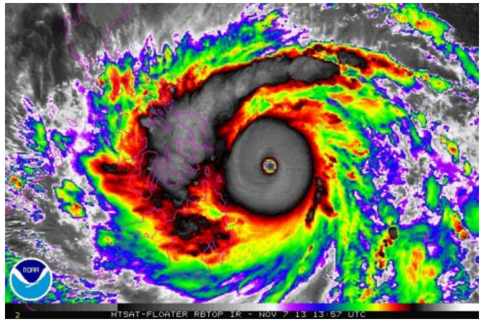
CausaLearn: Automated Framework for Scalable Streaming-based Causal Bayesian Learning using FPGAs

Bita Rouhani, Mohammad Ghasemzadeh, and Farinaz Koushanfar
ECE Department, UC San Diego
February 2018

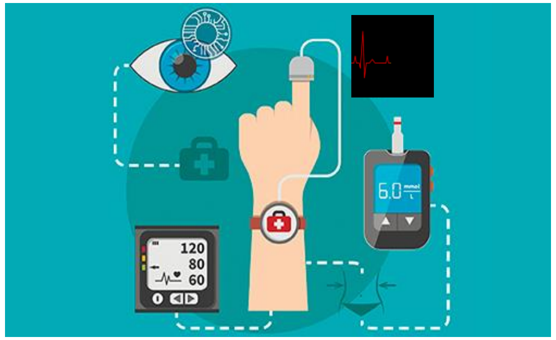
Time-series data with causality structure

Financial Data Analysis

Weather Forecasting

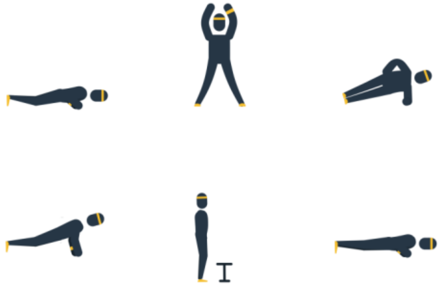


Health care



Geological Data Analysis

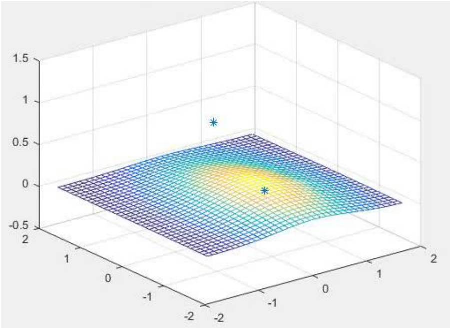
Activity Recognition



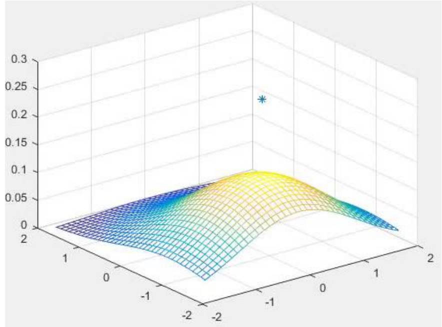
Speech Recognition

Markov Chain Monte Carlo (MCMC)

- MCMC is a commonly used method for probabilistic analysis of time-series data
- Algorithmic solutions to effectively capture **causality** structure of data involve **complex data flows** such as **gradient computation** (e.g., Hamiltonian MCMC)
- Existing MCMC hardware-accelerated tools are either:
  - Developed based on the assumption that data samples are ***independently and identically distributed*** to simplify the hardware implementation complexity (e.g., [1]), or



Prior work with i.i.d. assumption



CausaLearn

  - Developed for analyzing ***discrete random variables*** and are inapplicable to dynamic continuous variables (e.g., [2])

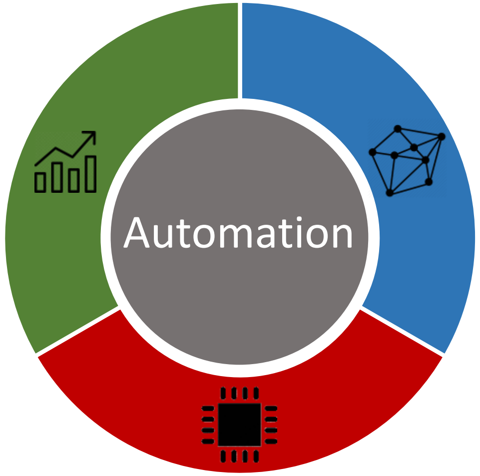
[1] Mingas et al., "Population-based MCMC on multi-core CPUs, GPUs and FPGAs", IEEE Transactions on Computers, 2016
[2] N. B. Asadi, et al., "Reconfigurable Computing for Learning Bayesian Networks," Field programmable gate arrays (FPGA), 2008.

Contributions

- ➢ Proposing CausaLearn, the **first end-to-end** framework that enables **real-time** multi-dimensional PDF approximation for **time-series** data with **causal** structure*

    - ➢ Devising the first **scalable** realization of Hamiltonian Markov Chain Monte Carlo to analyze **continuous** random variables on FPGA platforms

    - ➢ Developing an **automated customization** tool to optimize the underlying performance by **simultaneously** considering data, algorithm, and hardware characteristics



- ➢ Providing support for **streaming settings** in which the underlying PDF should be adaptively updated as data evolves over time

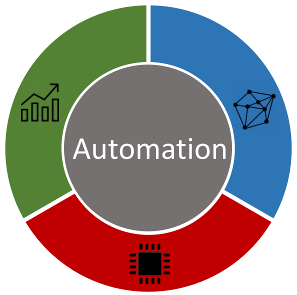
- ➢ Designing an **accompanying API** to ensure CausaLearn ease of use on various FPGAs

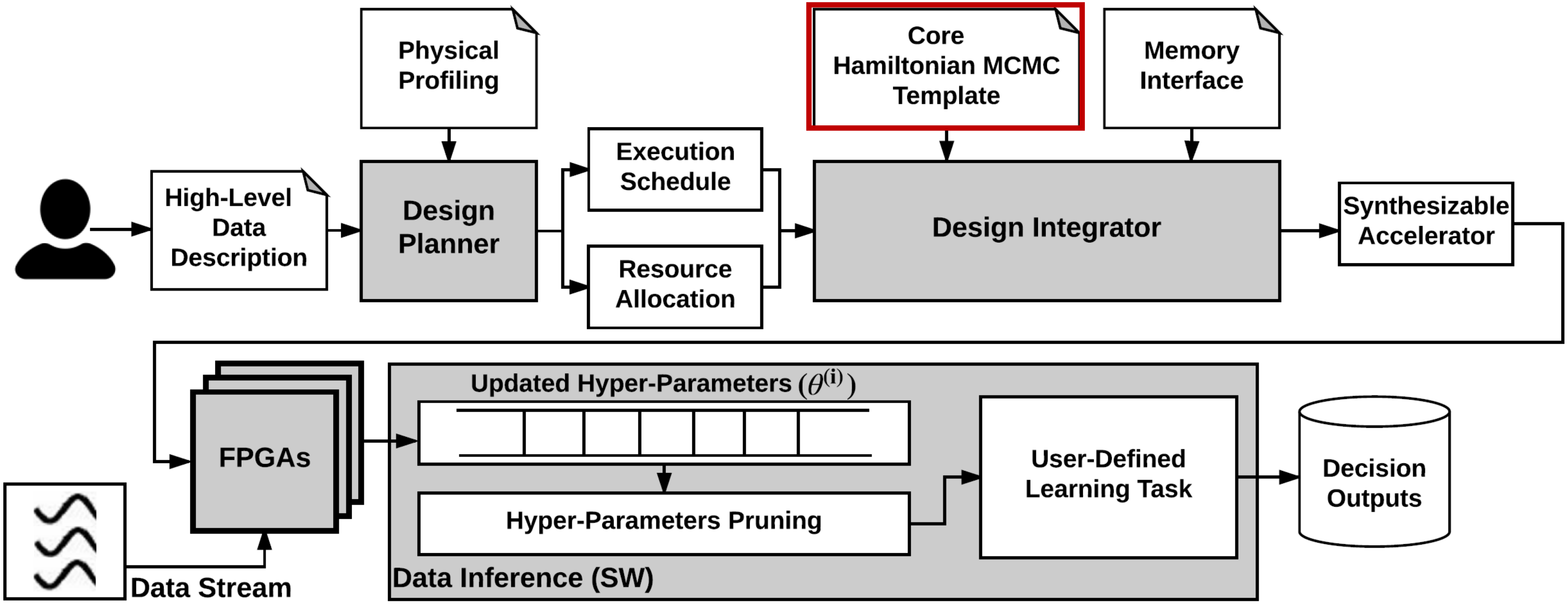
* B. Rouhani, M. Ghasemzadeh, and F. Koushanfar, U.S. Patent No. 62452880, 2017

