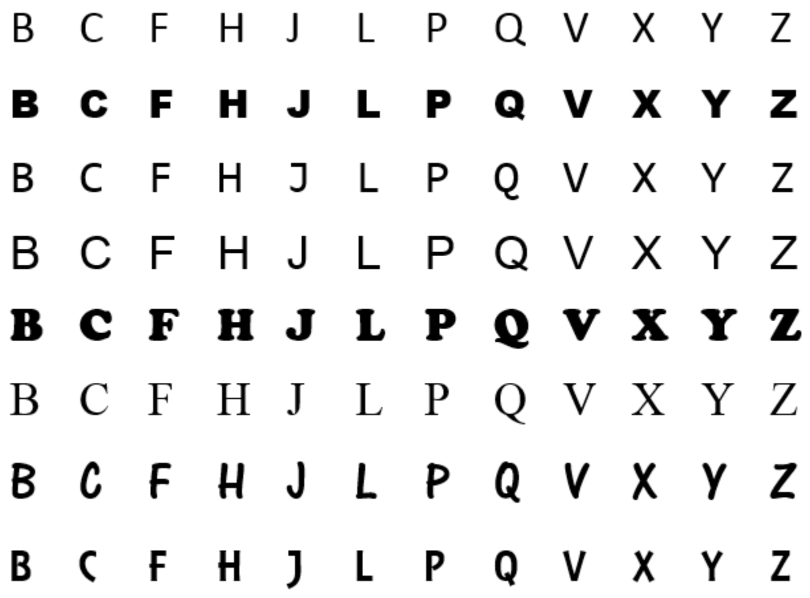


Figure 2.1 Prepared characters. Version without noise or other distortions.

letters were translated into datasets in *CSV* format. A long vector of zeros and ones (representing white or black, also according to some threshold) was concatenated to correct output. That kind of file was used to train the network.

The neural network was trained using following parameters:

- learning rate = 0.01, establishes how fast the weight changes during training process.
- momentum = 0.5, used for preventing network from converging to local minimum.
- back-propagation as a teaching algorithm.

Beside presented above "clean" sets, there are also prepared "noisy" ones. To be more precise - using written for that purpose Python's script, the pictures with some percentage of noise were generated as well as ones with some blank lines (number of lines also can be passed as an argument).

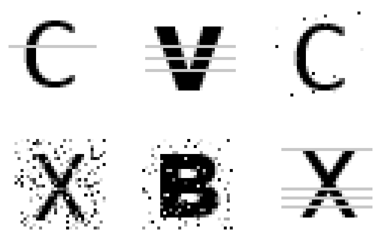


Figure 2.2 Examples of broken images.

2.4 Implementation

Core script for creating structure and proper set-up of designed network:

```python
#!/usr/bin/python

from pybrain.structure import FeedForwardNetwork
from pybrain.structure import FullConnection
from pybrain.structure import LinearLayer, SigmoidLayer, TanhLayer
from pybrain.tools.xml.networkwriter import NetworkWriter
from pybrain.tools.xml.networkreader import NetworkReader

class MyNet:

        def __init__(self, file='config.xml'):
                self.net = FeedForwardNetwork()
                self.file = file

        def constructNet(self, input, hidden, output):
                inputLayer = LinearLayer(input)
                hiddenLayer = TanhLayer(hidden)
                outputLayer = LinearLayer(output)

                self.net.addInputModule(inputLayer)
                self.net.addModule(hiddenLayer)
                self.net.addOutputModule(outputLayer)

                conn1 = FullConnection(inputLayer, hiddenLayer)
                conn2 = FullConnection(hiddenLayer, outputLayer)

                self.net.addConnection(conn1)
                self.net.addConnection(conn2)

        def setup(self):
                self.net.sortModules()

        def saveToFile(self,file='config.xml'):
                NetworkWriter.writeToFile(self.net, file)

        def loadFromFile(self, file='config.xml'):
                self.net = NetworkReader.readFrom(file)

if __name__ == "__main__":
        network = MyNet()
        network.constructNet(900,60,4)
        network.setup()

        network.saveToFile()
```

Beside presented core program, there were also written following scripts:

- train.py - network trainer, that using backpropagation algorithm performed teaching process with given parameters and storing results in xml file,
- liner.py - performs blanking lines in pictures,
- noiser.py - adds noise to picture - with some probability the pixel can be randomly set between white and black.
- setbuilder.py - transforms images in given folder (recursively) into a CSV file,
- validator.py - perfumes image translation into Morse code and validates the result. A set of images can be given. At the end the statistics of successful operations will be displayed.

