UDC 004.8 (045)
DOI: 10.18372/1990-5548.53.12151

[1]V. M. Sineglazov,
[2]M. O. Omelchenko,
[3]V. P. Hotsyanivskyy

COMPARATIVE CHARACTERISTICS OF KERAS AND LASAGNE MACHINE LEARNING PACKAGES

Educational & Scientific Institute of Information and Diagnostic Systems,
National Aviation University, Kyiv, Ukraine
E-mails: [1]svm@nau.edu.ua, [2]max.omel4enko@yandex.ua, [3]sttt912@yahoo.com

Abstract—*A comparative analysis of the Lasagne and Keras computer libraries has been performed to construct convolutional neural networks used in image processing systems. The carried out researches have allowed to define their advantages and disadvantages that will allow to make to researchers the correct choice at the decision of applied problems.*

Index Terms—Automation deep learning; image processing; convolutional neural networks.

I. INTRODUCTION

With the rapid growth of computerization and increased computing power, there is a problem of processing graphical information, which is needed to solve a number of problems. Information technology can help in processing these images.

For example, medical imaging of organs is the main source of information when diagnosis and treatment are provided, which give the main body of information about the patient and his illness.

When processing images it is necessary to solve a wide range of tasks, such as:

– improvement of image quality;
– pattern recognition;
– optimization of image parameters, etc.

Traditional approaches to addressing these problems do not always provide the necessary flexibility and many applications will benefit from the use of intelligent nonlinear system. The intelligent system is a nonlinear system that will better categorize data than commonly used linear methods.

Such a system is linear neural network, because it can make decisions based on the data found in the hidden patterns. A distinctive feature of neural networks is that they do not use any inference rules for diagnosis and learn to do it by examples.

Neural networks they are electronic models of the neural structure of the brain, which mainly learns from experience. A natural analog shows that the set of problems that are not yet subject to the resolution of existing computers, but can be effectively solved by convulated neural networks [1].

II. PROBLEM STATEMENT

Rolls neural network process image is not full and separate "chunks", consistently reducing its size or highlighting the most important characteristic signs, going to a new level of abstraction. These networks are formed so-called card signs that an outside observer seem blurred, distorted copies of the original image, but the neural network are fundamentally different meaning, because they contain the characteristic features of the desired areas.

The basic idea convolutional neural network is subdescretes alternating layers, layers of rolls and output layers. So combined together architectural ideas that achieve invariance to distortion (Fig. 1).

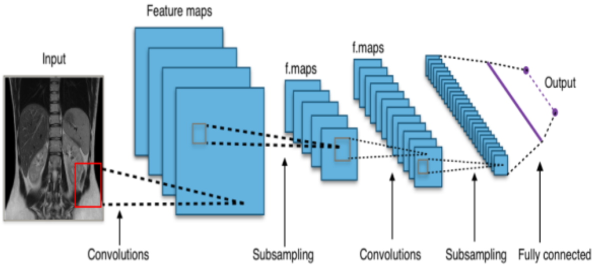

Fig. 1. Block diagram of the convolution neural network

Alternating layers allows card features make the cards the previous layer that promotes recognition of complex hierarchies of attributes. The model provides for the allocation of a large number of areas of the original image, and therefore, in this case, the rational use of computing power, able to quickly process large amounts of data.

There are currently dozens of libraries for working with neural networks, all of which are different in the implementation, but you can identify two basic approaches: imperative and symbolic.

Both approaches have their advantages and disadvantages. First of all, the imperative programs are more flexible, more visible and easier to set up. We can use all the richness of the programming language, and display intermediate results for debugging purposes. This flexibility is achieved, first of all, by the small constraints imposed on the interpreter, which should be ready for any further use of the variables.

On the other hand, the symbolic paradigm imposes more restrictions, but the calculations are more effective both in terms of memory and speed of execution: at the compilation stage, you can apply a number of optimizations, identify unused variables, perform part of the calculations, and so on.

We dwell on this in such detail, because the imperative paradigm is familiar to most programmers, while the symbolic may seem unusual, and Theano, an obvious example of a symbolic framework.

Theano is a library used to develop machine learning systems by itself and as a computational backend for higher-level libraries, such as Lasagne and Keras.

