Università degli Studi di Padova

Dipartimento di Fisica e Astronomia Galileo Galilei

Optimizing Tree Tensor Networks for classification on hardware accelerators

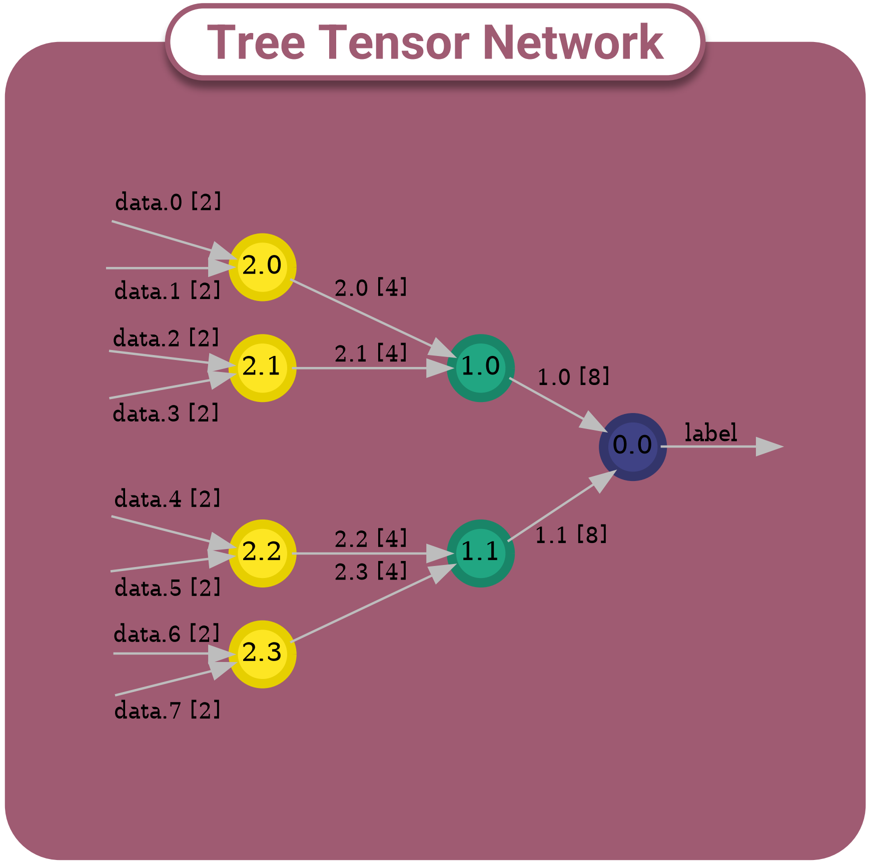
PhD Student

Alberto Coppi

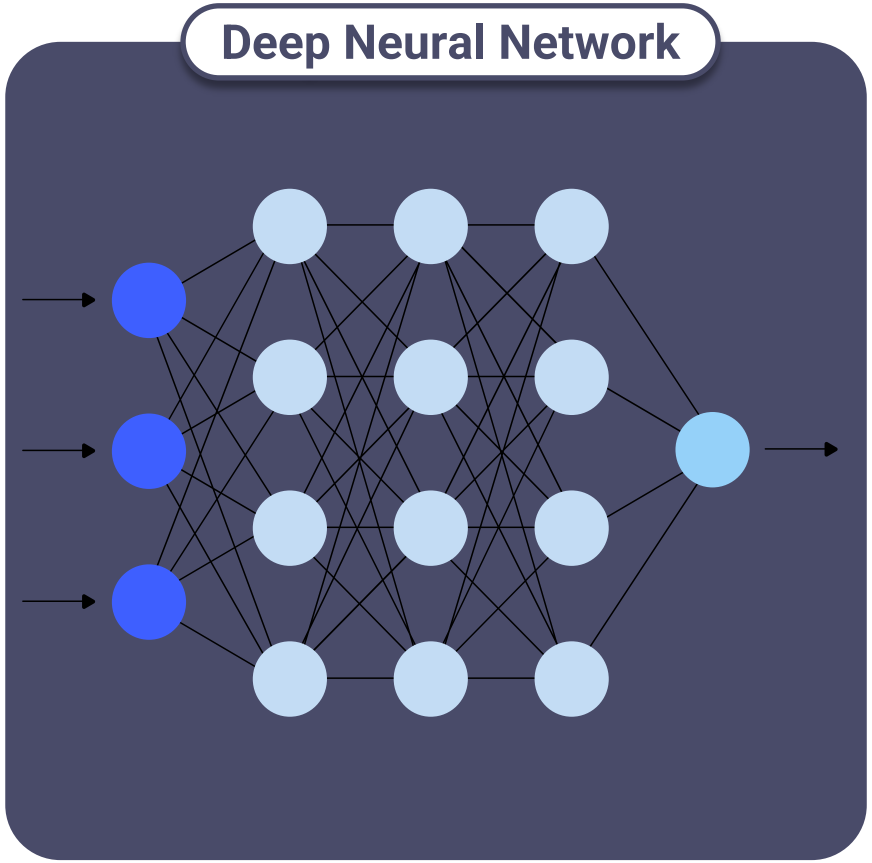
Workshop on Tensor Networks and (Quantum) Machine Learning for High-Energy Physics

Date: 05/11/2024

Tree Tensor Networks

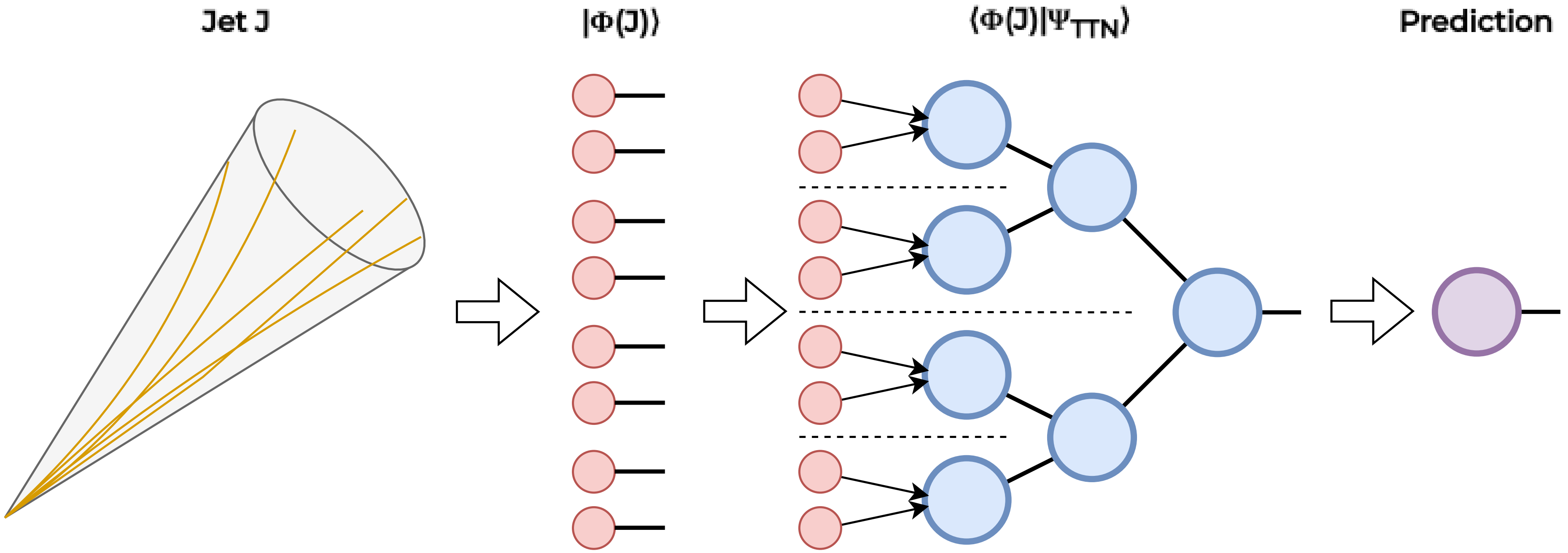


Tree Tensor Network



Deep Neural Network

Tree Tensor Networks classifiers



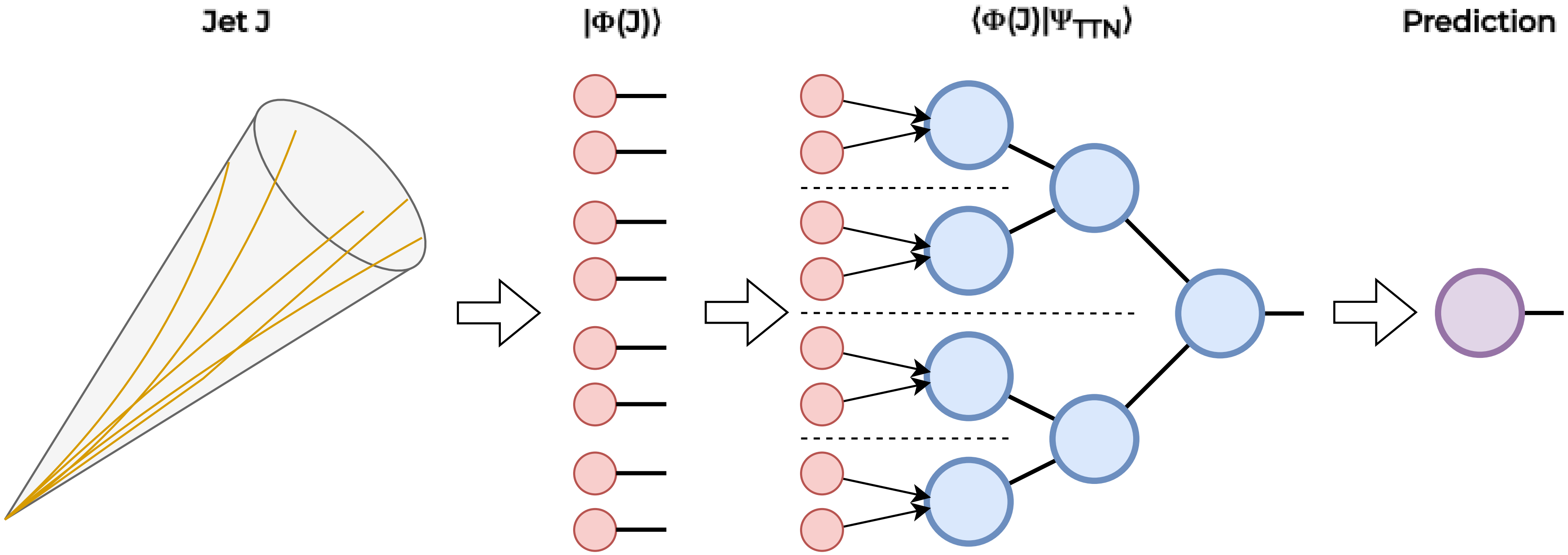
Jet J

$|\Phi(J)\rangle$

$\langle\Phi(J)|\Psi_{TTN}\rangle$

Prediction

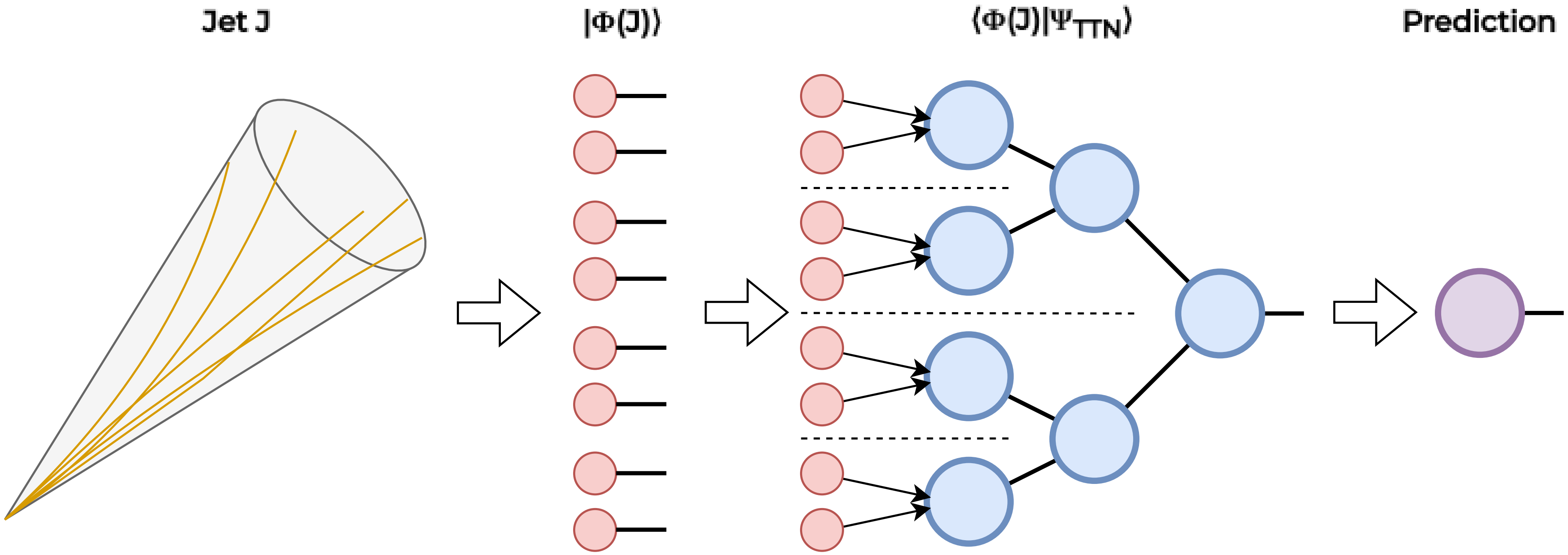
Tree Tensor Networks classifiers

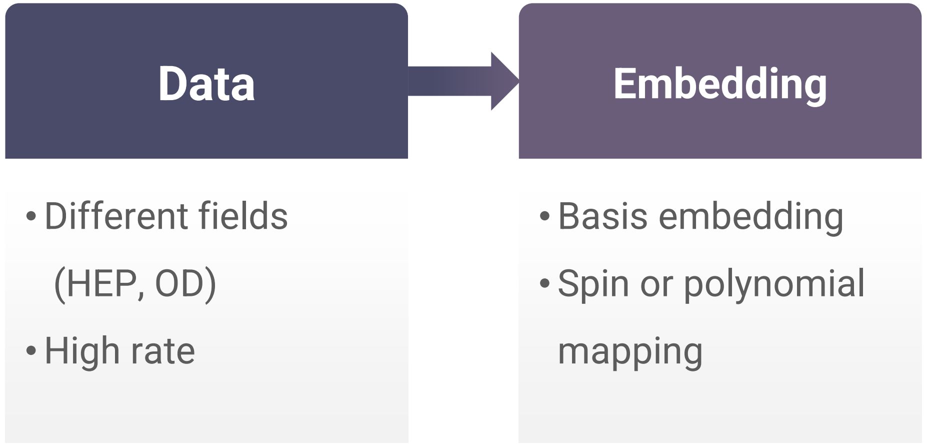


Data

- Different fields (HEP, OD)
- High rate

Tree Tensor Networks classifiers





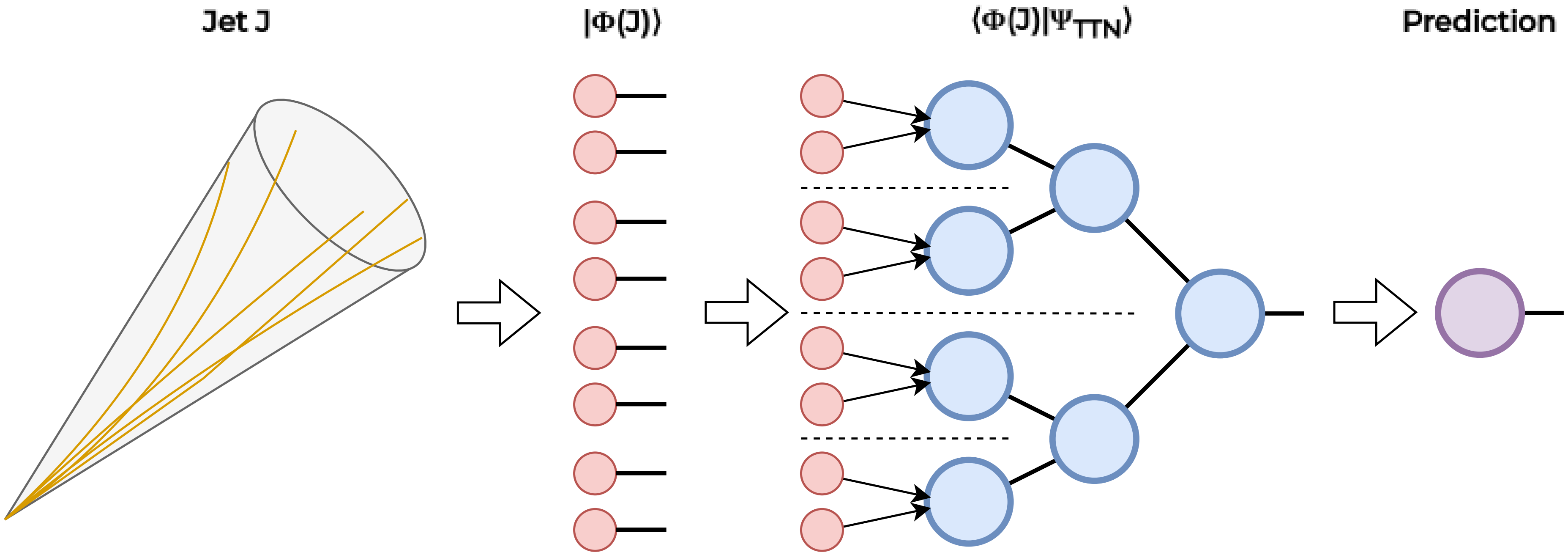
Data

- Different fields (HEP, OD)
- High rate

Embedding

- Basis embedding
- Spin or polynomial mapping

Tree Tensor Networks classifiers



Data

- Different fields (HEP, OD)
- High rate

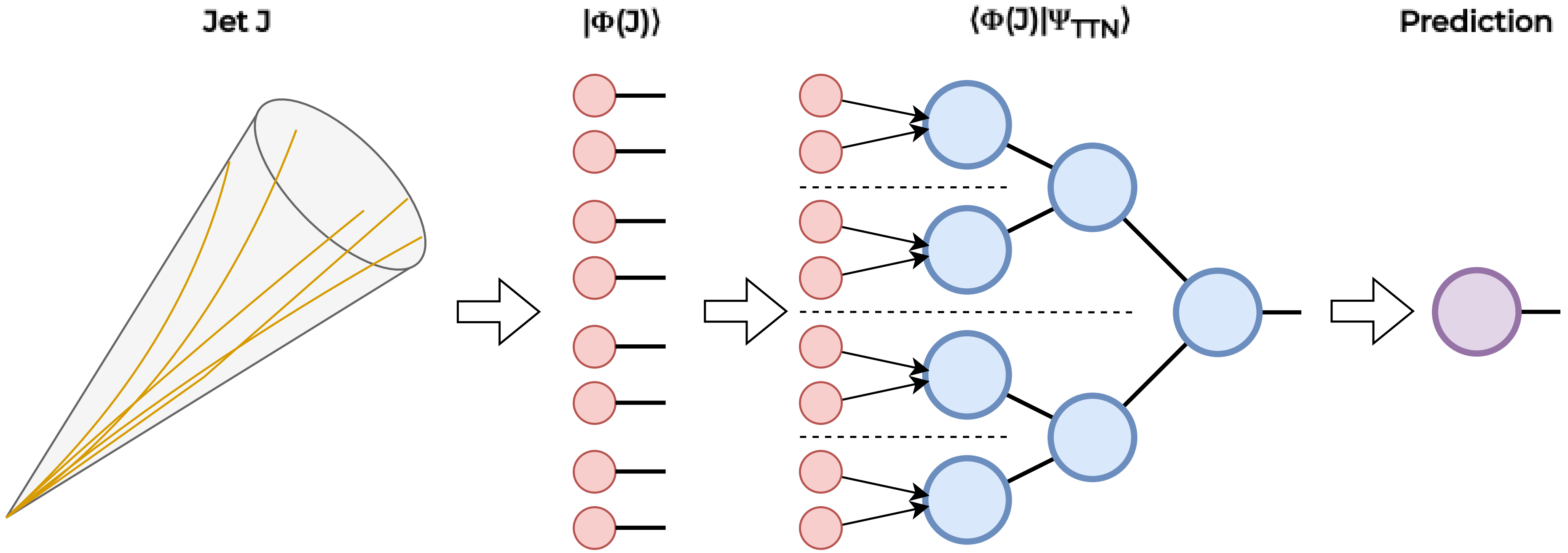
Embedding

- Basis embedding
- Spin or polynomial mapping

(Parallel) Contraction

- Calculate fidelity of TTN state over data
- Embarrassingly parallel operation

Tree Tensor Networks classifiers



Data → **Embedding** → **(Parallel) Contraction** → **Prediction**

Data

- Different fields (HEP, OD)
- High rate

Embedding

- Basis embedding
- Spin or polynomial mapping

(Parallel) Contraction

- Calculate fidelity of TTN state over data
- Embarrassingly parallel operation

Prediction

- Scalar or vector out
- Plug into loss function for optimisation

Tree Tensor Networks classifiers:

Why?

Software

Training

- Initialisation
- Optimisation:
  - Global SGD
  - Sweeping

Explainability

- Measurement of physical quantities

Hardware

FPGA

- Deterministic latency
- Limited resources

Use case

- **Jet tagging** for HEP experiments, e.g. CMS
- Currently done **offline** by complex ML models like ParticleNet on **already filtered data**
- Objective: deploy a **tagger online** in the L1 trigger of CMS experiment to improve selection efficiency

Tree Tensor Networks classifiers:

Why?

Software

Training

- Initialisation
- Optimisation:
    - Global SGD
    - Sweeping

Explainability

- Measurement of physical quantities

FPGA

Explainability

- The model can be "explained" through physical measurements
- ➢ **No black-boxes** for filtering out ~98% of data

Compressibility

- The number of parameters can be reduced, post-learning
- ➢ **Fit** the model **to limited resources** hardware

Speed

- Based on simple operations and **parallelizable**
- ➢ **Compliance with latency limits**

Tree Tensor Networks classifiers:

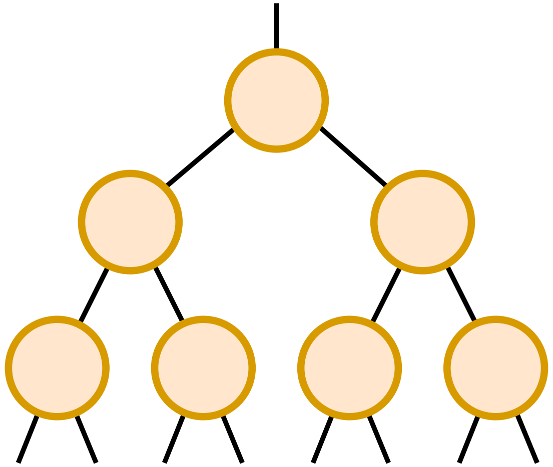
Training

Initialisation

Optimisation

Tree Tensor Networks classifiers:

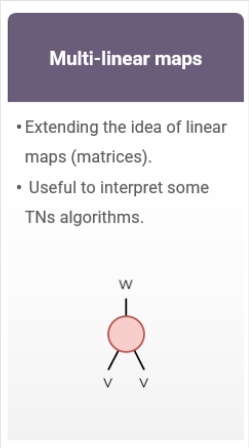
Initialisation



Randomly initialised

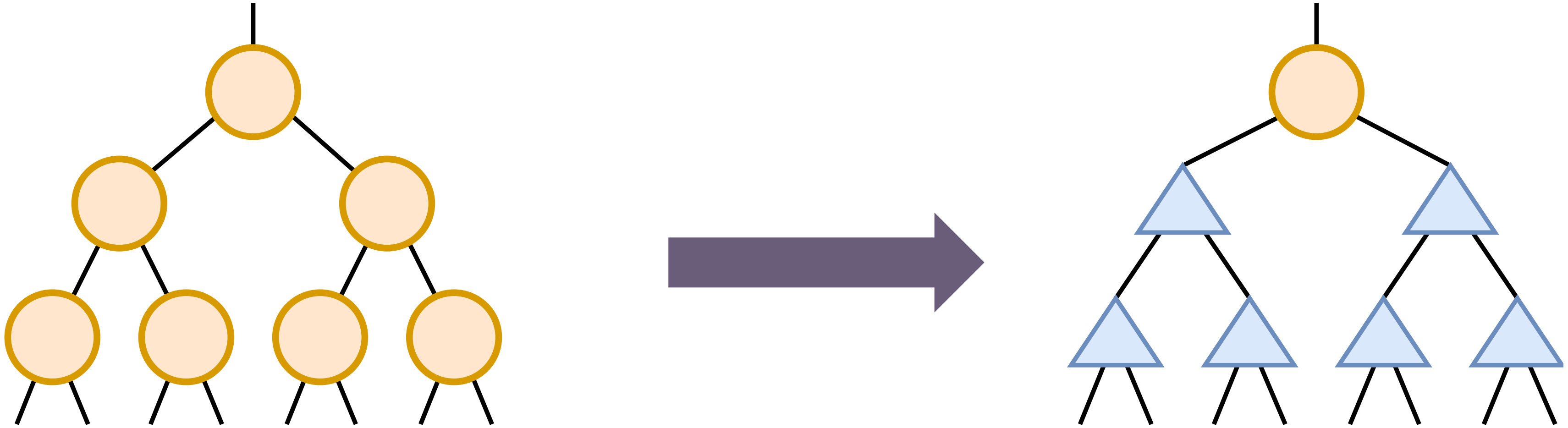
Tree Tensor Networks classifiers:

Initialisation



Multi-linear maps

- Extending the idea of linear maps (matrices).
- Useful to interpret some TNs algorithms.



