ICSC
Centro Nazionale di Ricerca in HPC,
Big Data and Quantum Computing

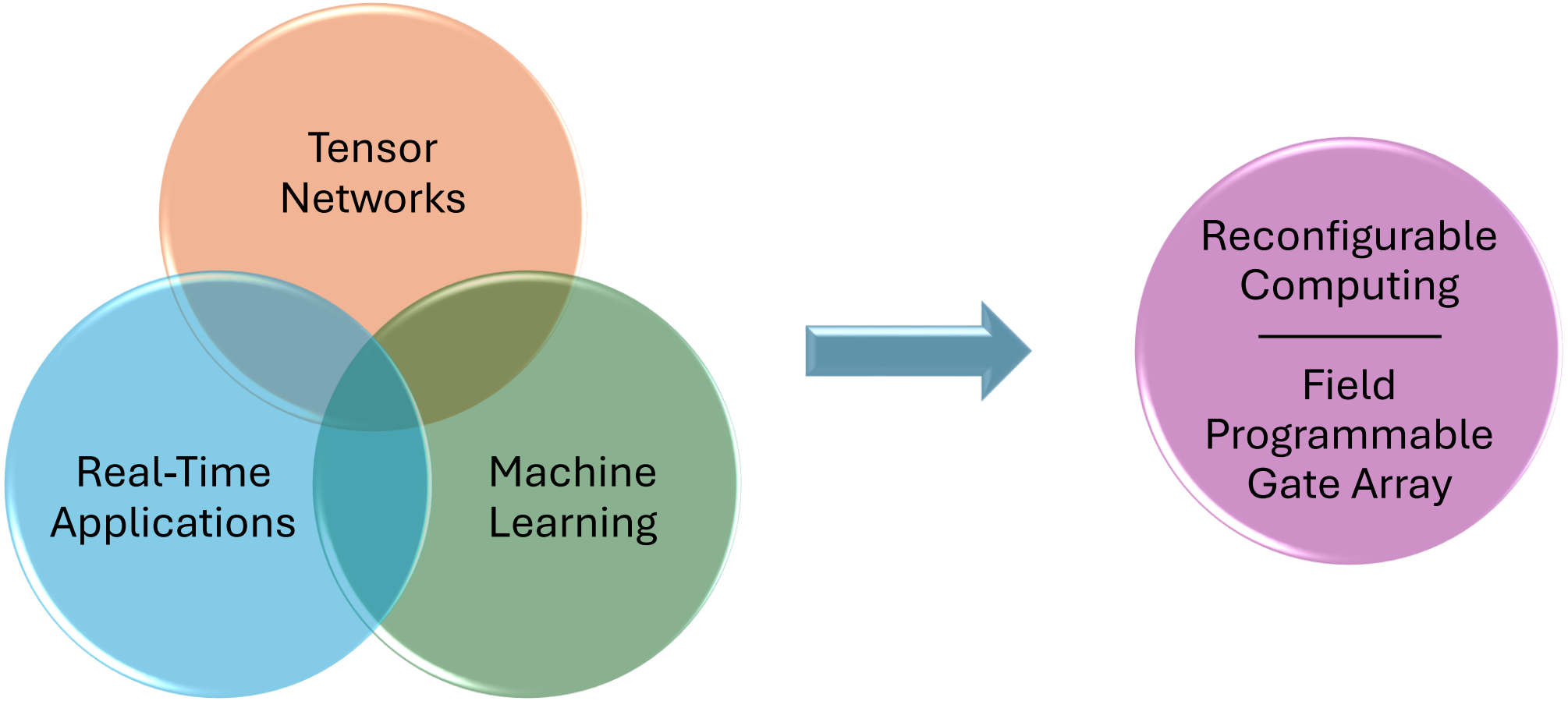
Università degli Studi di Padova

Hardware implementation of quantum machine learning predictors for ultra-low latency applications

Lorenzo Borella, Alberto Coppi, Jacopo Pazzini,
Andrea Stanco, **Andrea Triossi**, Marco Zanetti
University of Padova

"This work is partially supported by ICSC – Centro Nazionale di Ricerca in High Performance Computing, Big Data and Quantum Computing, funded by European Union – NextGenerationEU".

Introduction



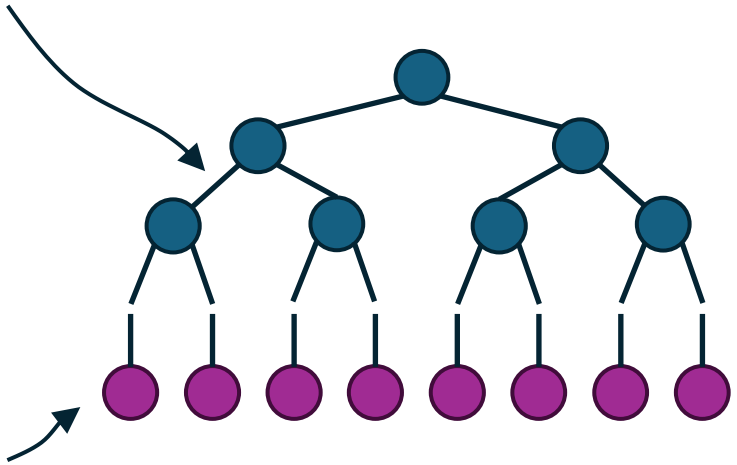
Tensor Networks

- Collection of tensors connected by contractions
- Intuitive graphical language
  - Tensors are notated by shapes
  - Indices are notated by lines emanating from these shapes
  - Connecting two index lines implies a contraction

- Approximate arbitrarily complex many-body quantum systems preserving its most important properties

Tree Tensor Networks for ML

- TN with a tree structure
- **Bond dimension** (χ_l) controls the expressivity of the TTN



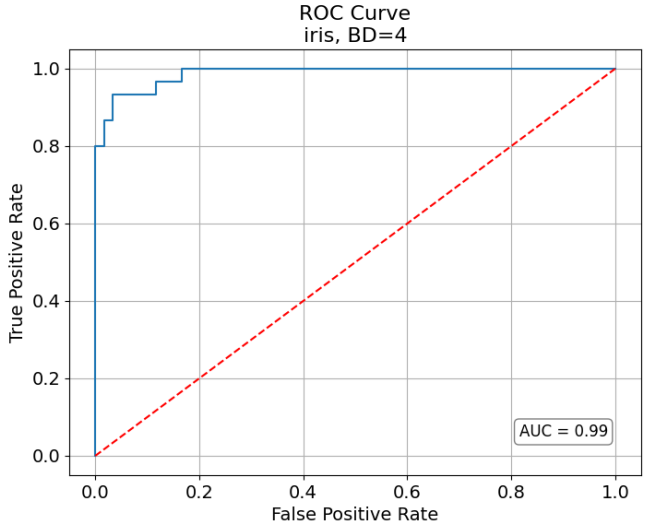
- **Input features** (N) are mapped in a higher dimensional space (D)
- Number of layers $L = \log_2 N$
- χ_l ideally scales with the layer as D^{2^l}
- Reduce the complexity of the problem artificially forcing $\chi_l = \min(D^{2^l}, \chi_0)$