What is CausaLearn?

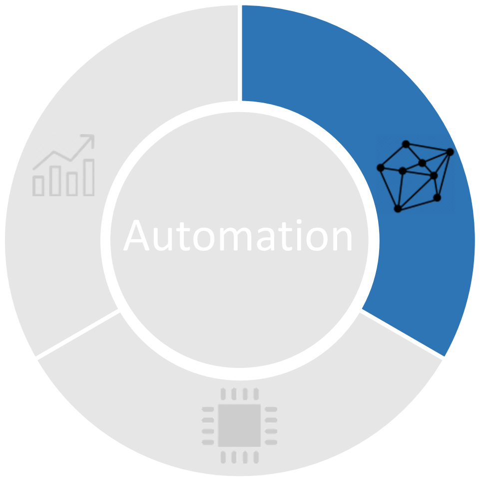
Global flow

Global flow

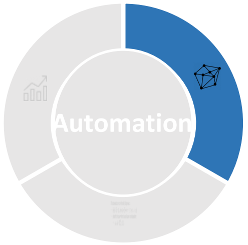

Automation


Physical
Profiling
Core
Hamiltonian MCMC
Template
Memory
Interface
High-Level
Data
Description
Design
Planner
Execution
Schedule
Resource
Allocation
Design Integrator
Synthesizable
Accelerator
FPGAs
Updated Hyper-Parameters ($\theta^{(i)}$)
Hyper-Parameters Pruning
User-Defined
Learning Task
Decision
Outputs
Data Stream
Data Inference (SW)

Hamiltonian MCMC



Objective function



- Time-series data can be represented as a pair of $(\boldsymbol{x}, \boldsymbol{y})$ values, where:
  - $\boldsymbol{x} = \{x_i = [x_{i1}, \dots, x_{id}]\}_{i=1}^{n}$ are the input **data features**
    - *d* is the feature space size
    - *n* specifies the number of data measurements that grows over time
  - $\boldsymbol{y} = [y_1, \dots, y_n]$ are the **observation values**
    - Each observation y_i can be either continuous as in regression tasks, or discrete as in classification applications

Objective function

Automation

$$y_i = f(x_i) + \varepsilon_i$$
$$\varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$$

- The observations are conditionally independent
  - $p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\sigma_n^2}) \sim \prod_{i=1}^{N} p(y_i|f_i, \sigma_n^2)$, where $\boldsymbol{f} = [f(x_1), \dots, f(x_n)]$
  - Note that the observations themselves are not independent
    - e.g., p(**y**) $\neq \prod_{i=1}^{N} p(y_i)$

$$f(\boldsymbol{x})|\gamma \sim \mathcal{GP}(m(\boldsymbol{x}), K(x, x'|\gamma))$$

- The latent function **f**(.) is defined as a Gaussian Processes (GP)
- GP is fully defined by its second order statistics

$$K_{ij}(\boldsymbol{x}) = \sigma_k^2 \, e^{\frac{-1}{2}(x_i - x_j)^T \Sigma^{-1} (x_i - x_j)}$$
$$\Sigma = diag[\mathcal{L}_1^2, \dots, \mathcal{L}_d^2]$$

We further assume a log-uniform prior for the variance parameter σ_k^2 and a multivariate Gaussian prior for the length-scale parameters

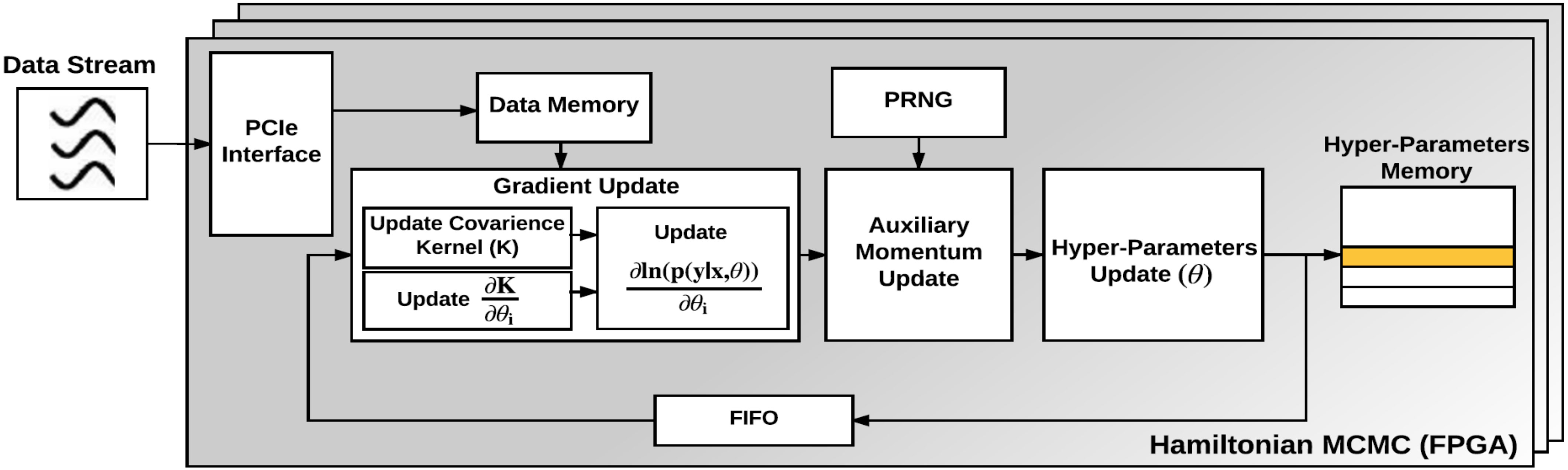
Hamiltonian MCMC

- To effectively capture the causality structure of the data, we need to **<u>fine-tune</u>** the underlying hyper-parameters *inline with the data arrival*