Randomly initialised

Each layer progressively project data into a lower dimensional space[1]

[1] E Miles Stoudenmire, Learning relevant features of data with multi-scale tensor networks, *Quantum Sci. Technol.* **3** 034003 (2018). https://doi.org/10.1088/2058-9565/aaba1a

Tree Tensor Networks classifiers:

Optimisation

Global SGD

Pros:

- Treat TTN parameters as classical NN parameters
- Exploit **gradient tracking automation** of PyTorch

Cons:

- Suffers from **barren plateaus**

VS

Sweeping

Pros:

- Decompose a large problem in many **smaller problems**
- More **stable and robust**, enabling training of larger models

Cons:

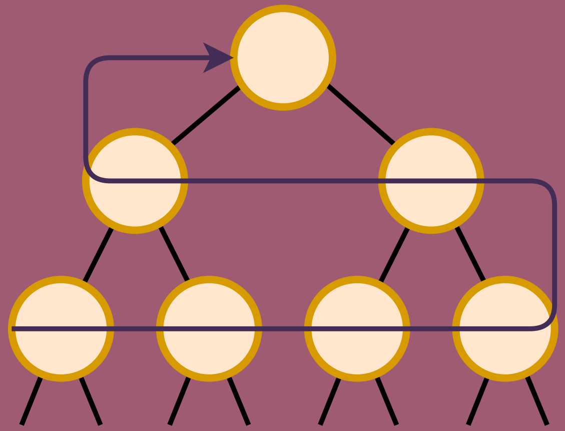
- Must be **manually implemented**, thus can be **slower**

Tree Tensor Networks classifiers:

Optimisation

VS

Sweeping



Pros:

- Decompose a large problem in many **smaller problems**
- More **stable and robust**, enabling training of larger models

Cons:

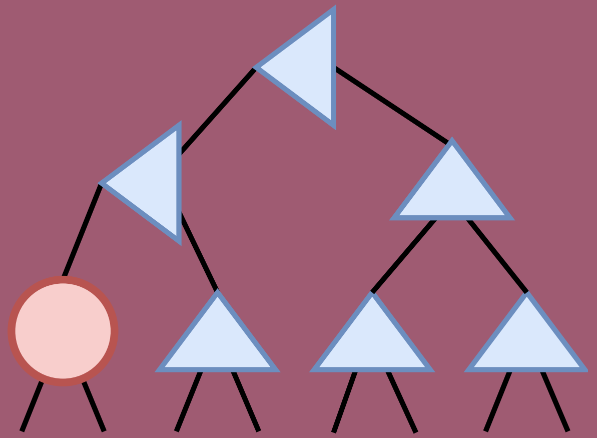
- Must be **manually implemented**, thus can be **slower**

Tree Tensor Networks classifiers:

Optimisation

VS

Sweeping



Pros:

- Decompose a large problem in many **smaller problems**
- More **stable and robust**, enabling training of larger models

Cons:

- Must be **manually implemented**, thus can be **slower**

Tree Tensor Networks classifiers:

Optimisation

Global SGD

Pros:

- Treat TTN parameters as classical NN parameters
- Exploit **gradient tracking automation** of PyTorch

Cons:

- Suffers from **barren plateaus**

VS

Sweeping

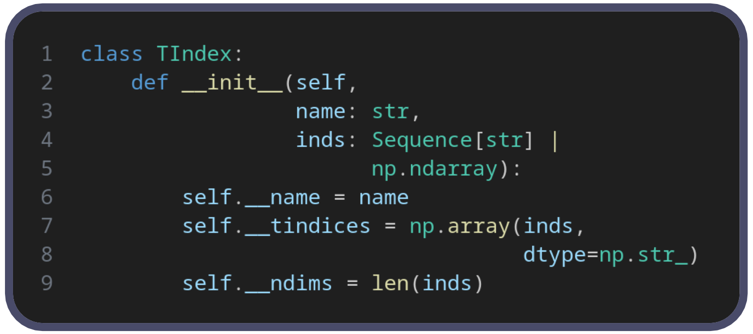
Pros:

- Decompose a large problem in many **smaller problems**
- More **stable and robust**, enabling training of larger models

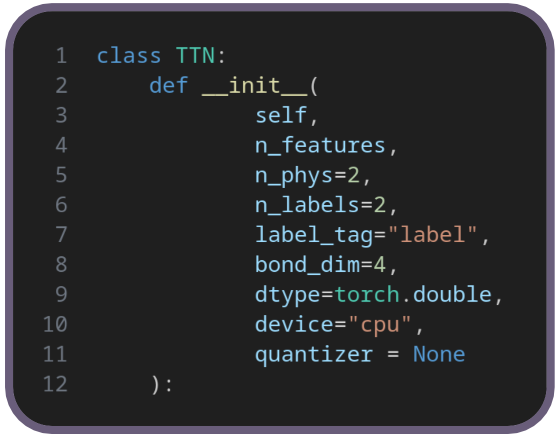
Cons:

- Must be **manually implemented**, thus can be **slower**

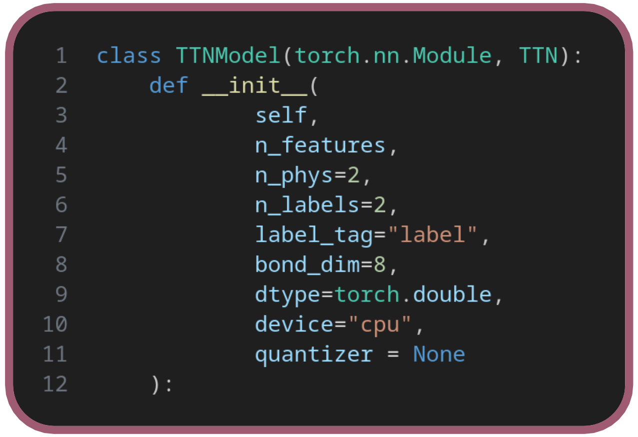
Software implementation



```python
class TIndex:
    def __init__(self,
                 name: str,
                 inds: Sequence[str] |
                       np.ndarray):
        self.__name = name
        self.__tindices = np.array(inds,
                                   dtype=np.str_)
        self.__ndims = len(inds)
```



```python
class TTN:
    def __init__(
            self,
            n_features,
            n_phys=2,
            n_labels=2,
            label_tag="label",
            bond_dim=4,
            dtype=torch.double,
            device="cpu",
            quantizer = None
    ):
```



```python
class TTNModel(torch.nn.Module, TTN):
    def __init__(
            self,
            n_features,
            n_phys=2,
            n_labels=2,
            label_tag="label",
            bond_dim=8,
            dtype=torch.double,
            device="cpu",
            quantizer = None
    ):
```

Characteristics

- **Open-source**
- Developed **from scratch**
- Based on **PyTorch**
- **Enables to train and explain a TTN ML model**

Structure

`TIndex`

- Base class for **tensor indexing**
- Can be used as **key** in dictionaries

`TTN`

- Fundamental class to construct a **full TTN**
- Equipped with methods to enable **TTN functionality**: **contraction**, initialization, derivative, **expectation value**, entanglement **entropy**, drawing

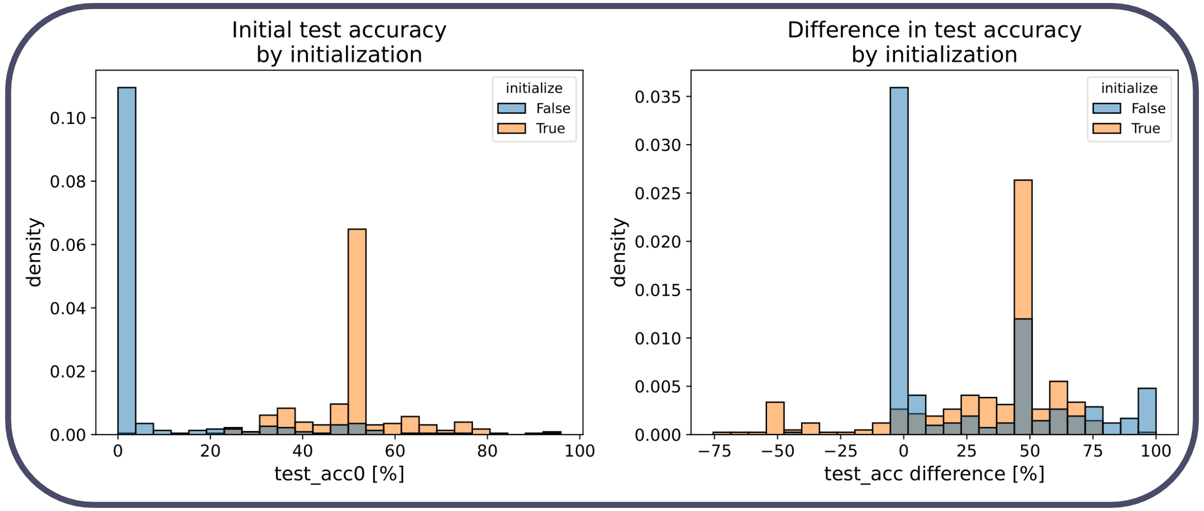
`TTNModel`

- Derived class of both `TTN` and `torch.nn.Model`
- This provides a ML based approach to optimise the TTN, with (an almost free) **SGD**
- Implements the **Sweeping** optimisation algorithm

Tests on hyperparameters

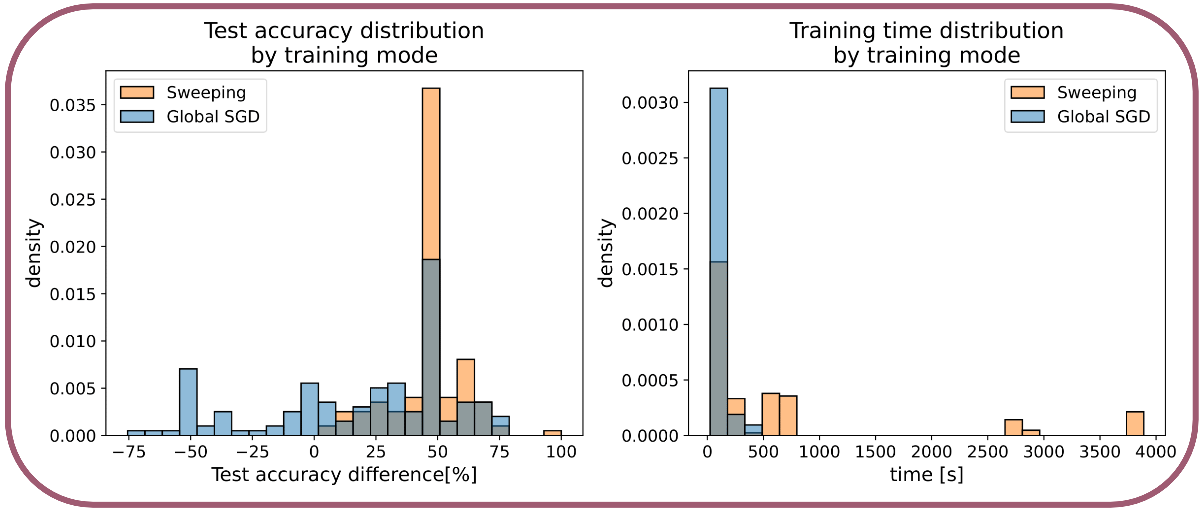
Initialisation on/off:

- The initialisation procedure moves the model towards optimum
- From there, the **training** procedure **is facilitated** to find the optimum



Global SGD vs. Sweeping:

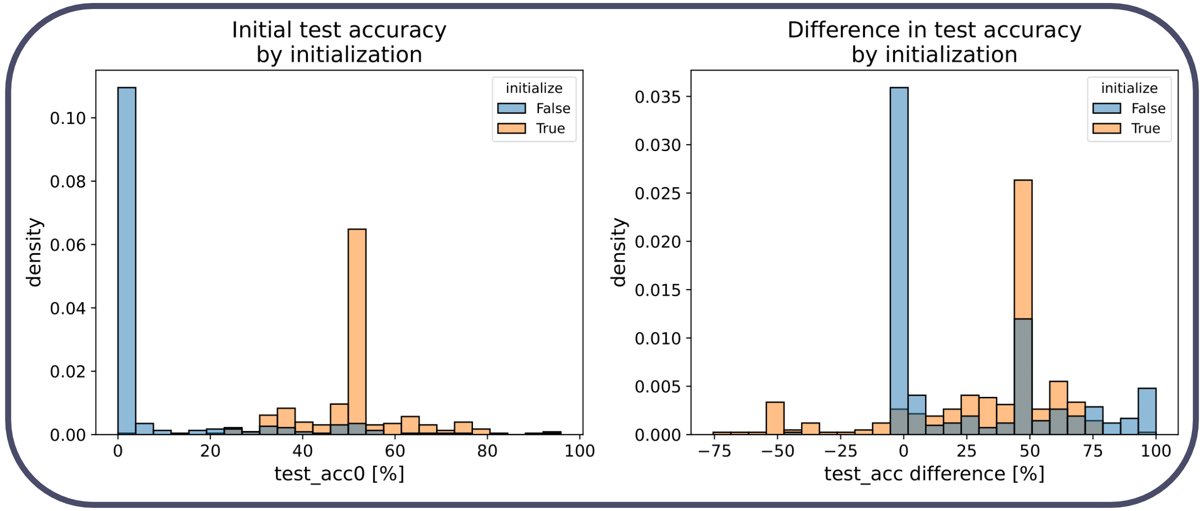
- **Sweeping is more stable** and robust, reaching the optimum more frequently, **but slower**
- Global SGD doesn't work for large models



Tests on hyperparameters

Initialisation on/off:

- The initialisation procedure moves the model towards optimum
- From there, the **training** procedure **is facilitated** to find the optimum

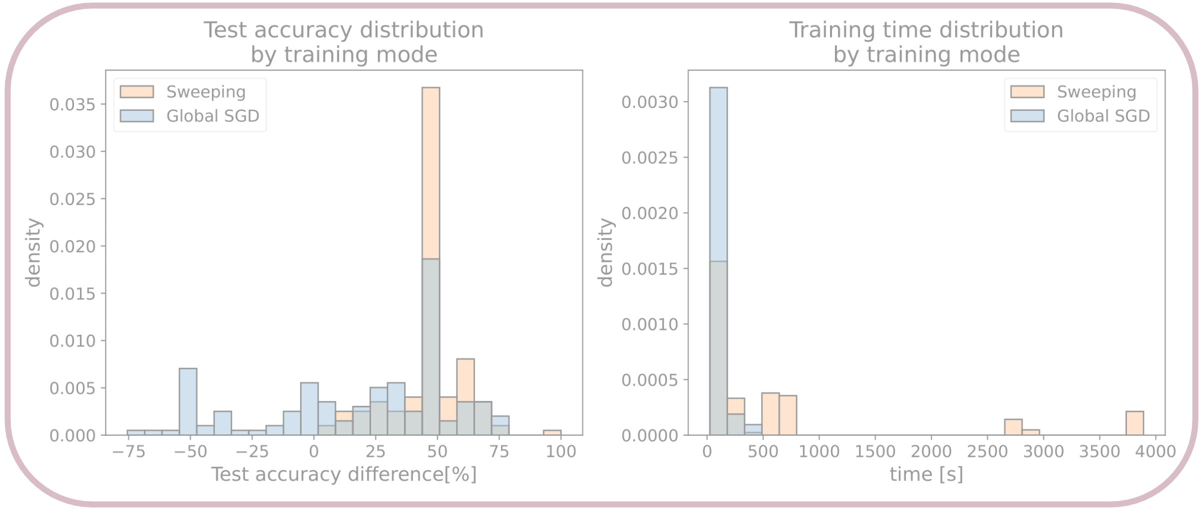


Initial test accuracy by initialization

Difference in test accuracy by initialization

Global SGD vs. Sweeping:

- **Sweeping is more stable** and robust, reaching the optimum more frequently, **but slower**
- Global SGD doesn’t work for large models



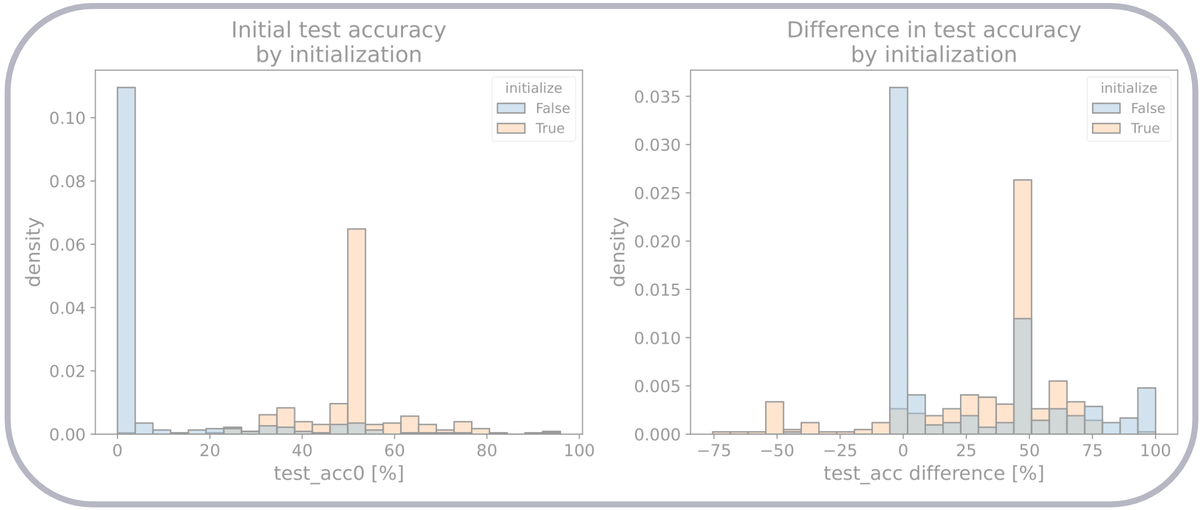
Test accuracy distribution by training mode

Training time distribution by training mode

Tests on hyperparameters

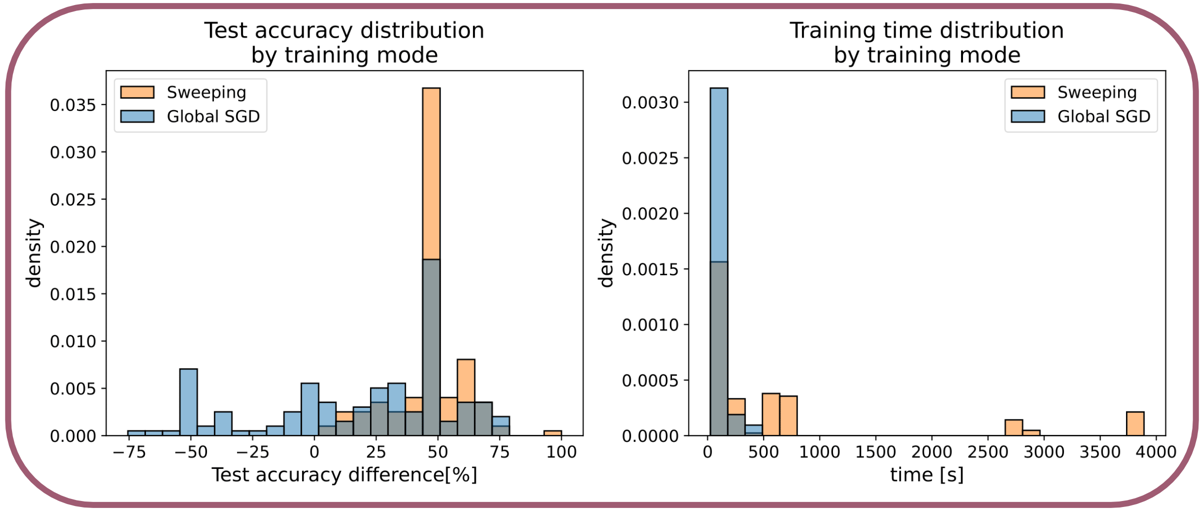
Initialisation on/off:

- The initialisation procedure moves the model towards optimum
- From there, the **training** procedure **is facilitated** to find the optimum



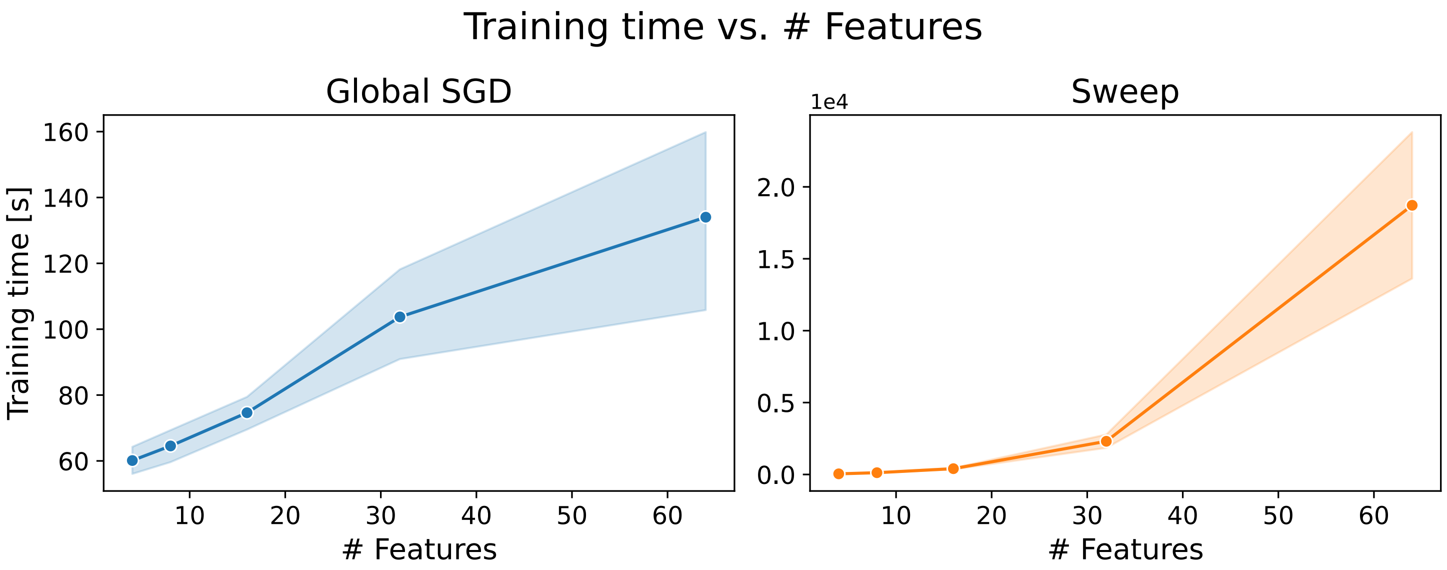
Global SGD vs. Sweeping:

- **Sweeping is more stable** and robust, reaching the optimum more frequently, **but slower**
- Global SGD doesn’t work for large models



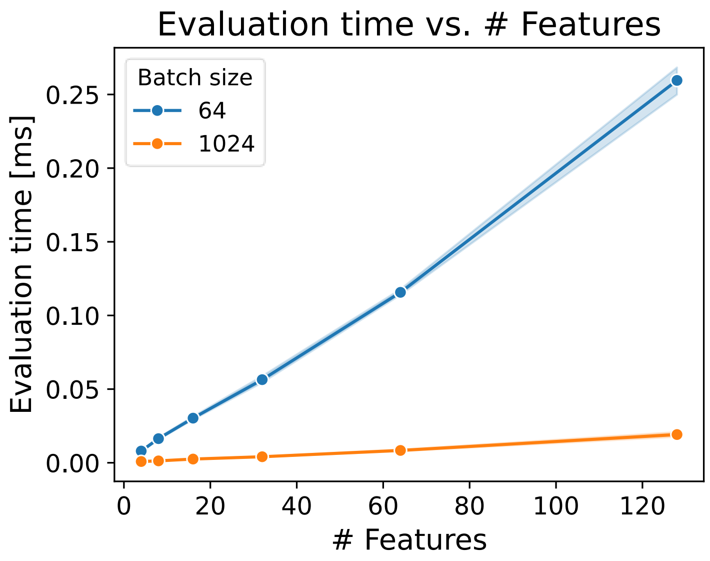
Tests on hyperparameters:

Performance



- Approximately linear dependence (thanks PyTorch)

- Expected at least quadratic dependence (the number of nodes is $\mathcal{O}(N^2)$)
- Actually worse



- Roughly linear dependence (NVIDIA GTX 1050)
- Greater batch size -> better exploitation of parallel computing capabilities (thanks PyTorch)

Tests on synthetic datasets

Iris

Commonly used in ML for benchmarking purposes

Characteristics

- 4 features
- 150 samples
- 3 classes:
  Iris-setosa, Iris-versicolor, Iris-virginica

Titanic

Commonly used in ML for benchmarking purposes

Characteristics

- 8 features (originally 13)
- 1043 samples (originally 1309)
- 2 classes:
  (not) survived

LHCb

Dataset coming from LHCb open data, already used in literature to test TTNs in ML [2]

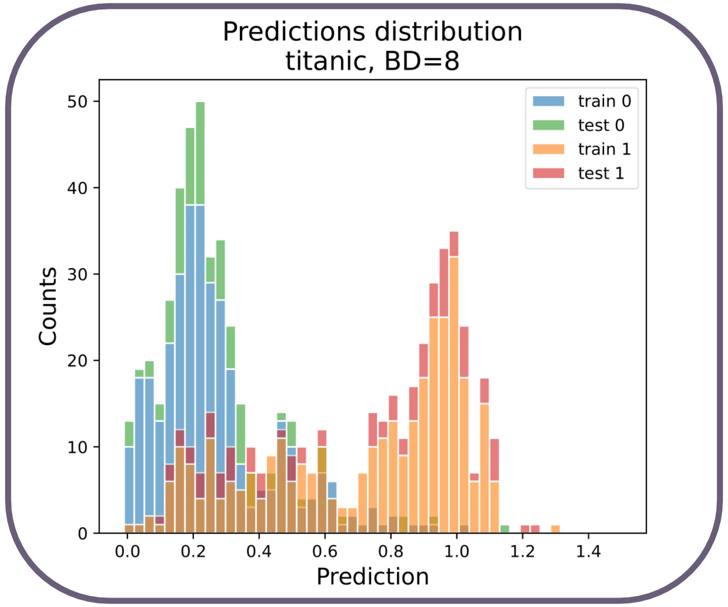
Characteristics

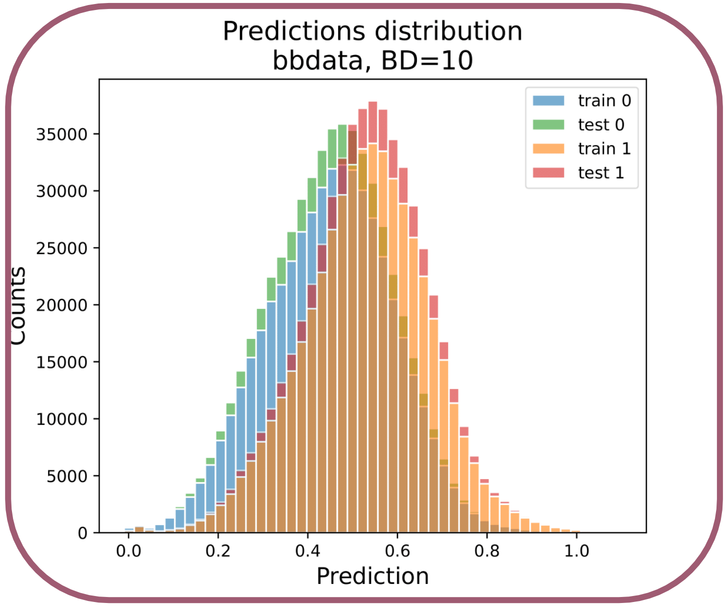
- 16 features
- $\sim 1.1 \times 10^6$ samples
- 2 classes:
  b vs $\bar{b}$

[2] Felser, T., Trenti, M., Sestini, L. *et al.* Quantum-inspired machine learning on high-energy physics data. *npj Quantum Inf* **7**, 111 (2021). https://doi.org/10.1038/s41534-021-00443-w

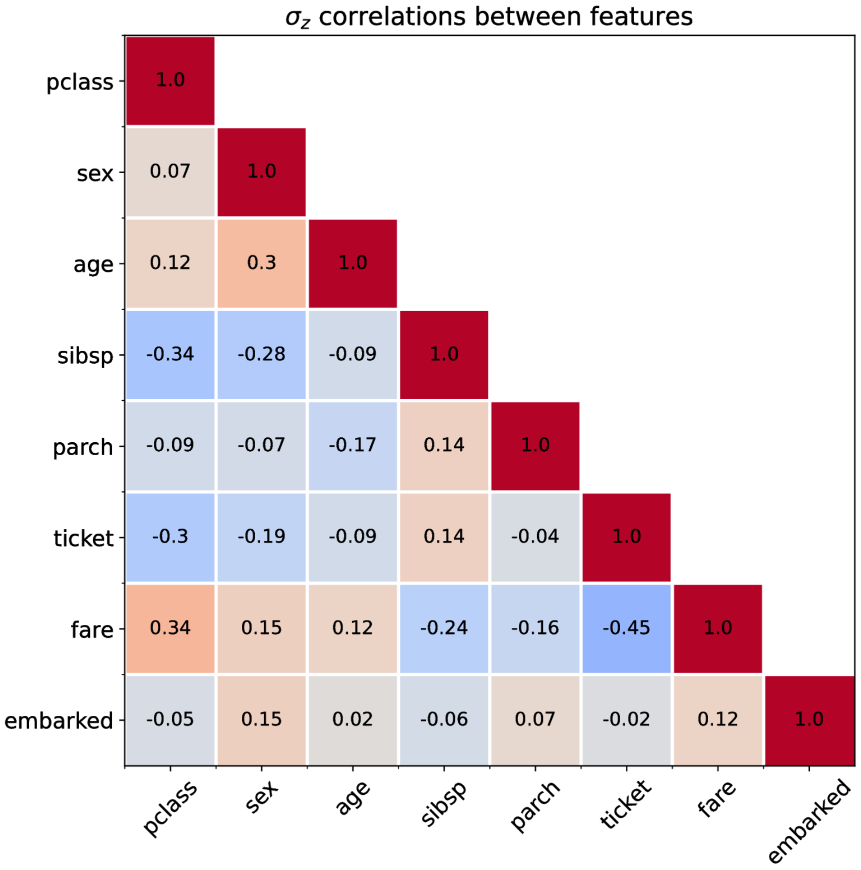
Tests on synthetic datasets

	Iris	Titanic	LHCb
Accuracy (Training/Test)	99,1% / 96,7%	80,9% / 77,0%	61,7% / 61,8%
AUC	1,0	0,83	0,66
Accuracy (smaller)	96,7% / 96,7%	79,3% / 74,1%	60,0% / 60,3%
AUC (smaller)	0,99	0,84	0,63