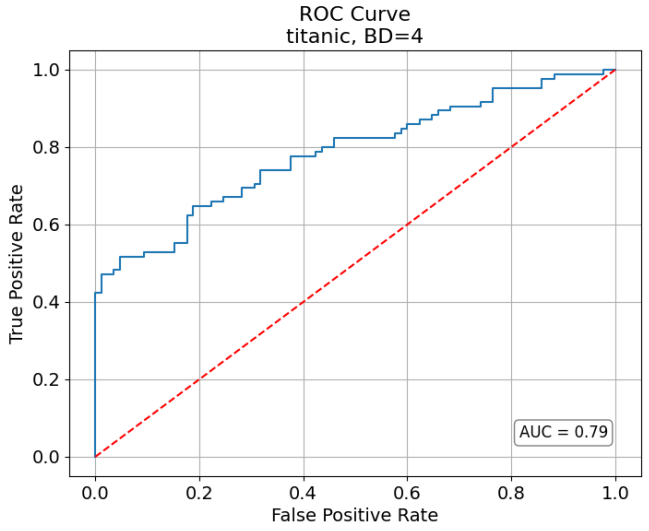
Methods

- Goal: binary classifier with ultra-low latency
- Training is done in software
- Weights are loaded in the dedicated hardware (FPGA) for inference
- Inference consists of only linear transformations
- FPGA is a programmable devices that combines an array of combinatorial logic blocks with a mesh of interconnections
  - Look-up tables, storage elements, fast carry chains
  - Dedicated hardware for specific functions (RAM, PLL, Ser/Des)
  - Digital Signal Processor (DSP) for arithmetic functions (adder, multiplier, accumulator)
  - Hardware Description Language (HDL) for circuit description
  - High degree of parallelization but limited resources

Training

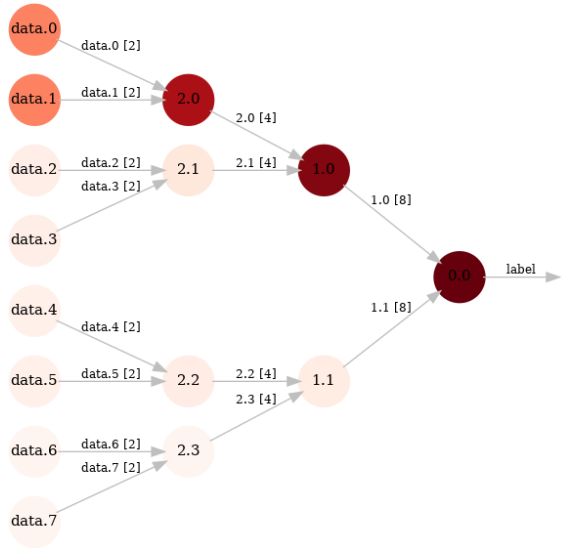
- Pythonic custom classes
- Class for representing TTN as a Torch NN module
  - ML approach for optimizing TTN with SGD
- Canonicalization
  - QR decomposition to isometrise towards a target node
- Tested on two datasets
  - Iris – N=4, D=2, χ_0=4 ⟶ 99% Accuracy
  - Titanic – N=8, D=2, χ_0=3,4,8,16 ⟶ 79% Accuracy
- Methods to measure physical quantities
  - Entropy of each link
  - Expectation value of a generic n-sites observable
- Ranking of the input features
  - Complexity reduction of the TTN

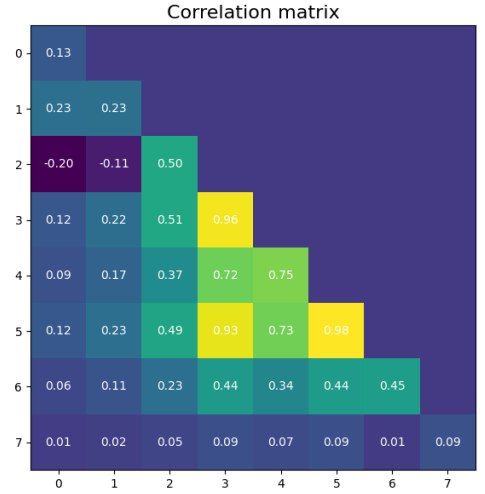




Training

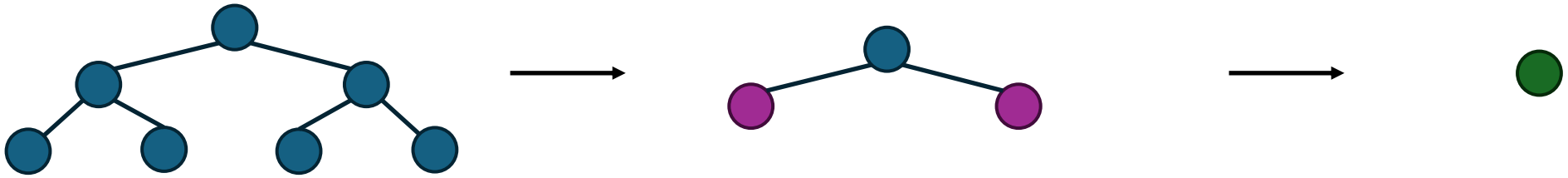
- Pythonic custom classes
- Class for representing TTN as a Torch NN module
  - ML approach for optimizing TTN with SGD
- Canonicalization
  - QR decomposition to isometrise towards a target node
- Tested on two datasets
  - Iris – N=4, D=2, χ_0=4 ⟶ 99% Accuracy
  - Titanic – N=8, D=2, χ_0=3,4,8,16 ⟶ 79% Accuracy
- Methods to measure physical quantities
  - Entropy of each link
  - Expectation value of a generic n-sites observable
- Ranking of the input features
  - Complexity reduction of the TTN





