UDC 004.8 (045)

[1]V. M. Sineglazov,
[2]D. S. Raduchych

# ANALYSIS OF USING SOFTWARE PACKAGES FOR IMAGING IN MEDICAL DIAGNOSTICS

Aviation Computer-Integrated Complexes Department, National Aviation University, Kyiv, Ukraine
E-mails: [1]svm@nau.edu.ua, [2]raduchich@i.ua

***Abstract****—The most popular software packages were analyzed using artificial neural networks. Libraries Treano and Torch were investigated.*

**Index Terms—**Automation; deep learning; image processing; medicine; neural networks; software.

## I. INTRODUCTION

The wide range of tasks is solving, with computer processing of medical images, such as improving the quality of images; calculation clinically important quantitative parameters; recognition and image compression.

The digital form facilitates the processing of image storage and transmission of medical image data. Information technology can assist in all phases of obtaining and processing of medical images

Medical image of organism is the main source of information in the diagnosis and treatment of which provide the bulk of information about the patient and his disease.

## II. NECESSITY OF IMAGE PROCESSING OF LIVER DISEASE

All variety of medical images, regardless of the way they are received may be classified as a two main groups: analog and matrix image.

By matrix images include those that are received by a computer. Matrix images are images obtained with CT, digital X-beamgraphy, MR imaging, EOM-scintigraphy with computer information processing, ultrasonic scanning. Since the basis of the matrix image is computerized technology, they are available for various processing on a computer.

Digital image processing can be used to:

– improving the image quality compensation system defects, recording and noise reduction;

– calculating clinically important quantitative parameters (distance, area, volume, etc.);

– facilitate interpretation (identification structure).

## III. STAGES OF PROCESSING MEDICAL IMAGES

Highly specialist spent from one to two hours per patient for diagnosing fibrosis, even if there are such costs of resource and a sufficient amount of information, doctors can diagnose fibrosis only on the second or third stage, making it impossible to timely treatment of fibrosis. Using neural networks for processing medical images gives us several advantages, including.

1. Neural networks are nonlinear systems to classify data much better than linear methods.
2. Neural networks can work with lots of data, which influence the decision of the diagnosis, a person can`t be estimated.
3. Neural networks are able to make decisions based on them discovered hidden patterns in multidimensional data.
4. Trained network demonstrated its capacity to identify and take into account the very complex relationships prognostic variables, including their triple bonds to increase the accuracy of foresight.
5. Ability of create a neural network architecture special for specific tasks.
6. Neural networks can be used to forecast the actions developed various treatments.
7. Network is no "human factor" in the neural.

## IV. PROBLEM STATEMENT

There are many software packages for processing information`s with neural networks. Each of these packages has its own functionality, however, these software tools implemented at the appropriate level of its functionality for model problems which are given in the documentation .Using these packages for solving real practical problems raises difficulties for the user. The aim is the objective research of the data packet and issuing recommendations for their use.

## V. THE ANALYSIS OF USED PACKAGES

Possibilities of neural networks implemented in a variety of different software products, is appropriate libraries for these programming languages like C / C ++, Pascal, Java, Scala, Python, R and Visual Basic. Developers and users of solutions based on neural networks show special attention to products based on open source technologies, as there is a synergistic effect of joint development work with open source technologies and tools for Big Data

Open Source, used in the scientific community and which absorbed the achievements of fundamental science.

There are many software tools to meet the challenges of deep learning. The "most popular" among them are in Table I.

The first six software libraries implementing the widest range of methods of deep learning. Developers provide opportunities to create fully connected neural networks (fully connected neural network, FC NN), convolutional neural networks (convolutional neural network, CNN), autoencoder (autoencoder, AE) and restricted Boltzmann machines (restricted Boltzmann machine, RBM).

It would be desirable to draw attention to the last four libraries. Despite the fact that they have less functionality and in some cases their simplicity helps achieve greater productivity.