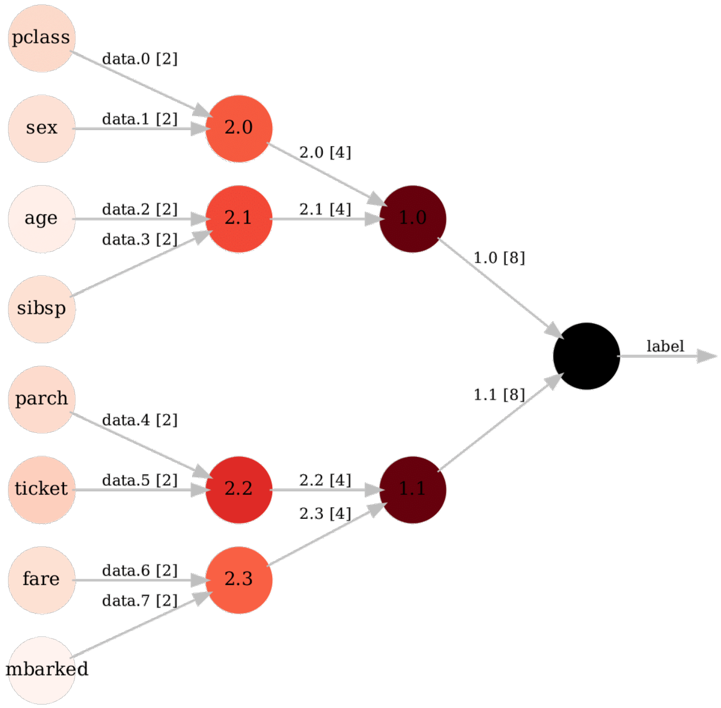

Predictions distribution
titanic, BD=8


Predictions distribution
bbdata, BD=10

Tests on synthetic datasets:
Explainability

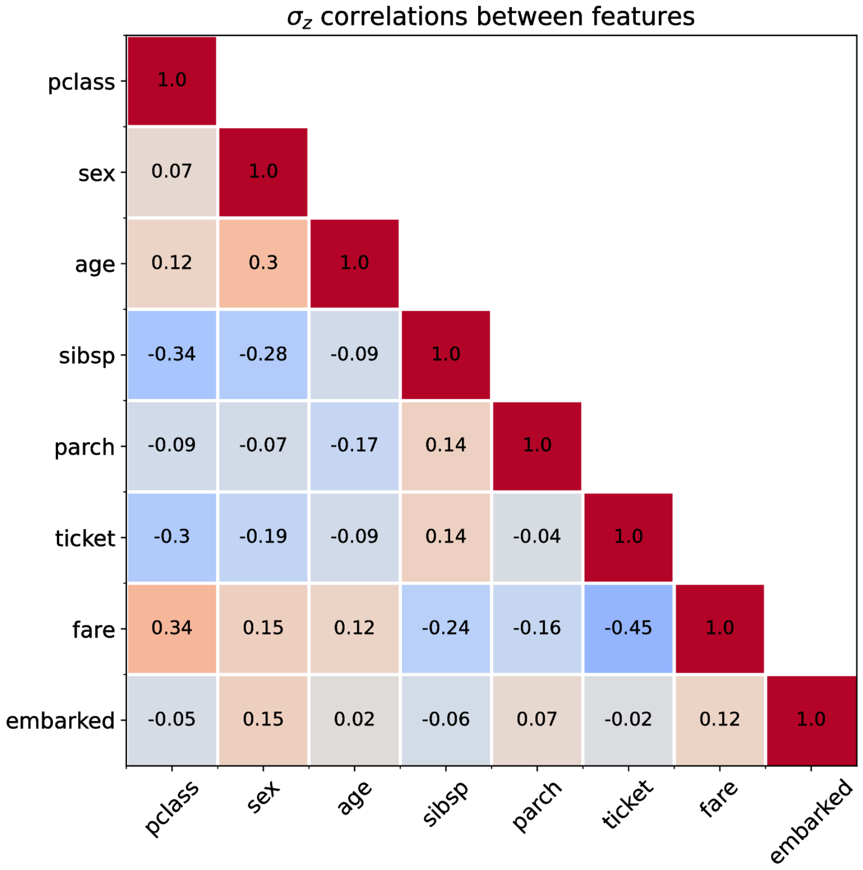


Two-sites σ_z correlation



Entanglement entropy

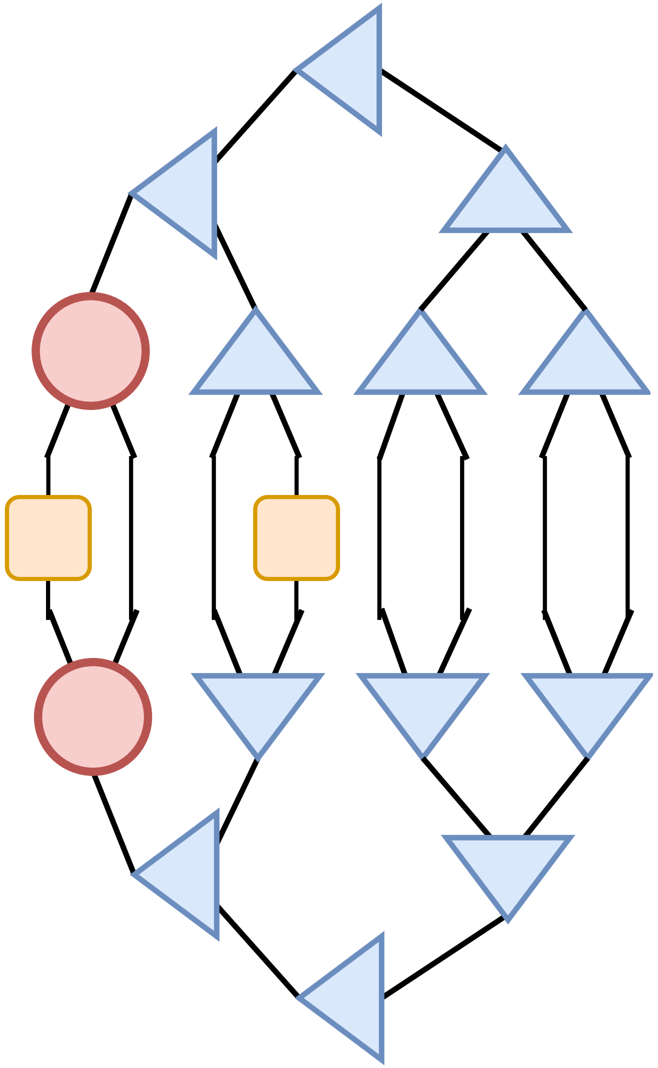
Tests on synthetic datasets:
Explainability



σ_z correlations between features

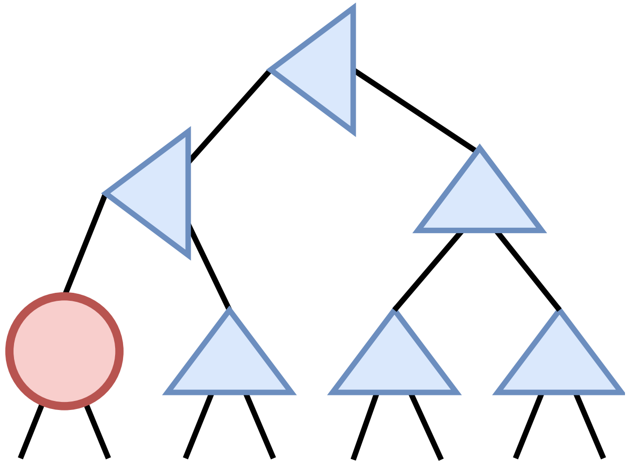
	pclass	sex	age	sibsp	parch	ticket	fare	embarked
pclass	1.0							
sex	0.07	1.0						
age	0.12	0.3	1.0					
sibsp	-0.34	-0.28	-0.09	1.0				
parch	-0.09	-0.07	-0.17	0.14	1.0			
ticket	-0.3	-0.19	-0.09	0.14	-0.04	1.0		
fare	0.34	0.15	0.12	-0.24	-0.16	-0.45	1.0	
embarked	-0.05	0.15	0.02	-0.06	0.07	-0.02	0.12	1.0

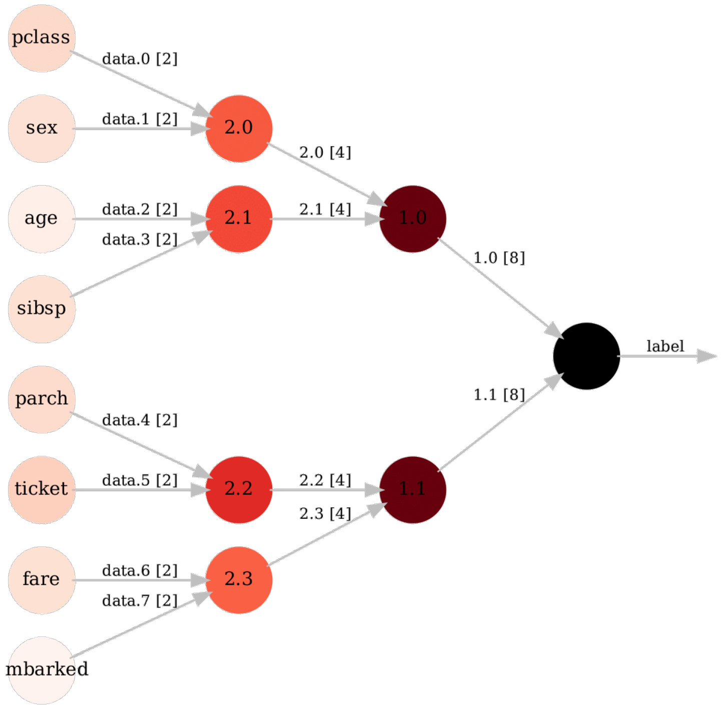
Two-sites σ_z correlation



Tests on synthetic datasets:

Explainability

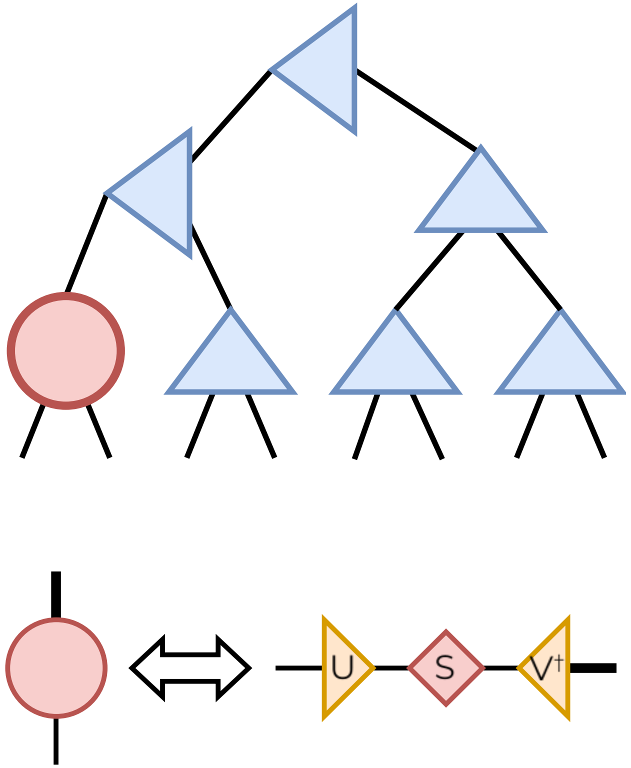


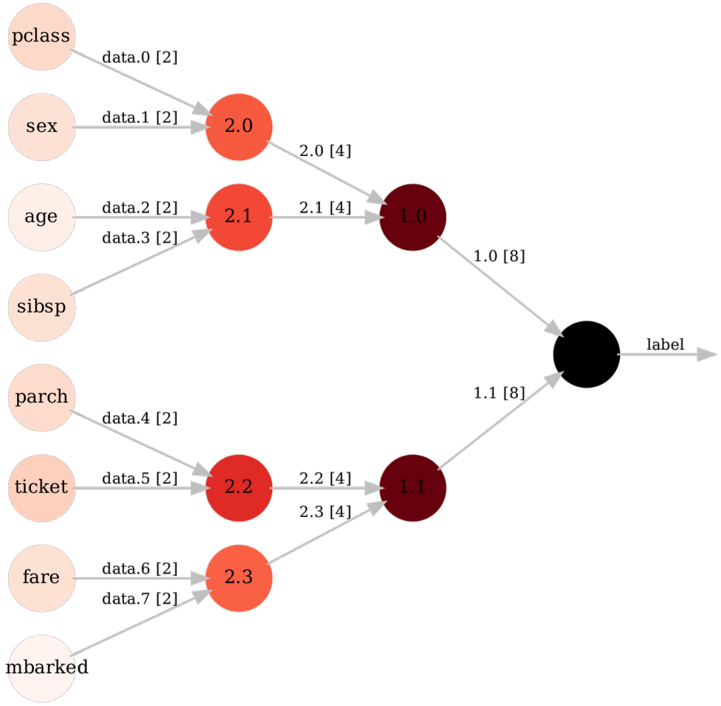


Entanglement entropy

Tests on synthetic datasets:

Explainability

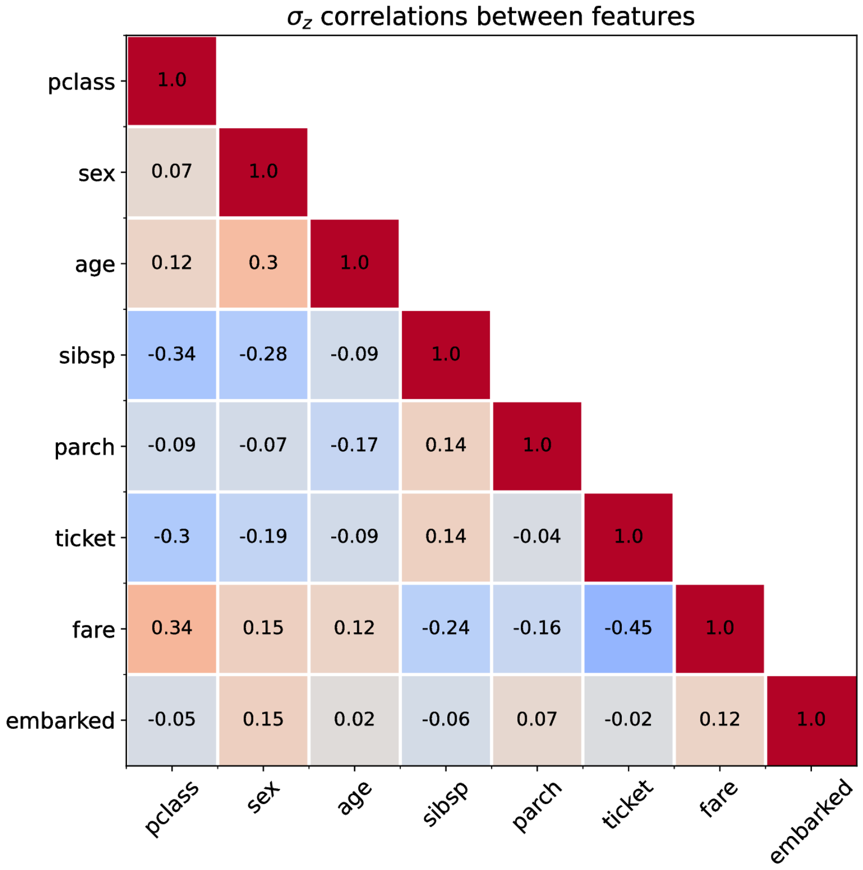




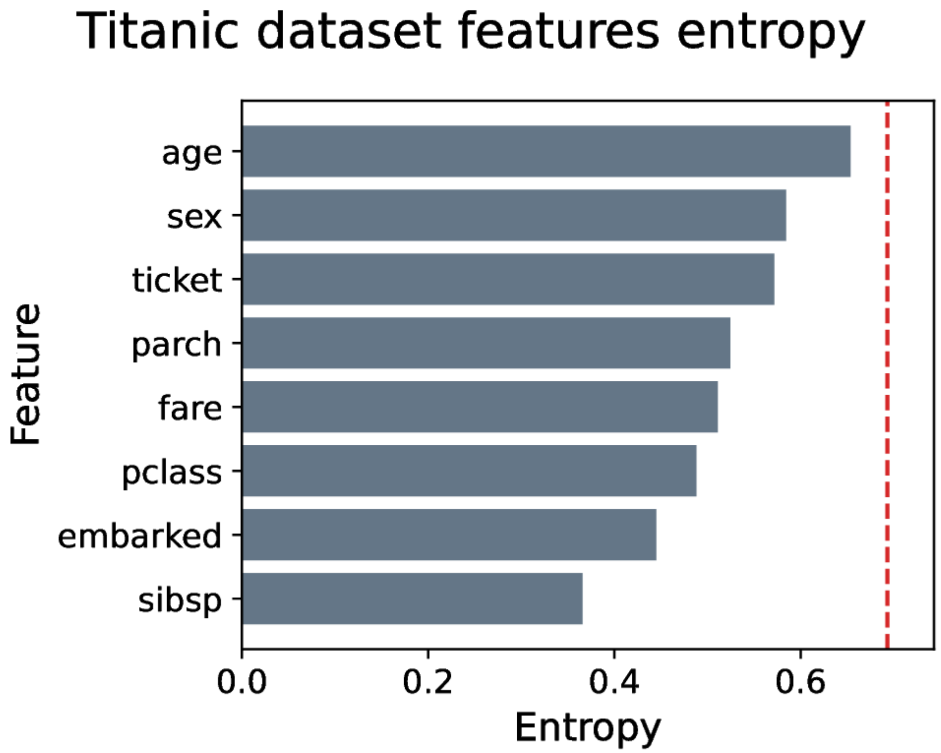
Entanglement entropy

Tests on synthetic datasets:

Explainability



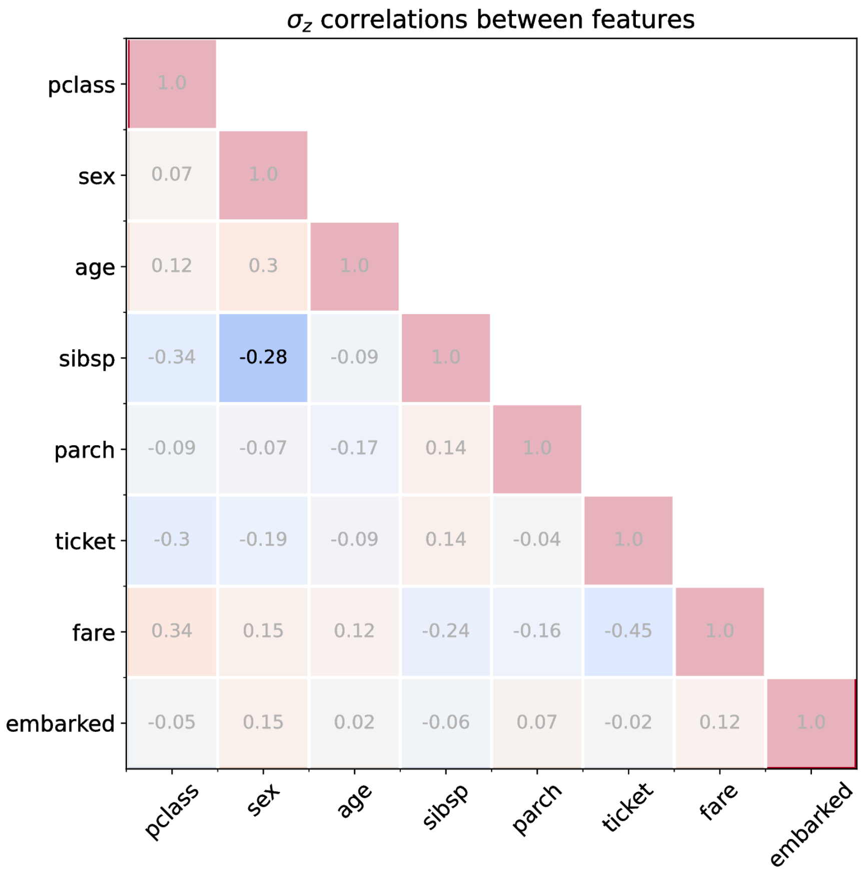
Two-sites σ_z correlation



Entanglement entropy

Tests on synthetic datasets:

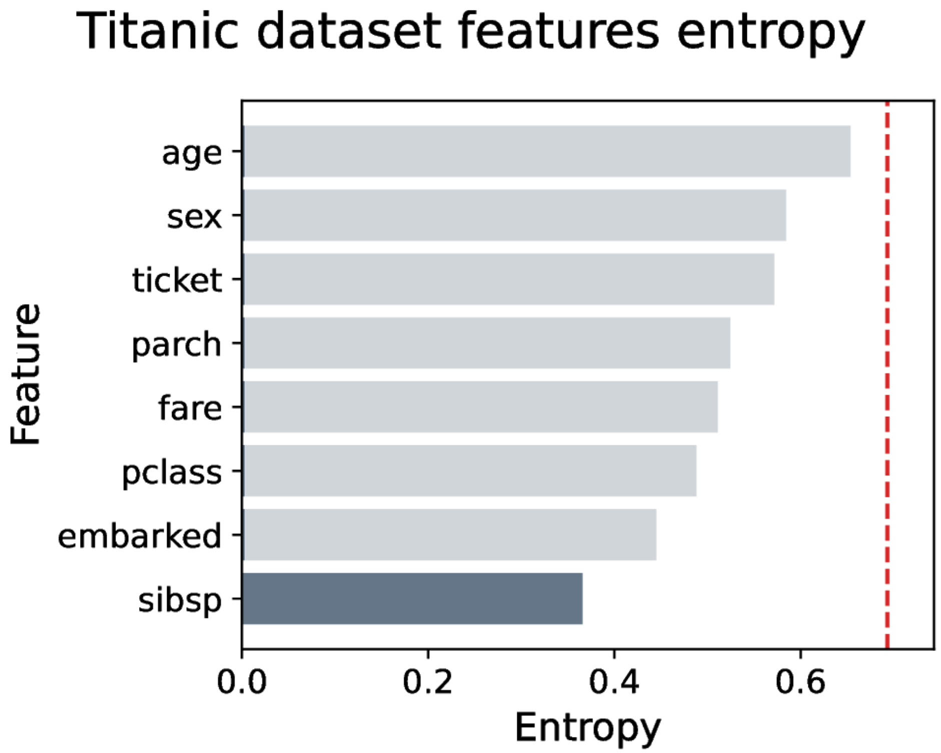
Explainability



σ_z correlations between features

	pclass	sex	age	sibsp	parch	ticket	fare	embarked
pclass	1.0							
sex	0.07	1.0						
age	0.12	0.3	1.0					
sibsp	-0.34	-0.28	-0.09	1.0				
parch	-0.09	-0.07	-0.17	0.14	1.0			
ticket	-0.3	-0.19	-0.09	0.14	-0.04	1.0		
fare	0.34	0.15	0.12	-0.24	-0.16	-0.45	1.0	
embarked	-0.05	0.15	0.02	-0.06	0.07	-0.02	0.12	1.0

Two-sites σ_z correlation

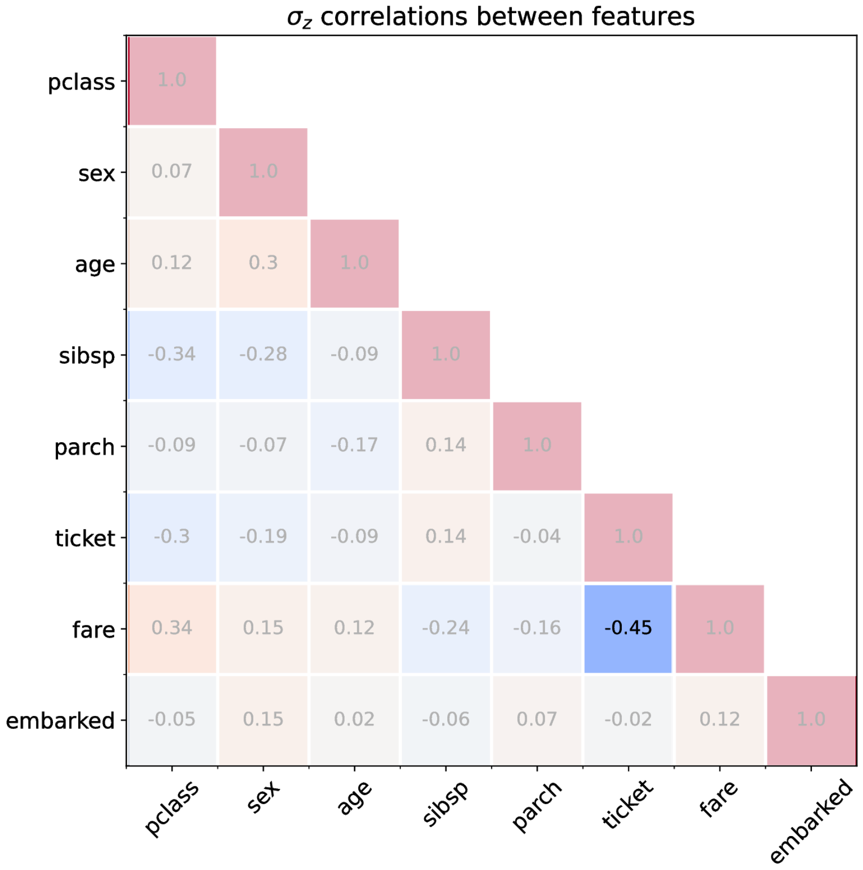


Titanic dataset features entropy

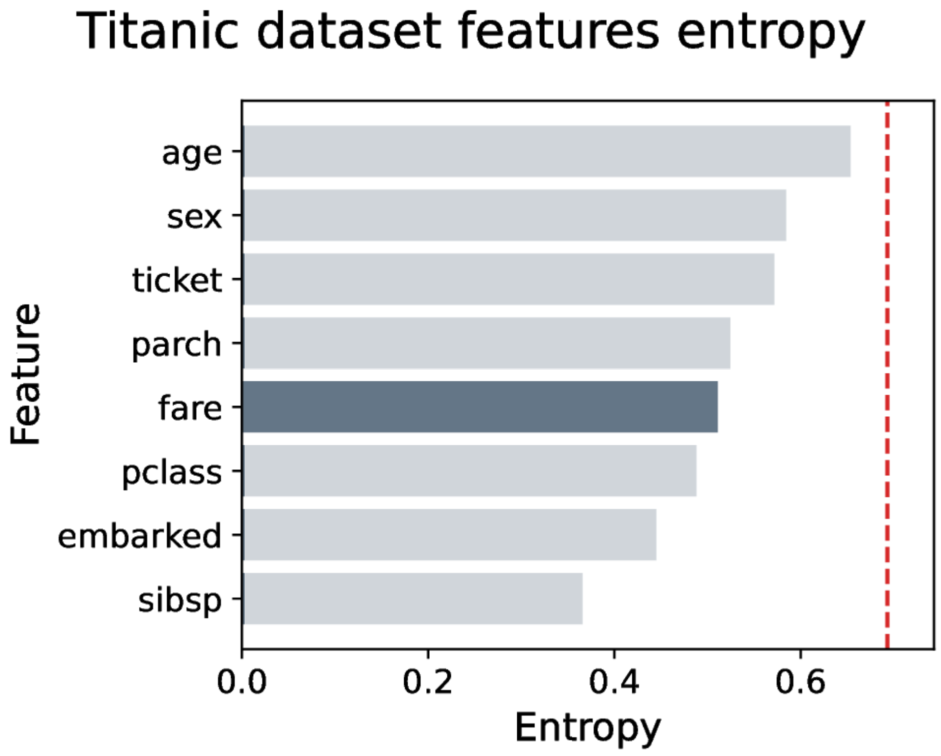
Entanglement entropy

Tests on synthetic datasets:

Explainability



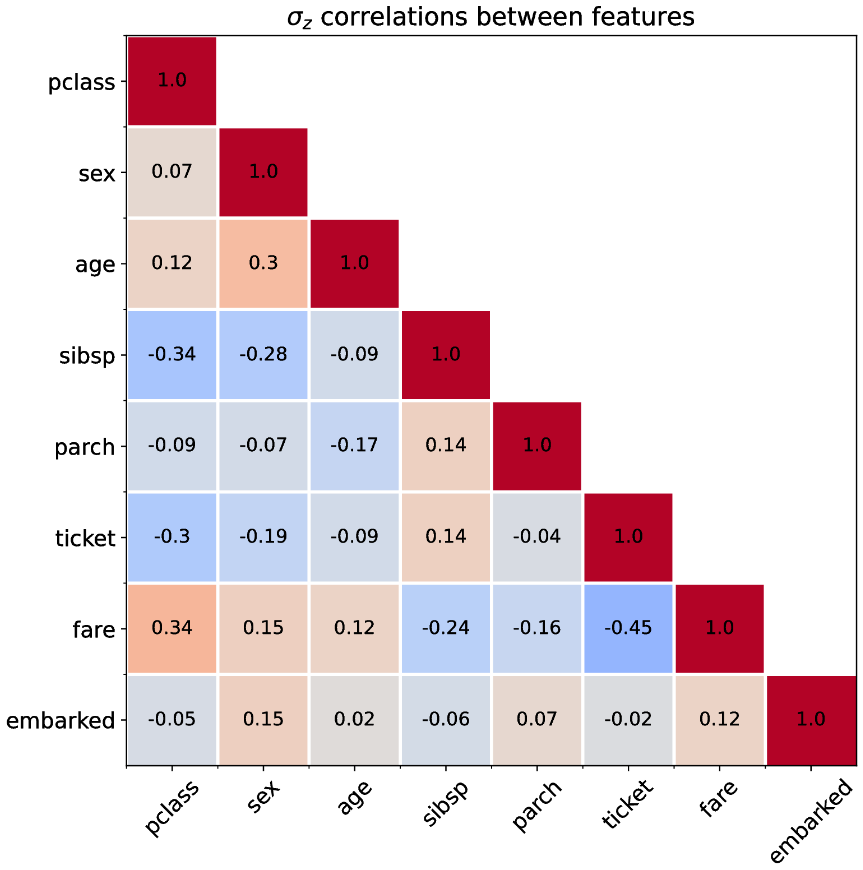
Two-sites σ_z correlation



Entanglement entropy

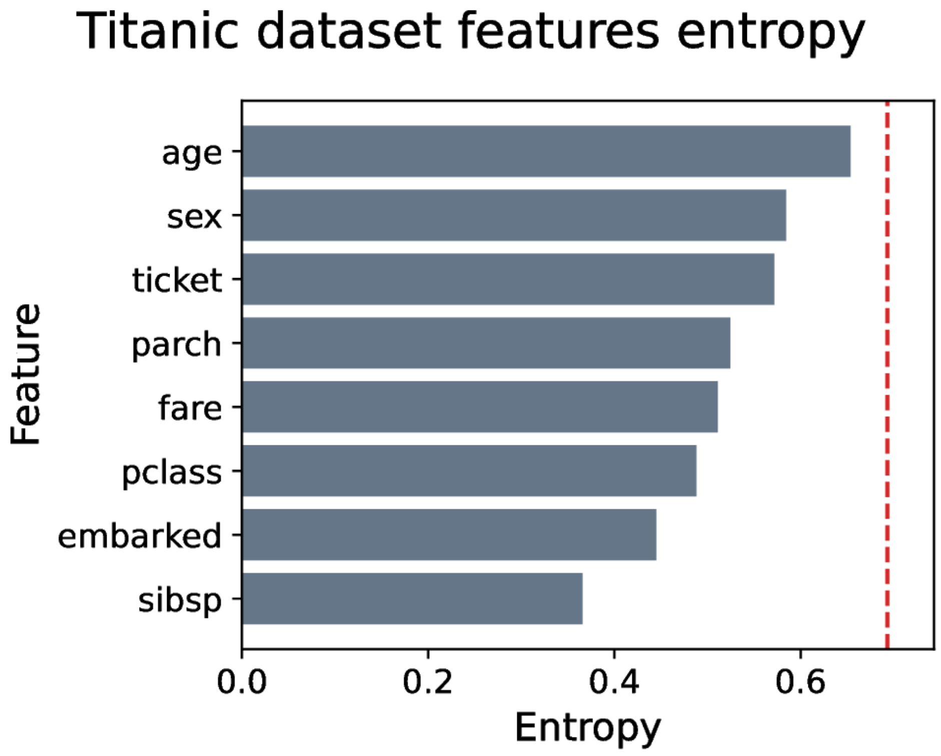
Tests on synthetic datasets:

Explainability



	pclass	sex	age	sibsp	parch	ticket	fare	embarked
pclass	1.0							
sex	0.07	1.0						
age	0.12	0.3	1.0					
sibsp	-0.34	-0.28	-0.09	1.0				
parch	-0.09	-0.07	-0.17	0.14	1.0			
ticket	-0.3	-0.19	-0.09	0.14	-0.04	1.0		
fare	0.34	0.15	0.12	-0.24	-0.16	-0.45	1.0	
embarked	-0.05	0.15	0.02	-0.06	0.07	-0.02	0.12	1.0

Two-sites σ_z correlation



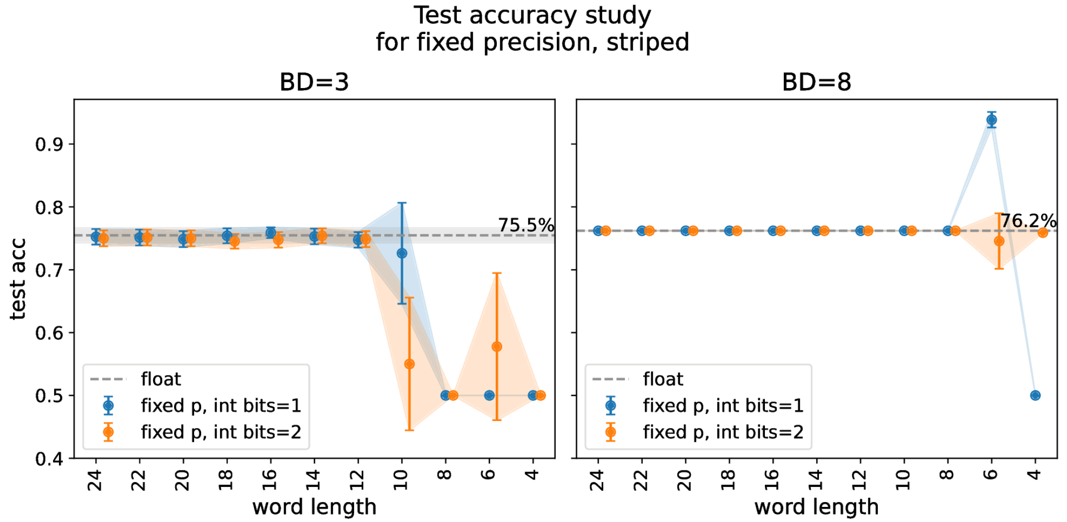
Entanglement entropy

Tests on synthetic datasets:

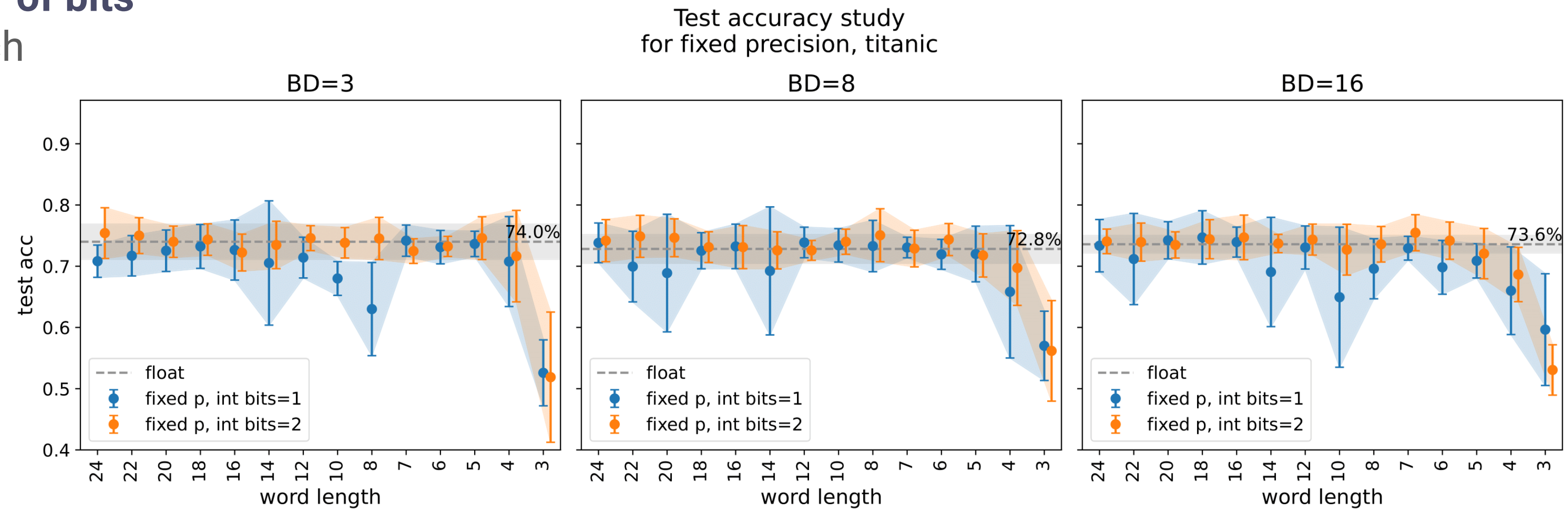
Towards hardware

- Different hardware implement **different logic**
- **Simulate model behaviour** in different numerical representations
- On FPGA, **APFP** representation

- Studied model performance **dependence on the number of bits** used, by means of QPyTorch
- Further compression



Test accuracy study
for fixed precision, striped



Test accuracy study
for fixed precision, titanic

Future perspectives

✓ **Hardware optimisation**

Optimise these algorithms exploiting TN properties for faster execution on specialized hardware.

✓ **Online jet tagging**

Deploy in the online selection algorithm a b-tagger.

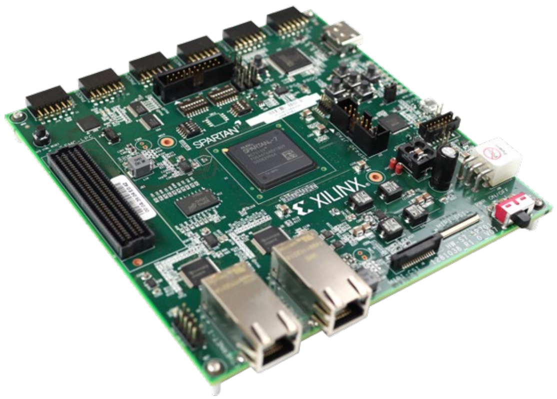
✓ **TTN based emulation and ML**

Implement the TTN ansatz both in the Quantum MATCHA and CHAI TEA applications to improve performance and accuracy of quantum computer emulations.

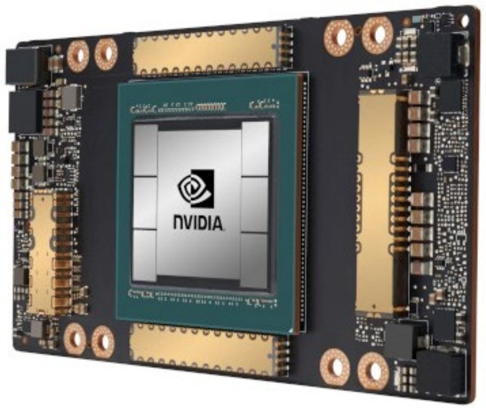
✓ **Extension to other ansatzes**

TTN may be not powerful enough for some specific tasks. Other ansatzes like MERA are worth exploring.

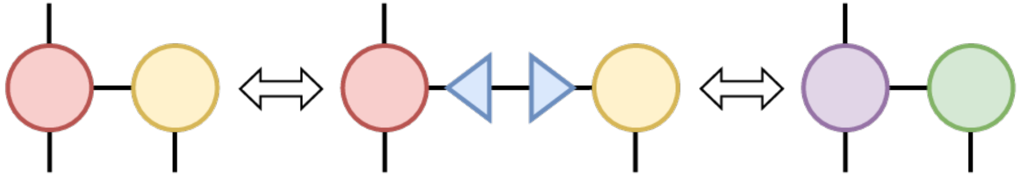
Hardware optimisation



FPGA



GPU



Gauge freedom

What

- Exploit the characteristics of different hardware for faster execution of computations
- Parallelism, different numerical representations (APFP, TF32)

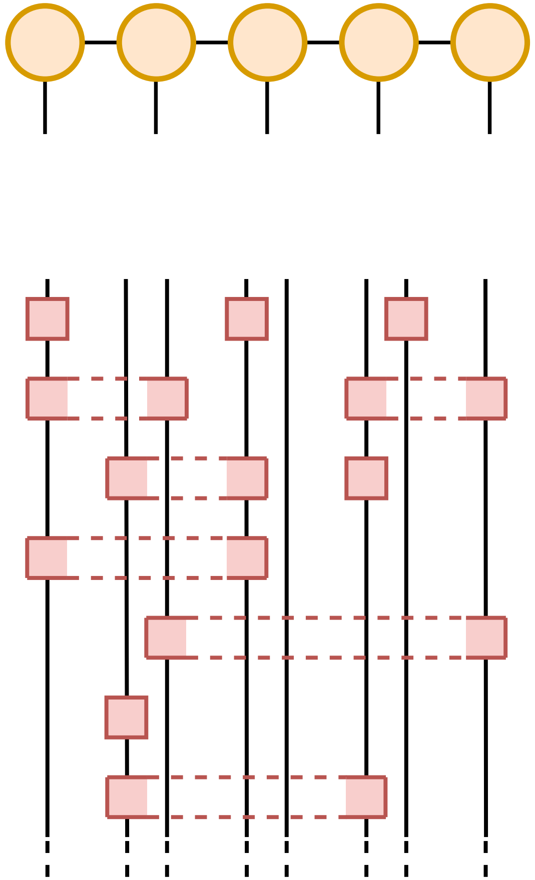
How

- Compressibility of TNs
- Gauge freedom

Why

- Enable fast, low-resource simulations, accelerating research into quantum algorithms and making it widely feasible.

TTN based emulation



MPS based emulation

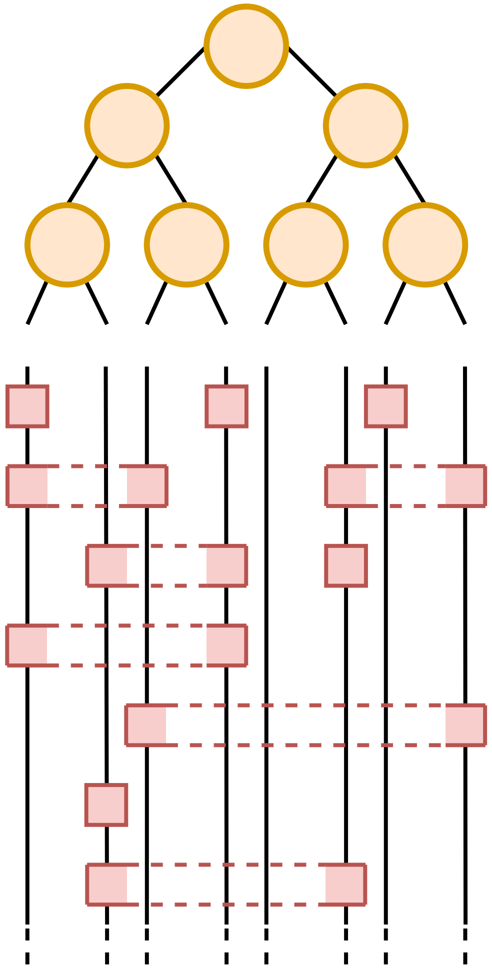
Pros

- **Well developed** technology, integrated in popular emulation libraries (Quantum MATCHA TEA, qiskit)
- **Straightforward** application of one and two qubits gates

Cons

- Supports only **linear topology** circuits
- Difficult representation of **long distance** interactions

TTN based emulation



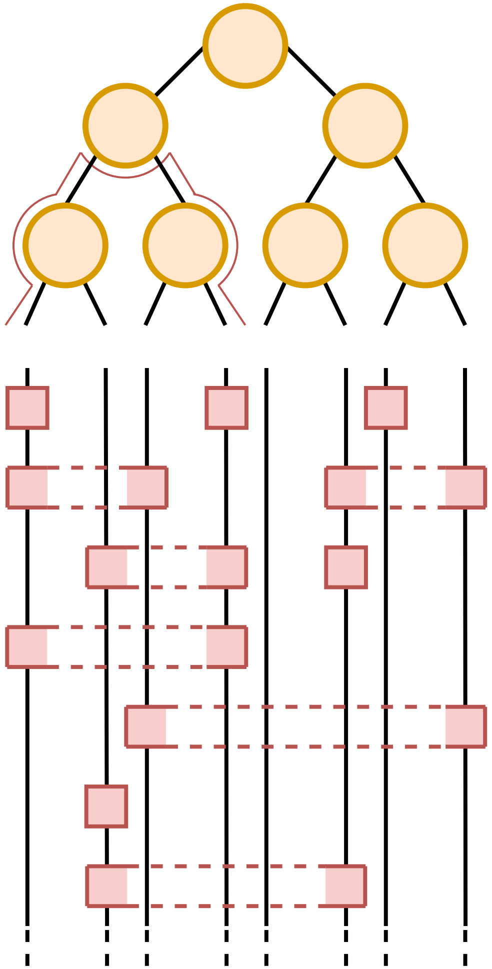
TTN based emulation

Pros

- Can handle **longer distance** interactions
- **Non-linear** circuits

P. Seitz, I. Medina, E. Cruz, Q. Huang, and C. B. Mendl, *Simulating quantum circuits using tree tensor networks,* Quantum 7, 964 (2023).

TTN based emulation



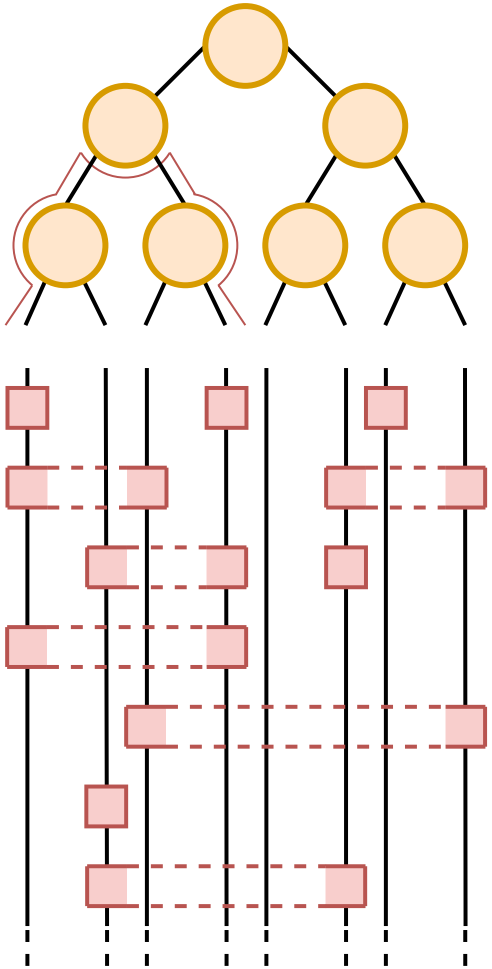
TTN based emulation

Pros

- Can handle **longer distance** interactions
- **Non-linear** circuits

P. Seitz, I. Medina, E. Cruz, Q. Huang, and C. B. Mendl, *Simulating quantum circuits using tree tensor networks*, Quantum 7, 964 (2023).

TTN based emulation



TTN based emulation

Pros

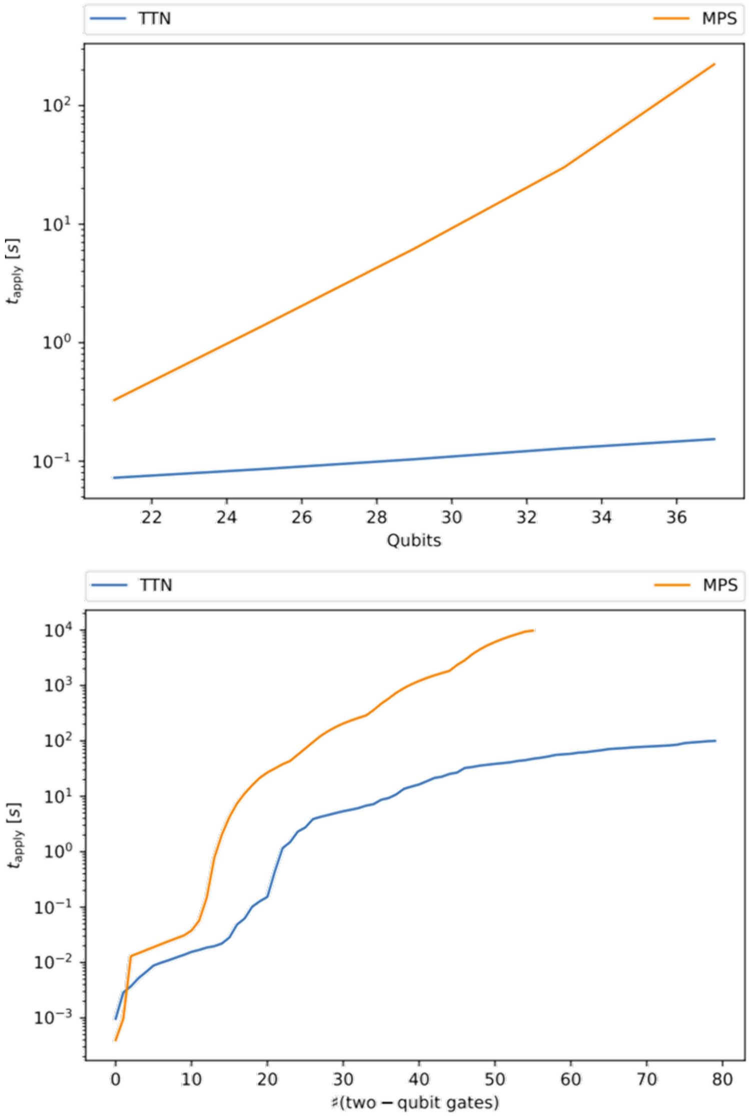
- Can handle **longer distance** interactions
- **Non-linear** circuits

Cons

- **More complex** operations
- Still difficulty for circuits with **high connectivity**

P. Seitz, I. Medina, E. Cruz, Q. Huang, and C. B. Mendl, *Simulating quantum circuits using tree tensor networks,* Quantum 7, 964 (2023).

TTN based emulation



TTN based emulation

Pros

- Can handle **longer distance** interactions
- **Non-linear** circuits

Cons

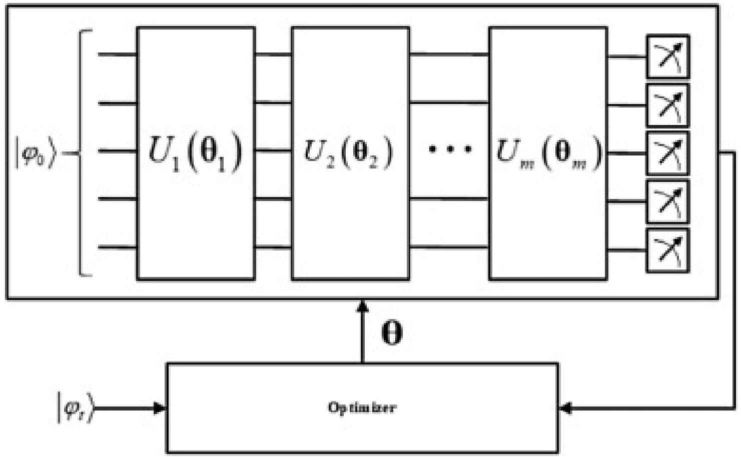
- **More complex** operations
- Still difficulty for circuits with **high connectivity**

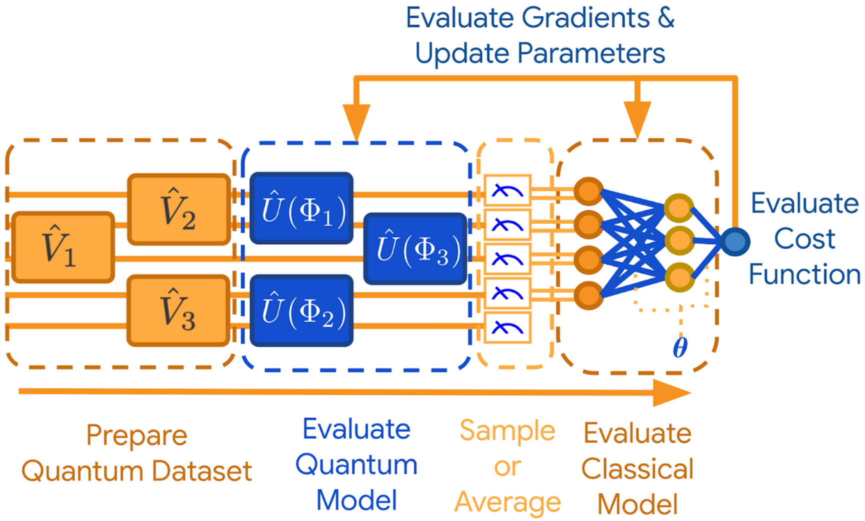
Tests

- Computation **time** and **resource** usage
- Final state **accuracy** for common quantum algorithms

Figures from: P. Seitz, I. Medina, E. Cruz, Q. Huang, and C. B. Mendl, *Simulating quantum circuits using tree tensor networks,* Quantum 7, 964 (2023).

TTN based ML





Quantum-Enhanced ML

What

- Quantum computer as part of a **hybrid quantum-classical** model
- Classical part optimizes **classical parameters** controlling the quantum part
- ➢ Expand CHAI TEA to use MATCHA TEA for hybrid quantum-classical ML. Expose as PyTorch layers

Why

- Offer a complete framework
- Competitive performances w.r.t. Tensorflow Quantum, possibly improving on TorchQauntum

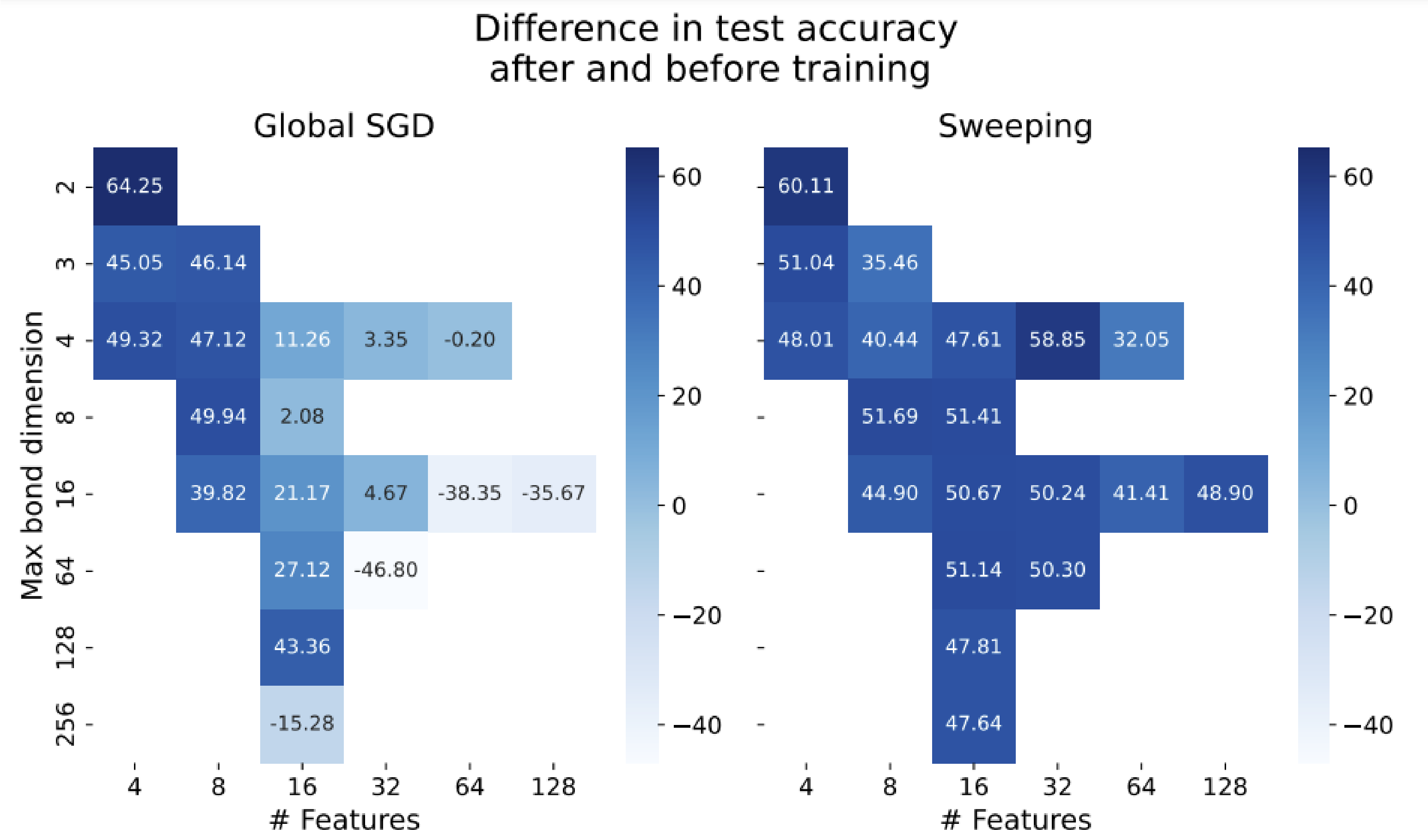
Top figure from: D. Peral-García, J. Cruz-Benito, F. J. García-Peñalvo, *Systematic literature review: Quantum machine learning and its applications*, Computer Science Review, Volume 51, 2024, 100619, ISSN 1574-0137.
Bottom figure from: Tensorflow Quantum.