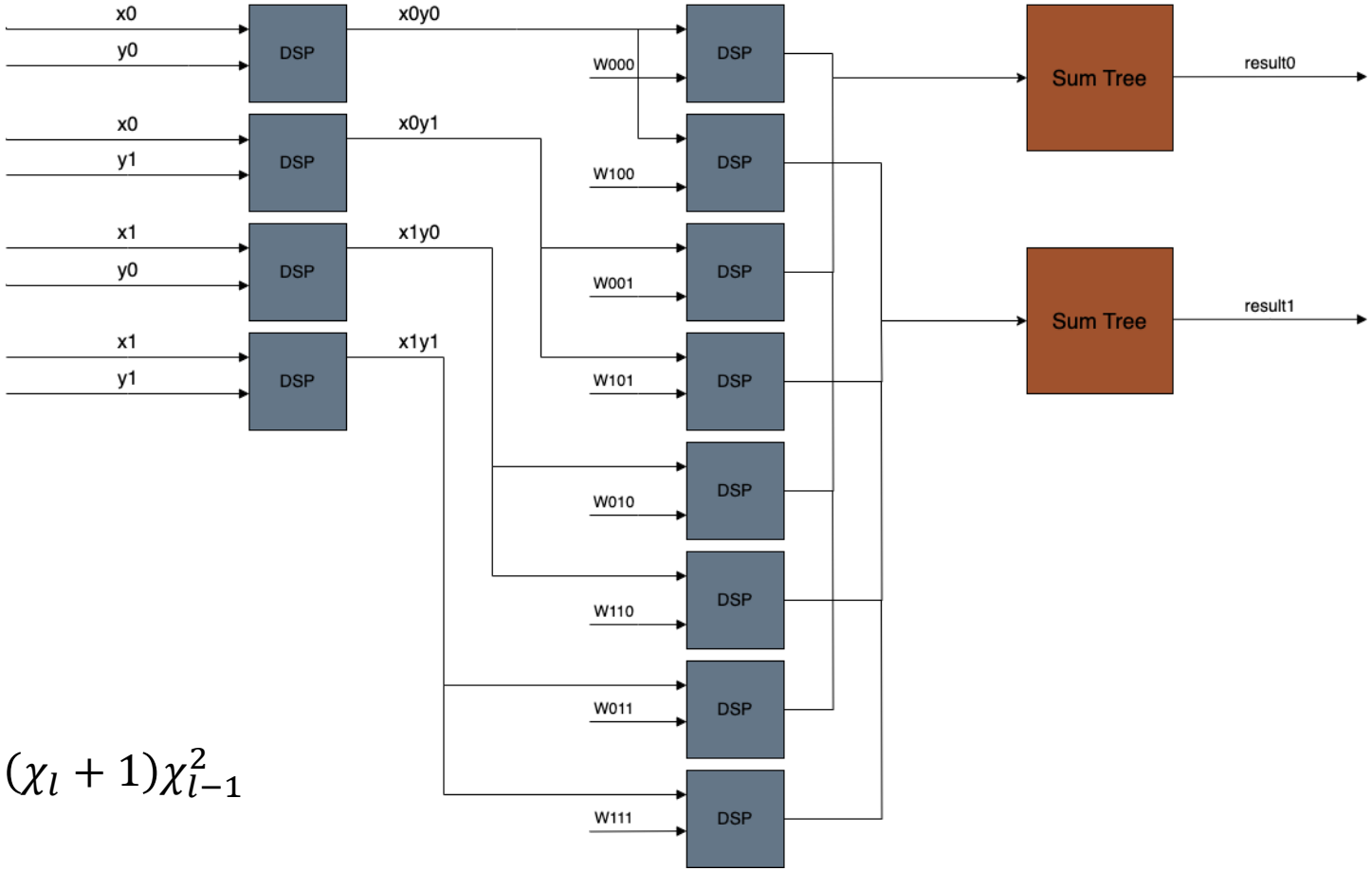
Tensor contraction

- Inference means to contract the full tree



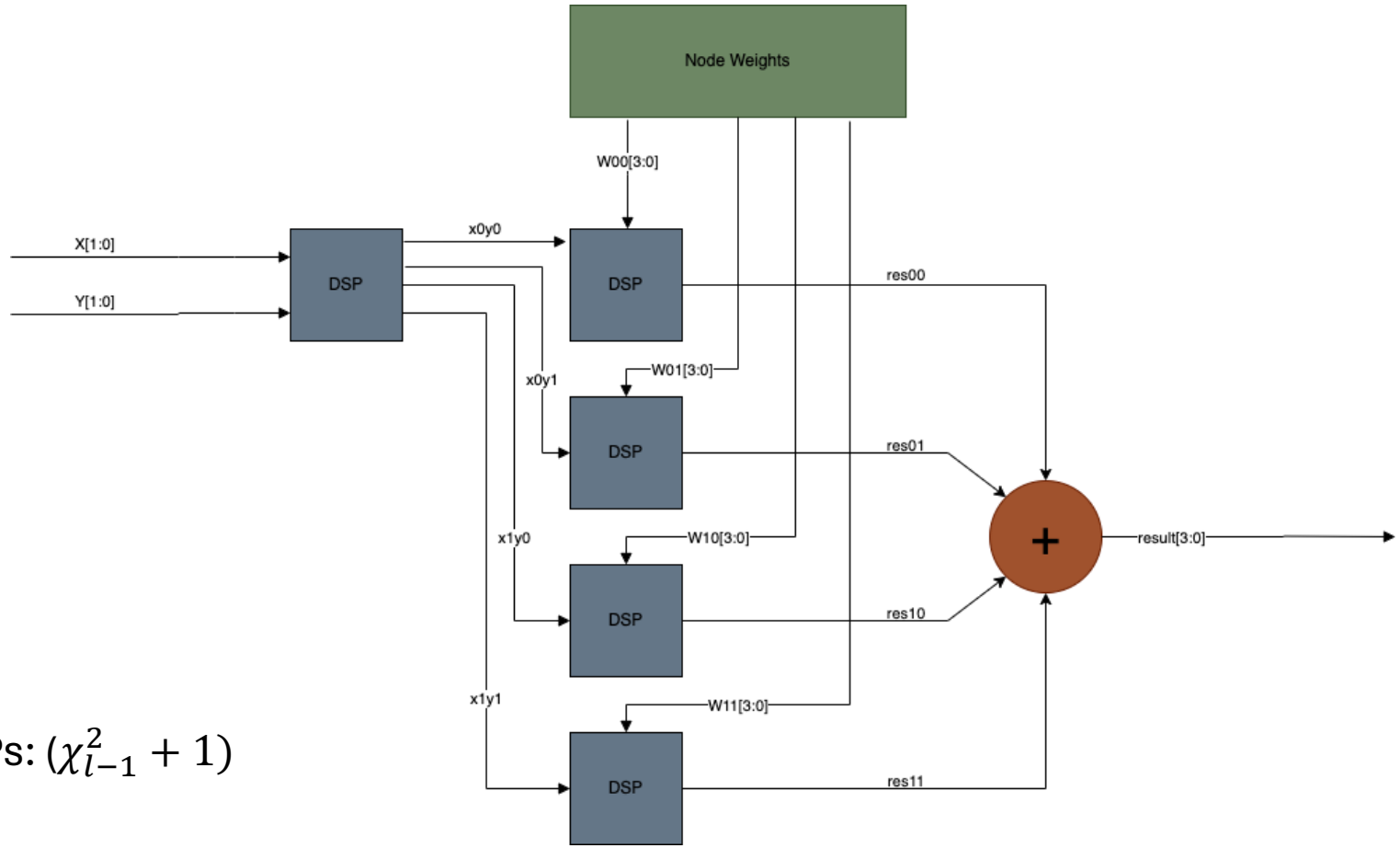
- Contraction operation in an order-3 tensor: $c_k = \sum_{ij} a_i b_j W_{ijk}$
- DSP has two inputs ⟶ two multiplication stages are needed
- We explored two degree of parallelism for the contraction
  - Full parallel – where the exploited number of DSPs is maximal
  - Partially parallel – where we introduce a reuse of the DSPs

