

Quantum Machine Learning applied to High Energy Physics

Searching for stop 4-body decay using Variational Quantum Classifiers

Diogo de Bastos | MEAFP-II

Can QC help HEP in the NISQ era?

One of the main objectives of [CERN Quantum Technology Initiative](#) (CERN QTI) is to investigate if quantum computing (QC) can be used in the field of high-energy physics (HEP).

Noisy Intermediate-Scale Quantum (NISQ) era:

- Noisy: imperfect control over qubits
- Intermediate: number of qubits ranging from 50 to a few hundred

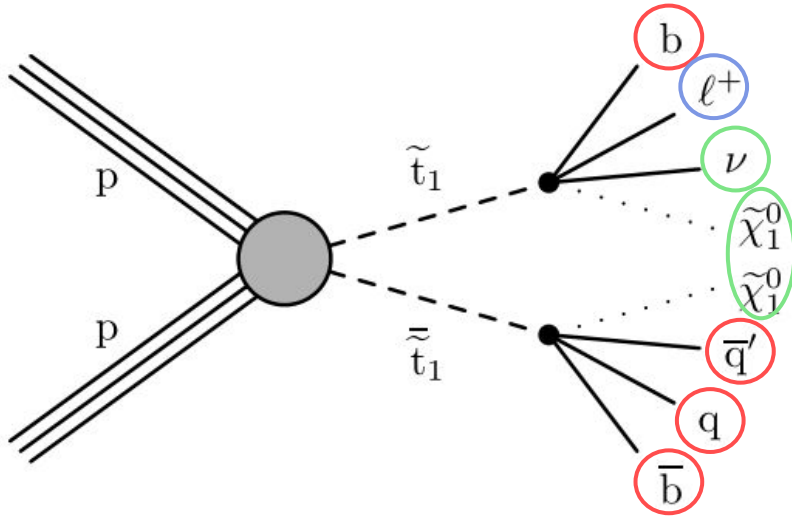
Today, we are going to explore **quantum machine learning** (QML) through the use of **Variational Quantum Circuits** (VQCs) using a simulated quantum computer applied to solve the problem of classification between **Supersymmetry** and **Standard Model** type of events.