Thank you for your attention

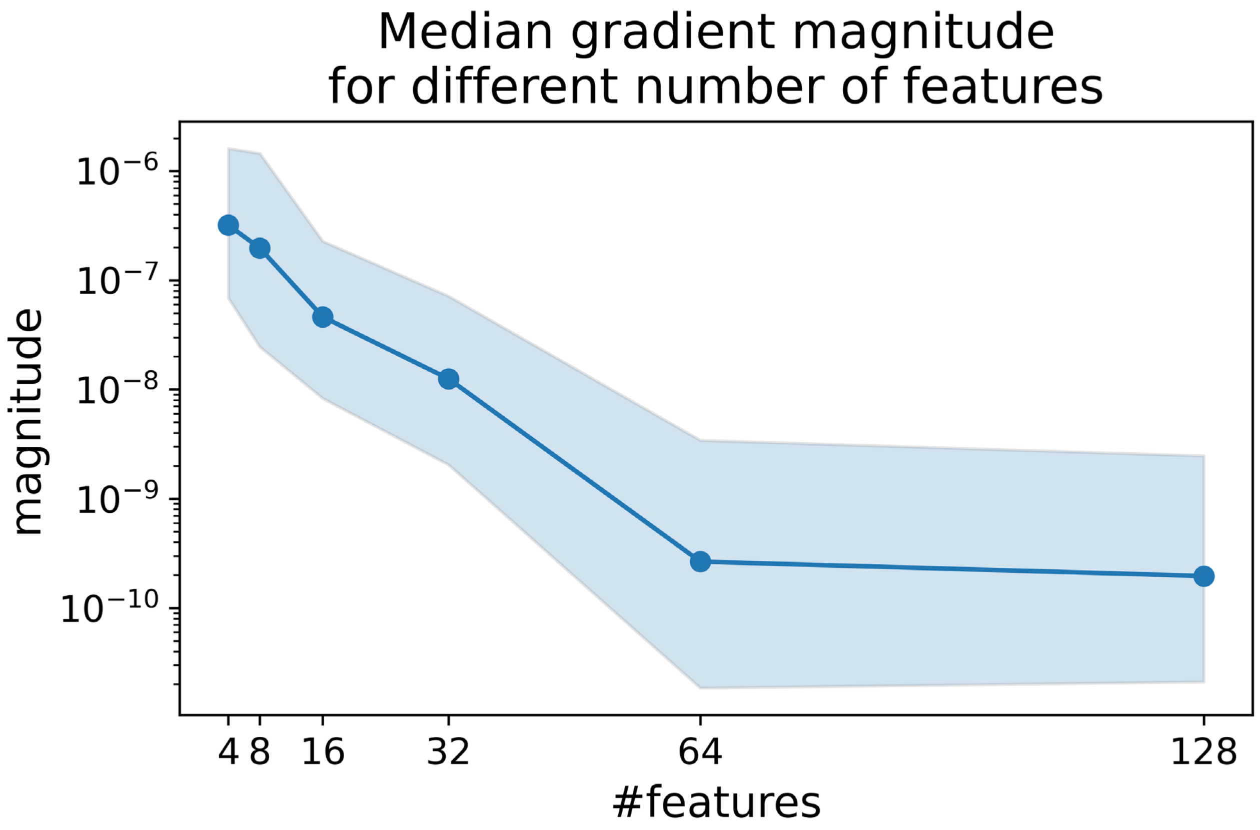
BACKUP

Software studies

Sweeping analysis



Barren plateaus

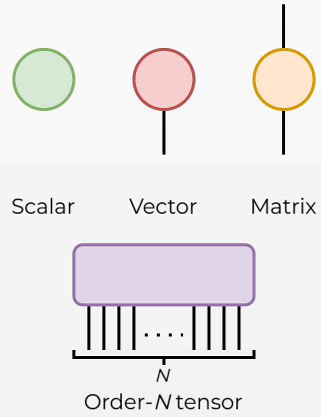


Tensors

Tensors

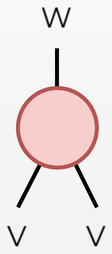
Multi-dimensional arrays

- Generalisation of the idea of vectors and matrices.
- Connected to software representation.



Multi-linear maps

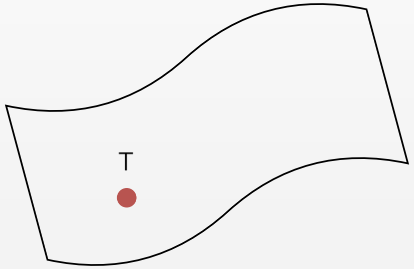
- Extending the idea of linear maps (matrices).
- Useful to interpret some TNs algorithms.



Elements of tensor product space

- Define properties such as order and shape.
- Connected to quantum many-body systems:

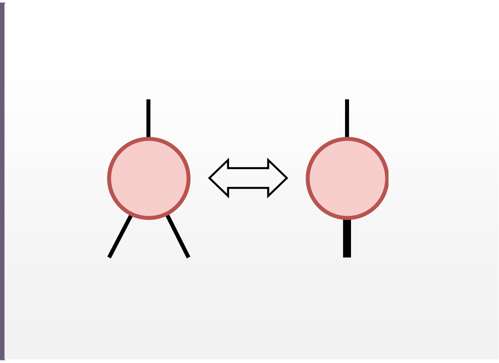
$$|\Psi\rangle = \sum_{\{i\}^N} \Psi_{\{i\}^N} |i_1\rangle \otimes \cdots \otimes |i_N\rangle$$



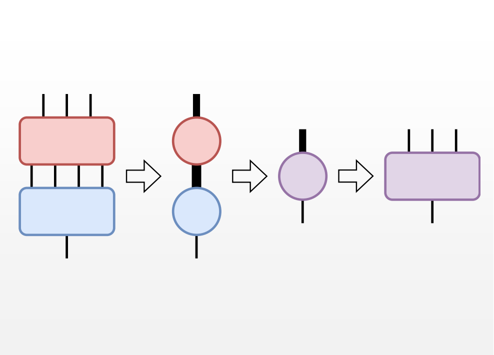
Tensor operations

Manipulations

Index fusing & splitting

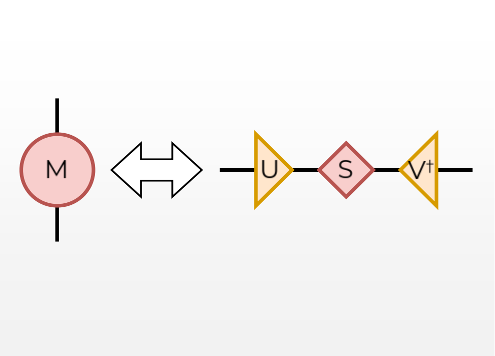


Contraction

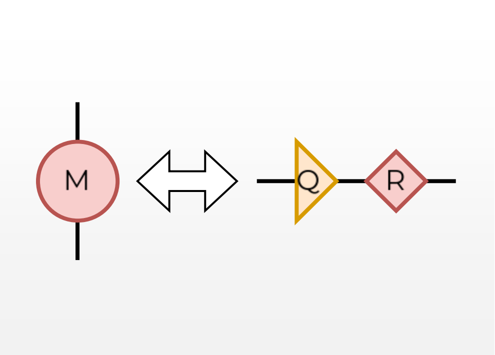


Decompositions

SVD / eigenvalue

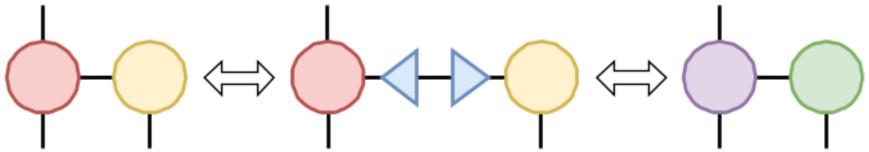


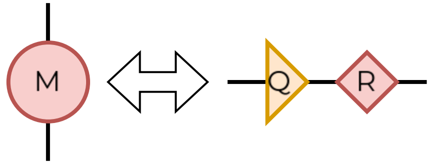
QR

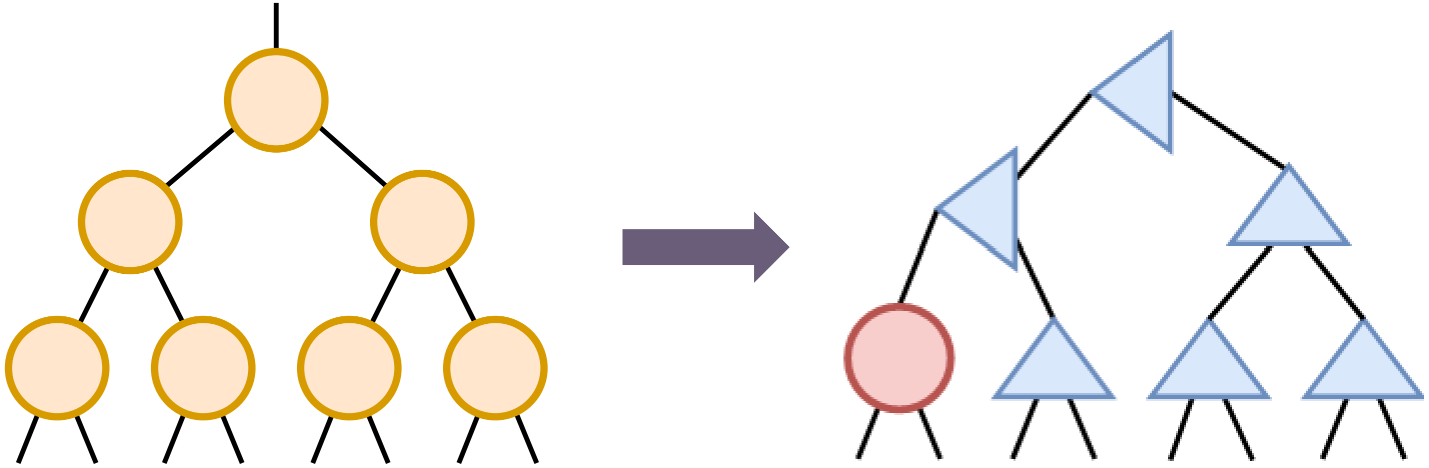


Isometrisation

Isometrisation



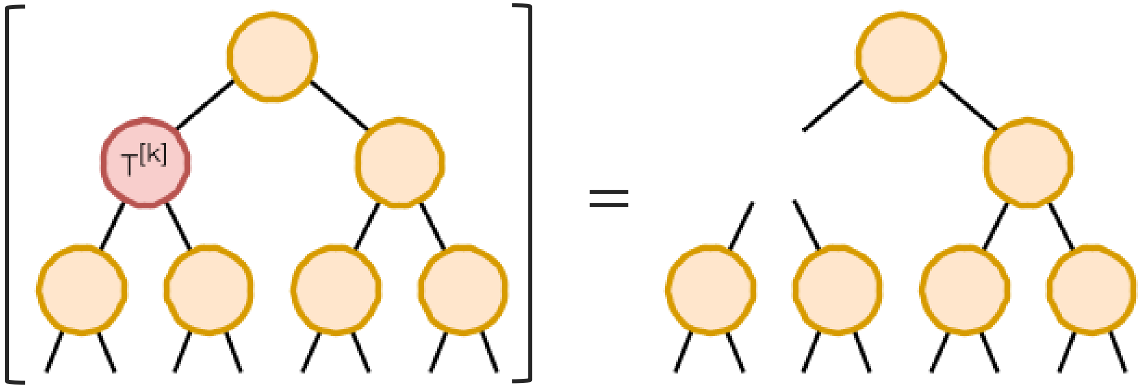




Differentiation

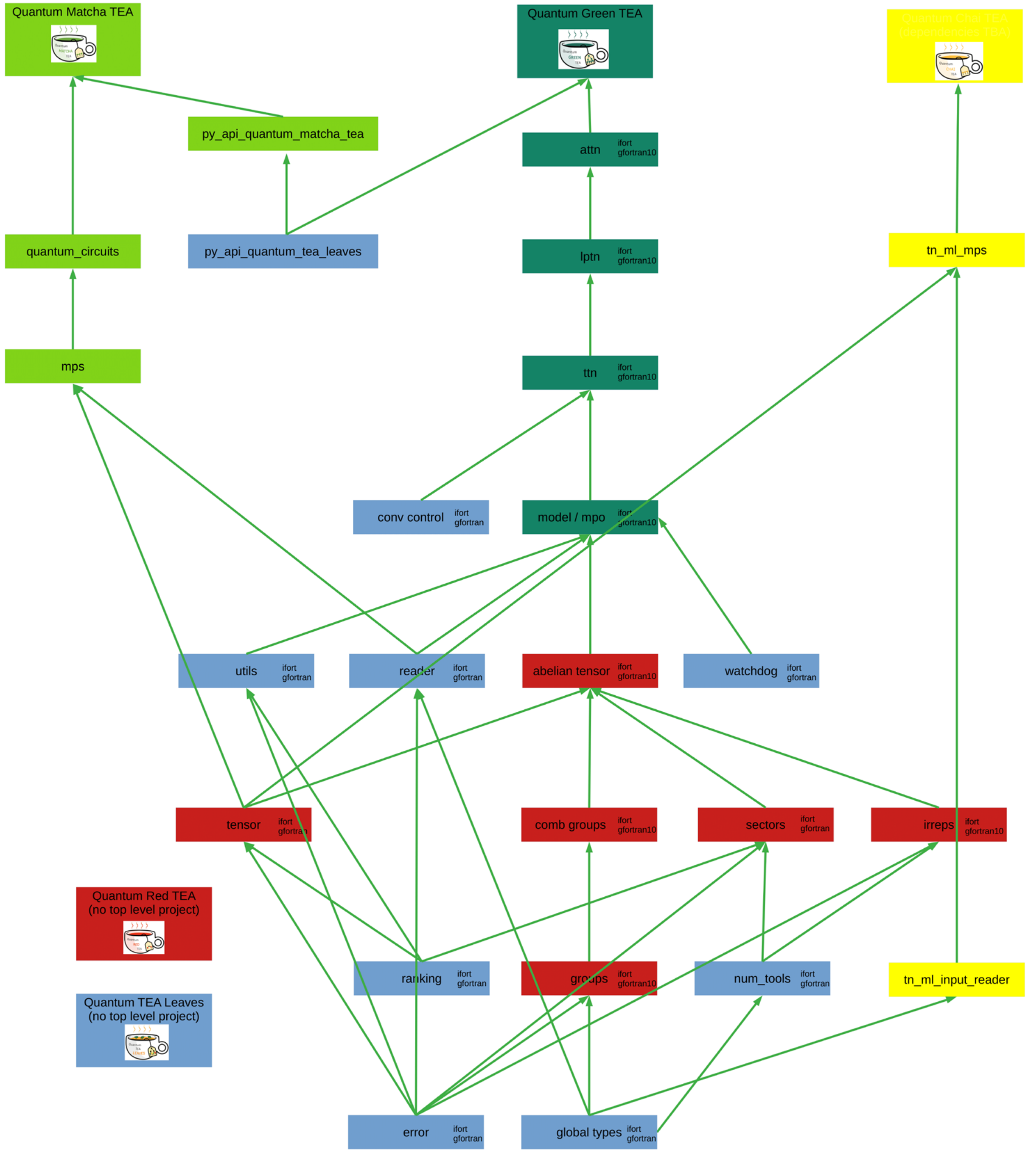
Derivative

$$\frac{\partial}{\partial T^{[k]}}$$

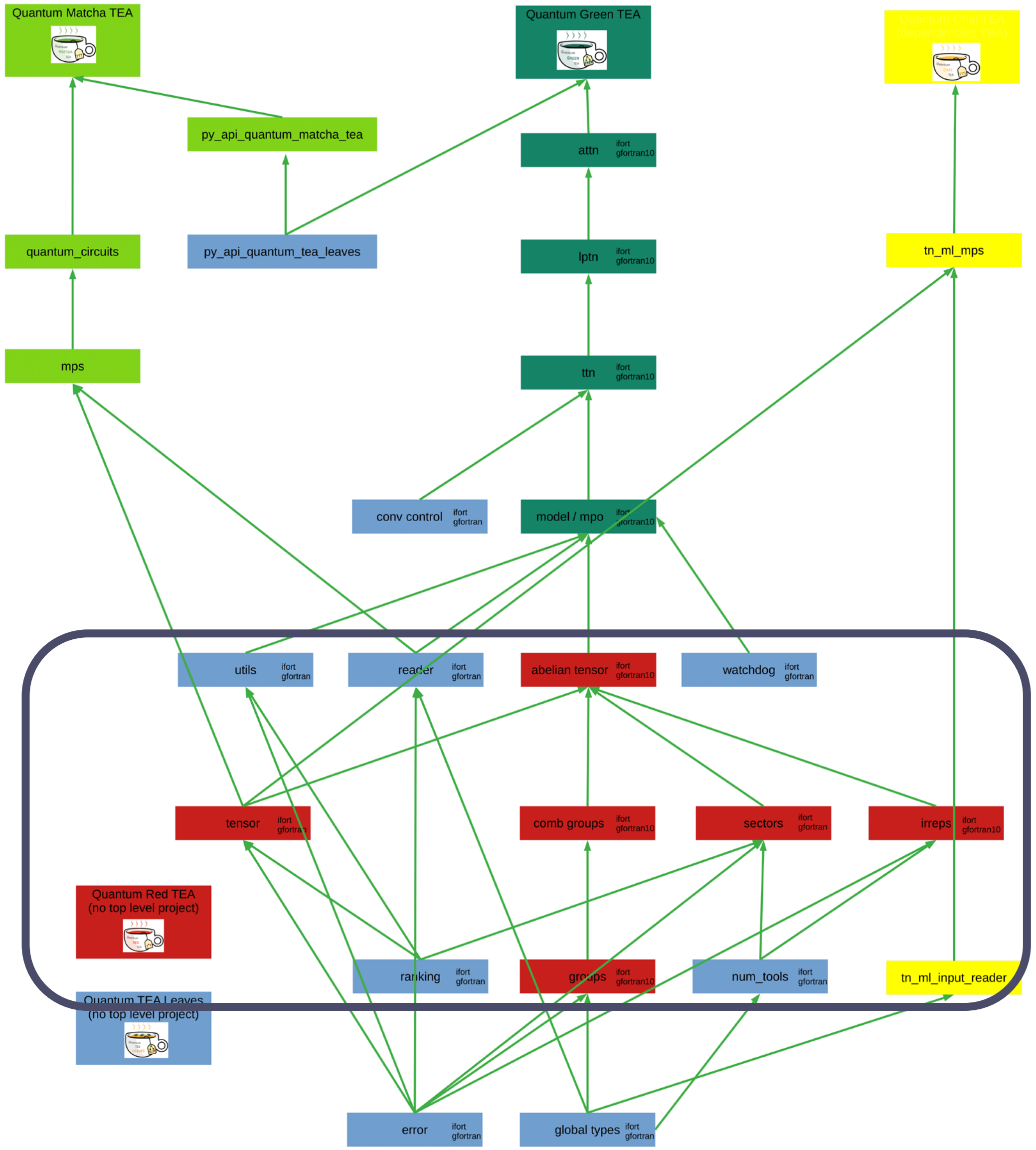


Quantum TEA

Quantum TEA library



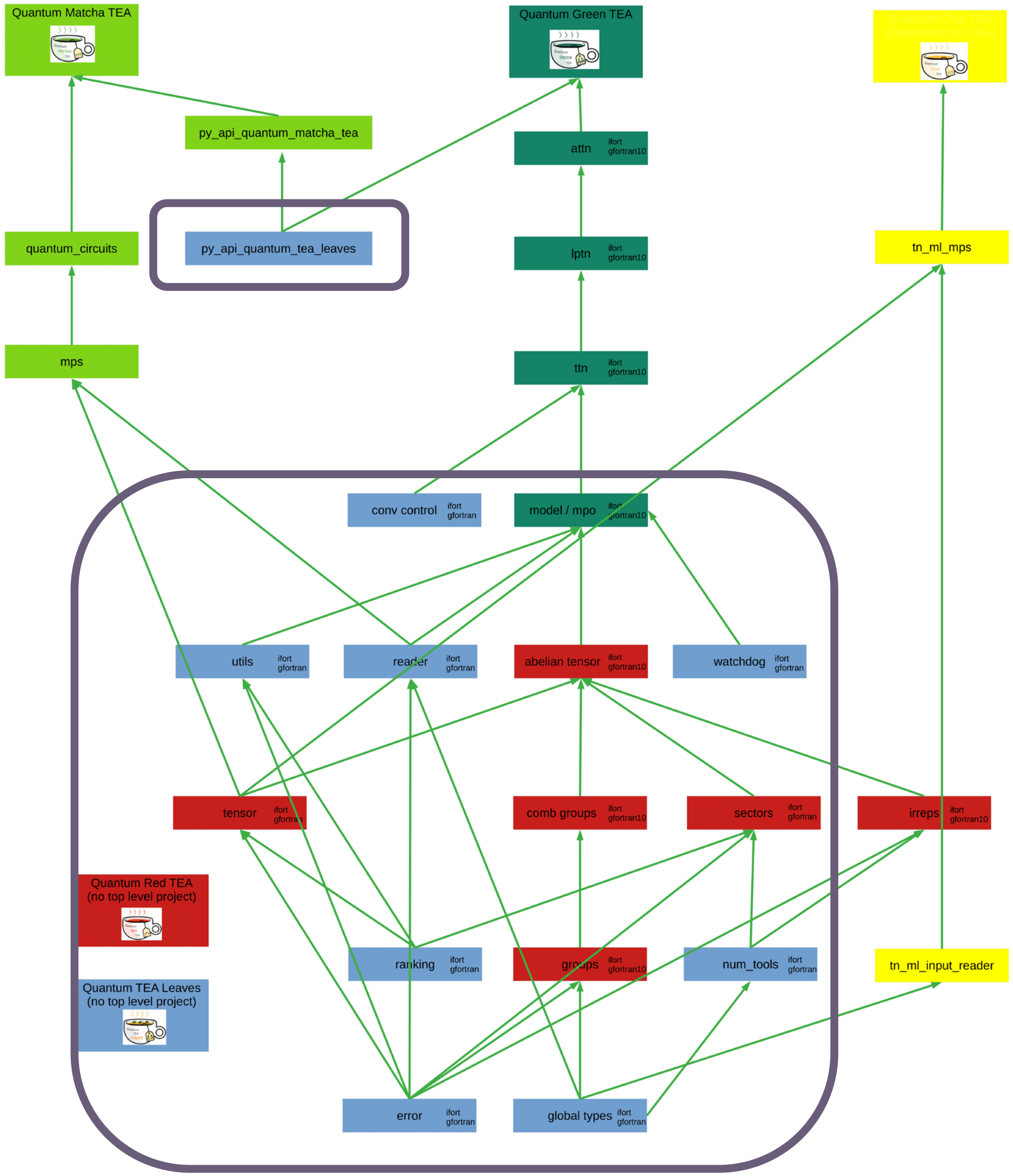
Quantum TEA library



Quantum RED TEA

Provide the interfaces to BLAS/LAPACK and CUDA for the higher-level applications.

Quantum TEA library



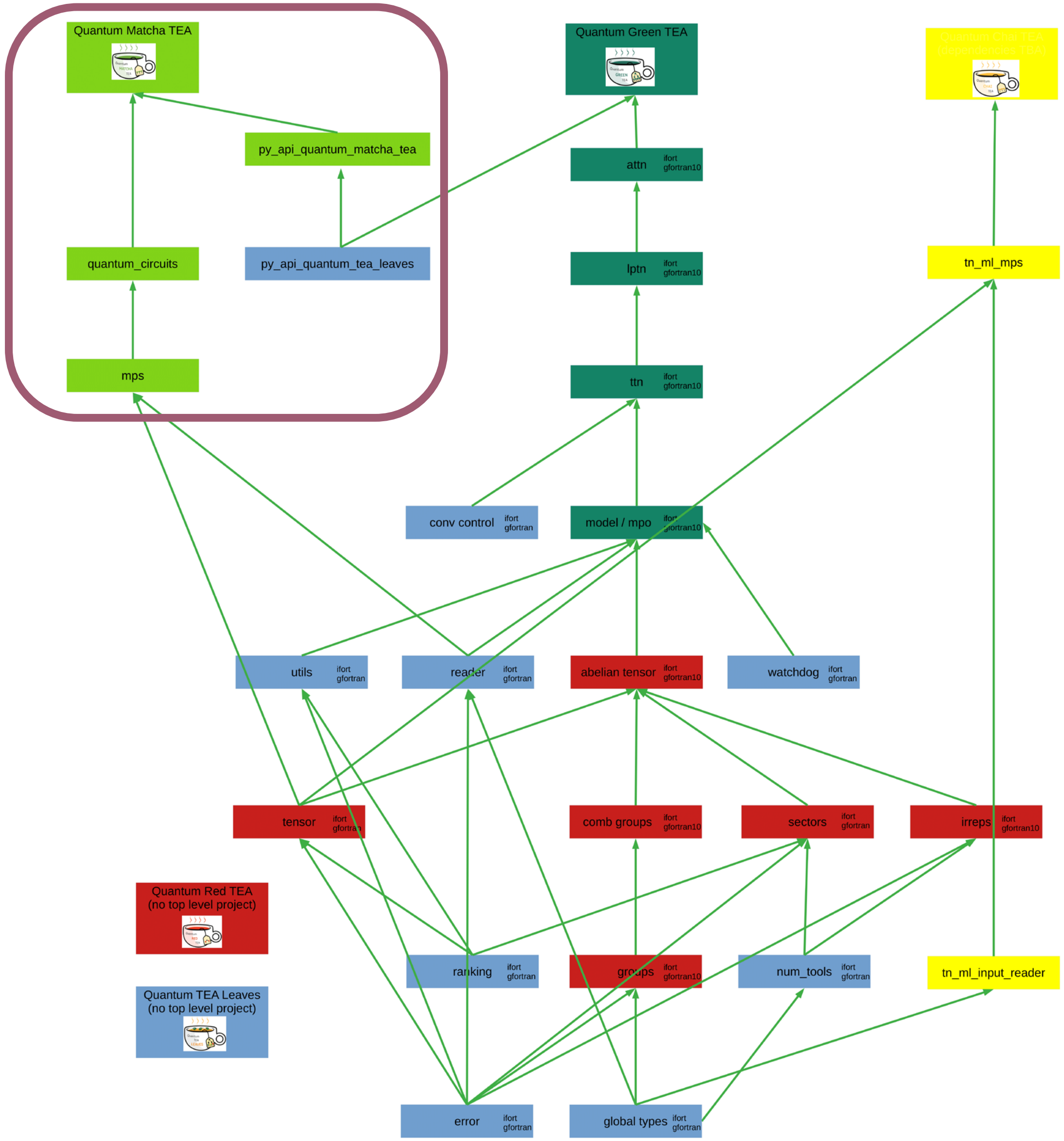
Quantum RED TEA

Provide the interfaces to BLAS/LAPACK and CUDA for the higher-level applications.

Quantum TEA LEAVES

Python solutions for common TN geometries, ground state search algorithms, time evolution via TDVP, Python-FORTRAN interfaces.

Quantum TEA library



Quantum RED TEA

Provide the interfaces to BLAS/LAPACK and CUDA for the higher-level applications.