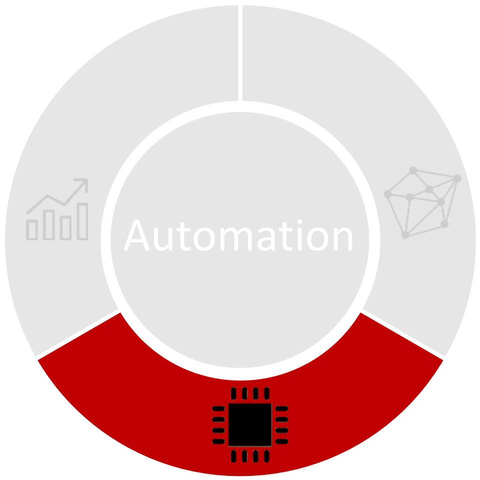
- Exploring the GP parameter space
  - **<u>Random-walk</u>**
    - Simple dataflow for hardware implementation
    - High cost of unnecessary space exploration in high-dimensional settings
  - **<u>Gradient-based Hamiltonian dynamics</u>**
    - Efficient sampling by moving towards the gradient of the model
    - Complex dataflow for hardware implementation

Hamiltonian MCMC block diagram

- Three main steps:
  I. Computing gradient of posterior distribution
  II. Updating auxiliary momentum variable
  III. Drawing new hyper parameter samples

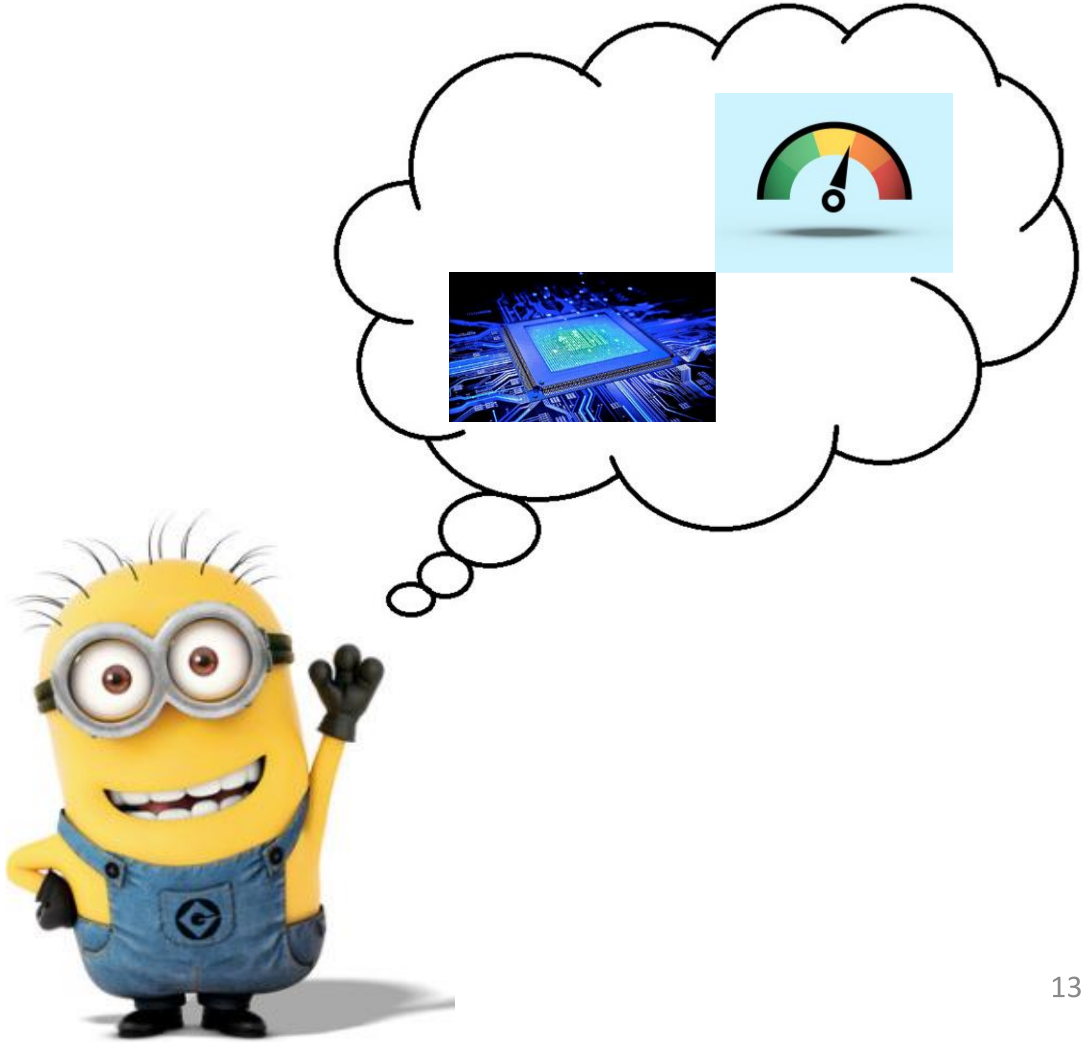


Hardware implementation



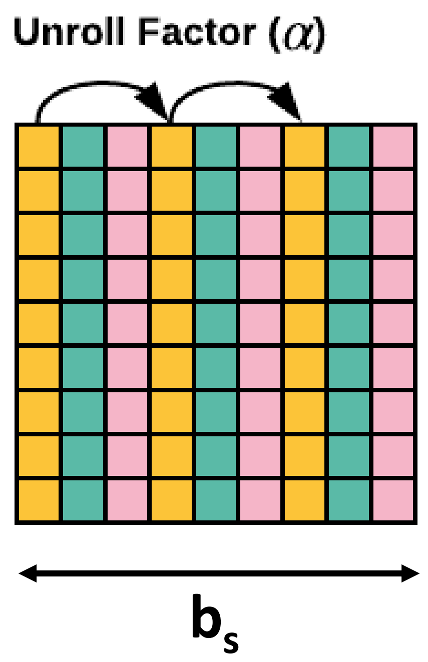
Hardware accelerator architecture

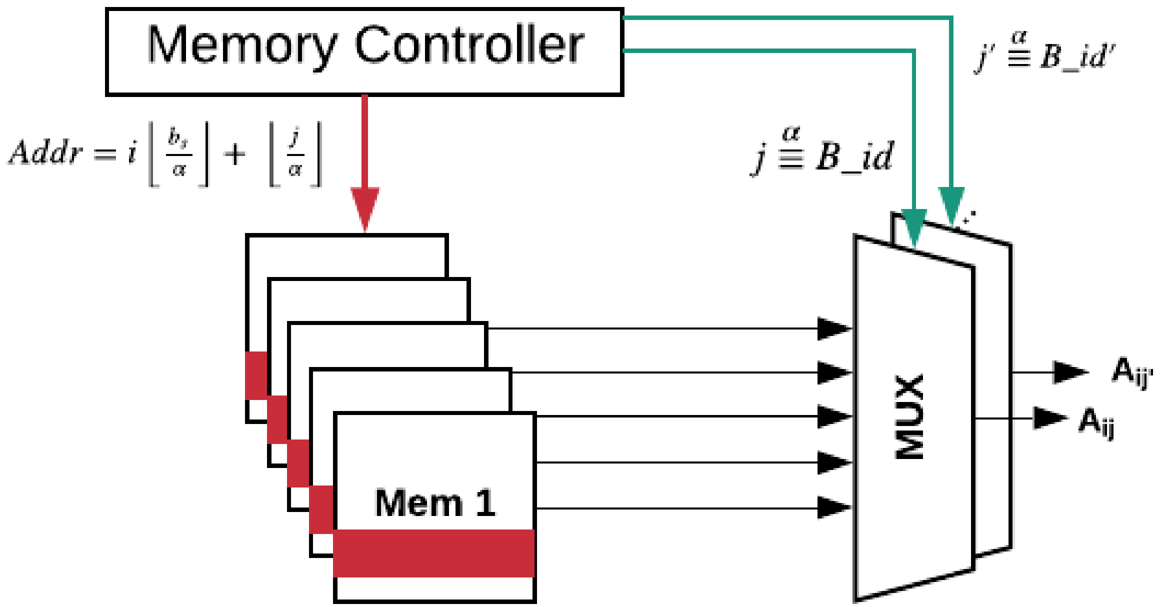
- Memory management
- Dot-product
  - Tree-based reduction
- Matrix inversion