Try it here!  <https://github.com/diogodebastos/StopQML>

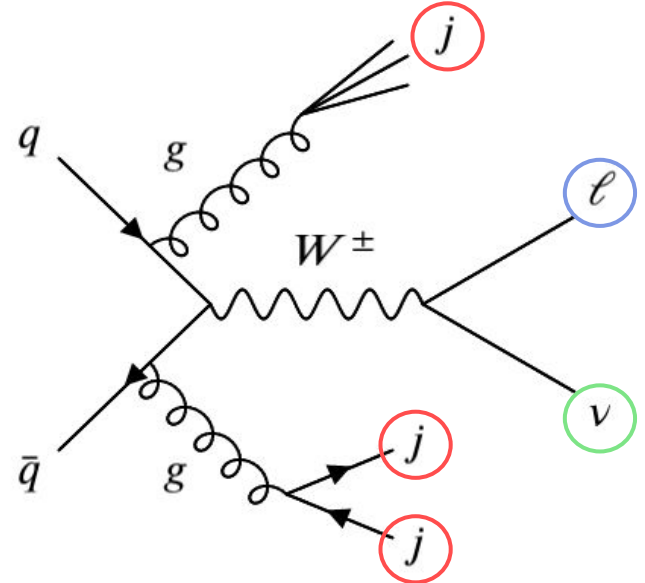
Searching for SUSY

Detector signature: 1 lepton + jets + MET

Signal: stop 4-body decay



Background: $W \rightarrow \ell + \nu + \text{jets}$



Let's build a **quantum model** to classify signal and background

Meet the qubit

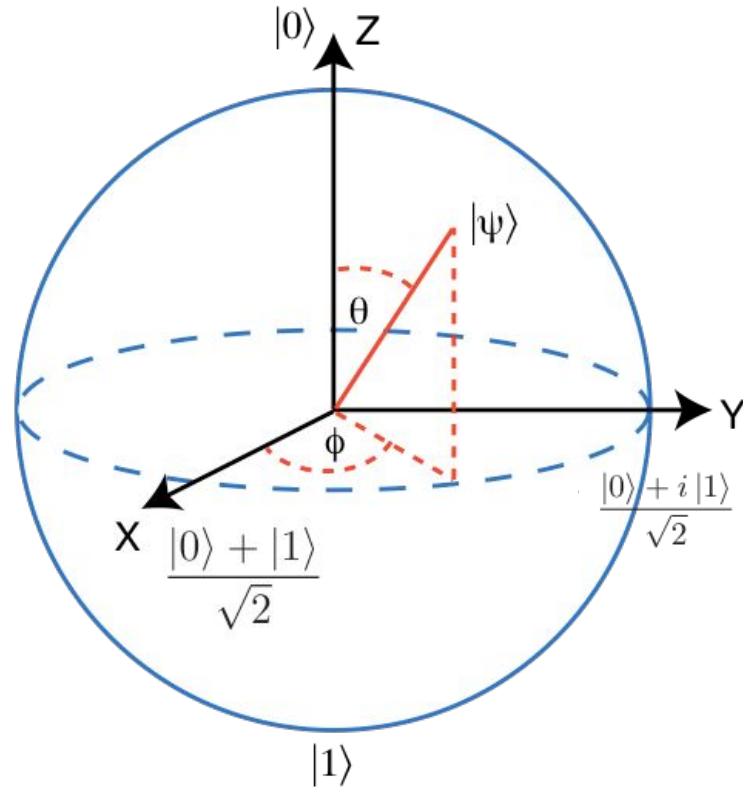
A quantum bit (qubit) is the basic unit of quantum information. We take advantage of properties of quantum mechanics computations such as **superposition** and **entanglement**.

A classic bit can only have a value of 1 **or** 0, a qubit can be in a superposition of 1 **and** 0. The qubit's 0 and 1 states are expressed as follows:

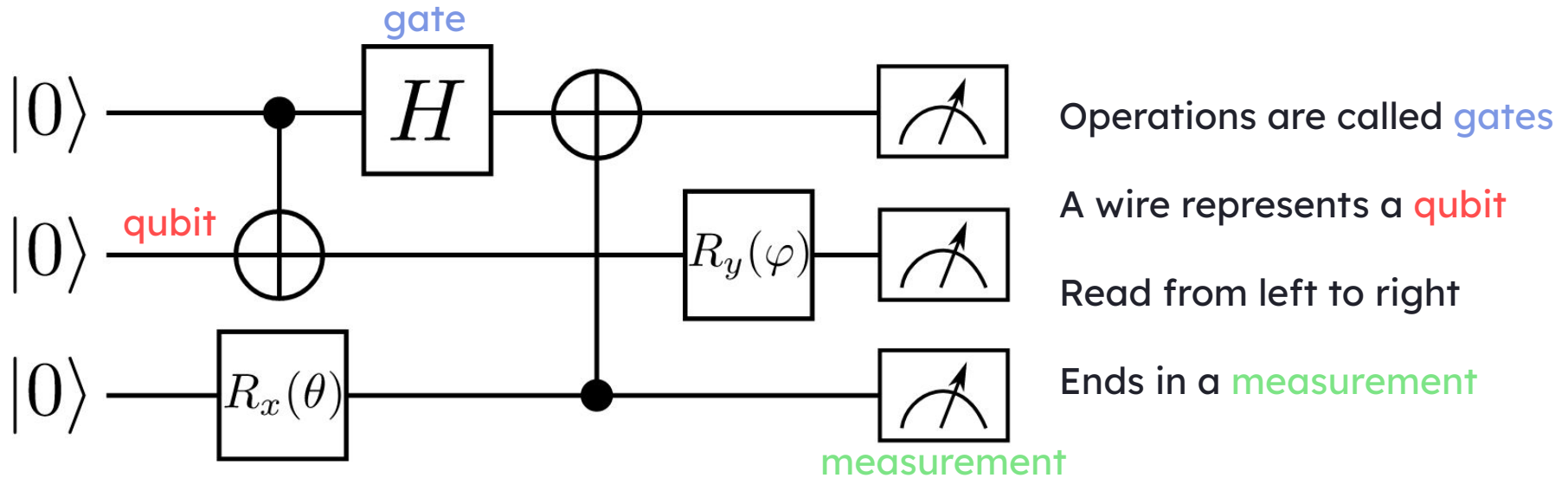
$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

Qubits can be in a superposition of states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha \begin{pmatrix} 1 \\ 0 \end{pmatrix} + \beta \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}, \alpha\alpha^* + \beta\beta^* = 1$$



Quantum Circuits



The circuit below has 3 qubits that start in state $|0\rangle$, applies 5 operations, and measures every qubit at the end.