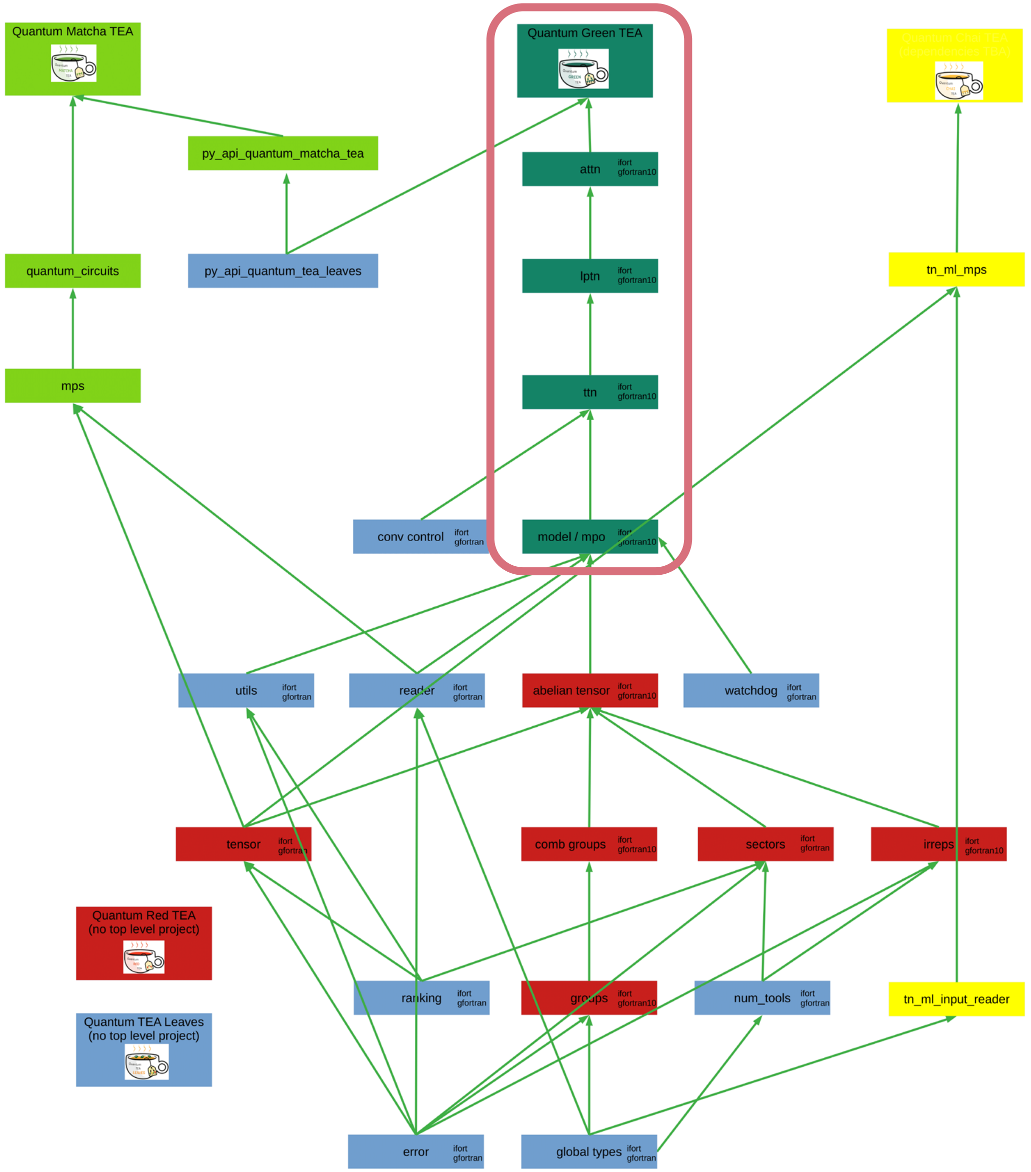
Quantum TEA LEAVES

Python solutions for common TN geometries, ground state search algorithms, time evolution via TDVP, Python-FORTRAN interfaces.

Quantum MATCHA TEA

Quantum computer emulator powered by **matrix product states**

Quantum TEA library



Quantum RED TEA

Provide the interfaces to BLAS/LAPACK and CUDA for the higher-level applications.

Quantum TEA LEAVES

Python solutions for common TN geometries, ground state search algorithms, time evolution via TDVP, Python-FORTRAN interfaces.

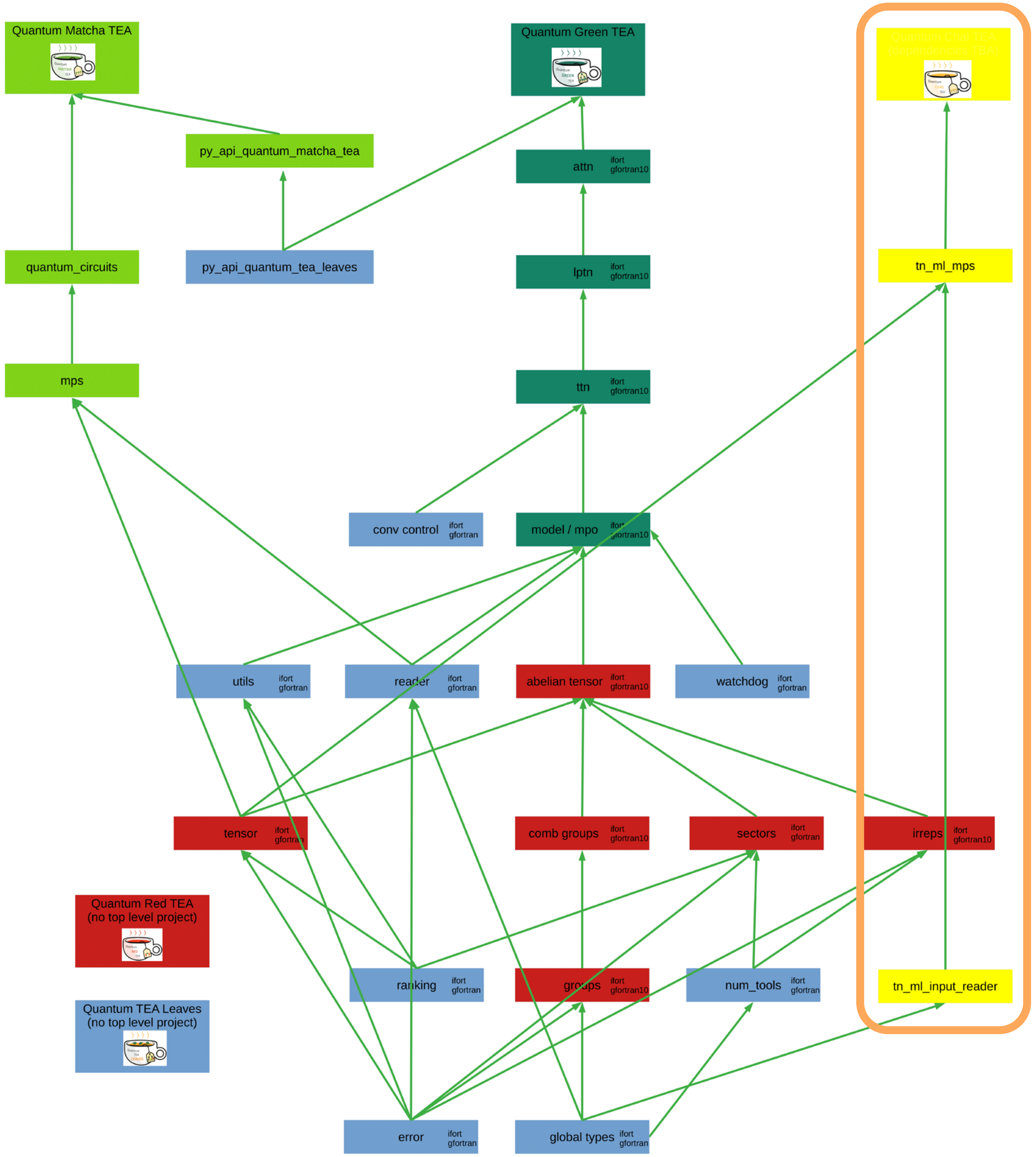
Quantum MATCHA TEA

Quantum computer emulator powered by **matrix product states**

Quantum GREEN TEA

Solves the static and time-dependent Schrödinger equation and Lindblad equation

Quantum TEA library



Quantum RED TEA

Provide the interfaces to BLAS/LAPACK and CUDA for the higher-level applications.

Quantum TEA LEAVES

Python solutions for common TN geometries, ground state search algorithms, time evolution via TDVP, Python-FORTRAN interfaces.

Quantum MATCHA TEA

Quantum computer emulator powered by **matrix product states**

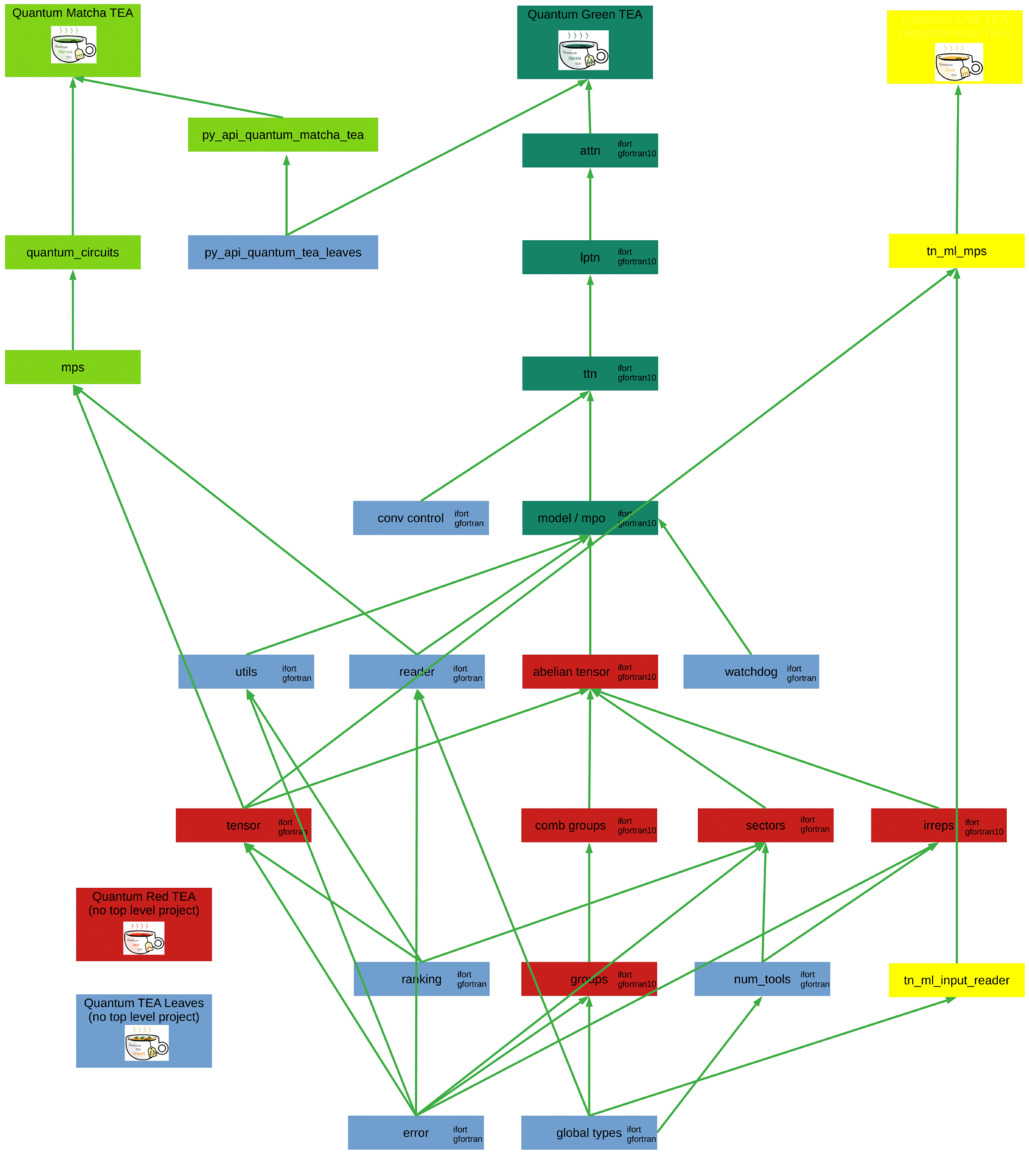
Quantum GREEN TEA

Solves the static and time-dependent Schrödinger equation and Lindblad equation

Quantum CHAI TEA

Contains the machine learning applications using TNs.

Quantum TEA library



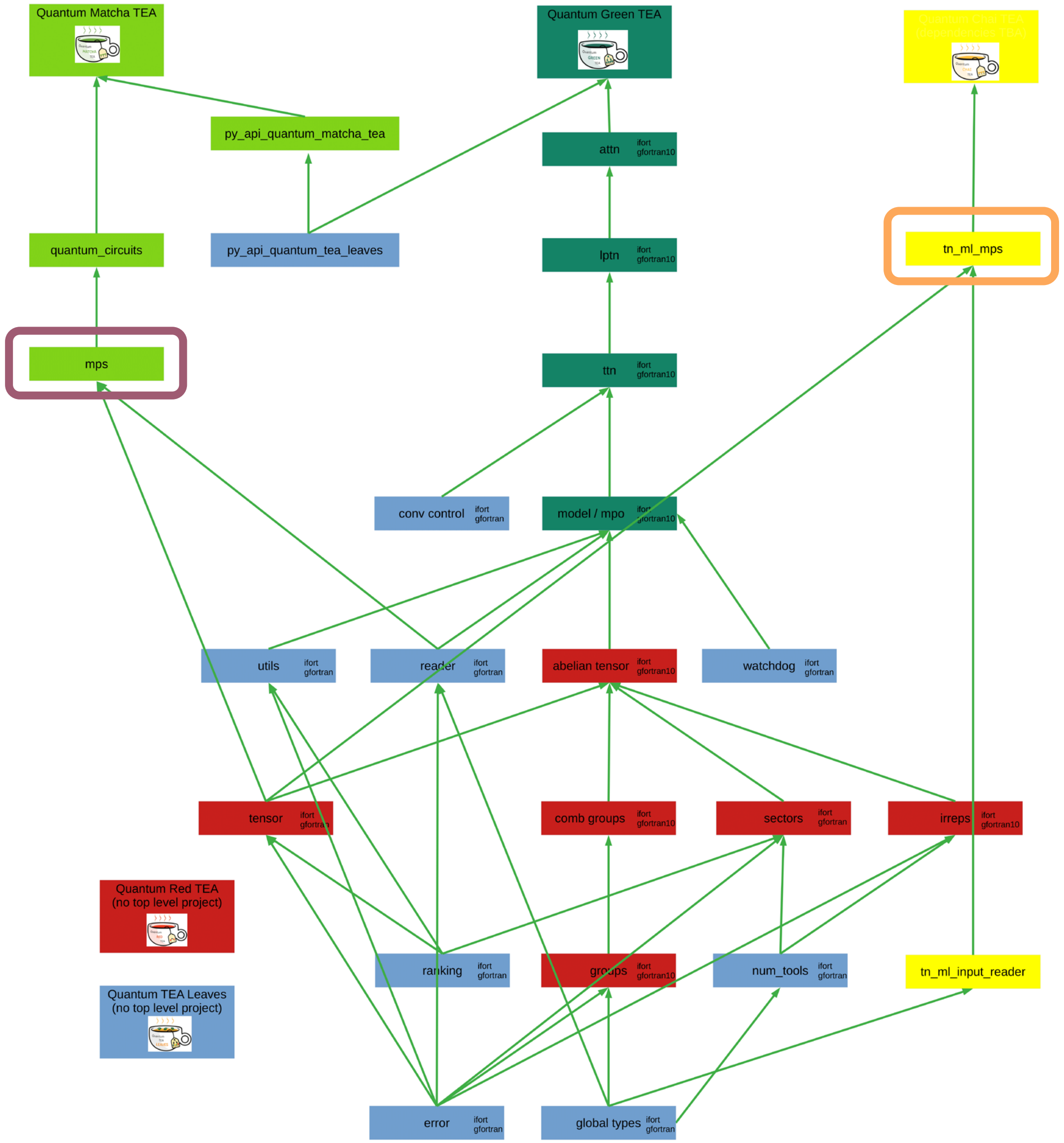
Quantum MATCHA TEA

Quantum computer emulator powered by **matrix product states**

Quantum CHAI TEA

Contains the machine learning applications using TNs.

Quantum TEA library



Quantum MATCHA TEA

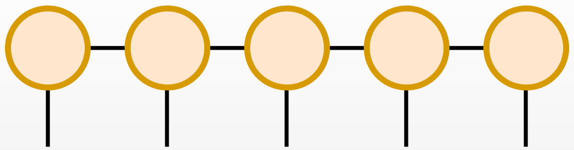
Quantum computer emulator powered by **matrix product states**

Quantum CHAI TEA

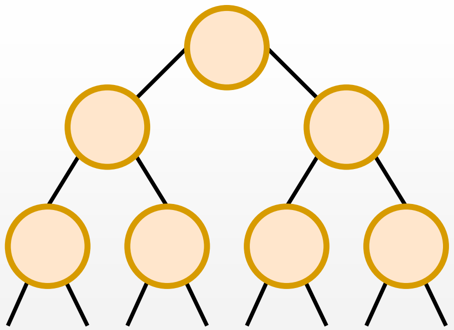
Contains the machine learning applications using TNs.

TN ansatzes

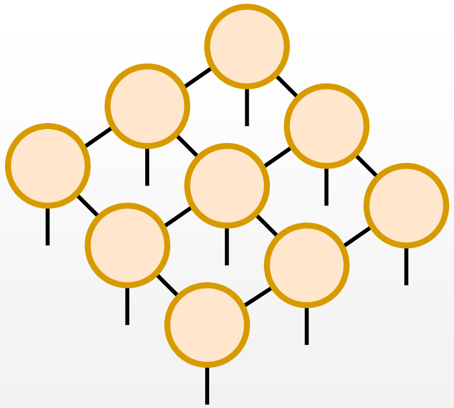
MPS



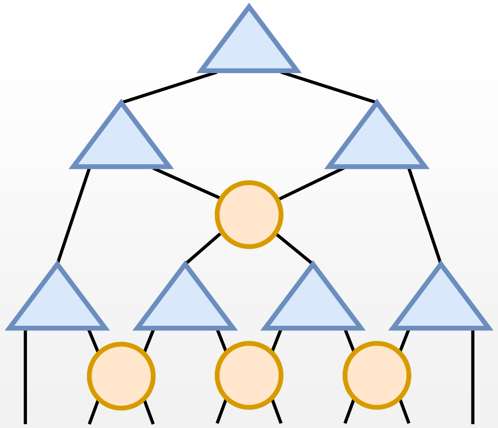
TTN



PEPS

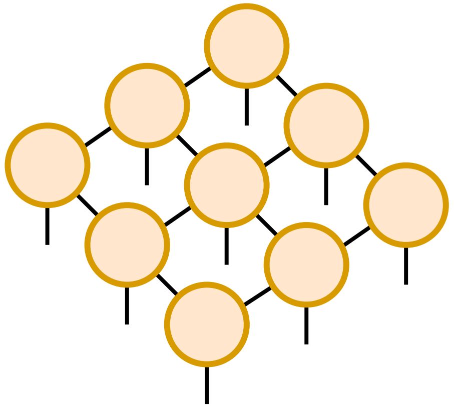


MERA

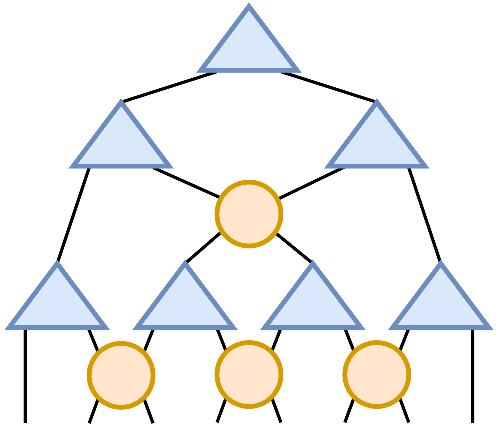


Extend to other ansatzes

PEPS



MERA



What

- Explore other, more complex ansatez like PEPS and MERA
- Integrate them in the Quantum TEA framework making the switch between different ansatzes effortless

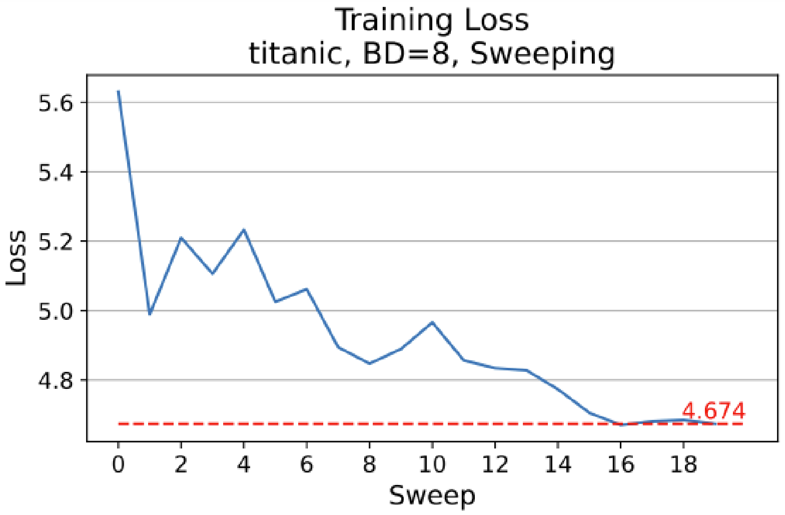
Why

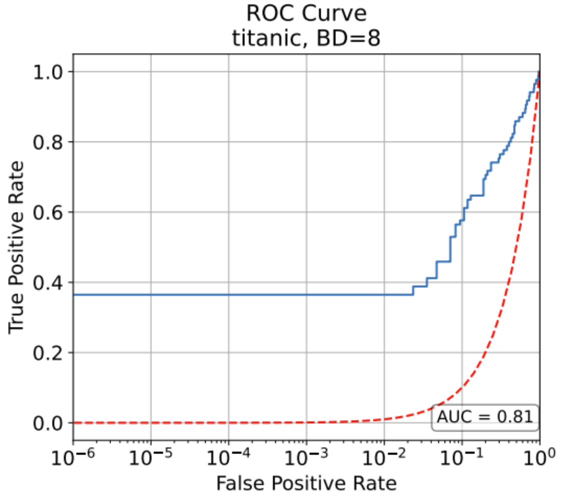
- Different geometries are guaranteed to capture higher amount of entropy and longer-range interactions
- Thought for non-linear systems → can stand higher connectivity