To this point, we already understand the main stages of creating machine learning systems with the help of Theano, such as:

– initialization of input variables;

– definition of the model;

– compilation of Theano-functions;

– cycle with optimizer steps.

III. Problem Solution

In the process of working with Theano framework [2], it was concluded that this is a great tool, but the development of machine learning systems with its help is difficult, so you should pay attention to such libraries as Lasagne and Keras, which can ease the process of developing the system(Table I).

Lasagne is a great library for developing neural networks that works as an add-on for Theano. Lasagne provides a set of ready components: layers, optimization algorithms, loss functions, parameter initializations, etc., while not hiding Theano on numerous layers of abstractions, which allows you to quickly edit the desired element of the program as needed by the programmer.

The project was launched by Sander Dieleman in September 2014. The library was developed by a team of eight people (Eric Battenberg, Sander Dieleman, Daniel Nouri, Eben Olson, Aäron van den Oord, Colin Raffel, Jan Schlüter, Søren Kaae Sønderby) and numerous additional members of the github community.

Lasagne modular library. That is, you can connect individual modules, roller coasters, optimizers, schematics of initialization, activation functions, regularization schemes. It provides different classes that represent the layers of the neural network. All of them are subclasses of the lasagne.layers base class.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. Use Keras if you need a deep learning library that:

– allows for easy and fast prototyping (through user friendliness, modularity, and extensibility);

– supports both convolutional networks and recurrent networks, as well as combinations of the two;

– runs seamlessly on CPU and GPU.

– Keras is compatible with: Python 2.7-3.6 [3].

User friendliness. Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.

Modularity. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.

Easy extensibility. New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

Work with Python. No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility (Figs 2 and 3).

When comparing packages, the following comparison critics were identified:

A) Functionality

Lasagne reveals Theano more than Keras. Because its main purpose is to use the Theano computing model to write symbolic expressions, using, and simplifying some elements of the work with it – to make it convenient to use the functions of Theano.

TABLE I. LAYERS OF KERAS AND LASAGNE MACHINE LEARNING PACKAGES

Lasagne layers	
InputLayer	This layer holds a symbolic variable that represents a network input. A variable can be specified when the layer is instantiated, else it is created.
DenseLayer	A fully connected layer.
Conv2DLayer	Performs a 2D convolution on its input and optionally adds a bias and applies an element wise nonlinearity.
TransposedConv2DLaye r	Performs the backward pass of a 2D convolution (also called transposed convolution, fractionally-strided convolution or deconvolution in the literature) on its input and optionally adds a bias and applies an element wise nonlinearity.
DilatedConv2DLayer	Performs a 2D convolution with dilated filters, then optionally adds a bias and applies an elementwise nonlinearity.
MaxPool2DLayer	Performs 2D max-pooling over the two trailing axes of a 4D input tensor.
Pool2DLayer	Performs 2D mean or max-pooling over the two trailing axes of a 4D input tensor.
Upscale2DLayer	Performs 2D upscaling over the two trailing axes of a 4D input tensor.
ConcatLayer	Concatenates multiple inputs along the specified axis. Inputs should have the same shape except for the dimension specified in axis, which can have different sizes.
BatchNormLayer	The input is normalized to zero mean and unit variance, and then linearly transformed. The crucial part is that the mean and variance are computed across the batch dimension, i.e., over examples, not per example.
Keras layers	
Conv2D	2D convolution layer (e.g. spatial convolution over images).This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs.
SeparableConv2D	Separable convolutions consist in first performing a depth wise spatial convolution (which acts on each input channel separately) followed by a point wise convolution which mixes together the resulting output channels. The depth_multiplier argument controls how many output channels are generated per input channel in the dept hwise step. Intuitively, separable convolutions can be understood as a way to factorize a convolution kernel into two smaller kernels, or as an extreme version of an Inception block.
UpSampling2D	Repeats the rows and columns of the data by size (0) and size (1) respectively.
MaxPooling2D	Max pooling operation for spatial data.
AveragePooling2	Average pooling operation for spatial data.
LocallyConnected2D	The LocallyConnected2D layer works similarly to the Conv2D layer, except that weights are unshared, that is, a different set of filters is applied at each different patch of the input.
BatchNormalization	Normalize the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1.
GaussianNoise	Apply additive zero-centered Gaussian noise. This is useful to mitigate over fitting (you could see it as a form of random data augmentation). Gaussian Noise (GS) is a natural choice as corruption process for real valued inputs. As it is a regularization layer, it is only active at training time.
AlphaDropout	Alpha Dropout is a Dropout that keeps mean and variance of inputs to their original values, in order to ensure the self-normalizing property even after this dropout. Alpha Dropout fits well to Scaled Exponential Linear Units by randomly setting activations to the negative saturation value.