TABLE I

"MOST POPULAR" AMONG SOFTWARE TOOLS

| **Name** | **Programming language** | **FCNN** | **CNN** | **AE** | **RBM** |
|---|---|---|---|---|---|
| DeepLearn Toolbox | Matlab | + | + | + | + |
| Theano | Python | + | + | + | + |
| Pylearn2 | Python | + | + | + | + |
| Deepnet | Python | + | + | + | + |
| Deepmat | Matlab | + | + | + | + |
| Torch | Lua, C | + | + | + | + |
| Darch | R | + | — | + | + |
| Caffe | C++, Python, Matlab | + | + | — | — |
| nnForge | C++ | + | + | — | — |
| CXXNET | C++ | + | + | — | — |
| Cuda-convnet | C++ | + | + | — | — |
| Cuda CNN | Matlab | + | + | — | — |

Based on the above information and recommendations to specialists for further review selected four libraries: Theano, Pylearn 2 – one of the most mature and complete functional libraries, Torch and Caffe – widely used Community. Each library is seen on the following schedule.

1. Brief background.
2. Technical features (operating system, programming language, depending).
3. Functionality.
4. An example of the formation of networks such as logistic regression.
5. Training and use of the constructed model for classification.

### *A. Library Theano*

Theano – an extension language of Python, to effectively calculate mathematical expressions containing multidimensional arrays. Theano was developed in the laboratory LISA for supporting the rapid development of machine learning algorithms.

The library implemented in Python, supported on operating systems Windows, Linux and Mac OS. The structure Theano is a compiler that converts mathematical expressions written in Python in efficient code in C or CUDA.

Theano provides a basic set of tools for configuration of neural networks and their training. Possible implementation multilayer fully connected network (Multi-Layer Perceptron), convolutional neural network (CNN), recurrent neural networks (Recurrent Neural Networks, RNN), auto coders and restricted Boltzmann machines. Additionally, activating various functions, including sigmoid, softmax-function, cross-entropy. During the study used a batch gradient descent (Batch SGD).

Think about a neural network configuration Theano. For your convenience, we realize class which will contain variables – the exercise parameters W, b and functions to work with them – counting answers Network (y = softmax (Wx + b)) and function errors. Then, to create neural network training function train_model. For her to describe methods that determine the function of error calculation rule gradients way to change neural network weights, size and location of the mini-batch sample (same images and answers for them). After determining the function of all parameters compiled and transmitted in a cycle of learning (Fig. 1).

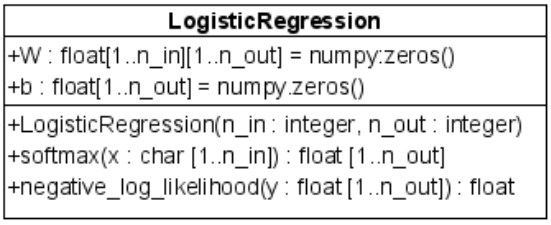


Fig. 1. The scheme of class for implementing neural network Theano

Software implementation class:

```
class LogisticRegression(object):

    def __init__(self, input, n_in, n_out):

        self.W = theano.shared(
```

```
                value=numpy.zeros((n_in, n_out), dtype=theano.config.floatX), name='W', borrow=True)
        self.b = theano.shared(value=numpy.zeros((n_out,), dtype=theano.config.floatX), name='b', borrow
=True)

        self.p_y_given_x = T.nnet.softmax(T.dot(input, self.W) + self.b)
        self.y_pred = T.argmax(self.p_y_given_x, axis=1)
        self.params = [self.W, self.b]

    def negative_log_likelihood(self, y):
        return -T.mean(T.log(self.p_y_given_x)[T.arange(y.shape[0]), y])
x = T.matrix('x')
y = T.ivector('y')
classifier = LogisticRegression(input=x, n_in=28 * 28, n_out=10)
cost = classifier.negative_log_likelihood(y)
g_W = T.grad(cost=cost, wrt=classifier.W)
g_b = T.grad(cost=cost, wrt=classifier.b)
updates = [(classifier.W, classifier.W - learning_rate * g_W),  (classifier.b, classifier.b - learning_r
ate * g_b)]
train_model = theano.function(
    inputs=[index], outputs=cost, updates=updates,
    givens={
        x: train_set_x[index * batch_size: (index + 1) * batch_size],
        y: train_set_y[index * batch_size: (index + 1) * batch_size]
    }
)
```

It is easy to see that the process of creating the model and determining its parameters requires writing code volume. The library is low-level. It should be noted its flexibility and the availability and feasibility of using its own components.