Full parallel contraction



Number of DSPs: $(\chi_l + 1)\chi_{l-1}^2$

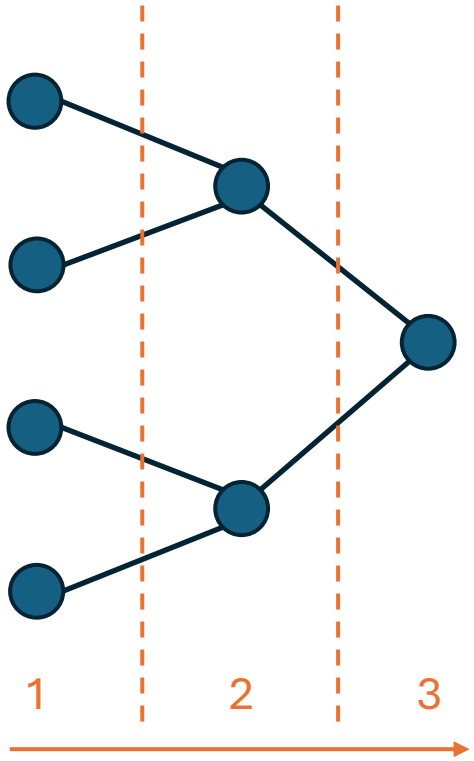
Partially parallel contraction



Number of DSPs: $(\chi_{l-1}^2 + 1)$

Execution flow

- Feature map implemented in LUTs
- Node computation is independent
- Each layer is computed in parallel
- Layers are fully pipelined
- After full contraction, output is a scalar



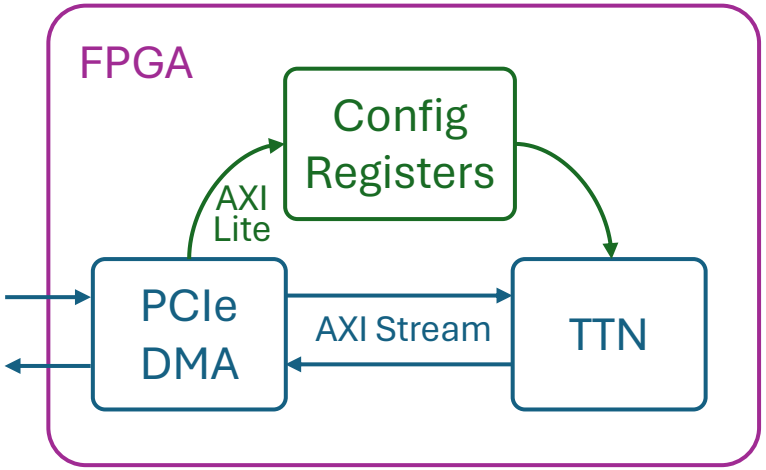
Hardware setup

Server

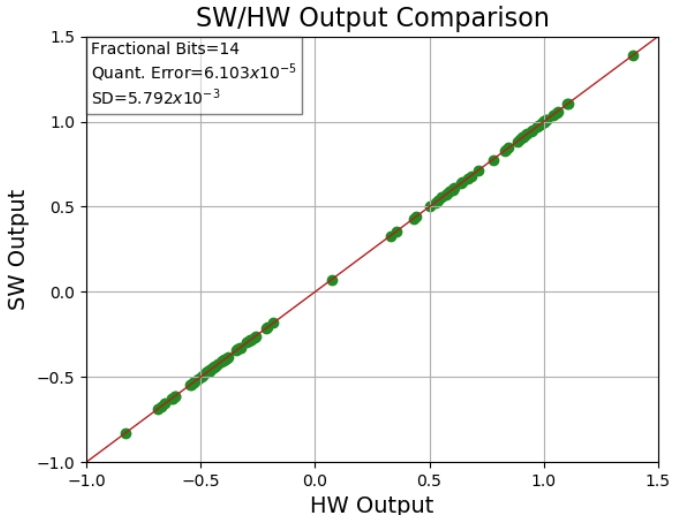
PCIe

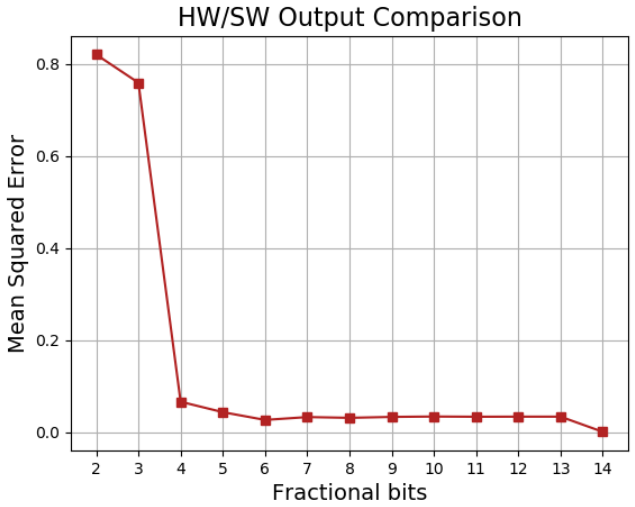
KCU1500
Kintex Ultrascale

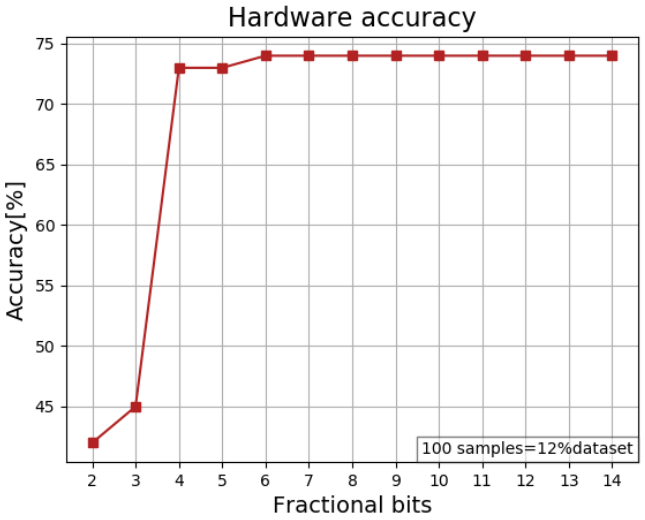
- TTN deployed on hardware accelerator
- Offloading of the TTN inference
- Application on top of Xilinx DMA drivers