Memory management

- CausaLearn facilitates matrix-based computations by:
  - Enabling concurrent access to multiple elements within a matrix
  - Preventing diverse/complex memory access pattern (**universal indexing**)

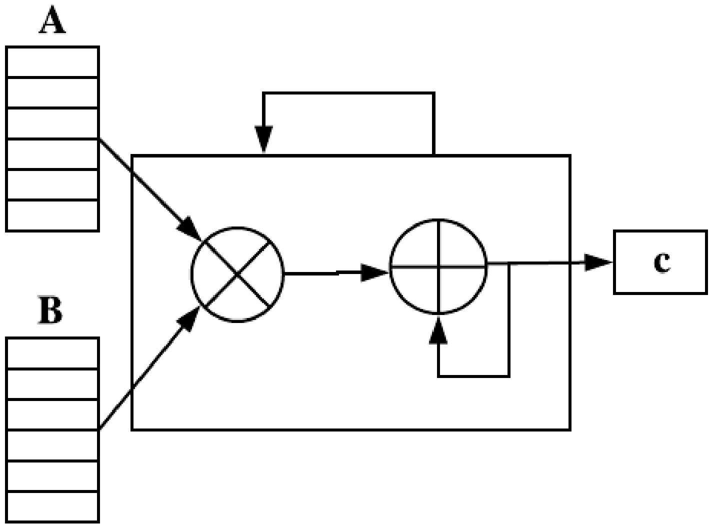




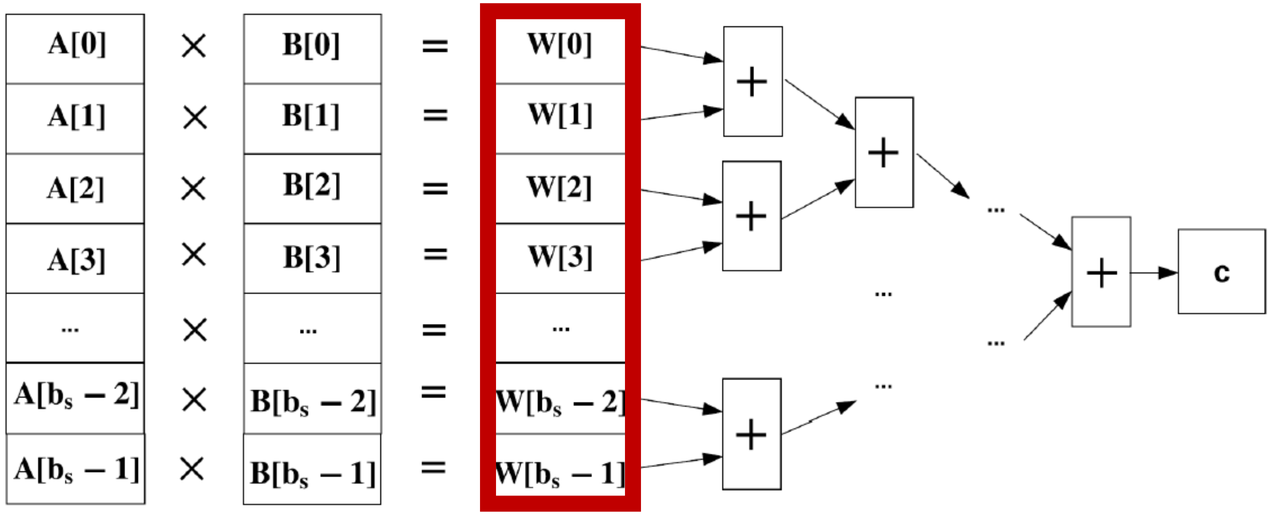
Tree-based reduction

- Matrix-based computations requires frequent dot product operations:

$$c \mathrel{+}= A[i] \times B[i]$$



Conventional sequential approach

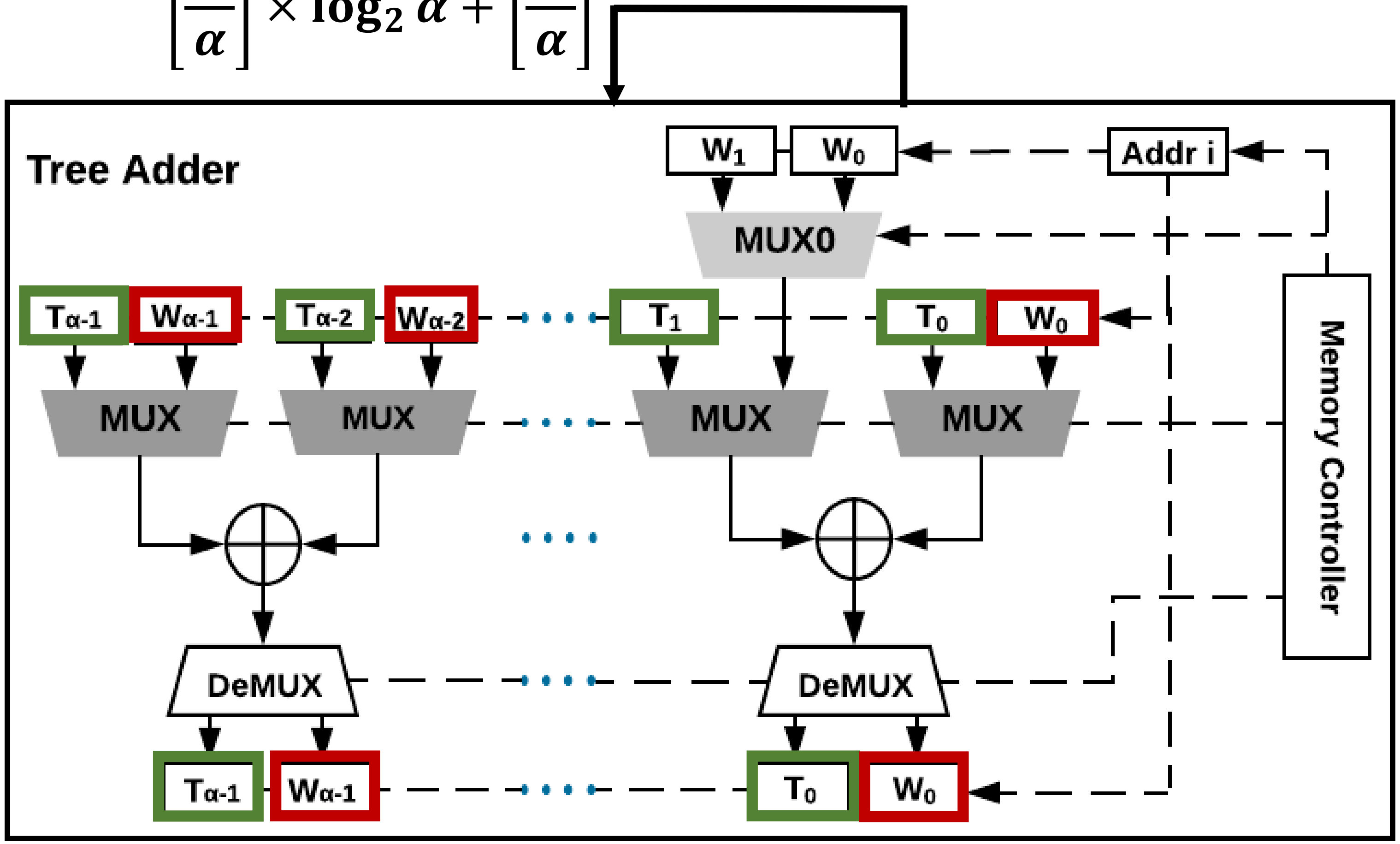


Tree-based approach

Tree-base adder

- The number of floating-point adders is equivalent to the unrolling factor α

$$\left[\frac{b_s}{\alpha}\right] \times \log_2 \alpha + \left[\frac{b_s}{\alpha}\right]$$



