INTERNSHIP PROPOSAL:

Title :

Implementing a differentiable programming framework based on linear logic and functional programming

Topic :

Functional programming languages, deep learning, linear logic, lambda-calculus.

City and country :

Paris - France (IRIF)

Team or project in the lab :

IRIF (participation to the ANR project PPS)

Name and mail of the advisor :

Michele Pagani, pagani@irif.fr

Other possible collaborators will be:

Alois Brunel, alois.brunel@gmail.com

Damiano Mazza, damiano.mazza@lipn.univ-paris13.fr

Name and mail of the head of the department :

Magniez Frederic, magniez@irif.fr

General presentation of the topic (roughly 5 to 10 lines) :

In the last few years, a new research is flourishing at the interface between the programming languages and the deep learning communities, starting from the understanding that complex neural networks may in fact be expressed more synthetically and modularly in terms of actual programs, with branching, recursive calls or even higher-order primitives, like list combinators such as map or fold.

Can the logical structure of the functional primitives allow for a mathematical understanding of deep neural nets? Which kind of preprocessing or code analysis can be performed in order to make more efficient learning algorithms? All these questions spurred the development of a new, fast-growing line of research going under the name of differentiable programming.

Objective of the internship (roughly 10 to 20 lines) :

One crucial step for efficiently training deep neural networks is to compute the gradient of functions specified by a certain class of first-order programs, called computational graphs. There is nowadays an impressive body of work in both industry and academia for developing techniques and software systems for

automatically computing such gradients. At the industrial level, let us mention the deep learning frameworks TensorFlow [1] and Pytorch [2]. At a more academic level we mention automatic differentiation, given a general theory encompassing most of these techniques [3].
Some works have also been proposed in order to adapt these algorithms to functional languages, in particular we mention the Lantern framework [4]. The benefit of the latter approach is to describe automatic differentiation as program transformations. We also defined in [5,6] a transformation which, in contrast to [4], is purely functional and effect-free, based on a simply typed lambda-calculus endowed with a linear negation type construct. In this internship, we propose to study in detail the mechanisms of automatic differentiation in PyTorch, to analyse the possible improvements offered by a pure, simply typed functional programming framework, and/or to develop a first implementation of our transformation in Python and to compare its performance with the available deep learning frameworks.

Bibliographic references :

[1] M. Abadi et al., “TensorFlow: A System for Large-Scale Machine Learning”. OSDI2016. See also: https://www.tensorflow.org
[2] A. Paszke et al., “Automatic differentiation in PyTorch”, 2017. See also: https://pytorch.org
[3] A.G. Baydin, B.A. Pearlmutter, A.A. Radul, J.M. Siskind, “Automatic Differentiation in Machine Learning: a Survey”, Journal of Machine Learning Research 2017
[4] F.W.X. Wu, G.M. Essertel, J.M. Decker, T. Rompf, “Demystifying Differentiable Programming: Shift/Reset the Penultimate Backpropagator”, ICFP 2019, see also: https://feiwang3311.github.io/Lantern/
[5] A. Brunel, D. Mazza, M. Pagani, "Backpropagation in the Simply Typed Lambda-calculus with Linear Negation", POPL 2020, preprint available at: https://arxiv.org/pdf/1909.13768.pdf
[6] D. Mazza, M. Pagani, "Automatic Differentiation in PCF", POPL 2021, preprint available at:https://arxiv.org/abs/2011.03335

Expected ability of the student :

Good knowledge of at least one among: Python, OCaml.
Expertise in one or more of the following topics: functional programming, lambda-calculus, deep learning, compilation techniques.