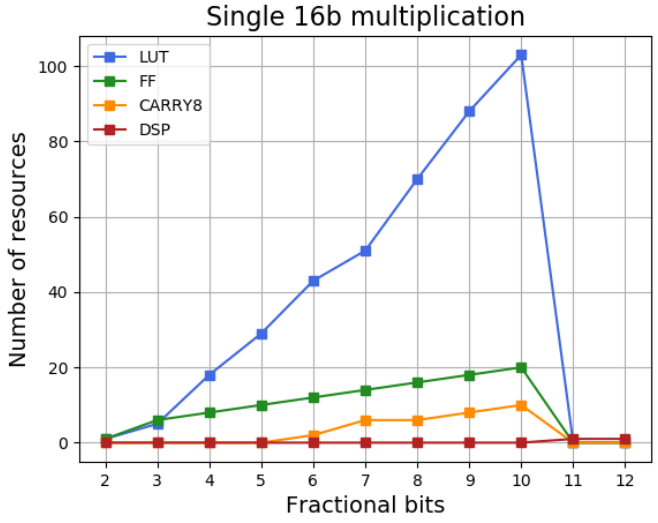


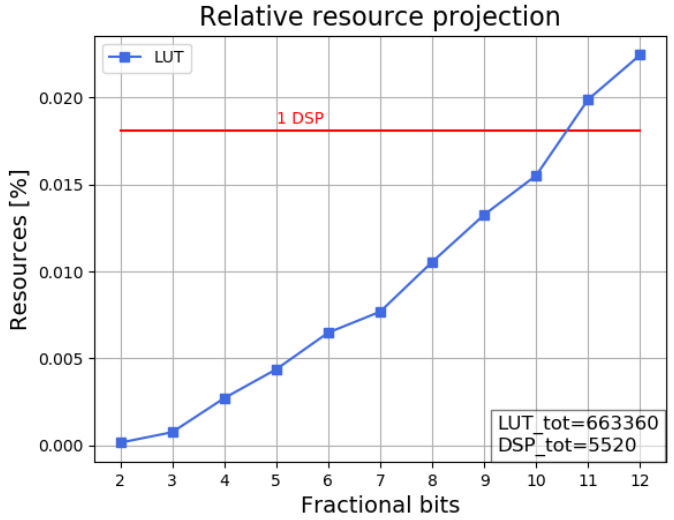
Hardware validation





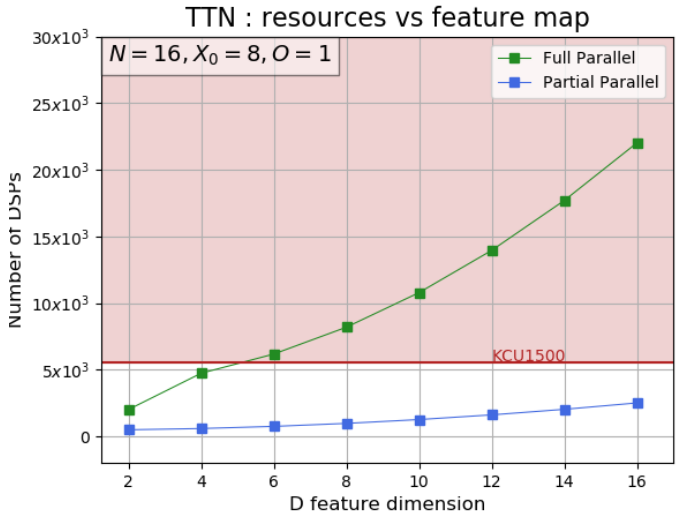


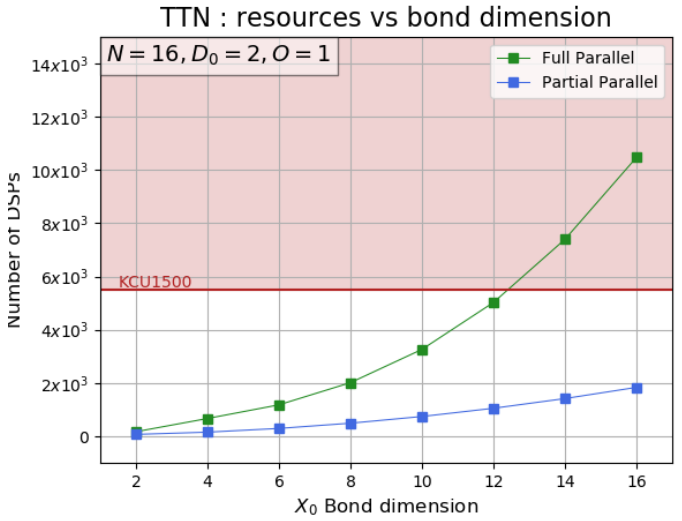


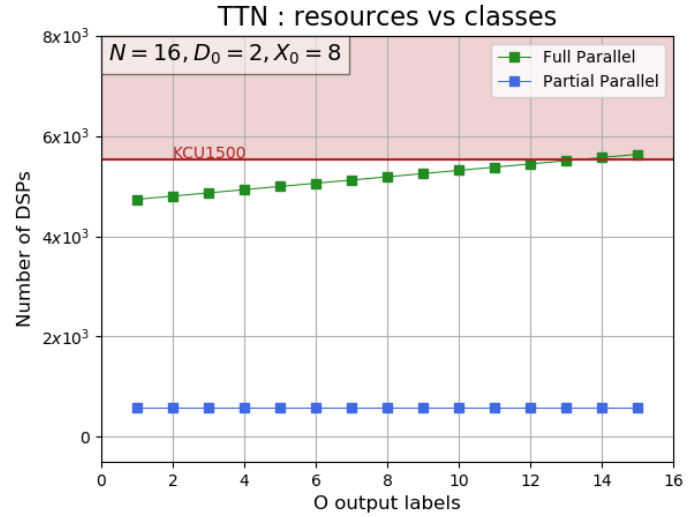


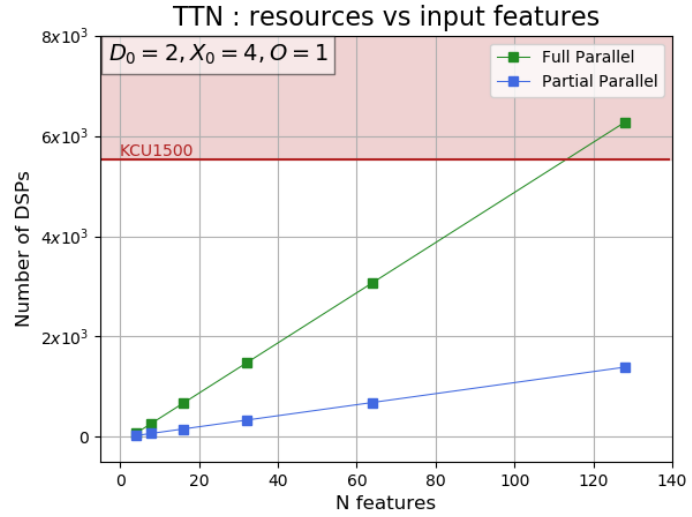
- 16-bit fixed point `<2,14>` data type
- Software/Hardware perfect matching
- Several weight-quantization alternatives tested
- DSP/LUT vs quantization shows promising results