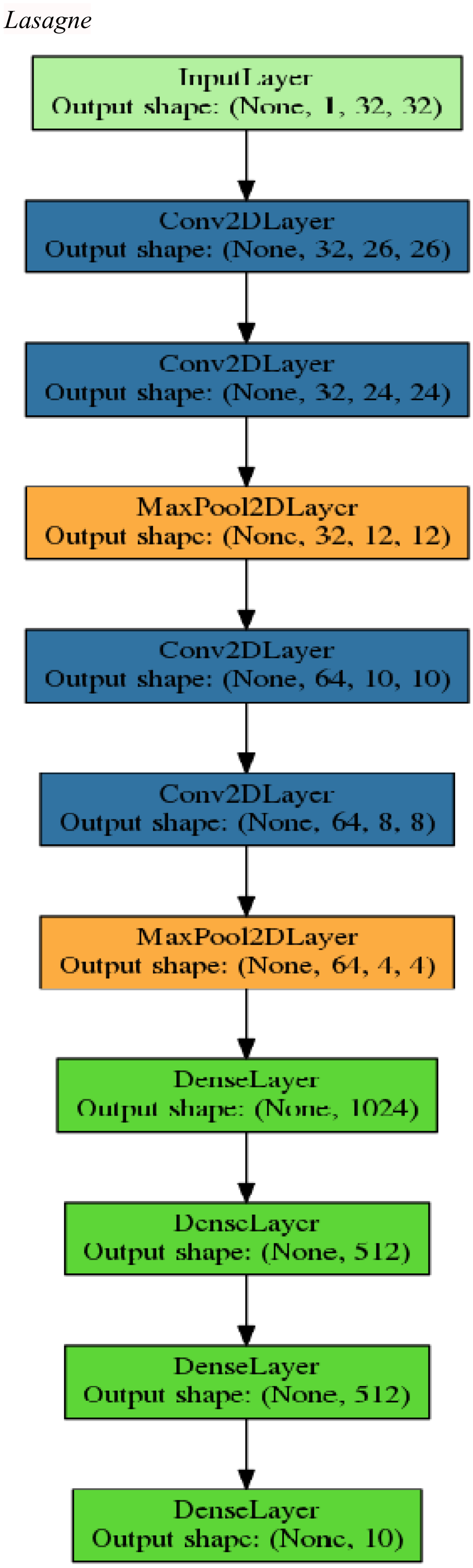
Lasagne
InputLayer
Output shape: (None, 1, 32, 32)
Conv2DLayer
Output shape: (None, 32, 26, 26)
Conv2DLayer
Output shape: (None, 32, 24, 24)
MaxPool2DLayer
Output shape: (None, 32, 12, 12)
Conv2DLayer
Output shape: (None, 64, 10, 10)
Conv2DLayer
Output shape: (None, 64, 8, 8)
MaxPool2DLayer
Output shape: (None, 64, 4, 4)
DenseLayer
Output shape: (None, 1024)
DenseLayer
Output shape: (None, 512)
DenseLayer
Output shape: (None, 512)
DenseLayer
Output shape: (None, 10)