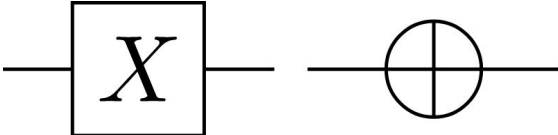
Quantum computation is all about manipulating qubit states by using gates and seeing the outcome of such manipulations by measuring the qubits.

A Brief introduction to Quantum Operations

Flipping qubits!

Quantum operations involve multiplication by matrices that send valid, normalized quantum states to other normalized quantum states.

The simplest operation: $X|1\rangle = |0\rangle$,
 $X|0\rangle = |1\rangle$

X is known as **bit flip**, **Pauli-X** or **NOT** gate: $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ or 

Let's see if this operation works...

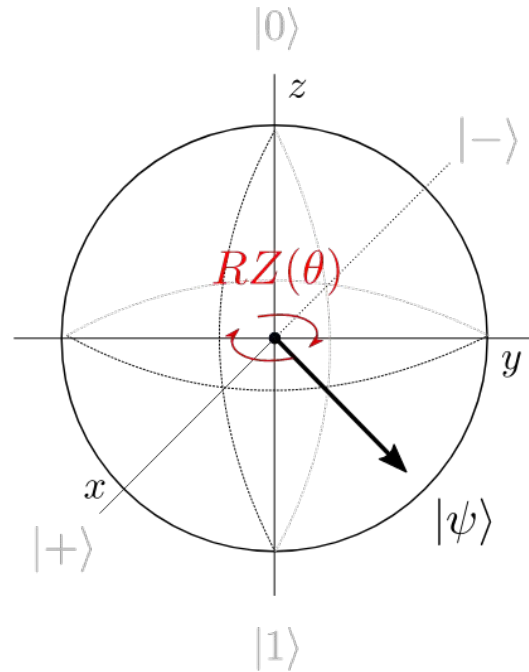
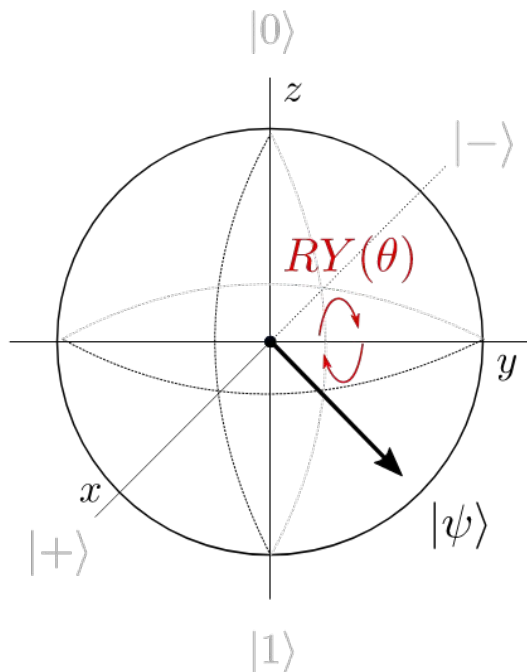
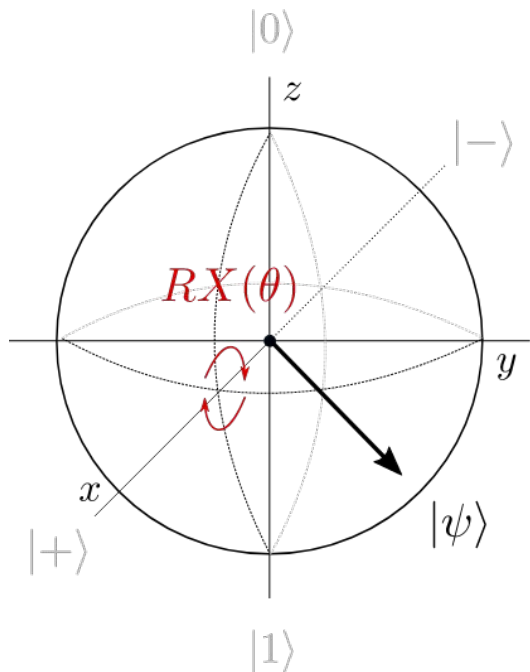
$$X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} = |1\rangle$$

We've **flipped** a quantum bit!