Resources









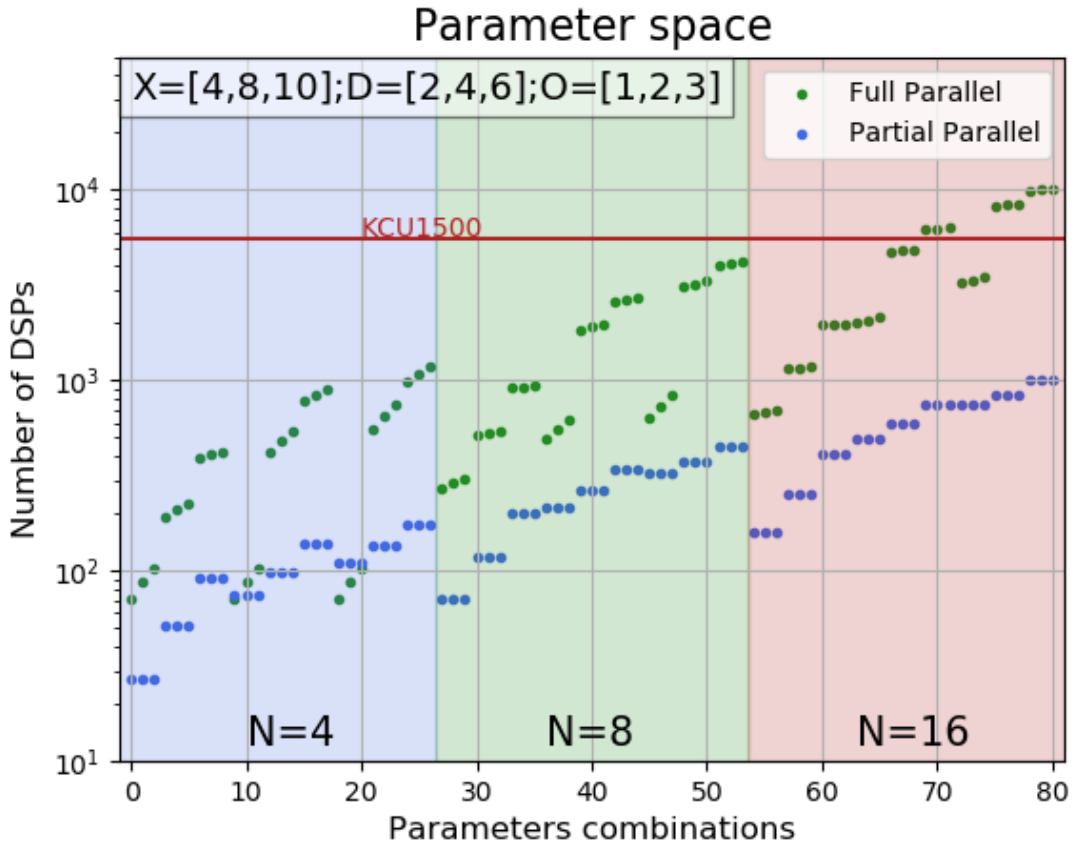
Total number of DSPs

$$\sum_{l=1}^{L} \chi_{l-1}^2 (\chi_l + 1) \frac{N}{2^l}$$

$$\sum_{l=1}^{L} (\chi_{l-1}^2 + 1) \frac{N}{2^l}$$

$O > 1$ for multi-class prediction

Resources

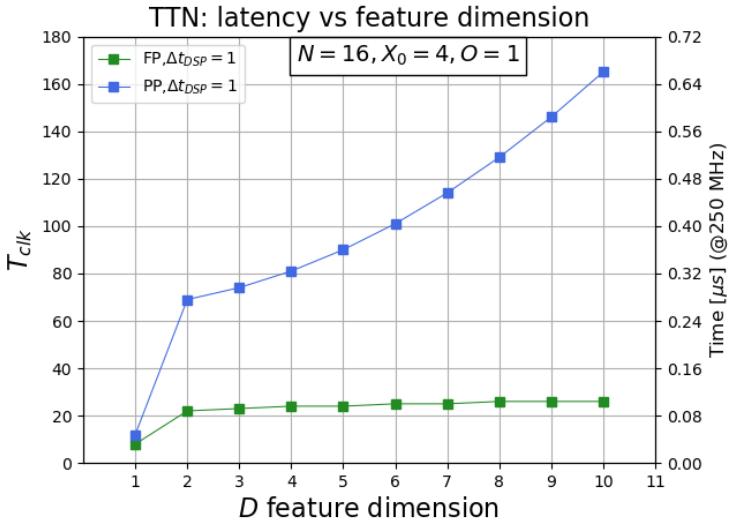


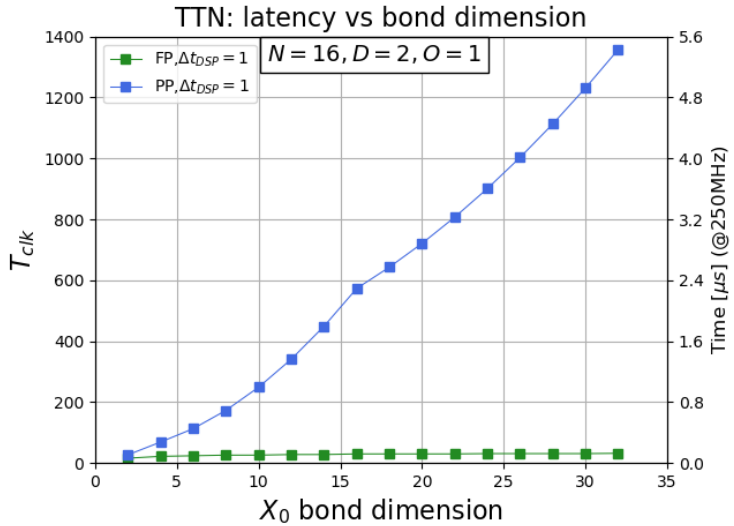
Total number of DSPs

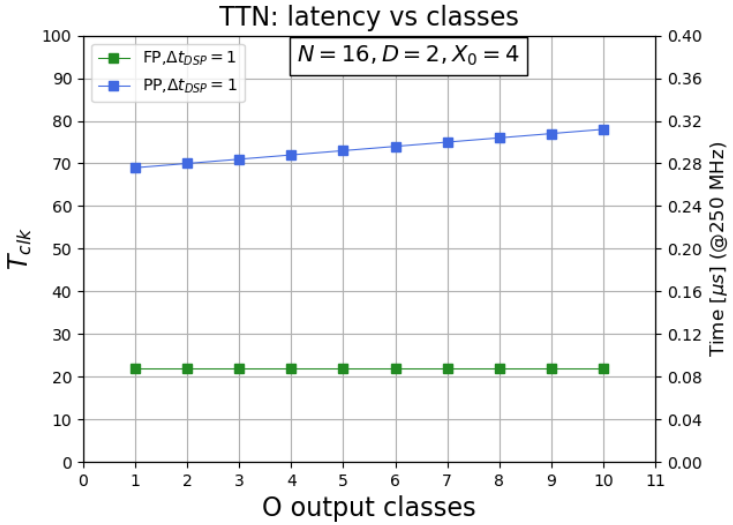
$$\sum_{l=1}^{L} \chi_{l-1}^{2} (\chi_l + 1) \frac{N}{2^l}$$

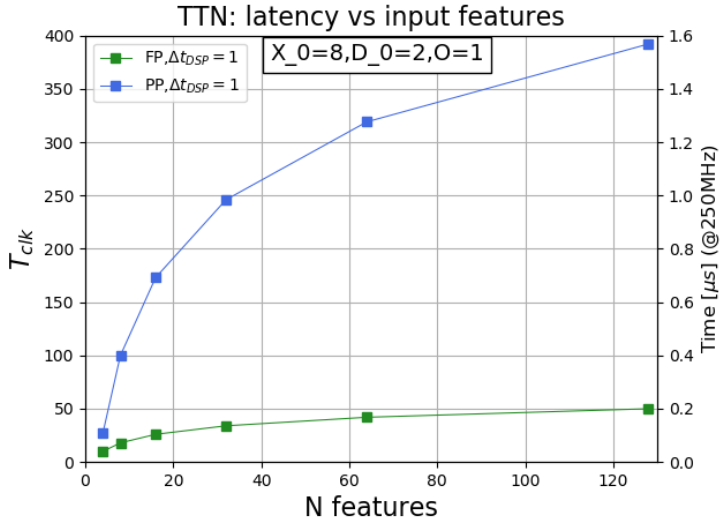
$$\sum_{l=1}^{L} (\chi_{l-1}^{2} + 1) \frac{N}{2^l}$$