Matrix inversion

- Computing the inverse of the covariance kernel $\boldsymbol{K}$ is a key step in finding the gradient direction in each iteration

- Let us consider a linear equation as the following:

$$V = K^{-1}B$$

QR decomposition

$$V = (QR)^{-1}B$$

$$V = R^{-1}Q^{T}\,B$$

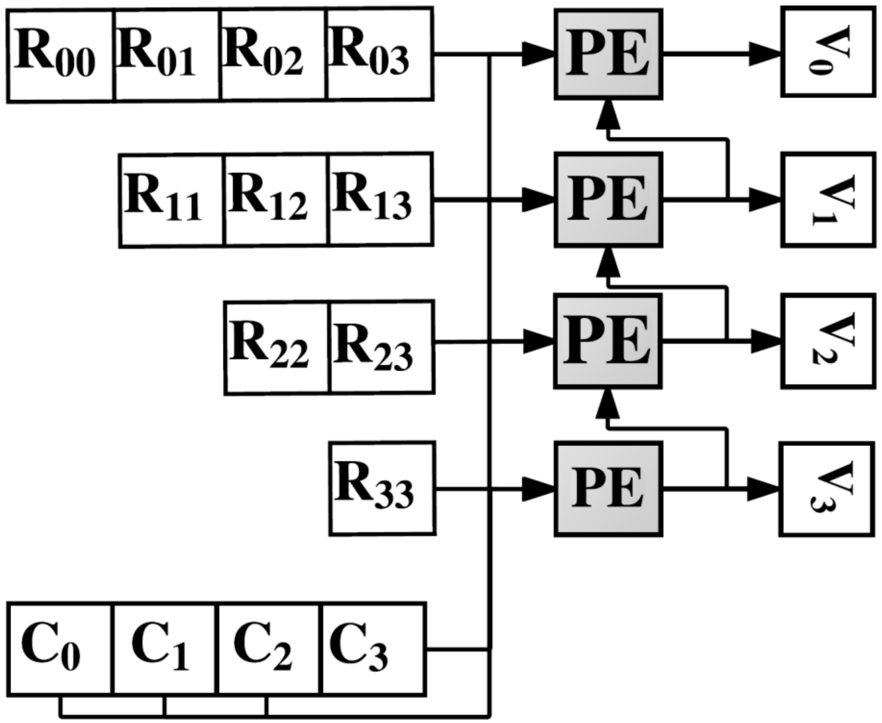
$$\underbrace{R}V = \underbrace{Q^{T}B}_{C}$$

$$\begin{bmatrix} \bullet & \bullet & \bullet & \bullet & \bullet \\ & \bullet & \bullet & \bullet & \bullet \\ & & \bullet & \bullet & \bullet \\ & 0 & & \bullet & \bullet \\ & & & & \bullet \end{bmatrix}$$

Matrix inversion

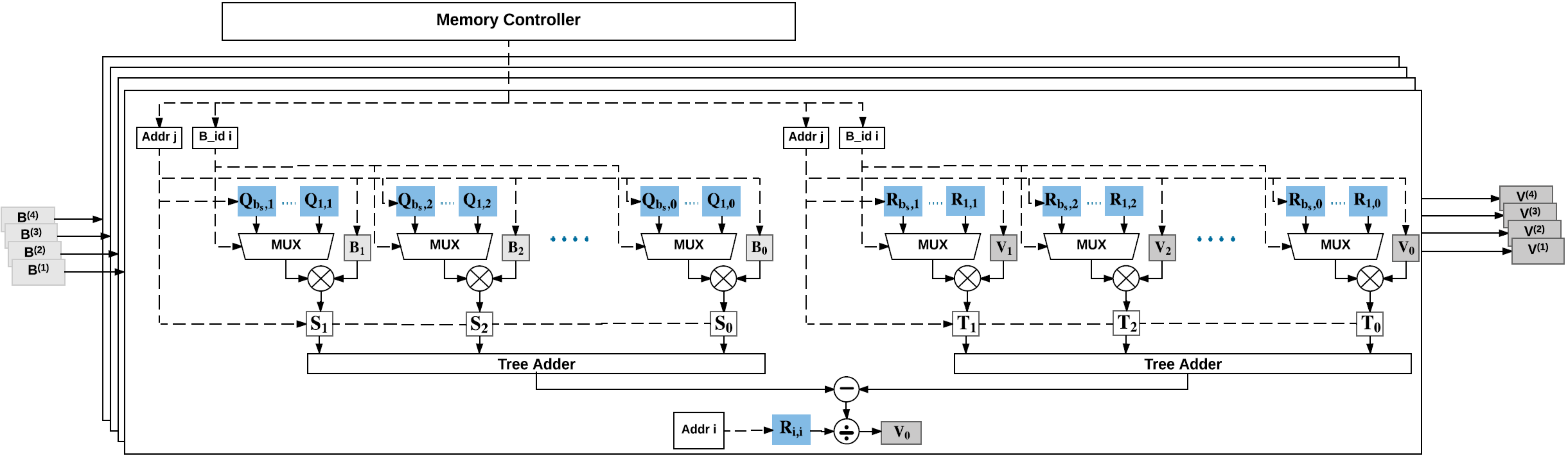
- Computing the inverse of the covariance kernel $\boldsymbol{K}$ is a key step in finding the gradient direction in each iteration
- Let us consider a linear equation as the following:

$$R_{bs\times bs}V_{bs\times 1} = C_{bs\times 1}$$

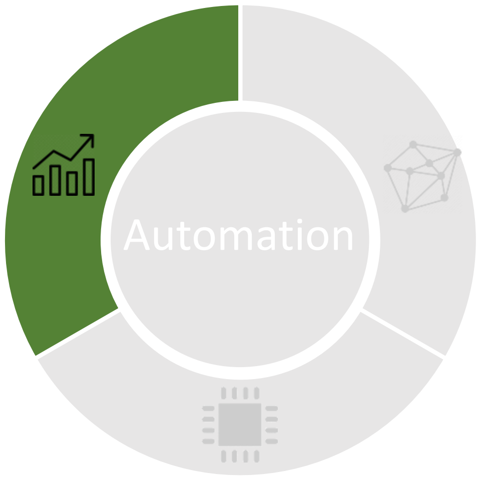


Matrix inversion architecture

- Concurrent computation of both sides of equation $RV = Q^T B$
  - Cyclic interleaving of all matrices
  - Universal memory access signals
  - Data parallelism to maximize throughput

