Technical University of Denmark - DTU compute

Pre-print, ODL survey

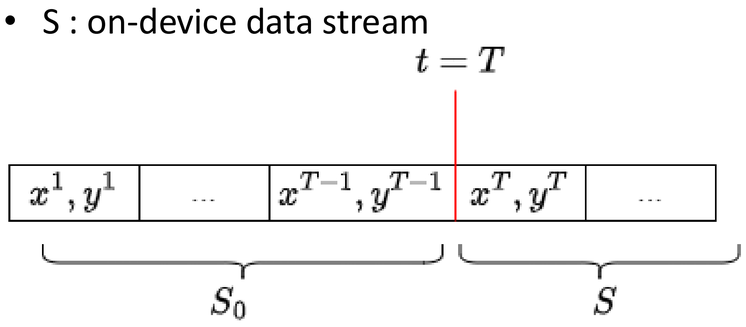
LBP: Formalizing the on-device learning problem

Massimo Pavan[1], Luca Pezzarossa[2], Xenofon Fafoutis[3]
{mapav[1], lpez[2], xefa[3]}@dtu.dk
Technical University of Denmark – DTU compute

There is little agreement on what doing On-Device Learning (ODL) means, and many ODL works don't consider the constraints of real-world conditions: **limited memory, computation**, and **label availability,** data as a **stream**, **no model validation**. We propose a formalization of the ODL problem that works with different environments and constraints, with the goal of designing reliable solutions in real-world conditions.

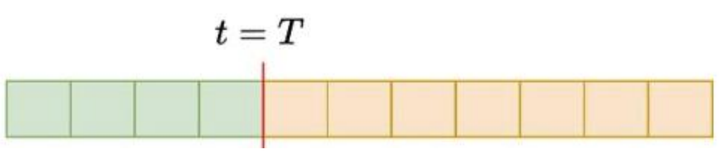
Components of the ODL problem

- Tiny device ***D***
- An ODL solution ***A***
- A data generating process ***P***
  - Provides a pair (x^t, y^t) at each t
  - Sample from $p^t(x,y)$
  - (x^t, y^t) represents the data of a sensor + label
  - At t=T A is deployed on **D**
- S_0 : off-device (pre-)training set
- S : on-device data stream



3 Environments:

- Single change Environments



$$p^{T-1}(x,y) \neq p^T(x,y)$$

- 2 Non stationary Environments

$$\exists t > T \quad p^{t-1}(x,y) \neq p^t(x,y).$$

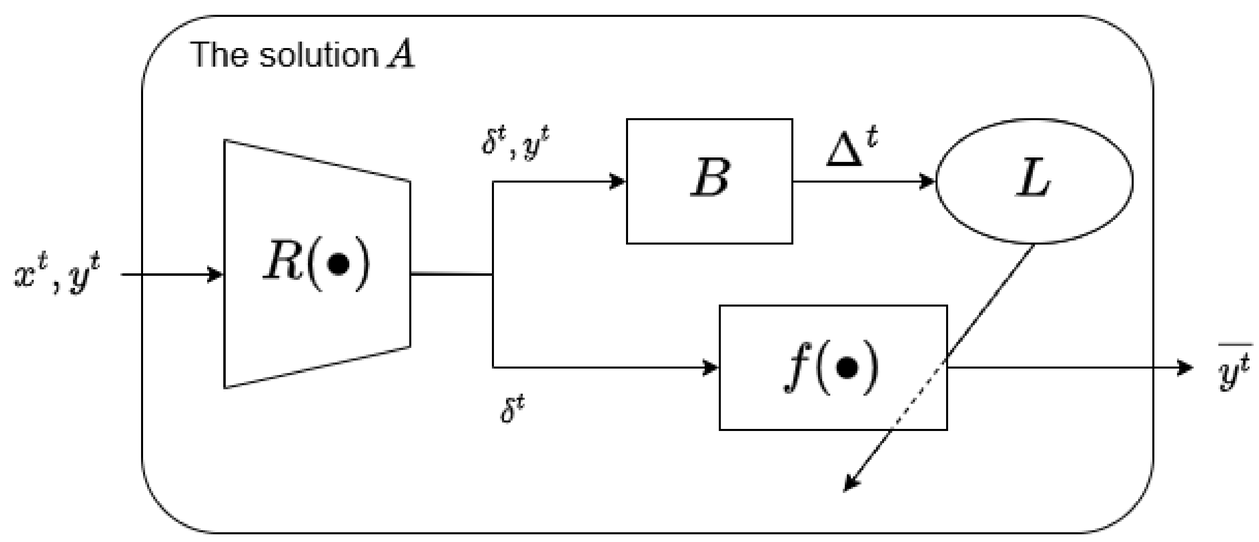
  - Concept Drift Environments
    - Focus on fast adaptation (plasticity)
  - Continual Learning Environments
    - Focus on retaining knowledge (Stability)

Evalutation of the solutions:

- Difficult to validate solutions on-device
- Use multiple "hold-out" data-streams S, available off-device
- Over-time performance monitoring
- Environment specific metrics

Goal: Develop a complete ODL solution ***A*** featuring:

- A feature extraction block ***R***
- A data management policy ***B***
- The ML classifier **f**
- A Learning Mechanism ***L***



Respecting hard constraints on Execution time and memory.

Future Developments: A Library for designing reliable ODL solutions

Problems/Gap in the literature:

- Extremely difficult to compare ODL solutions, due to variability of environments, algorithms and hardwares employed
- Fragmented pipelines, each component optimized independently from each other

We are working on a Library that:

- Helps finding the best **joint** configuration of *R, f, L* and *B.*
- Provides worst-case Memory/Computational requirements for solutions
- Focus on reproducibility (Provides Datasets, baseline implementation for components, metrics and evaluation methods for different environments)
- Focus on extendability (Design easily just a new component, or the complete pipeline, and share them with the community!)

Feedback: What features would make you interested in using it?

The NeAIxt project has received funding from the European Union's Horizon Europe research and innovation programme under the HORIZON-JU-Chips-2024-1-IA grant agreement No 101194172. Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union or CHIPS JU. Neither the European Union nor the granting authority can be held responsible for them. The project is also supported by its members France, Italy, Czech Republic, Germany, Sweden, Denmark, Greece, Spain, Netherlands, Portugal, Turkey, Switzerland.

Contacts

Linkedin

Embedded Systems Engineering - DTU