Chapter 3

Tests

Network is build and trained. The learned datasets are well recognized (100% in any case).
The tests were executed using noisy images. But the clue is to compare results between them. In order to do that, two versions of neural network were prepared: one uses sigmoid as an activation function in hidden layer, the other one - hyperbolic tangent.

Presented here results will establish which one is better in case of presented project.

3.1 Images with noise

The most common distortion is white noise. To check network beahaviour, the pictures with some noisiness were generated. Then using validator, the tests were performed and statistics gathered

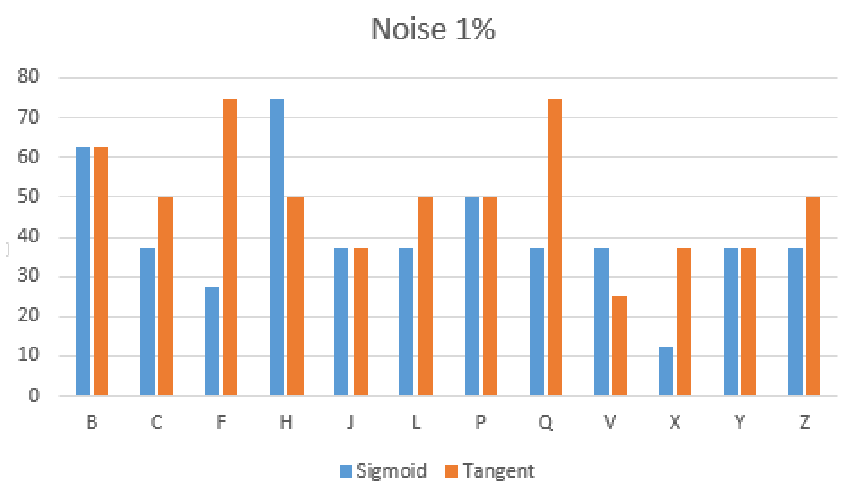


Figure 3.1 Experiment - build dataset with noise = 0.01