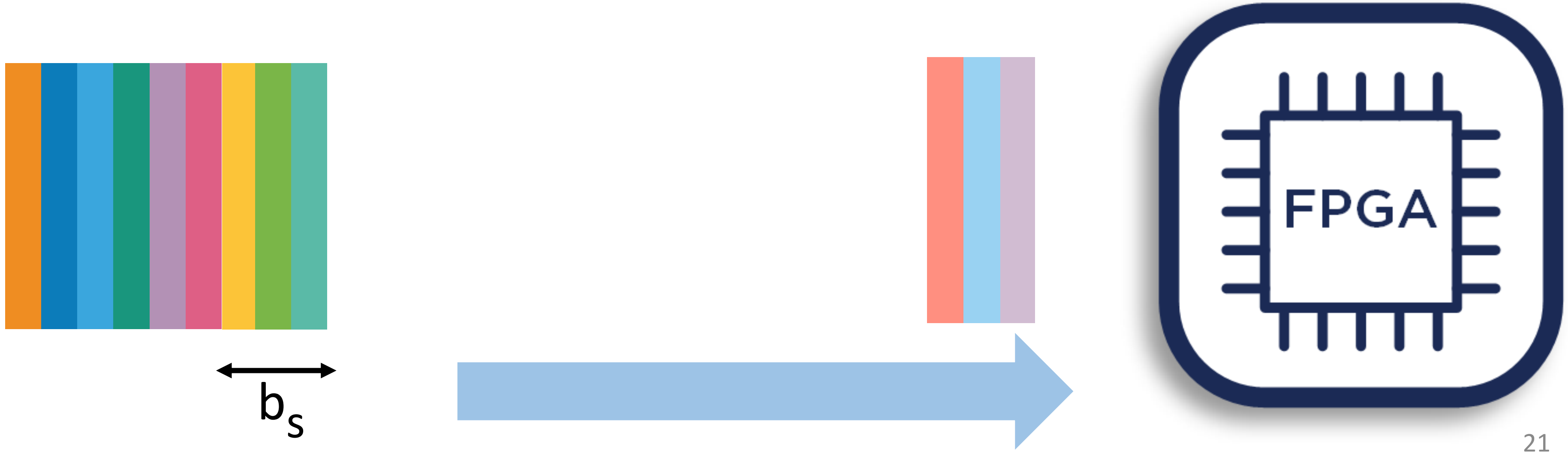


Data customization

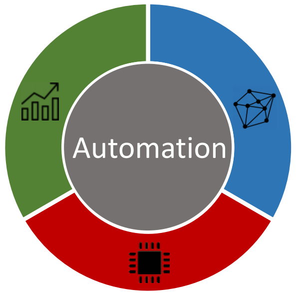


Batch optimization

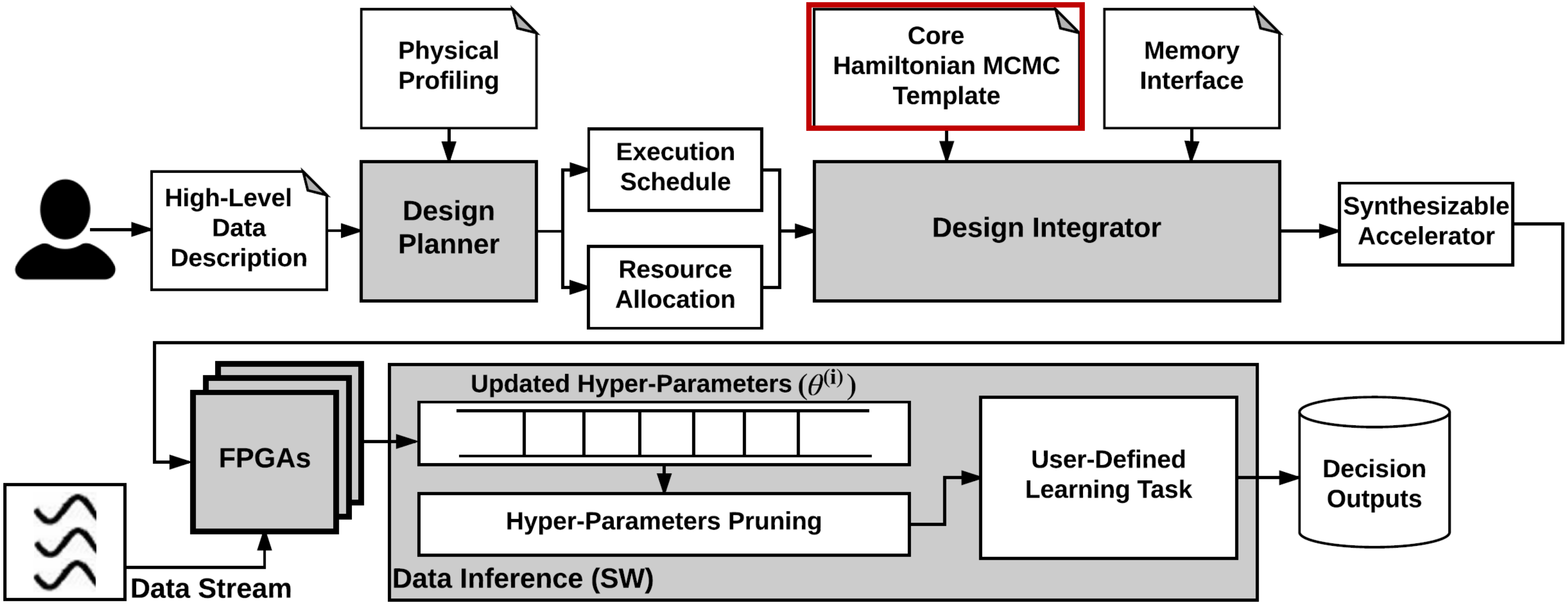
- The input data evolves over time
  - Breaking down the input data into batches that fit the memory budget
- The underlying batch size is a design parameter
  - Trade-off between system throughput and MCMC mixing time



Global flow

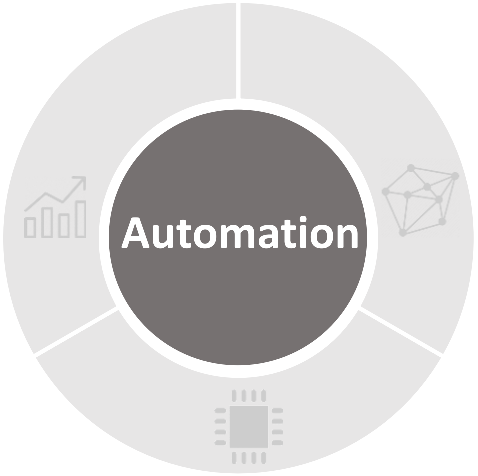


Automation

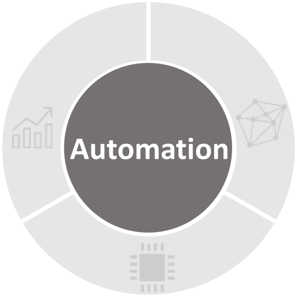


Physical Profiling
Core Hamiltonian MCMC Template
Memory Interface
High-Level Data Description
Design Planner
Execution Schedule
Resource Allocation
Design Integrator
Synthesizable Accelerator
Updated Hyper-Parameters ($\theta^{(i)}$)
FPGAs
Hyper-Parameters Pruning
User-Defined Learning Task
Decision Outputs
Data Stream
Data Inference (SW)

Automation



Design planner



- Automated design space exploration
  - **Batch size $\boldsymbol{b_s}$**
    - Karush-Kuhn-Tucker (KKT) optimization