Latency







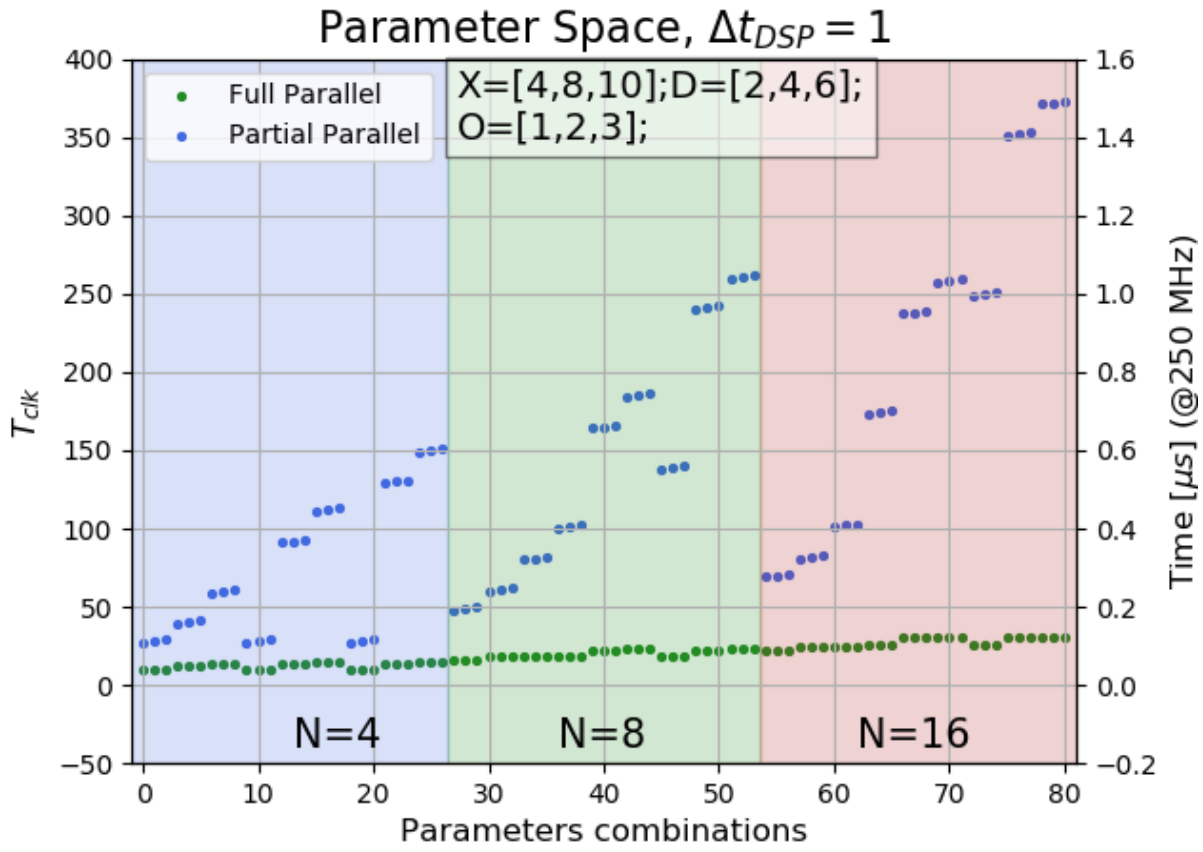


Total latency

$$\Delta_{DSP} \sum_{l=1}^{L} 2 + \log_2(\chi_{l-1}^2)$$

$$\Delta_{DSP} \sum_{l=1}^{N} \chi_{l-1}^2 + \chi_l + 1$$

Latency



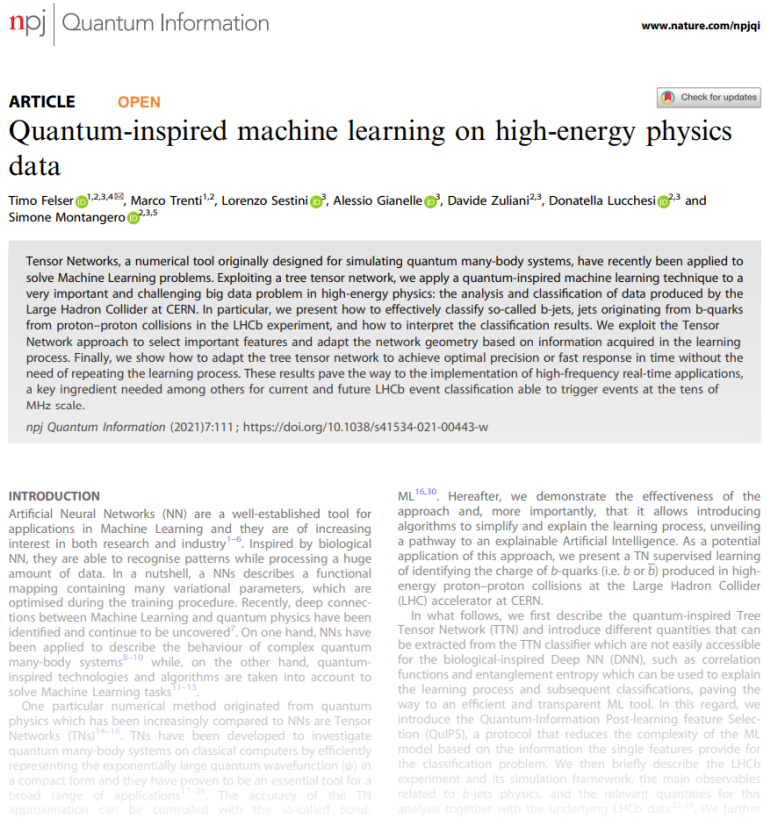
Total latency

$$\Delta_{DSP} \sum_{l=1}^{L} 2 + \log_2(\chi_{l-1}^2)$$

$$\Delta_{DSP} \sum_{l=1}^{N} \chi_{l-1}^2 + \chi_l + 1$$

Forthcoming works

- Validation with a real dataset from LHCb for b-tagging
- Integration in trigger system of HEP experiment
- Moving towards a higher-level description (HLS) ⟶ Future integration to existing ML frameworks?
- Online training for fast calibration of the network parameters



npj | Quantum Information www.nature.com/npjqi

ARTICLE OPEN Check for updates

Quantum-inspired machine learning on high-energy physics data

Timo Felser[1,2,3,4✉], Marco Trenti[1,2], Lorenzo Sestini[3], Alessio Gianelle[3], Davide Zuliani[2,3], Donatella Lucchesi[2,3] and Simone Montangero[2,3,5]

Tensor Networks, a numerical tool originally designed for simulating quantum many-body systems, have recently been applied to solve Machine Learning problems. Exploiting a tree tensor network, we apply a quantum-inspired machine learning technique to a very important and challenging big data problem in high-energy physics: the analysis and classification of data produced by the Large Hadron Collider at CERN. In particular, we present how to effectively classify so-called b-jets, jets originating from b-quarks from proton–proton collisions in the LHCb experiment, and how to interpret the classification results. We exploit the Tensor Network approach to select important features and adapt the network geometry based on information acquired in the learning process. Finally, we show how to adapt the tree tensor network to achieve optimal precision or fast response in time without the need of repeating the learning process. These results pave the way to the implementation of high-frequency real-time applications, a key ingredient needed among others for current and future LHCb event classification able to trigger events at the tens of MHz scale.

npj Quantum Information (2021)7:111 ; https://doi.org/10.1038/s41534-021-00443-w

INTRODUCTION