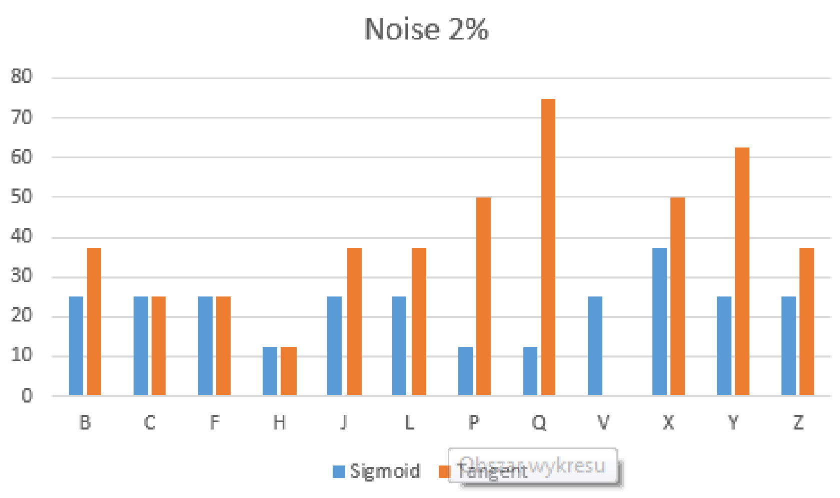


Figure 3.2 Experiment - build dataset with noise = 0.02

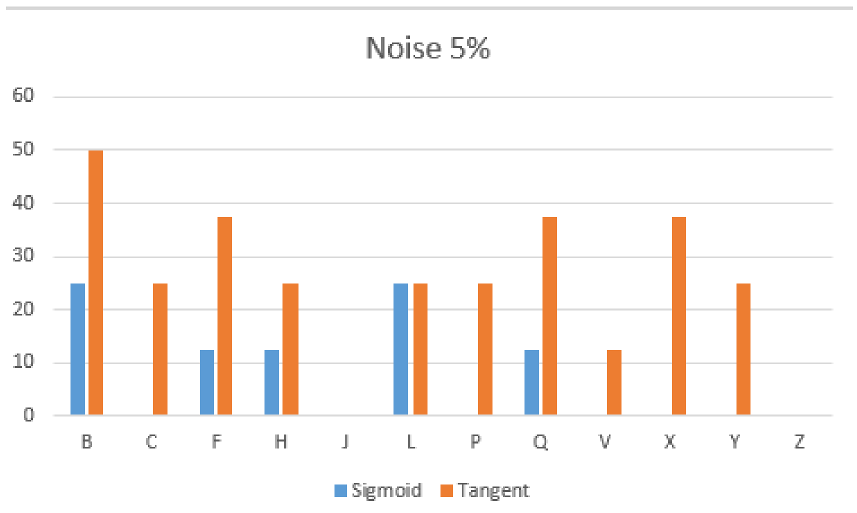


Figure 3.3 Experiment - build dataset with noise = 0.05

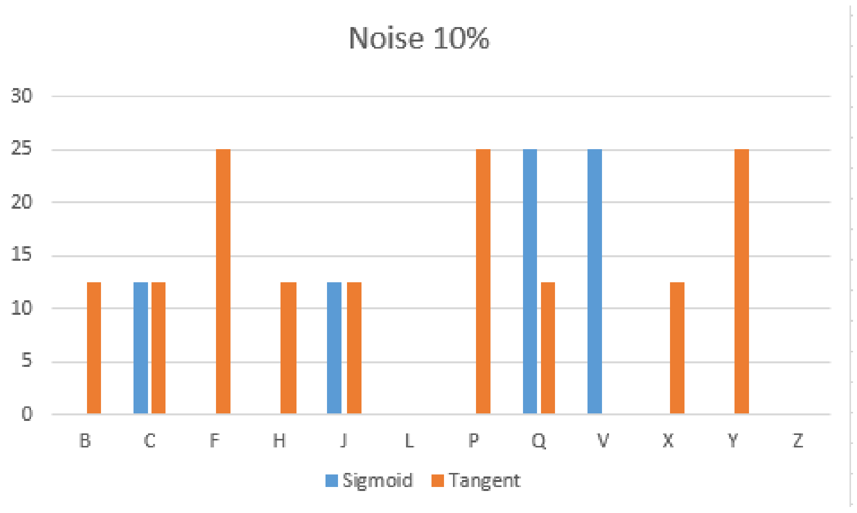


Figure 3.4 Experiment - build dataset with noise = 0.10

Results
As the picture presents, in most cases the hyperbolic tangent is better. When sigmoid cannot recognize the letters, the hyperbolic tangent still is doing correctly.

The first letter that is not recognized is 'V'.

3.2 Images with missing lines

That kind of distortion can be noticed e.g. using printer. Some lines can be not covered and we receive white lines. Will the network be able to recognize such a thing?

