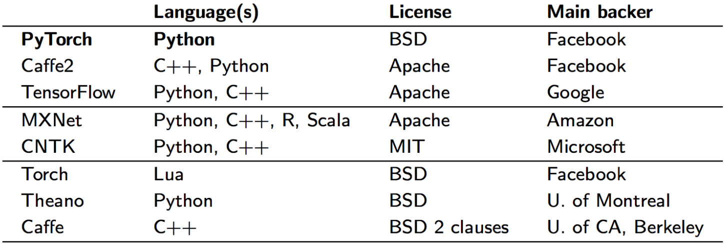
Introduction to Deep Learning with R

Sophie Donnet for SatRdays

02/2018

What are solutions for Deep Learning in R

The success of deep neural networks has led to a wide range of frameworks and libraries for different programming languages. Examples include Caffee, Theano, Torch and Tensor Flow, among others (see Figure).



	Language(s)	License	Main backer
PyTorch	**Python**	BSD	Facebook
Caffe2	C++, Python	Apache	Facebook
TensorFlow	Python, C++	Apache	Google
MXNet	Python, C++, R, Scala	Apache	Amazon
CNTK	Python, C++	MIT	Microsoft
Torch	Lua	BSD	Facebook
Theano	Python	BSD	U. of Montreal
Caffe	C++	BSD 2 clauses	U. of CA, Berkeley

Figure 1: The main codes of Deep Learning

Solutions in R

Many packages, which implement the paradigm of neural networks or link with an existing library in another language.

Without being exhaustive:

- `nnet`, the good old multi-layered perceptrons with the nnet package (available in the basic version of R), maintained by the legendary Brian Ripley,
- `RSNNS` offers an interface with the Stuttgart Neural Network Simulator (SNNS).
- `FCNN4R` provides an interface with the FCNN library allows user-expandable artificial neural networks,
- `rnn` implements recurrent neural networks,
- `deepnet` (feed-forward neural network, restricted Boltzmann machine, deep belief network, stacked autoencoders)
- `RcppDL`, denoising autoencoder, stacked denoising autoencoder, restricted Boltzmann machine, deep belief network

More solutions in R

- `h2o`, feed-forward neural network, deep autoencoders
- `DeepNet` has also implemented Deep Neural Networks, Deep Belief Networks and Restricted Boltzmann Machines
- `mxNet` contains pre-formed templates for object recognition.
- `kerasR` package, which was written by Taylor Arnold.
- `keras` of RStudio is an API (Application programming interface) of high level neural networks developed in order to allow a fast experimentation*

About Keras

- Keras is written in Python
- Keras supports both convolutional and recurrent networks (as well as combinations of both),
- Seamlessly runs on CPUs and GPUs
- Can run on multiple backends (optimizer), including TensorFlow (https: //tensorflow.rstudio.com/), CNTK and Theano.

About the `keras` R package

`keras` R package allows you to enjoy the benefit of R programming while having access to the capabilities of the Python Keras package.

Some vital links on the subject:

- The dedicated website: https://keras.rstudio.com/
- The cheat-sheet of Rstudio on the subject: https://github.com/rstudio/cheatsheets/raw/master/keras.pdf
- For a great introduction https://www.datacamp.com/community/tutorials/keras-r-deep-learning

Example 1 : MNIST database

The MNIST database (modified database of the National Institute of Standards and Technology) is an extensive database of handwritten figures commonly used for the training of various image processing systems. The database is also widely used for training and testing in the field of machine learning. It was created by "remixing" the samples from the original NIST datasets. The creators felt that since the NIST training dataset came from US Census Bureau employees, while the test dataset was from US students, it was not well suited to the experiments of the US Census Bureau. machine learning. In addition, black-and-white NIST images were normalized to fit in a 28x28-pixel, anti-aliasing bounding box, which introduced grayscale levels. ! [Extracted from the MNIST data set] (./ images / digits.png)

Data definition

```r
#loading the keras inbuilt mnist dataset
data <- dataset_mnist()

#separating train and test file
train_x <- data$train$x
train_y <- data$train$y
test_x <- data$test$x
test_y <- data$test$y

rm(data)

# converting a 2D array into a 1D array for feeding into th
train_x <- array(train_x, dim = c(dim(train_x)[1], prod(dim
test_x <- array(test_x, dim = c(dim(test_x)[1], prod(dim(te

image(matrix(train_x[2,],28,28,byrow=T), axes = FALSE,col=g
```