Results on synthetic datasets

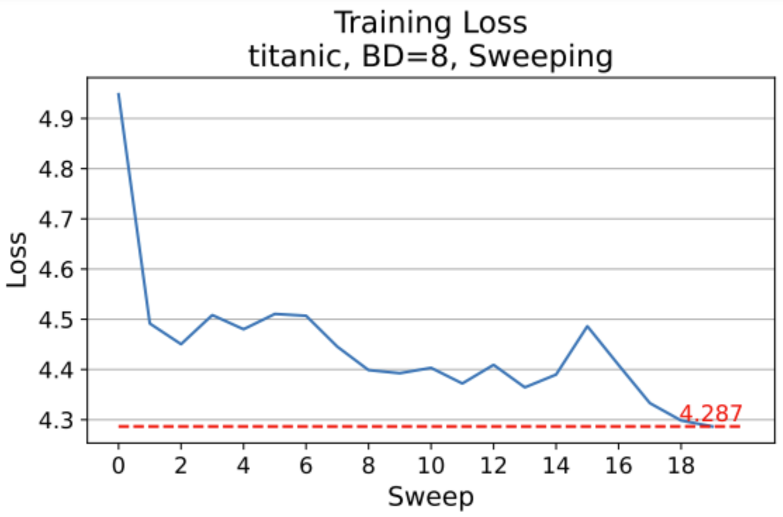
Titanic

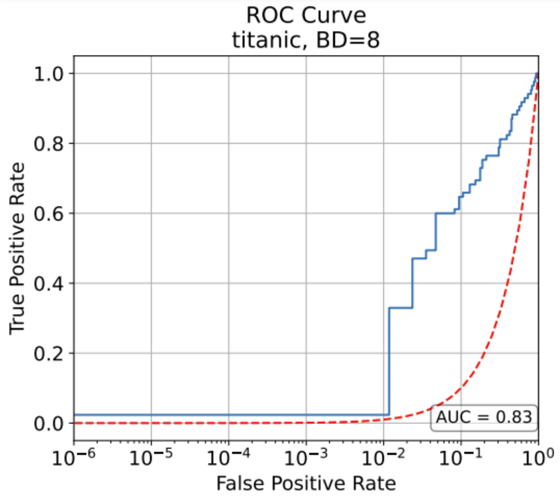
spin


Training Loss
titanic, BD=8, Sweeping


ROC Curve
titanic, BD=8

poly

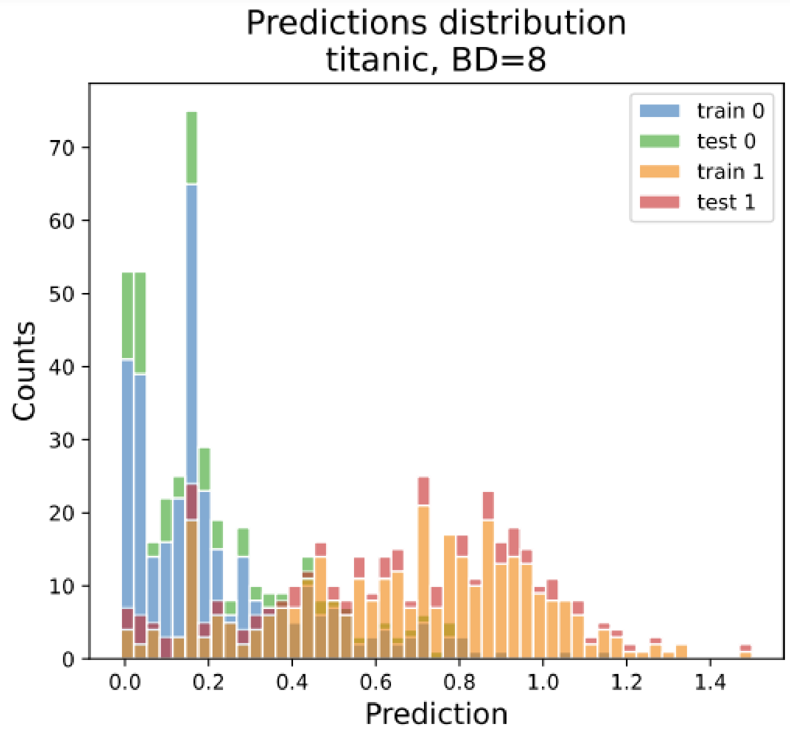

Training Loss
titanic, BD=8, Sweeping


ROC Curve
titanic, BD=8

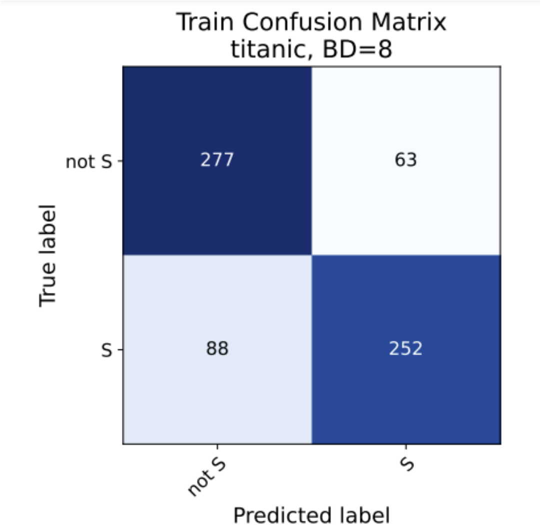
Titanic

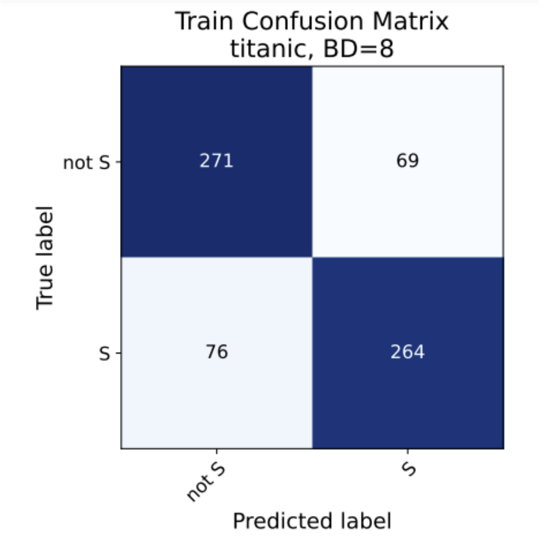
spin

poly



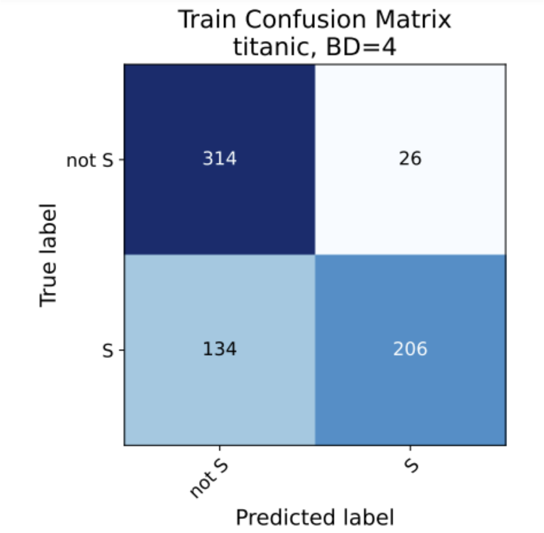
Titanic

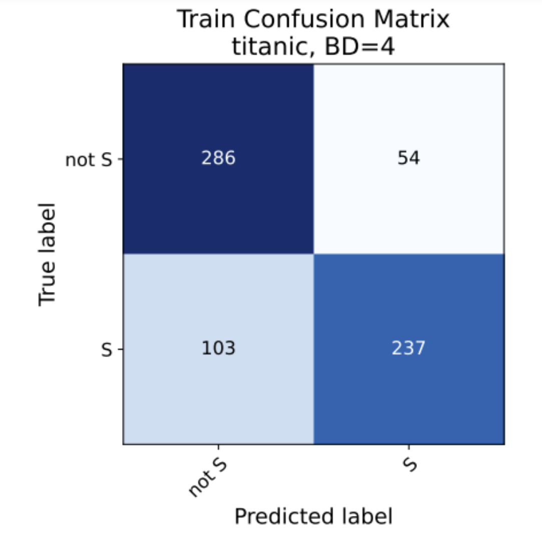

Train Confusion Matrix
titanic, BD=8


Train Confusion Matrix
titanic, BD=8

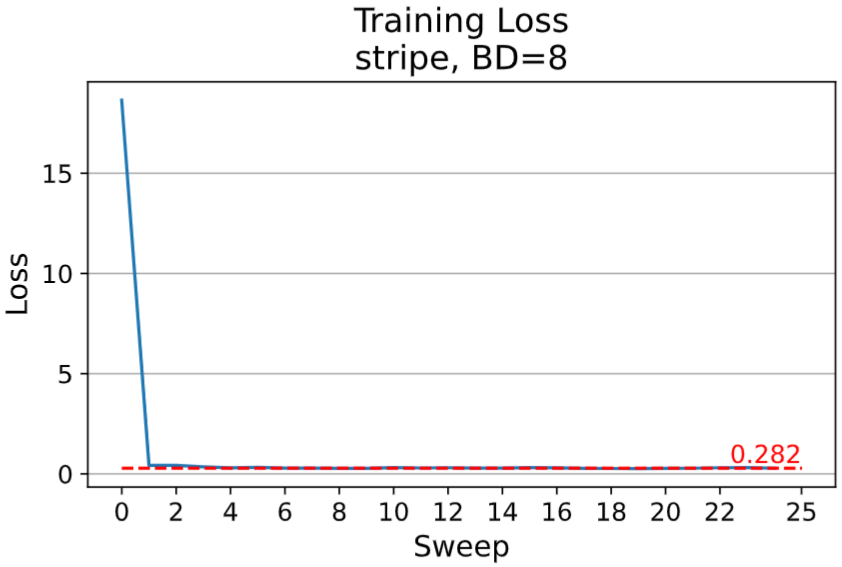
spin

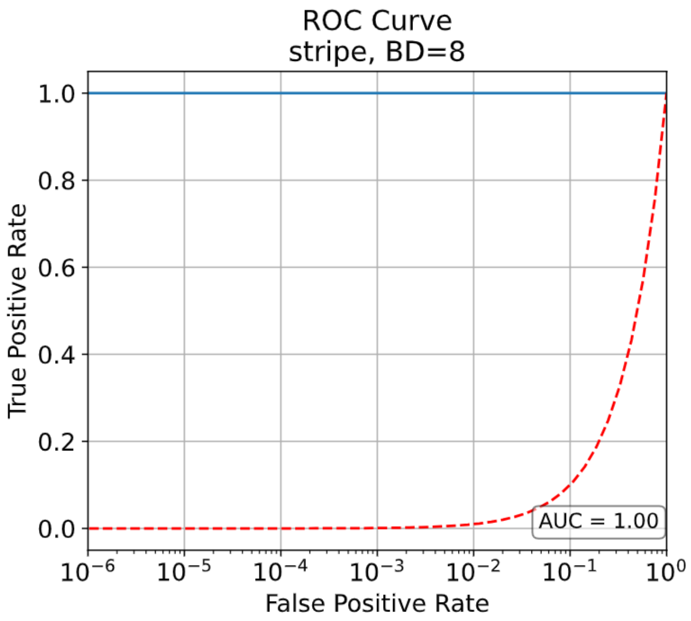
poly

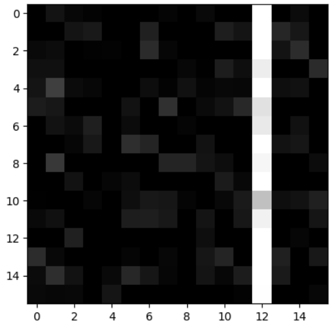

Train Confusion Matrix
titanic, BD=4


Train Confusion Matrix
titanic, BD=4

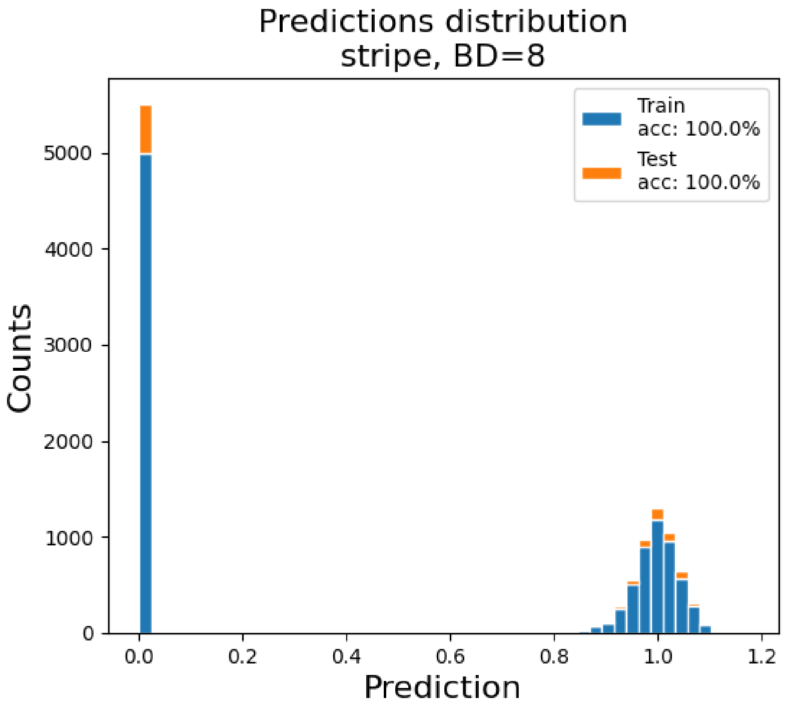
Striped images

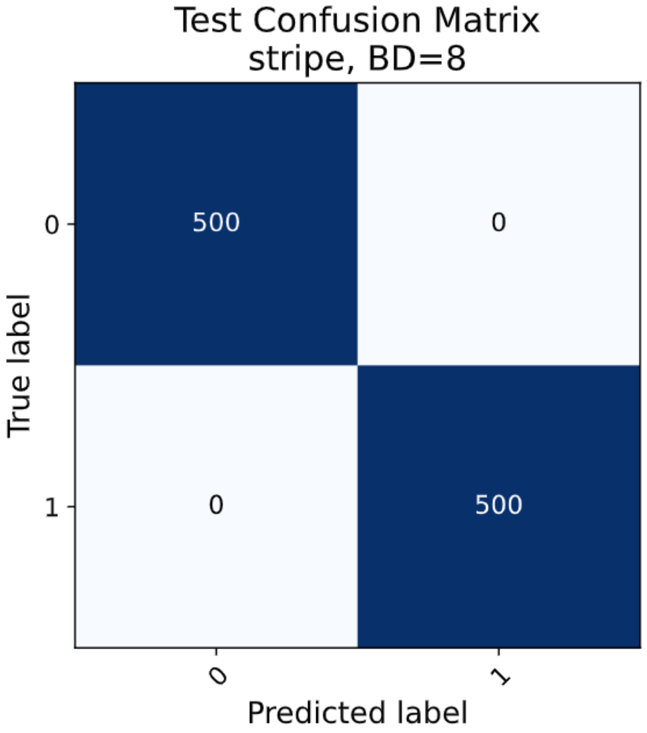






Striped images

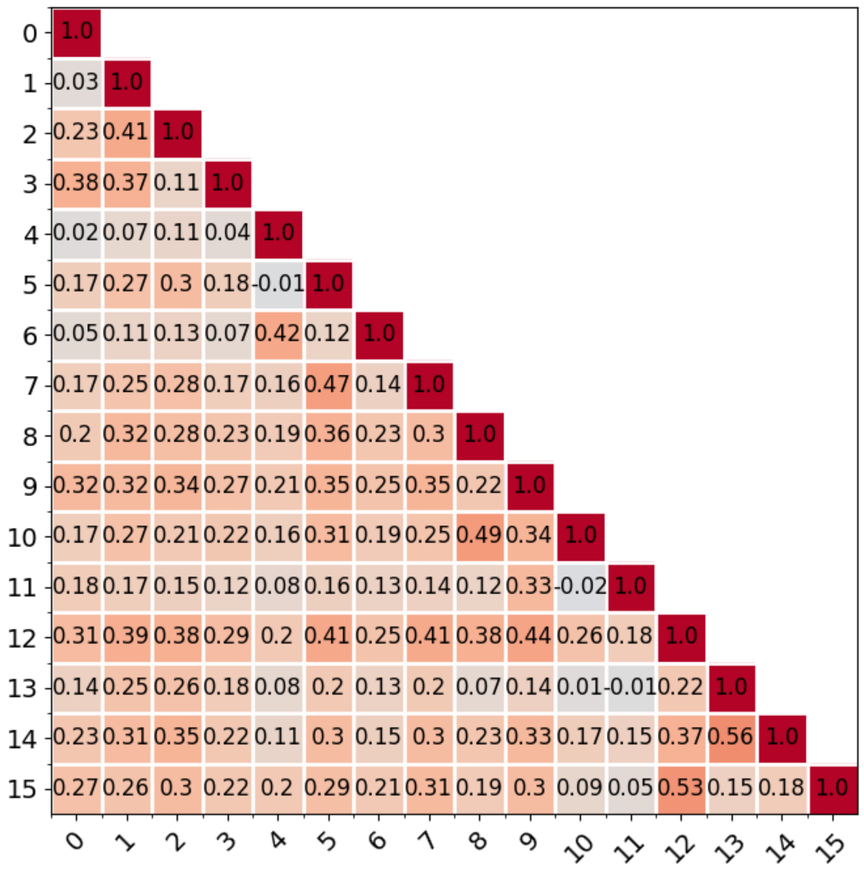




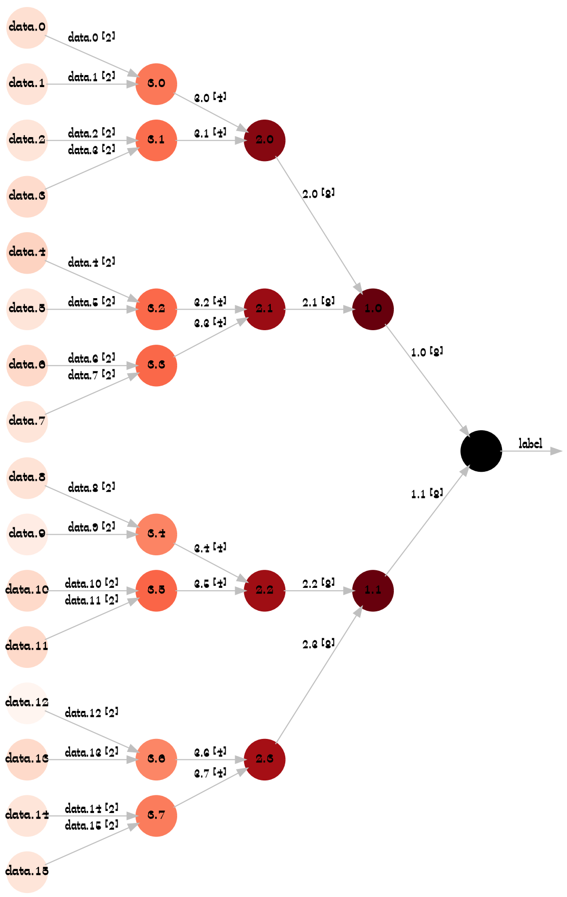
Striped images

σ_z correlations between features

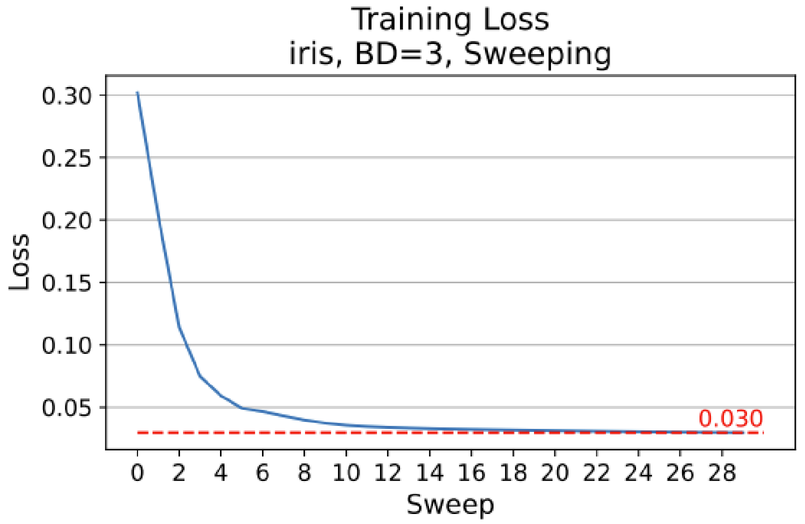
Vertical

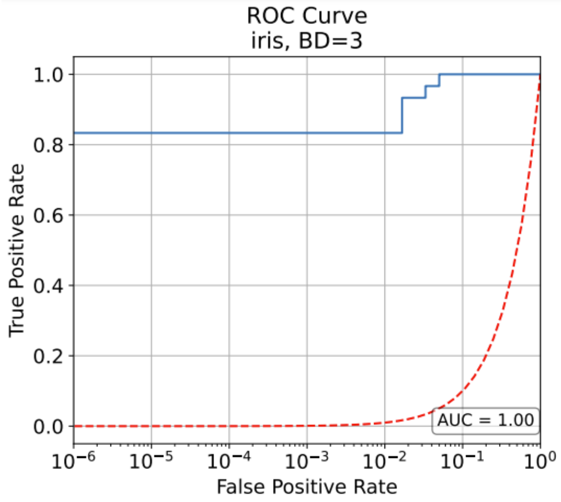


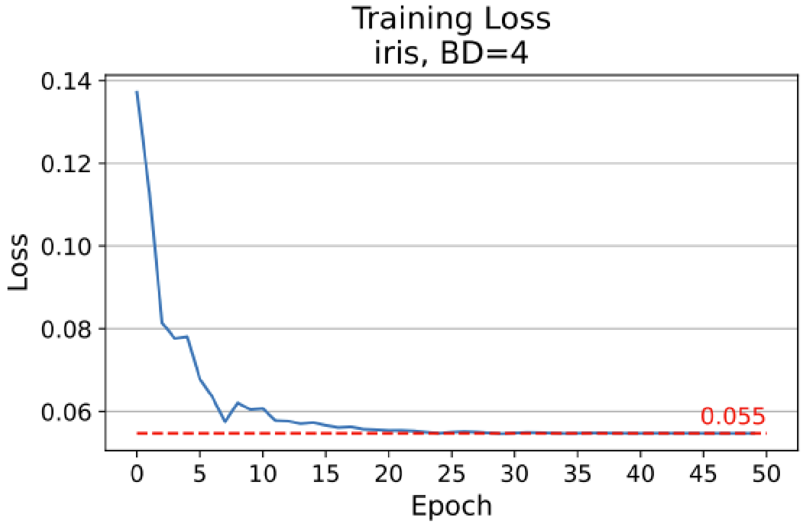
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0															
1	0.03	1.0														
2	0.23	0.41	1.0													
3	0.38	0.37	0.11	1.0												
4	0.02	0.07	0.11	0.04	1.0											
5	0.17	0.27	0.3	0.18	-0.01	1.0										
6	0.05	0.11	0.13	0.07	0.42	0.12	1.0									
7	0.17	0.25	0.28	0.17	0.16	0.47	0.14	1.0								
8	0.2	0.32	0.28	0.23	0.19	0.36	0.23	0.3	1.0							
9	0.32	0.32	0.34	0.27	0.21	0.35	0.25	0.35	0.22	1.0						
10	0.17	0.27	0.21	0.22	0.16	0.31	0.19	0.25	0.49	0.34	1.0					
11	0.18	0.17	0.15	0.12	0.08	0.16	0.13	0.14	0.12	0.33	-0.02	1.0				
12	0.31	0.39	0.38	0.29	0.2	0.41	0.25	0.41	0.38	0.44	0.26	0.18	1.0			
13	0.14	0.25	0.26	0.18	0.08	0.2	0.13	0.2	0.07	0.14	0.01	-0.01	0.22	1.0		
14	0.23	0.31	0.35	0.22	0.11	0.3	0.15	0.3	0.23	0.33	0.17	0.15	0.37	0.56	1.0	
15	0.27	0.26	0.3	0.22	0.2	0.29	0.21	0.31	0.19	0.3	0.09	0.05	0.53	0.15	0.18	1.0

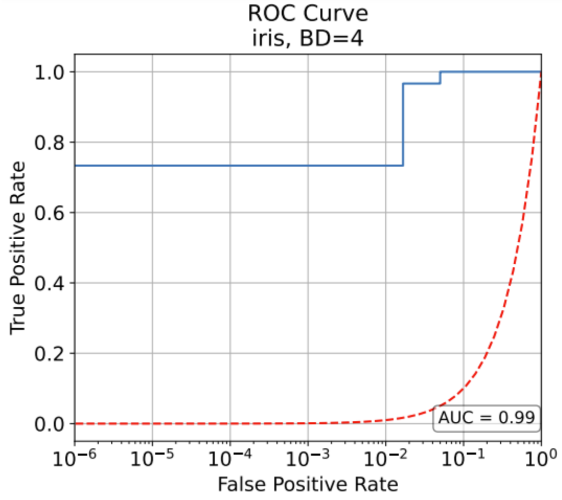


Iris

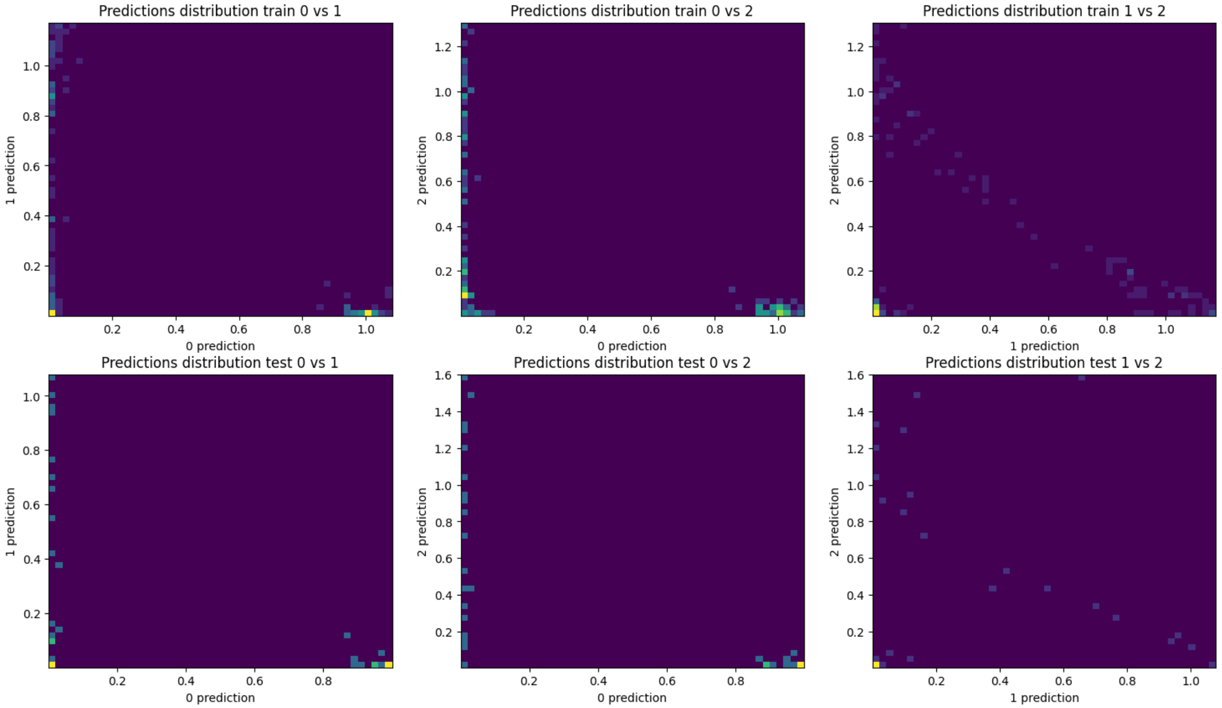

Training Loss
iris, BD=3, Sweeping
Loss
0.30
0.25
0.20
0.15
0.10
0.05
0.030
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28
Sweep


ROC Curve
iris, BD=3
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
AUC = 1.00
10^{-6} 10^{-5} 10^{-4} 10^{-3} 10^{-2} 10^{-1} 10^{0}
False Positive Rate

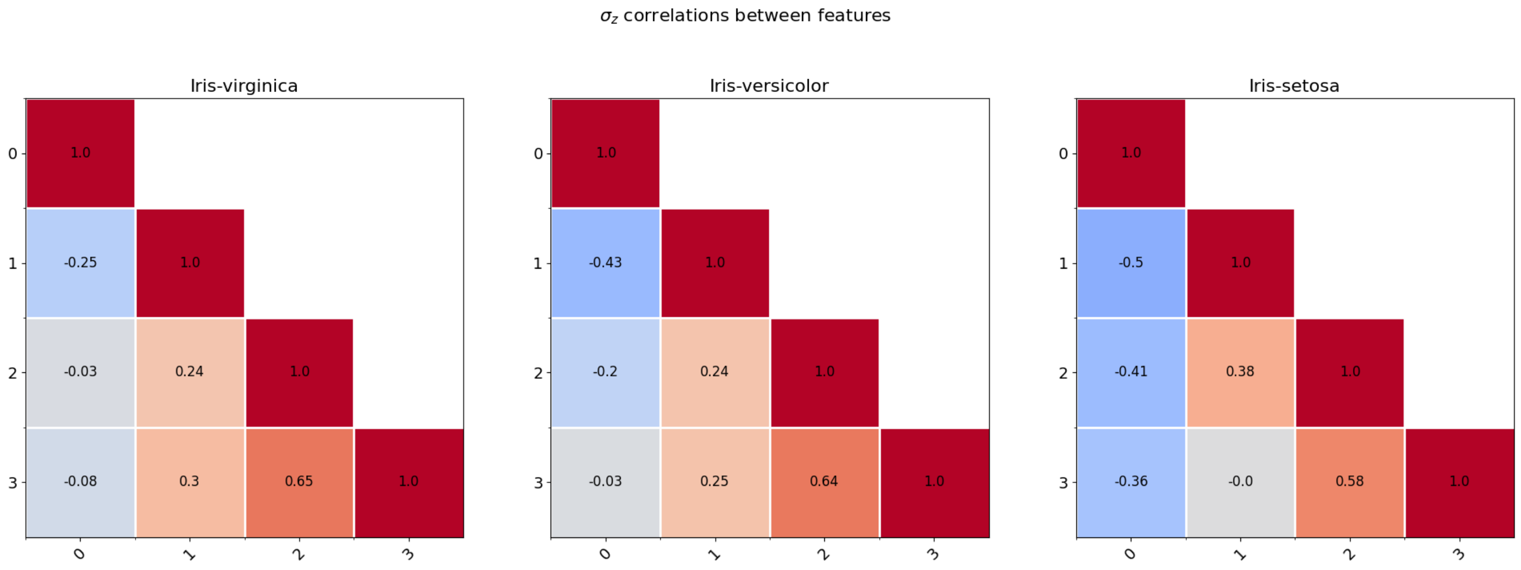

Training Loss
iris, BD=4
Loss
0.14
0.12
0.10
0.08
0.06
0.055
0 5 10 15 20 25 30 35 40 45 50
Epoch

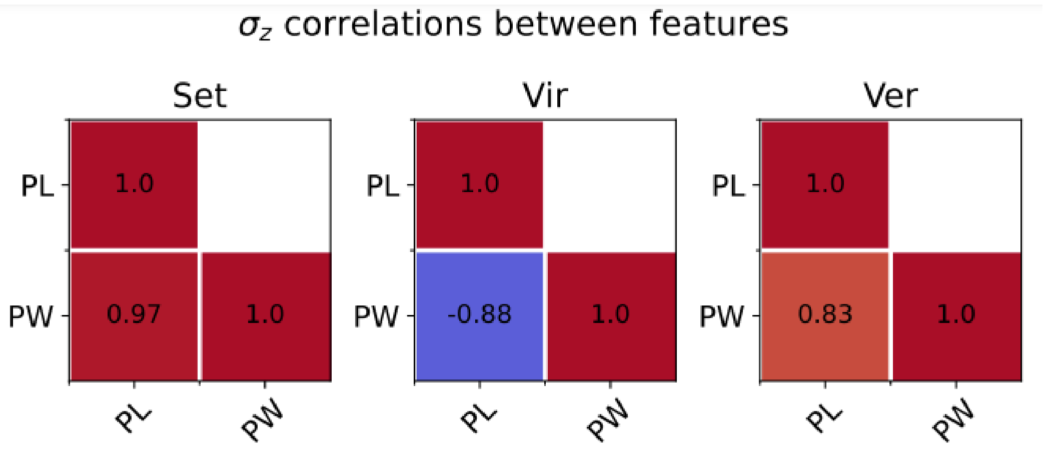

ROC Curve
iris, BD=4
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
AUC = 0.99
10^{-6} 10^{-5} 10^{-4} 10^{-3} 10^{-2} 10^{-1} 10^{0}
False Positive Rate

Iris

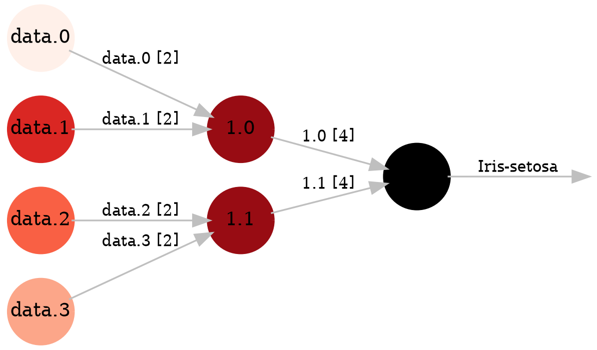

Predictions distribution train 0 vs 1
Predictions distribution train 0 vs 2
Predictions distribution train 1 vs 2
Predictions distribution test 0 vs 1
Predictions distribution test 0 vs 2
Predictions distribution test 1 vs 2
0 prediction
1 prediction
2 prediction
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

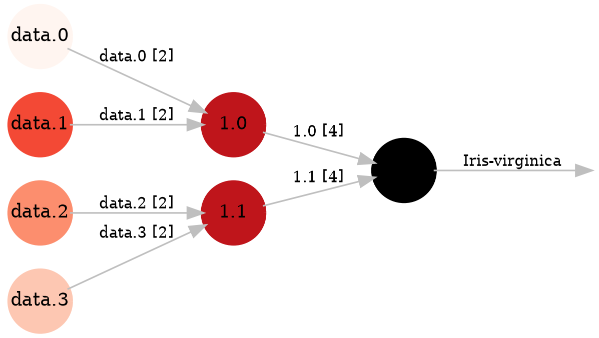
Iris

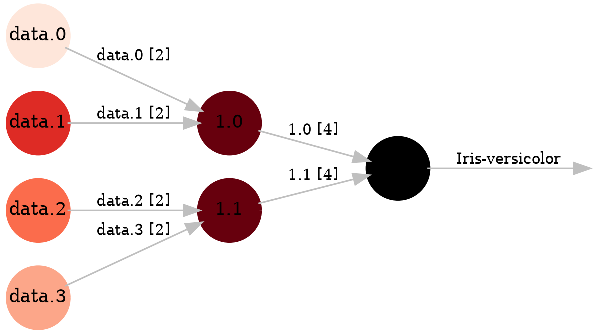

σ_z correlations between features
Iris-virginica
Iris-versicolor
Iris-setosa
0
1
2
3
1.0
-0.25
1.0
-0.03
0.24
1.0
-0.08
0.3
0.65
1.0
1.0
-0.43
1.0
-0.2
0.24
1.0
-0.03
0.25
0.64
1.0
1.0
-0.5
1.0
-0.41
0.38
1.0
-0.36
-0.0
0.58
1.0