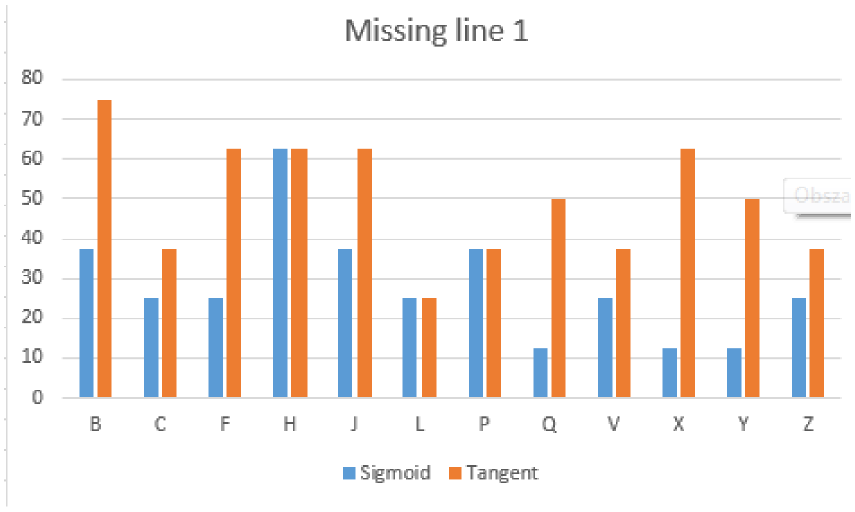


Figure 3.5 One blank line

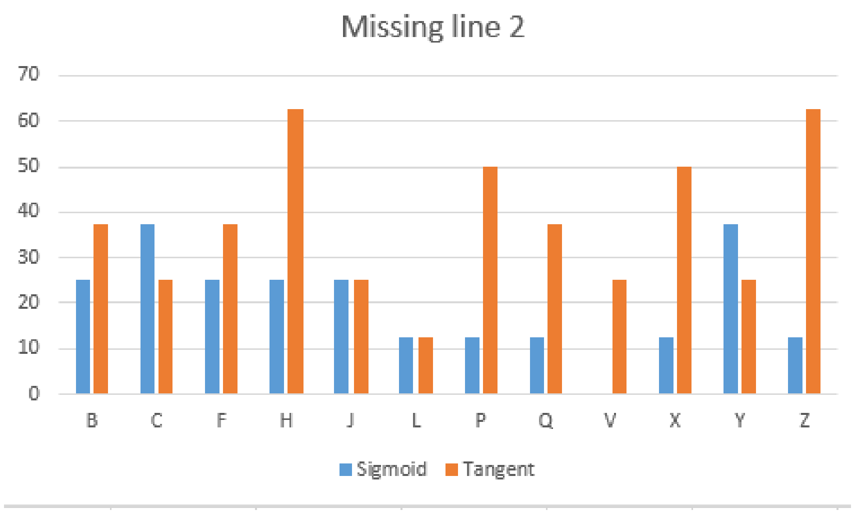


Figure 3.6 Two blank lines

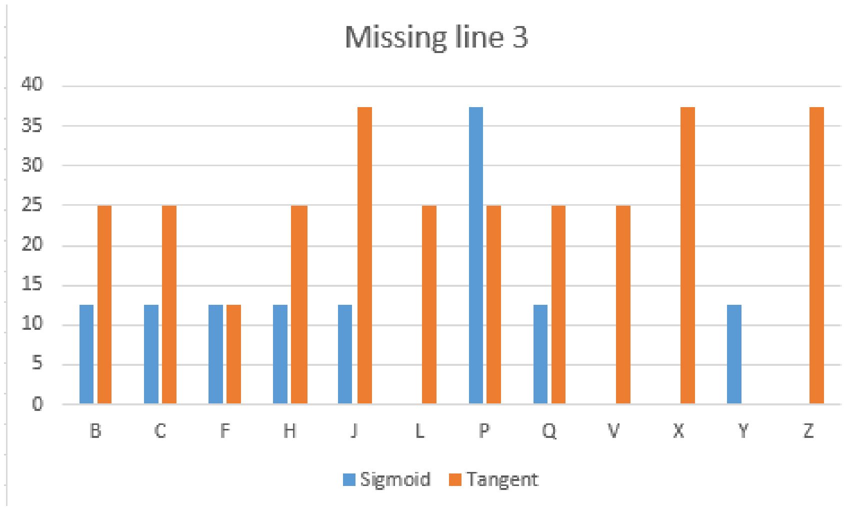


Figure 3.7 Three lines that interrupts recognition.

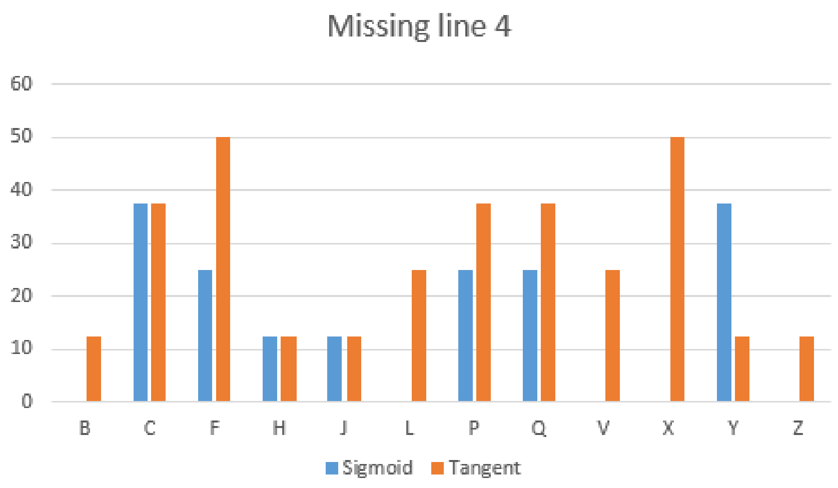


Figure 3.8 Four lines - algorithm overloading.