Rotations, rotations, rotations: RX, RZ & RY

A qubit can be represented in 3D-space (Bloch sphere)

RX, RZ, and RY rotate the qubit's state vector about the appropriate axis



The importance of qubit rotations

Rotations become very useful in the context of **QML** because it means we can manipulate the qubits as a function of a parameter (angle θ) in order to find some kind of optimum in the quantum algorithm we'll design

The matrix representation of $RX(\theta)$ is:

$$RX(\theta) = \begin{pmatrix} \cos(\frac{\theta}{2}) & -i \sin(\frac{\theta}{2}) \\ -i \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) \end{pmatrix}$$

Multiple qubit states

A single classical bit has 2 possible states, a qubit has 2 complex amplitudes.

In a similar manner, **2 classical bits** have **4 possible states**: 00 01 10 11

To describe a state of **2 qubits**, **4 complex amplitudes** are required:

$$|\psi\rangle = \psi_{00}|00\rangle + \psi_{01}|01\rangle + \psi_{10}|10\rangle + \psi_{11}|11\rangle = \begin{bmatrix} \psi_{00} \\ \psi_{01} \\ \psi_{10} \\ \psi_{11} \end{bmatrix} \leftarrow \text{state vector}$$

such that:

$$|\psi_{00}|^2 + |\psi_{01}|^2 + |\psi_{10}|^2 + |\psi_{11}|^2 = 1$$

Representing multiple qubit states

Let's suppose we have two separated qubits, $|a\rangle$ and $|b\rangle$:

$$|a\rangle = \begin{bmatrix} a_0 \\ a_1 \end{bmatrix}, |b\rangle = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

Using the kronecker product, the collective state (or the state a 2 qubits circuit):

$$|b\rangle \otimes |a\rangle = |ba\rangle = \begin{bmatrix} b_0 \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \\ b_1 \times \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} b_0 a_0 \\ b_0 a_1 \\ b_1 a_0 \\ b_1 a_1 \end{bmatrix} = b_0 a_0 |00\rangle + b_0 a_1 |01\rangle + b_1 a_0 |10\rangle + b_1 a_1 |11\rangle$$

If we have **n** qubits, we will need to keep track of **2ⁿ** complex amplitudes.

This is the reason quantum computers are so difficult to simulate A modern laptop can simulate a general quantum state of around 20 qubits, but simulating 100 qubits is too difficult even for the largest supercomputers!

Introduction to entanglement

We saw gates that act only on single qubits, to complete all the ingredients we need **gates that can act on multiple qubits** in order to entangle them.

The idea being **entanglement** is that knowing the state of a qubit, means knowing the state of another qubit. And the idea behind the **CNOT** gate is that we can use a qubit to control the state of another one. When this 2 concepts are merged together we can entangle qubits. Let's see how that's done.

By definition, a state is entangled if it cannot be described as a tensor product of individual qubit states. An entangled state can only be described by specifying the full state. One example of an entangled state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

An entangled state

In order to describe the previous state using the kronecker tensor we need to find a_0, a_1, b_0 and b_1 such that:

$$b_0 a_0 = 1$$

$$b_0 a_1 = 0$$

$$b_1 a_0 = 0$$

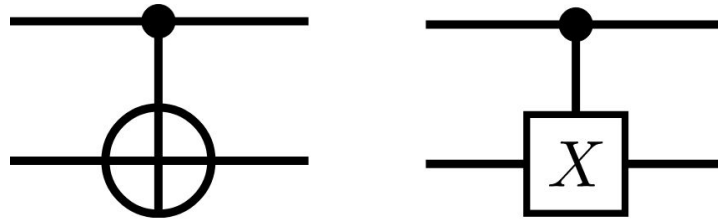
$$b_1 a_1 = 1$$

Each variable appears in two equations, one of which is equal to 1, and the other 0. But for any of the 0 ones to be true, we'd need at least one variable to be 0, and that would immediately contradict the other equation it is part of where the solution is 1.