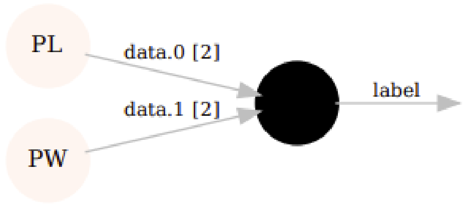

σ_z correlations between features
Set
Vir
Ver
PL
PW
1.0
0.97
1.0
1.0
-0.88
1.0
1.0
0.83
1.0

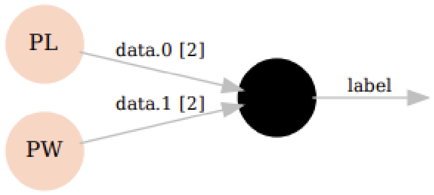
Iris

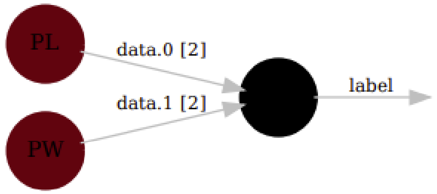

data.0
data.1
data.2
data.3
data.0 [2]
data.1 [2]
data.2 [2]
data.3 [2]
1.0
1.1
1.0 [4]
1.1 [4]
Iris-setosa


data.0
data.1
data.2
data.3
data.0 [2]
data.1 [2]
data.2 [2]
data.3 [2]
1.0
1.1
1.0 [4]
1.1 [4]
Iris-virginica


data.0
data.1
data.2
data.3
data.0 [2]
data.1 [2]
data.2 [2]
data.3 [2]
1.0
1.1
1.0 [4]
1.1 [4]
Iris-versicolor

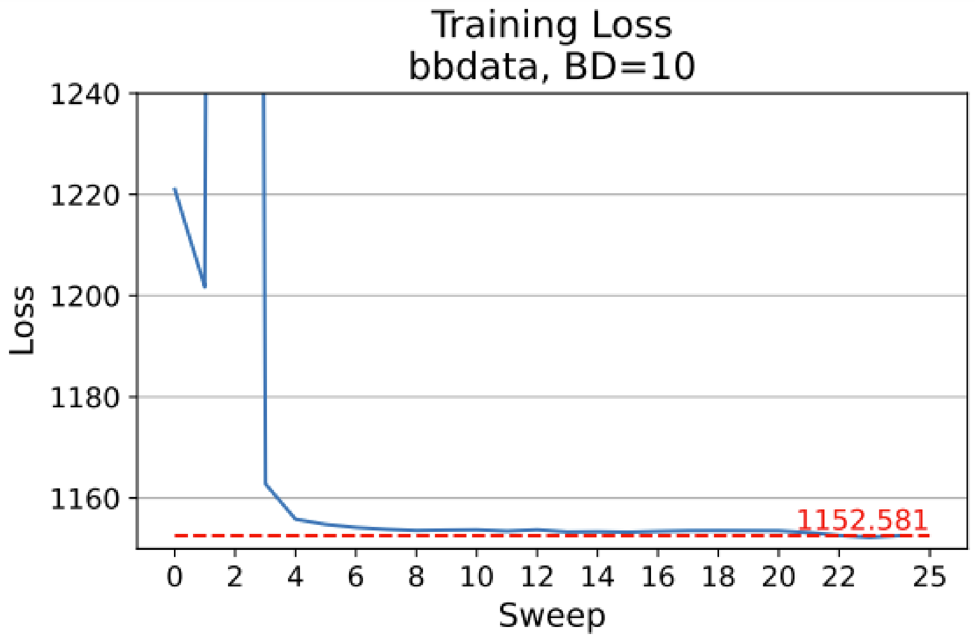

PL
PW
data.0 [2]
data.1 [2]
label

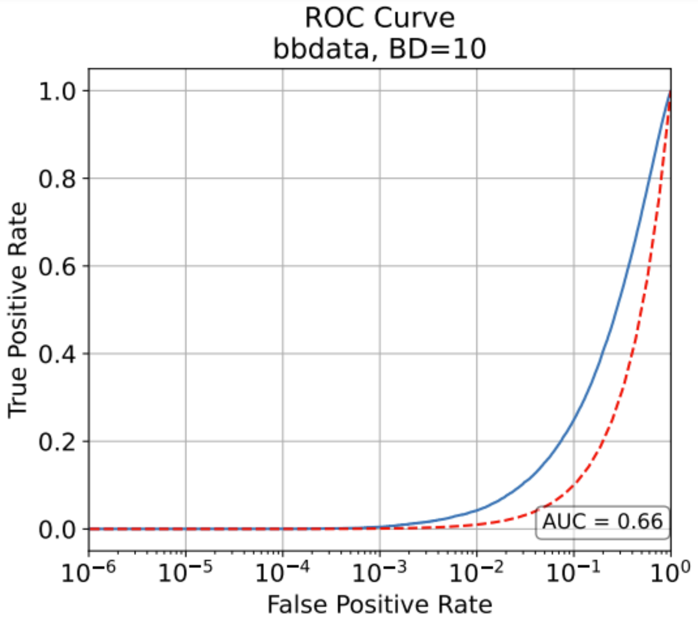

PL
PW
data.0 [2]
data.1 [2]
label


PL
PW
data.0 [2]
data.1 [2]
label

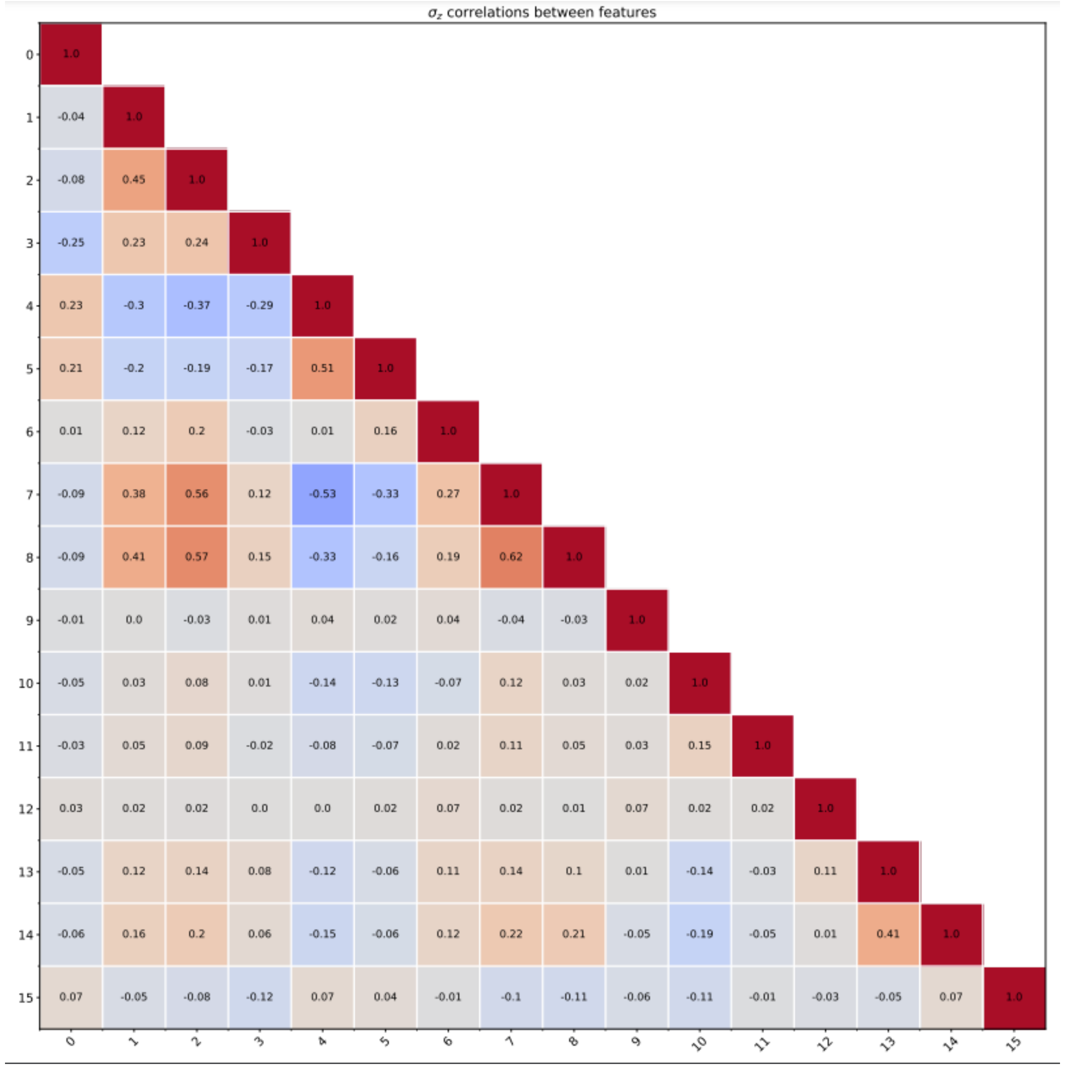
LHCb

LHCb


Training Loss
bbdata, BD=10
Loss
1240
1220
1200
1180
1160
1152.581
0
2
4
6
8
10
12
14
16
18
20
22
25
Sweep

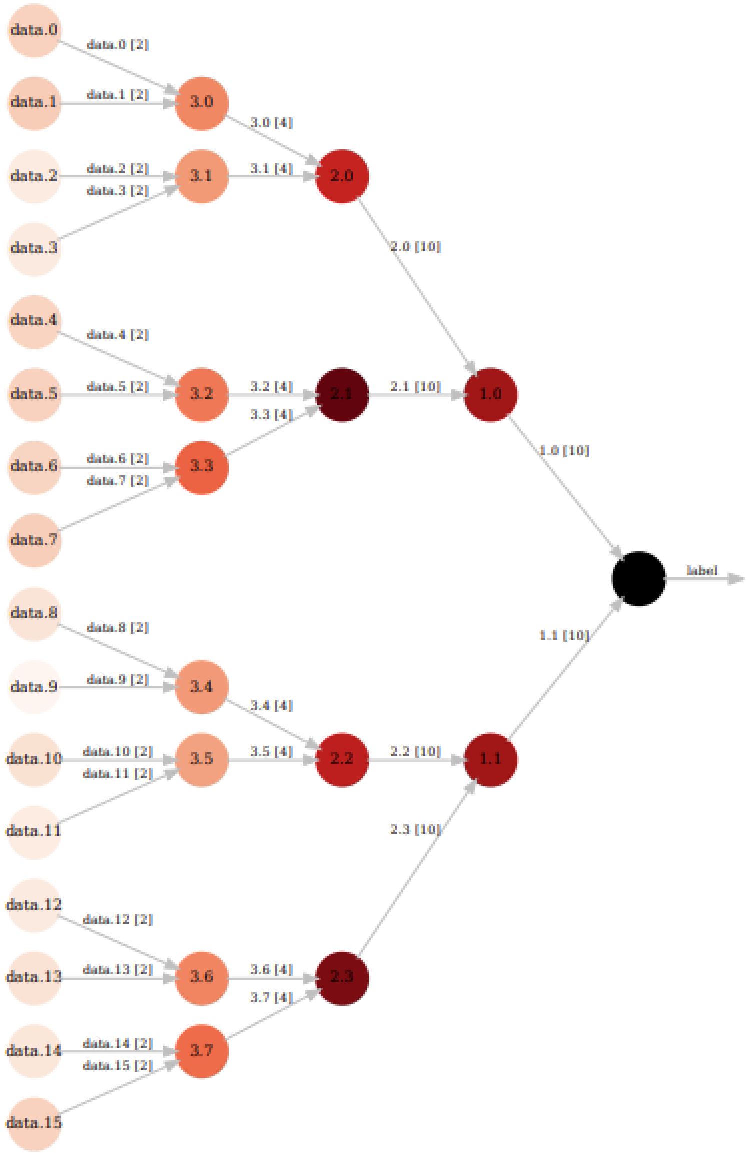

ROC Curve
bbdata, BD=10
True Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
AUC = 0.66
10^{-6}
10^{-5}
10^{-4}
10^{-3}
10^{-2}
10^{-1}
10^{0}
False Positive Rate

LHCb

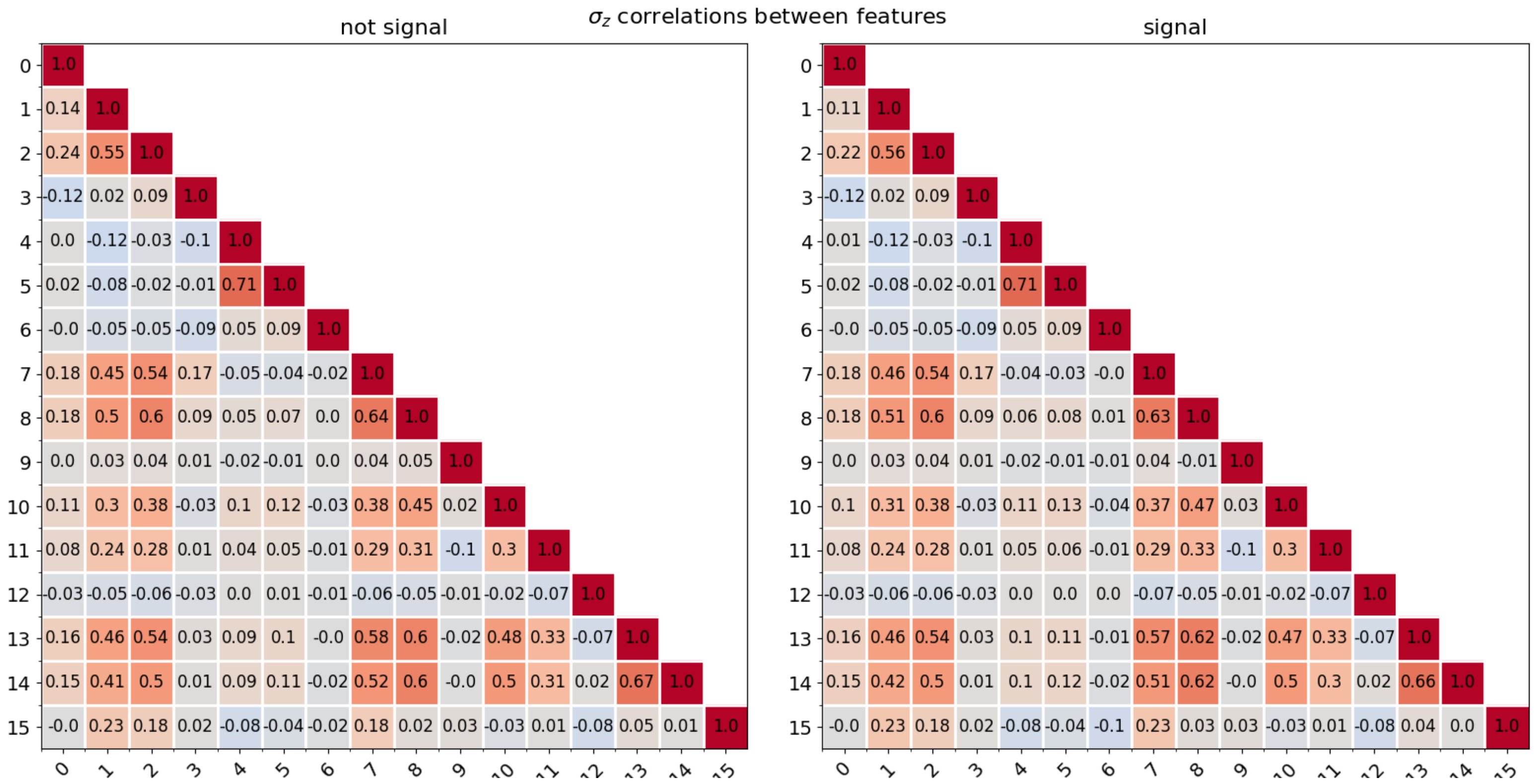


σ_z correlations between features

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0															
1	-0.04	1.0														
2	-0.08	0.45	1.0													
3	-0.25	0.23	0.24	1.0												
4	0.23	-0.3	-0.37	-0.29	1.0											
5	0.21	-0.2	-0.19	-0.17	0.51	1.0										
6	0.01	0.12	0.2	-0.03	0.01	0.16	1.0									
7	-0.09	0.38	0.56	0.12	-0.53	-0.33	0.27	1.0								
8	-0.09	0.41	0.57	0.15	-0.33	-0.16	0.19	0.62	1.0							
9	-0.01	0.0	-0.03	0.01	0.04	0.02	0.04	-0.04	-0.03	1.0						
10	-0.05	0.03	0.08	0.01	-0.14	-0.13	-0.07	0.12	0.03	0.02	1.0					
11	-0.03	0.05	0.09	-0.02	-0.08	-0.07	0.02	0.11	0.05	0.03	0.15	1.0				
12	0.03	0.02	0.02	0.0	0.0	0.02	0.07	0.02	0.01	0.07	0.02	0.02	1.0			
13	-0.05	0.12	0.14	0.08	-0.12	-0.06	0.11	0.14	0.1	0.01	-0.14	-0.03	0.11	1.0		
14	-0.06	0.16	0.2	0.06	-0.15	-0.06	0.12	0.22	0.21	-0.05	-0.19	-0.05	0.01	0.41	1.0	
15	0.07	-0.05	-0.08	-0.12	0.07	0.04	-0.01	-0.1	-0.11	-0.06	-0.11	-0.01	-0.03	-0.05	0.07	1.0



LHCb



σ_z correlations between features

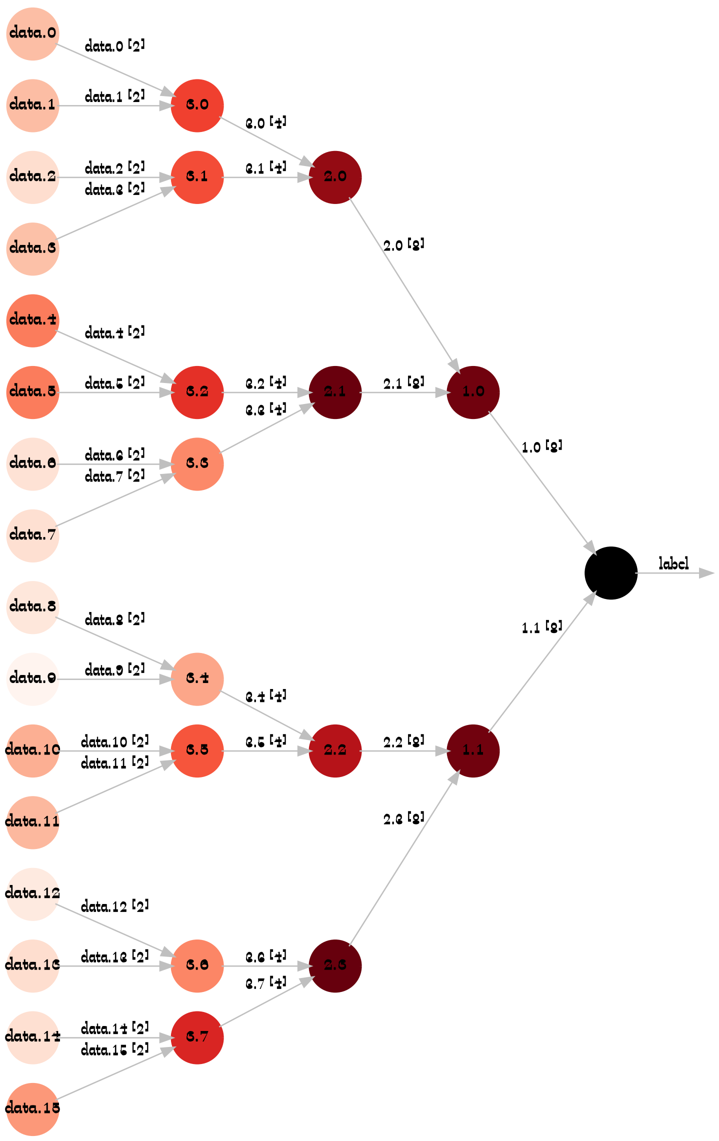
not signal

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0															
1	0.14	1.0														
2	0.24	0.55	1.0													
3	-0.12	0.02	0.09	1.0												
4	0.0	-0.12	-0.03	-0.1	1.0											
5	0.02	-0.08	-0.02	-0.01	0.71	1.0										
6	-0.0	-0.05	-0.05	-0.09	0.05	0.09	1.0									
7	0.18	0.45	0.54	0.17	-0.05	-0.04	-0.02	1.0								
8	0.18	0.5	0.6	0.09	0.05	0.07	0.0	0.64	1.0							
9	0.0	0.03	0.04	0.01	-0.02	-0.01	0.0	0.04	0.05	1.0						
10	0.11	0.3	0.38	-0.03	0.1	0.12	-0.03	0.38	0.45	0.02	1.0					
11	0.08	0.24	0.28	0.01	0.04	0.05	-0.01	0.29	0.31	-0.1	0.3	1.0				
12	-0.03	-0.05	-0.06	-0.03	0.0	0.01	-0.01	-0.06	-0.05	-0.01	-0.02	-0.07	1.0			
13	0.16	0.46	0.54	0.03	0.09	0.1	-0.0	0.58	0.6	-0.02	0.48	0.33	-0.07	1.0		
14	0.15	0.41	0.5	0.01	0.09	0.11	-0.02	0.52	0.6	-0.0	0.5	0.31	0.02	0.67	1.0	
15	-0.0	0.23	0.18	0.02	-0.08	-0.04	-0.02	0.18	0.02	0.03	-0.03	0.01	-0.08	0.05	0.01	1.0

signal

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1.0															
1	0.11	1.0														
2	0.22	0.56	1.0													
3	-0.12	0.02	0.09	1.0												
4	0.01	-0.12	-0.03	-0.1	1.0											
5	0.02	-0.08	-0.02	-0.01	0.71	1.0										
6	-0.0	-0.05	-0.05	-0.09	0.05	0.09	1.0									
7	0.18	0.46	0.54	0.17	-0.04	-0.03	-0.0	1.0								
8	0.18	0.51	0.6	0.09	0.06	0.08	0.01	0.63	1.0							
9	0.0	0.03	0.04	0.01	-0.02	-0.01	-0.01	0.04	-0.01	1.0						
10	0.1	0.31	0.38	-0.03	0.11	0.13	-0.04	0.37	0.47	0.03	1.0					
11	0.08	0.24	0.28	0.01	0.05	0.06	-0.01	0.29	0.33	-0.1	0.3	1.0				
12	-0.03	-0.06	-0.06	-0.03	0.0	0.0	0.0	-0.07	-0.05	-0.01	-0.02	-0.07	1.0			
13	0.16	0.46	0.54	0.03	0.1	0.11	-0.01	0.57	0.62	-0.02	0.47	0.33	-0.07	1.0		
14	0.15	0.42	0.5	0.01	0.1	0.12	-0.02	0.51	0.62	-0.0	0.5	0.3	0.02	0.66	1.0	
15	-0.0	0.23	0.18	0.02	-0.08	-0.04	-0.1	0.23	0.03	0.03	-0.03	0.01	-0.08	0.04	0.0	1.0

LHCb


data.0
data.1
data.2
data.3
data.4
data.5
data.6
data.7
data.8
data.9
data.10
data.11
data.12
data.13
data.14
data.15
data.0 [2]
data.1 [2]
data.2 [2]
data.3 [2]
data.4 [2]
data.5 [2]
data.6 [2]
data.7 [2]
data.8 [2]
data.9 [2]
data.10 [2]
data.11 [2]
data.12 [2]
data.13 [2]
data.14 [2]
data.15 [2]
3.0
3.1
3.2
3.3
3.4
3.5
3.6
3.7
3.0 [4]
3.1 [4]
3.2 [4]
3.3 [4]
3.4 [4]
3.5 [4]
3.6 [4]
3.7 [4]
2.0
2.1
2.2
2.3
2.0 [8]
2.1 [8]
2.2 [8]
2.3 [8]
1.0
1.1
1.0 [8]
1.1 [8]
label

GPU

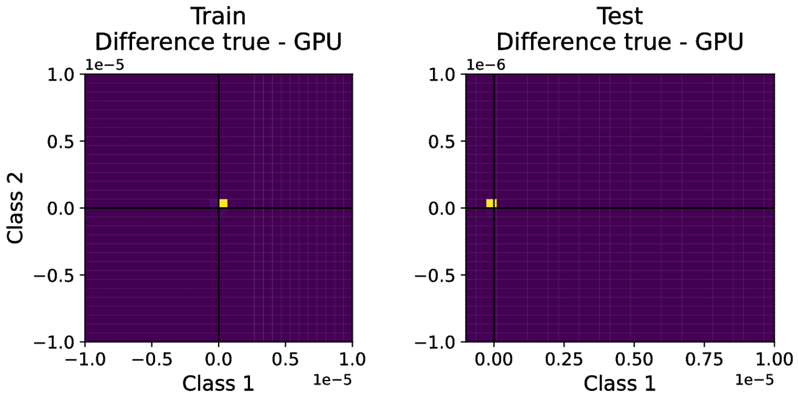
Classifiers on hardware accelerators:

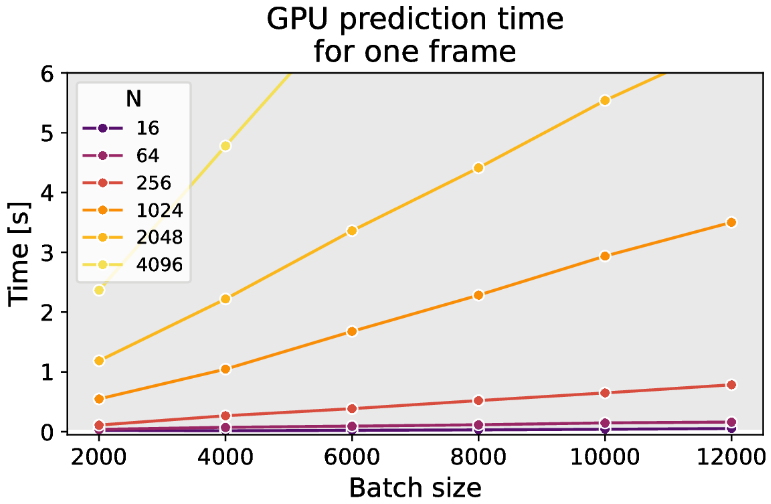
GPU

- GPU predictor implemented as part of **internship** for Tensor AI Solutions
- Based on **cuTENSOR**, a CUDA library for tensor contraction on NVIDIA GPUs
- **Tested** on the **trained models** mentioned before
- Tested on **FSOCO dataset** for traffic cones detection

Results:

- ➢ Perfect **match between** software and hardware **outputs**
- ➢ **Partial compliance** with video frame-rates

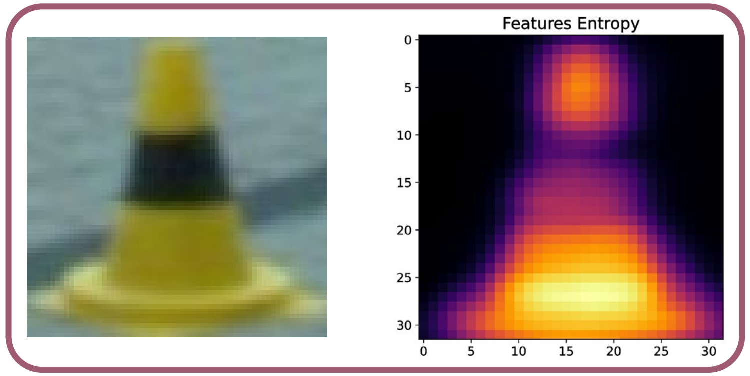




Classifiers on hardware accelerators:

GPU

- Trained a model on **FSOCO dataset**, containing high-quality camera images with cones delimited by bounding boxes
- Traffic cones detection performed through **sliding windows** technique
- **Features entropy** explain what the model learned
- Model shows **promising results** in object identification, but further refinements are needed


Features Entropy