Fig. 2. Structure of convolutional neural networks

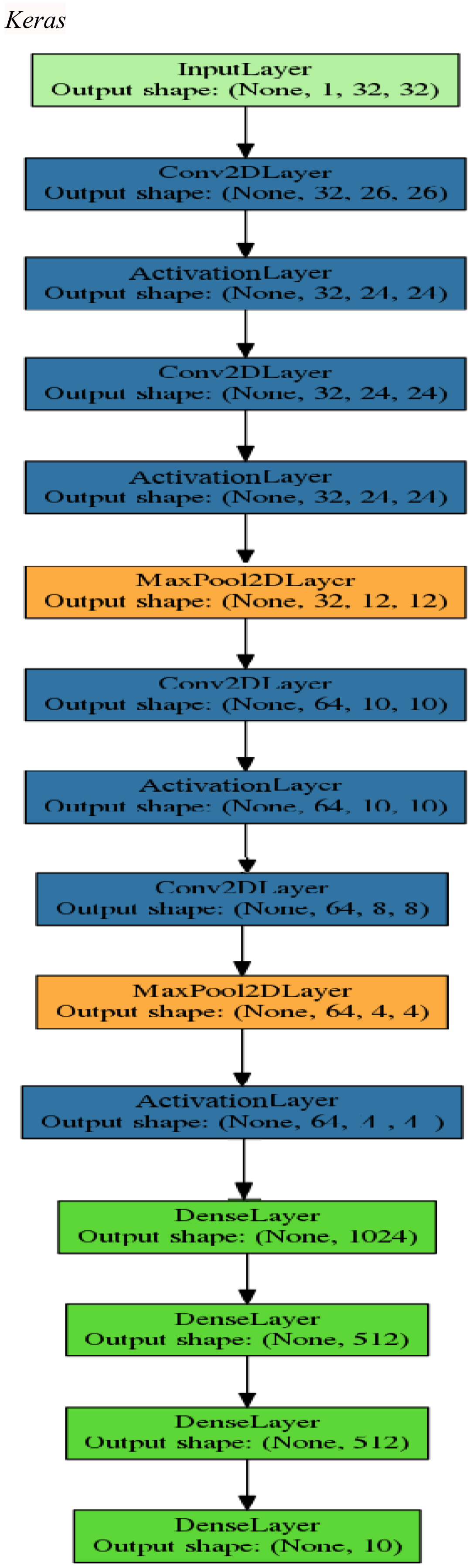
Keras
InputLayer
Output shape: (None, 1, 32, 32)
Conv2DLayer
Output shape: (None, 32, 26, 26)
ActivationLayer
Output shape: (None, 32, 24, 24)
Conv2DLayer
Output shape: (None, 32, 24, 24)
ActivationLayer
Output shape: (None, 32, 24, 24)
MaxPool2DLayer
Output shape: (None, 32, 12, 12)
Conv2DLayer
Output shape: (None, 64, 10, 10)
ActivationLayer
Output shape: (None, 64, 10, 10)
Conv2DLayer
Output shape: (None, 64, 8, 8)
MaxPool2DLayer
Output shape: (None, 64, 4, 4)
ActivationLayer
Output shape: (None, 64, 4 , 4)
DenseLayer
Output shape: (None, 1024)
DenseLayer
Output shape: (None, 512)
DenseLayer
Output shape: (None, 512)
DenseLayer
Output shape: (None, 10)

Fig. 3. Structure of convolutional neural networks

Another purpose is to very effectively write the methods of Theano. Kerass's main goal is to help people who do not know Theano write their first neural network within a few days. It is less flexible and less extensible than Lasagne (Tables II and III).

TABLE II. THE FIRST FIVE TRENING VALUES

Epoch	Time, s	Training error	Error function	PKR*
Lasagne				
1	119	2.318504	0.134	0.157
2	117	2.300581	0.166	0.146
3	119	2.292756	0.206	0.191
4	120	2.255572	0.248	0.232
5	119	2.200565	0.265	0.253
Keras				
1	153	1.877125	0.306	0.438
2	162	1.569455	0.421	0.513
3	145	1.442584	0.466	0.583
4	156	1.352436	0.515	0.581
5	155	1.283857	0.566	0.608

TABLE III. AVERAGE VALUE OVER 250 EPOCH

Lasagne		Keras	
Time	126	Time	149.2
Error function	0.983545	Error function	0.4509
Training error	0.0483	Training error	0.4548
PKR	0.0758	PKR	0.148

*The percentage of correct recognition.

B) Convenience

Lasagne is less convenient than Keras, since its main task is to simplify the writing of the code, while retaining the functionality of Théano.

Writing neural networks with Keras is extremely convenient, since it is the convenience of developers put in the first place, even if it affects the functionality of Théano.

C) Speed and precision

When studying both networks, the sample cifar 10 was used. The following network structures were selected for the test.

IV. CONCLUSIONS

With Lasagne / Keras you can easily create new networks and change existing ones. The Lasagne / Keras configuration options show that it is very easy to modify existing networks, as well as to custom-oriented user-defined data.