Therefore, there is no solution, it's not possible to describe this state as two separate qubits, so **they are entangled!**

The CNOT gate

The controlled-NOT or CNOT gate can make an entangled state by performing the Pauli X operation on one qubit (target) depending on the state of another (control) and is represented in a quantum circuit as:



The state of the control qubit (dot) does not change, but its state is the one that determines whether the operation is performed on the target qubit. In matrix form:

$$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

CNOT in action

Let's say we have qubits $|q_0\rangle$ and $|q_1\rangle$:

$$|q_0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$$
$$|q_1\rangle = |0\rangle$$

the full state: $|q_0 q_1\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |01\rangle)$ which is not entangled. Let's apply CNOT

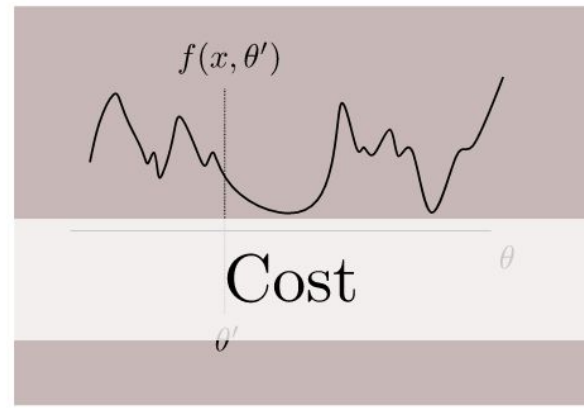
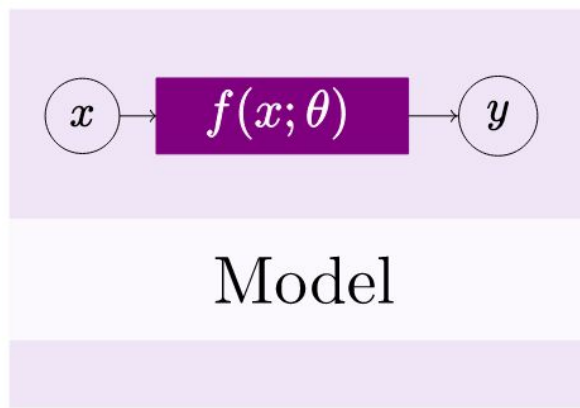
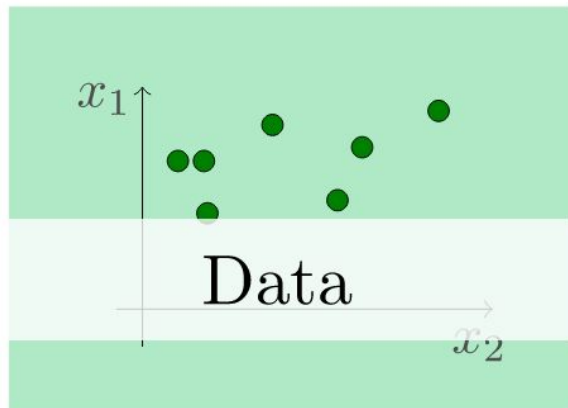
$$CNOT|q_0 q_1\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$$

Entangled state!

Note: this is known as the $|+\rangle$ state in the Bloch sphere and can be obtained by applying an Hadamard gate to $|0\rangle$

Quantum Machine Learning

Machine learning (general) approach

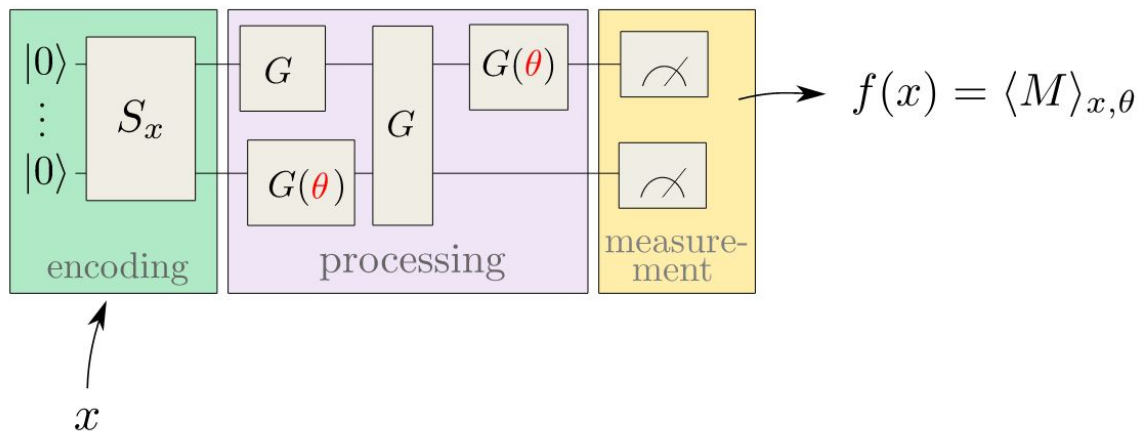


“Use data samples to construct a model that minimizes cost of unseen data.”