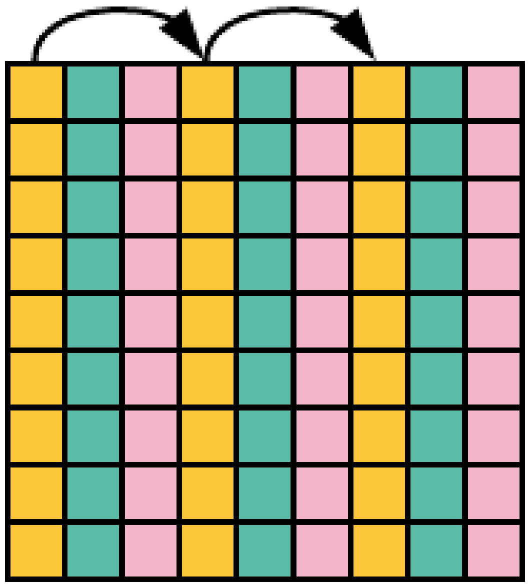
$$\underset{b_s}{\text{minimize}}\,(MC\ mixing\ time)\ \ s.t.\ \ T^{comm} + T^{comp} \leq\ T_{user}$$

$$M_{CausaLearn} \leq\ M_{user}$$

Design planner

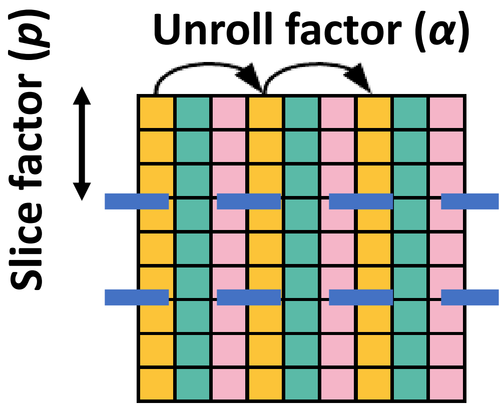
- Automated design space exploration
  - **Batch size b_s**
    - Karush-Kuhn-Tucker (KKT) optimization
  - **Unroll factor α**
    - Avoid excessive partitioning to registers

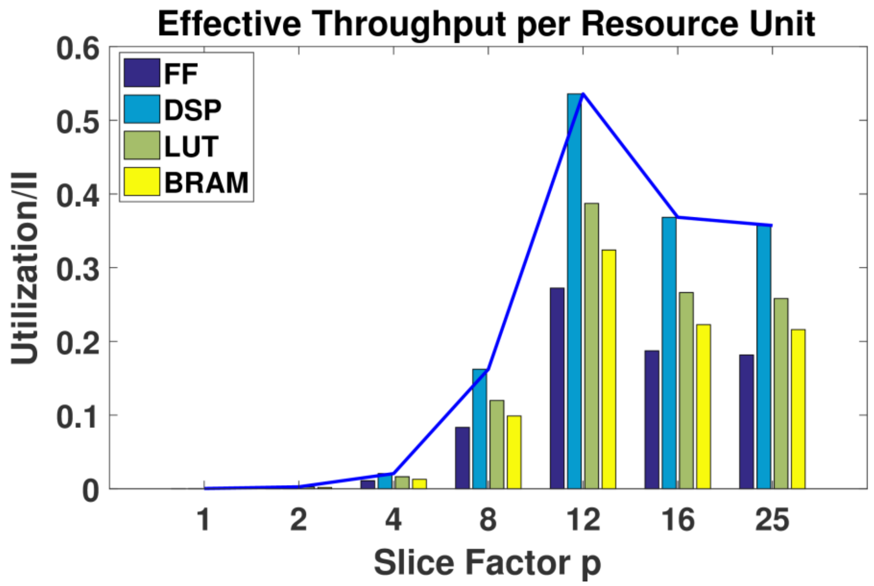
Unroll factor (α)



Design planner

- Automated design space exploration
  - **Batch size b_s**
    - Karush-Kuhn-Tucker (KKT) optimization
  - **Unroll factor α**
    - Avoid excessive partitioning to registers
  - **Slice factor p**
    - Maximizing throughput per resource unit

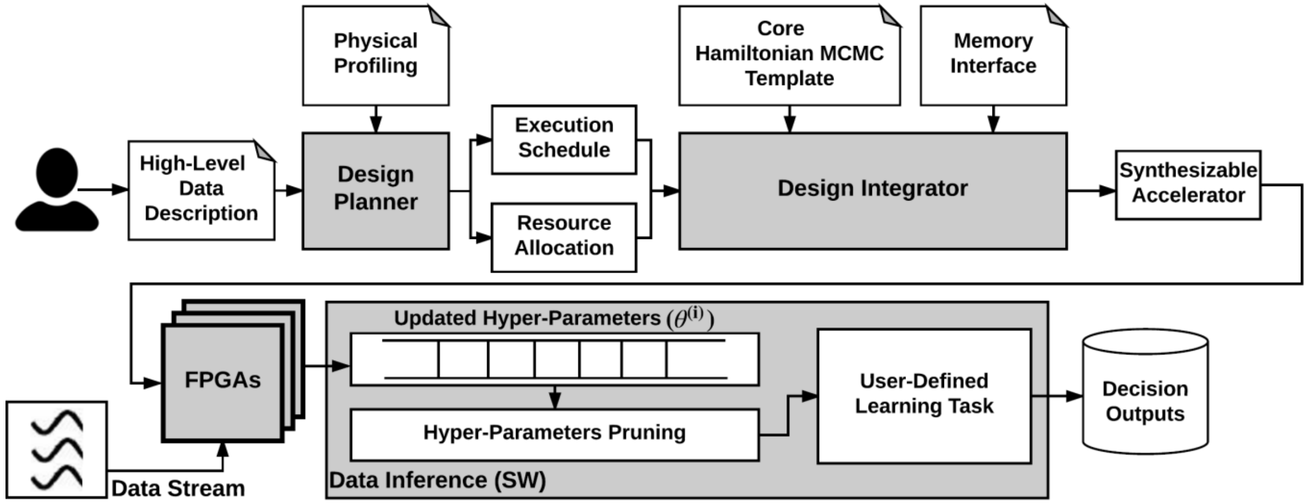




Design integrator

- Customizing the H_MCMC template with in accordance to the execution schedule
- Adding memory interface to communicate with the host CPU
  - Leveraging PCI express IP[1]

[1] Xillybus FPGA IP core for easy DMA, xillybus.com



CausaLearn evaluation

Example data applications evaluated so far ….

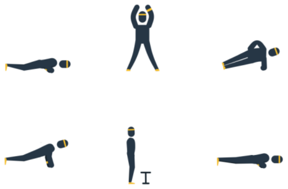
- **Dow-Jones index stock data**

  Task: data regression

  Predicting the percentage of return for each of the 30 involving stocks

  Data: daily stock data for 30 companies over 6 month

- **Activity recognition**



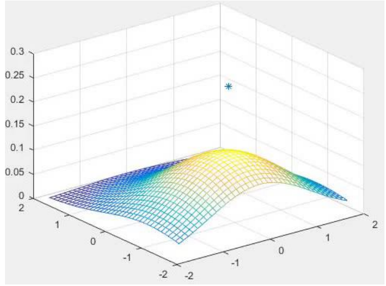
  Task: data classification

  Recognition of 12 different daily activities

  Data: recorded sensor body motion and vital signs at a sampling rate of 50Hz

---

- **Synthetic time-variant data**



  Task: data regression

  Data: 2-D regression data for visualization purposes

Example platforms evaluated so far ….