*B. Library Torch*

Torch – library for scientific computing with broad support machine learning algorithms. Developed Idiap Research Institute, New York University and NEC Laboratories America, starting in 2000, Licensed under BSD.

Library language Lua is implemented using C and CUDA. Quick Lua scripting language together with technology SSE, OpenMP, CUDA allow Torch show a good rate compared to other libraries. Currently supported operating systems Linux, FreeBSD, Mac OS X. The core modules also work on Windows. Depending Torch packages are imagemagick, gnuplot, nodejs, npm and others.

The library consists of a set of modules, each of which is responsible for the different stages of work on neural networks. For example, the module provides nn. neural network configuration (defining layers and their parameters), the module provides optim implementation of various optimization methods used for teaching and gnuplot provides data visualization (graph, display images, and so on. D.). Installing additional modules can extend the functionality of the library.

Torch allows you to create complex neural networks through the mechanism of containers. Container – a class that combines neural network components declared in a common configuration that may subsequently be transferred to the learning process. Part of neural networks can not only be fully connected or convolutional layers, but the activation function or error, and ready containers. Torch allows you to create the following layers.

– Full coupling layer (Linear).

– Function activation: the hyperbolic tangent (Tanh), choose the minimum (Min) or maximum (Max), softmax-function (SoftMax) and others.

– Rolls layers: convolution (Convolution), thinning (SubSampling), spatial association (MaxPooling, AveragePooling, LPPooling), (SubtractiveNormalization).

Features errors: medium square error (MSE), the cross-entropy (CrossEntropy) and others.

When training can use the following optimization techniques:

– stochastic gradient descent (SGD);

– averaged stochastic gradient descent (Averaged SGD);

– algorithm Broydena – Fletcher – Goldfarb – Shanna (L-BFGS);

– conjugate gradient method (Conjugate Gradient, CG).

Consider the process of configuring the neural network Torch. You must first declare a container, then add the layers. The order of adding layers is important because the output ($n$–1)th layer to the $n$th input.

```
regression = nn.Sequential()
regression:add(nn.Linear(784,10))
regression:add(nn.SoftMax())
loss = nn.ClassNLLCriterion()
```

Using neural networks and learning.

1. Loading input function X. torch.load (path_to_ready_dset) allows you to load pre-prepared dataset in text or binary format. Typically, this Lua-table consists of three fields: the size of the data and labels. If the finished dataseta not, you can use the standard functions of language Lua (eg, io.open (filename [, mode])) or packet functions of libraries Torch (eg, image.loadJPG (filename)).

2. Defining network response to input X:

```
Y = regression:forward(X)
```

3. Calculate error function $E$ = loss (Y, T), in this case the likelihood function.

```
E = loss:forward(Y,T)
```

4. Calculate gradients according to the algorithm back-propagation.

```
dE_dY = loss:backward(Y,T)
regression:backward(X,dE_dY)
```

Now gather everything together. To teach a neural network library Torch, you must write your own training cycle. It declare a special function (closure) which will calculate response network errors and determine the value transfer gradients and refer the circuit in the gradient descent function to update the weights of the network.

```
w, dE_dw = regression:getParameters()

local eval_E = function(w)
    dE_dw:zero()
    local Y = regression:forward(X)
    local E = loss:forward(Y,T)
    local dE_dY = loss:backward(Y,T)
    regression:backward(X,dE_dY)
    return E, dE_dw
end
optim.sgd(eval_E, w, optimState)
```

where optimState – gradient descent parameters (learningRate, momentum, weightDecay etc.).

It is easy to see that the ads procedure as the procedure training takes less than 10 lines of code, which indicates the ease of use of the library. This library allows to work with neural networks at a very low level.

Saving and loading network trained by using special features:

```
torch.save(path, regression)
net = torch.load(path)
```

After loading the network can be used for classification or additional training. If you need to know to which class belongs element sample, then simply follow the passage on the network and calculate the output:

```
result = net:forward(sample)
```

## VI. Conclusion

Summarizing, we can say that the most mature library is Torch. This Theano not yield her on many criteria (see Table I), so we can not exclude the possibility of her further use.