Artificial Neural Networks (NN) are a well-established tool for applications in Machine Learning and they are of increasing interest in both research and industry[1–6]. Inspired by biological NN, they are able to recognise patterns while processing a huge amount of data. In a nutshell, a NNs describes a functional mapping containing many variational parameters, which are optimised during the training procedure. Recently, deep connections between Machine Learning and quantum physics have been identified and continue to be uncovered[7]. On one hand, NNs have been applied to describe the behaviour of complex quantum many-body systems[8–10] while, on the other hand, quantum-inspired technologies and algorithms are taken into account to solve Machine Learning tasks[11–13].

One particular numerical method originated from quantum physics which has been increasingly compared to NNs are Tensor Networks (TNs)[14–16]. TNs have been developed to investigate quantum many-body systems on classical computers by efficiently representing the exponentially large quantum wavefunction $|\psi\rangle$ in a compact form and they have proven to be an essential tool for a broad range of applications[7,17–26]. The accuracy of the TN approximation can be controlled with the so-called bond-

ML[16,30]. Hereafter, we demonstrate the effectiveness of the approach and, more importantly, that it allows introducing algorithms to simplify and explain the learning process, unveiling a pathway to an explainable Artificial Intelligence. As a potential application of this approach, we present a TN supervised learning of identifying the charge of b-quarks (i.e. b or $\bar{b}$) produced in high-energy proton–proton collisions at the Large Hadron Collider (LHC) accelerator at CERN.

In what follows, we first describe the quantum-inspired Tree Tensor Network (TTN) and introduce different quantities that can be extracted from the TTN classifier which are not easily accessible for the biological-inspired Deep NN (DNN), such as correlation functions and entanglement entropy which can be used to explain the learning process and subsequent classifications, paving the way to an efficient and transparent ML tool. In this regard, we introduce the Quantum-Information Post-learning feature Selection (QuIPS), a protocol that reduces the complexity of the ML model based on the information the single features provide for the classification problem. We then briefly describe the LHCb experiment and its simulation framework, the main observables related to b-jets physics, and the relevant quantities for this analysis together with the underlying LHCb data[31,32]. We further