- Hardware evaluation platforms

Platform1: Virtex VC707
On-chip memory: 4.6 MB

Platform1: Zynq ZC702
On-chip memory: 0.6 MB

Platform1: Virtex VCU108
On-chip memory: 7.6 MB

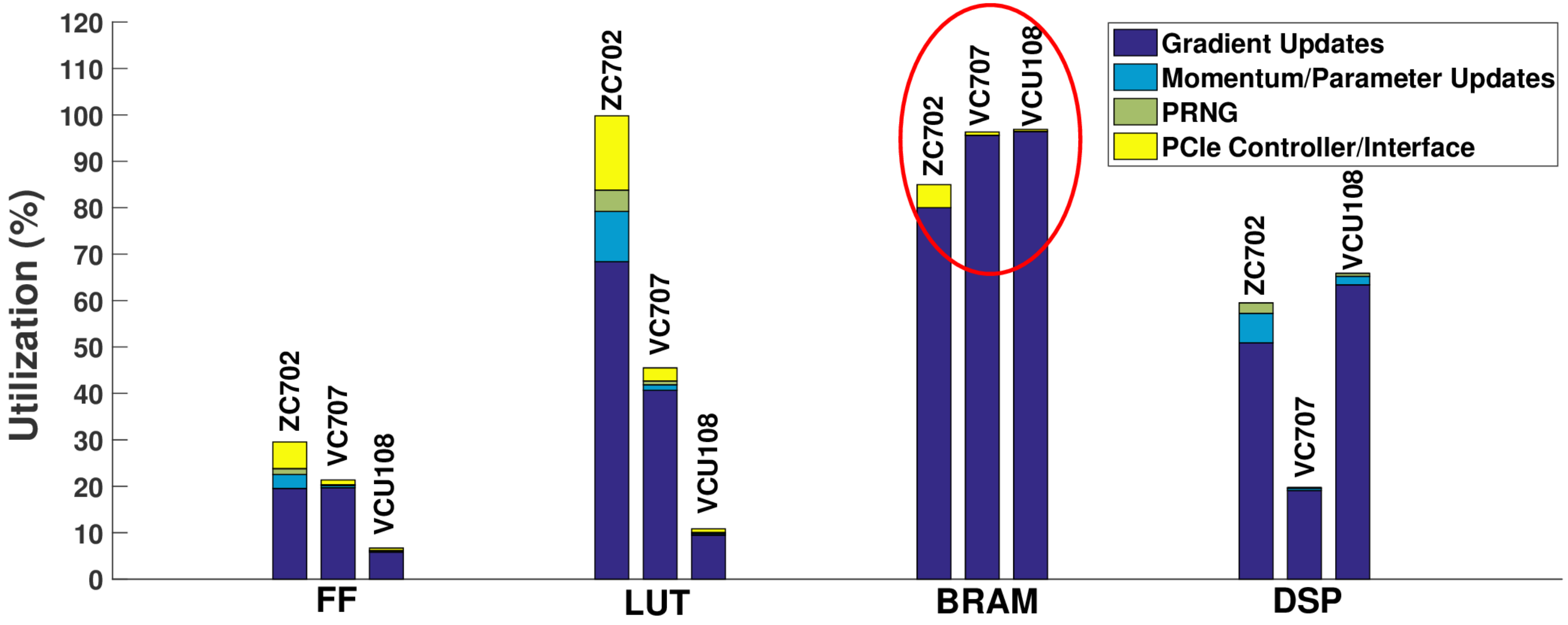
- Software evaluation platform

Processor: 2.4GHz Intel core i5-6300U
Memory: 8 GB

Eigen Library

Resource utilization

- Automated customization to **maximally exploits on-chip memory**
  - For a given dataset, each platform has its own batch size, unroll factor, and slice factor



Runtime and energy efficiency

- CausaLearn achieves up to **320x** runtime and **400x** energy improvement compared to a highly-optimized software solution running on a 2.4GHz Intel core i5-6300U

# of data samples*	SW Runtime per iteration C++	CausaLearn Runtime Improvement Virtex7	CausaLearn Energy Improvement Viretx7
256	113.01 sec	2.4x	4.1x
512	902.98 sec	9.6x	16.5x
1024	143.35 min	42.7x	65.9x
2048	33.52 hr	**320.2x**	**398.9x**

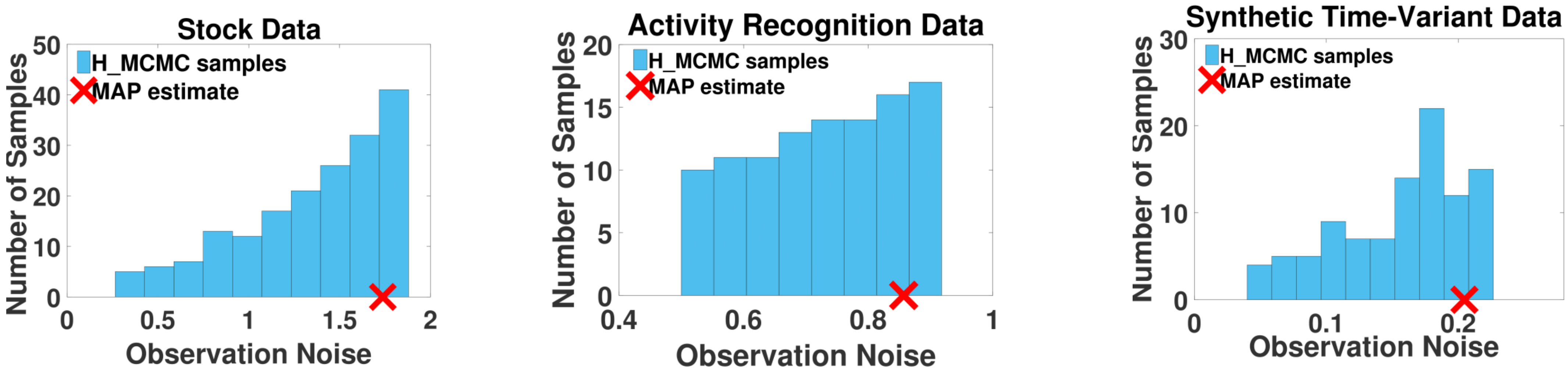
* $|\theta|$ = 10 for this experiment

Practical design experiments

- CausaLearn posterior distribution samples closely follow the optimal Maximum A Posterior (MAP) solution

$$\underset{\theta}{\operatorname{argmax}} \ln\big(p(y|x,\theta)\big) + \ln(p(\theta))$$

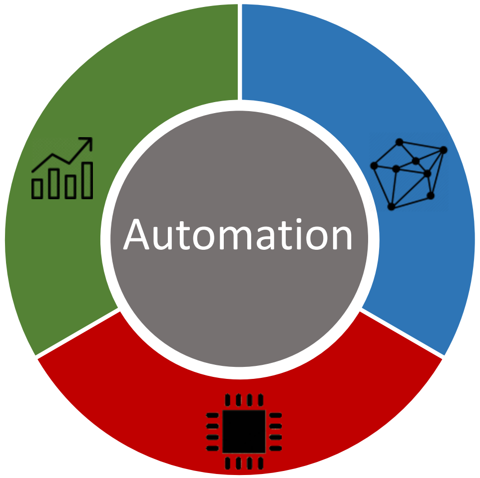
- Example posterior distribution samples for observation noise variance



Summary

- Proposing CausaLearn, the **first end-to-end** framework that enables **real-time** multi-dimensional PDF approximation using Hamiltonian MCMC*

- Developing an **automated** tool to optimize the physical performance by considering data, algorithm, and hardware characteristics



- Designing an **accompanying API** to ensure CausaLearn ease of use on various FPGAs

- Providing support for **streaming data**

* B. Rouhani, M. Ghasemzadeh, and F. Koushanfar, U.S. Patent No. 62452880, 2017