## References

[1] V. D. Kustikova and P. N. Druzhkov, “A Survey of Deep Learning Methods and Software for Image Classification and Object Detection.” *In: Proc. of the 9th Open German-Russian Workshop on Pattern Recognition and Image Understanding*, 2014.

[2] Y. Le Cun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision.” *In: Proc. of the IEEE Int. Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 253–256.

[3] M. Hayat, M. Bennamoun, and S. An, “Learning Non-Linear Reconstruction Models for Image Set Classification.” *In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2014.

Received October 03, 2016.

**Sineglazov Viktor.** Doctor of Engineering Science. Professor.
Educational and Scientific Institute of Information and Diagnostic Systems, National Aviation University, Aviation Computer-Integrated Complexes Department, Kyiv, Ukraine.
Education: Kiev Polytechnic Institute. Kyiv, Ukraine, (1973).
Research interests: Air Navigation, Air Traffic Control, Identification of Complex Systems, Wind/solar power plant.
Publications: more than 600 papers.
E-mail: svm@nau.edu.ua

**Denys Raduchych**. Student.
Educational and Scientific Institute of Information and Diagnostic Systems, National Aviation University, Aviation Computer-Integrated Complexes Department, Kyiv, Ukraine.
Research interests: Artificial Neural Network.
E-mail: raduchich@i.ua

**В. М. Синєглазов, Д. С. Радучич. Аналіз використання програмних пакетів для обробки зображень у медичній діагностиці**

Проаналізовано найпопулярніші пакети програмного забезпечення для реалізації обробки інформації з використанням штучних нейронних мереж. Досліджено бібліотеки Treano і Torch.

**Ключові слова:** автоматизація; глибоке навчання; обробка зображень; медицина; пакети програмного забезпечення; штучні нейронні мережі.

**Синєглазов Віктор Михайлович.** Доктор технічних наук. Професор.
Навчально-науковий інститут інформаційно-діагностичних систем, кафедра авіаційних комп'ютерно-інтегрованих комплексів, Національний авіаційний університет, Київ, Україна.
Освіта: Київський політехнічний інститут. Київ, Україна, (1973).
Напрям наукової діяльності: аеронавігація, управління повітряним рухом, ідентифікація складних систем, вітроенергетичні установки, штучні нейронні мережі.
Кількість публікацій: більше 600 наукових робіт.
E-mail: svm@nau.edu.ua

**Радучич Денис Сергійович** . Студент.
Навчально-науковий інститут інформаційно-діагностичних систем, кафедра авіаційних комп'ютерно-інтегрованих комплексів, Національний авіаційний університет, Київ, Україна.
Напрям наукової діяльності: штучні нейронні мережі.
E-mail: raduchich@i.ua

**В. М. Синеглазов, Д. С. Радучич. Анализ использования программных пакетов для обработки изображений в медицинской диагностике**

Проанализированы самые популярные пакеты программного обеспечения для реализации обработки информации с использованием искусственных нейронных сетей. Исследованы библиотеки Treano и Torch.

**Ключевые слова**: автоматизация; глубокое обучение; обработка изображений; медицина, пакеты программного обеспечения; искусственные нейронные сети.

**Синеглазов Виктор Михайлович**. Доктор технических наук. Профессор.
Учебно-научный институт информационно-диагностических систем, кафедра авиационных компьютерно-интегрированных комплексов, Национальный авиационный университет, Киев, Украина.
Образование: Киевский политехнический институт. Киев, Украина, (1973).
Направление научной деятельности: аэронавигация, управление воздушным движением, идентификация сложных систем, ветроэнергетические установки, искусственные нейронные сети.
Количество публикаций: более 600 научных работ.
E-mail: svm@nau.edu.ua

**Радучич Денис Сергеевич**. Студент.
Учебно-научный институт информационно-диагностических систем, кафедра авиационных компьютерно-интегрированных комплексов, Национальный авиационный университет, Киев, Украина.
Направление научной деятельности: искусственные нейронные сети.
E-mail: raduchich@i.ua