At the same time, Lasagne reveals Theano more than Keras. Because its main purpose is to use the Theano computing model to write symbolic expressions, using, and simplifying some elements of the work with it – to make it convenient to use the functions of Theano. Another purpose is to very effectively write the methods of Theano.

Moreover, the main purpose of Keras is to help people who do not know Theano write their first neural network in a few days. It is less flexible and less extensible than Lasagne.

REFERENCES

[1] A. N. Gordan, D. A. Rossiev, S. E. Gilev, et al. “NeuroComp” group: neural-networks software and its application. Russian Academy of Sciences, Krasnoyarsk Computing Center. Preprint no. 8. Krasnoyarsk, 1995. (in Russian)

[2] T. Lisman and R. J. Porte. Towards a rational use of low-molecular-weight heparin in patients with cirrhosis. Liver Int., London: Science of Medicine 2011.

[3] M. Riesenhuber. Object recognition in cortex: Neural mechanisms, and possible roles for attention Department of Neuroscience Georgetown University Medical Center, Washington, DC, 2007.

Received April 26, 2017

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

Omelchenko Maksym. Student.
Educational & Research Institute of Information and Diagnostic Systems, National Aviation University, Kyiv, Ukraine.
Research area: Artificial Neural Network.
Publications: 1.
E-mail: max.omel4enko@yandex.ua

Hotsyanivskyy Volodymyr. Student.
Educational & Research Institute of Information and Diagnostic Systems, National Aviation University, Kyiv, Ukraine.

Research area: Artificial Neural Network.
Publications: 1.
E-mail: sttt912@yahoo.com

В. М. Синєглазов, М. О. Омельченко, В. П. Хоцянівський. Порівняльні характеристики пакетів Keras та Lasagne для машиного навчання

Виконано порівняльний аналіз комп'ютерних бібліотек Lasagne і Keras для побудови згорткових нейронних мереж, які використовуються в інтелектуальних системах обробки зображень. Проведено дослідження, які дозволяють визначити переваги і недоліки названих бібліотек, це дає можливість користувачам зробити правильний вибір вирішуючи прикладні задачі.

Ключові слова: автоматизація глибокого навчання; обробка зображення; згорткові нейронні мережі.

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

Омельченко Максим Олегович. Студент.
Навчально-науковий Інститут інформаційно-діагностичних систем, Національний авіаційний університет, Київ, Україна.
Напрям наукової діяльності: штучні нейронні мережі.
Кількість публікацій: 1.
E-mail: max.omel4enko@yandex.ua

Хоцянівський Володимир Петрович. Студент.
Навчально-науковий Інститут інформаційно-діагностичних систем, Національний авіаційний університет, Київ, Україна.
Напрям наукової діяльності: штучні нейронні мережі.
Кількість публікацій: 1.
E-mail: sttt912@yahoo.com

В. М. Синеглазов, М. О. Омельченко, В. П. Хоцяновский. Сравнительные характеристики пакетов Keras и Lasagne для машинного обучения

Выполнен сравнительный анализ компьютерных библиотек Lasagne и Keras для построения сверточных нейронных сетей используемых в интеллектуальных системах обработки изображений. Проведены исследования, которые позволяют определить преимущества и недостатки названных библиотек, это дает возможность пользователям сделать правильный выбор при решении прикладных задач.

Ключевые слова: автоматизация глубокого обучения; обработка изображения; сверточные нейронные сети.

Синеглазов Виктор Михайлович. Доктор технических наук. Профессор.
Учебно-научный институт информационно-диагностических систем, Национальный авиационный университет, Киев, Украина.
Образование: Киевский политехнический институт.
Направление научной деятельности:
Количество публикаций: более 600 научных работ.
E-mail: svm@nau.edu.ua

Омельченко Максим Олегович. Студент.
Учебно-научный институт информационно-диагностических систем, Национальный авиационный университет, Киев, Украина.
Направление научной деятельности: искусственные нейронные сети.
Количество публикаций: 1.
E-mail: max.omel4enko@yandex.ua

Хоцяновский Владимир Петрович. Студент.
Учебно-научный институт информационно-диагностических систем, Национальный авиационный университет, Киев, Украина.
Направление научной деятельности: искусственные нейронные сети.
Количество публикаций: 1.
E-mail: sttt912@yahoo.com