Results

As in the previous test - also here the tangent gets proper results more often, but still there are letters when the sygmoid is simply better.
Tangent can recognize most of weakly-noised images.

Chapter 4

Extras

Beside implementing neural network and performing tests, also the visual effect can be shown.

Using Raspberry Pi ?? there was implemented simple script that takes Morse codes as parameter and using LED transmitts light signals.

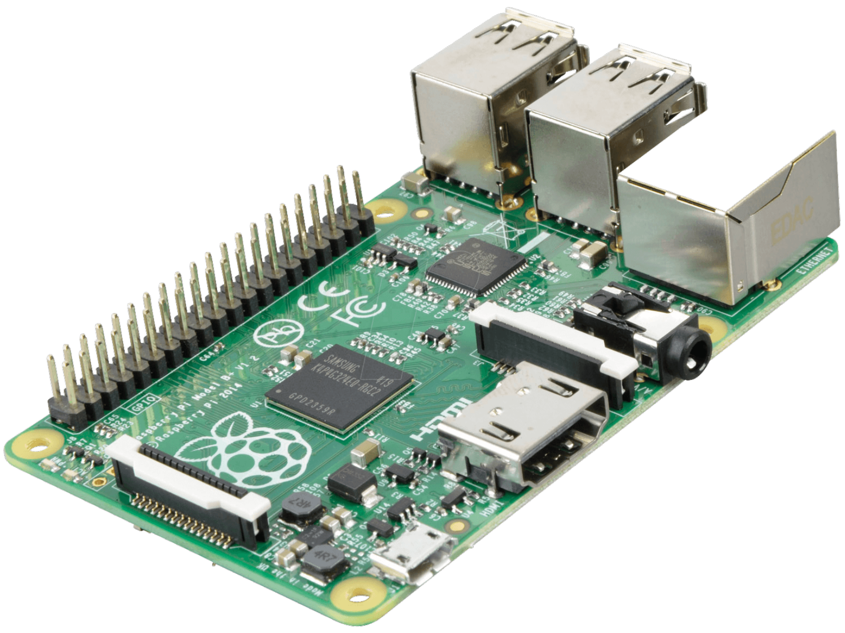


Figure 4.1 Raspberry PI model B+.

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import sys

PIN = 7
dot_length = 0.3
dash_length = dot_length*3
pause = dot_length
```

```
def blink(timer):

        GPIO.output(PIN, True)
        time.sleep(timer)
        GPIO.output(PIN, False)
        time.sleep(pause)

def dot():
        print 'dot'
        blink(dot_length)

def dash():
        print 'dash'
        blink(dash_length)

def control(arr):
        for e in arr:
                if e == '0':
                        dot()
                if e == '1':
                        dash()

def setup():
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(PIN, GPIO.OUT)
        GPIO.setwarnings(False)

def process():
        list = [sys.argv[1],sys.argv[2],sys.argv[3],sys.argv[4]]
        return list

def main():

        setup()
        if len(sys.argv)==5:
                code = process()
                control(code)

main()
```

It can be connected to neural network (e.g. using tcp messages in ZMQ protocol). That way, there was obtained practical system that not only recognizes the characters, but after translating them into Morse codes, it can also send visual signals.

Chapter 5

Conclusions

Despite well-known structure of neural network type, solutions has to be independently decided.
Even small things such as hidden layer activation function can has a huge impact on results.

In presented translation system had better results using hyperbolic tangent. And it was not so well trained as sygmoid, but still achieved better.
Training can last long in order to achieve proper results (the longest round of 10 000 epochs took almost 8 hours).
The neural network can be applied in simple projects. The example can be presented here Morse transmitter based on RPi. It does not require much work, but shows practical approach and possibility of application.

Final remarks

To build working network and achieve good results the tests should be done. They can help to establish if activation function is properly set or not.
Time should also be given to train the network. It is process that will determine the error of network, so it is worth to spend some time on it.

Bibliography

[1] Softcomputing course slides, Jacek Mazurkiewicz

[2] pyBrain documentation and tutorial, CogBotLab & Idsia

[3] Artificial Neural Networks, Wikibooks

[4] PILLOW - fork od Python Imaging Library documentation.