Variational Quantum Circuits - VQCs

Let's train quantum computers like we train neural networks (NNs)

PHYSICAL CIRCUIT



Encoding: sample data into “qubit form”

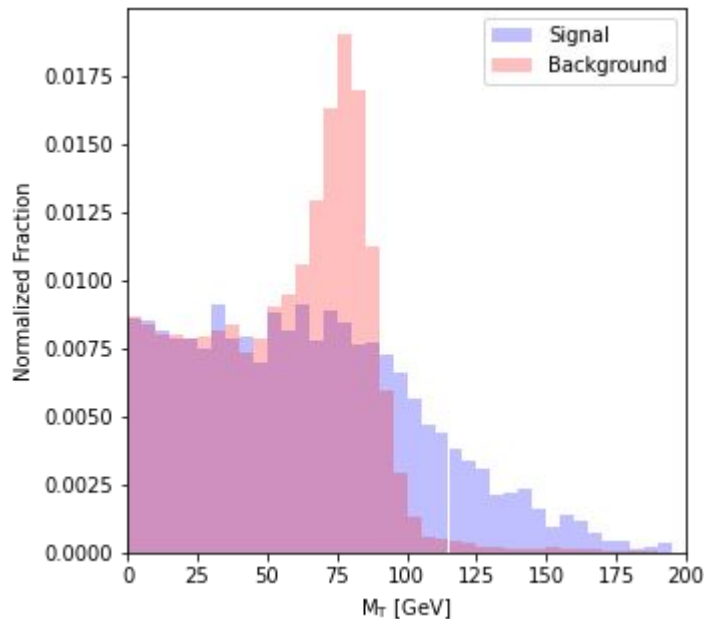
Processing: perform operations that depend on the parameter θ .

The model has to be trainable, composable and differentiable

Measurement: measure the qubits final state and compare to labeled data.

Measure a cost function

Understanding the samples



Signal: stop 4-body decay

Training label: **1**

Background: $W \rightarrow \ell + \nu + \text{jets}$

Training label: **-1**

Selection before training:

- Exactly 1 lepton
- Soft leptons: $p_T(\ell) > 5(3.5)$ GeV for e(μ)
- For $\Delta m \leq 60$ GeV: $p_T(\ell) < 30$ GeV
- $H_T > 200$ GeV
- $p_T^{\text{miss}} > 280$ GeV
- $p_T^{\text{ISR}} > 110$ GeV

Training features: $p_T(\ell)$, p_T^{miss} , p_T^{ISR} , M_T , H_T and B-jet discriminant

Training setup

Training features: $p_T(\ell)$, p_T^{miss} , p_T^{ISR} , M_T , H_T and B-jet discriminant

Features are normalized to XS / number of MC generated events

Using 512 events of background and signal to train

Features are then scaled from 0 to π in order to be transformed into the angle θ used in the **Encoding** phase of the VQC

Using $N(\text{qubits}) = N(\text{features}) == 5$

Split data randomly (80/20 %) into **train+validation** and **test** datasets

Building a trainable quantum circuit: Variational Quantum Circuit

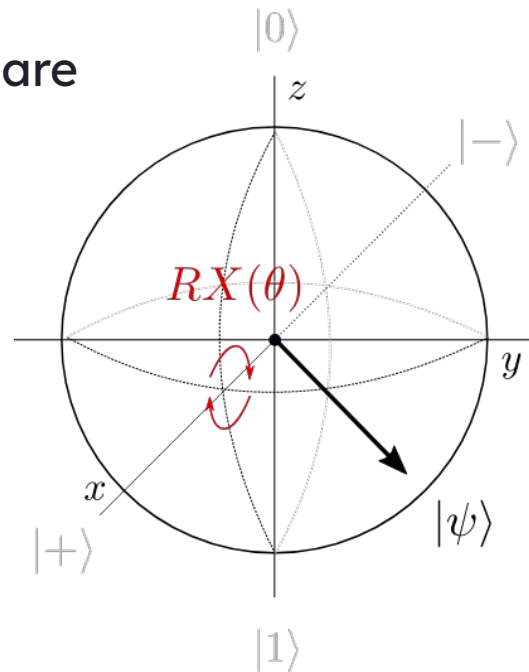
Encoding: Angle Embedding

Encodes **N** features into the **rotation angles** of **n qubits**, where $N \leq n$.

This is the first step into turning our data into quantum information.

5 independent qubits, one for each feature, are set to the $|0\rangle$ state. Then, we apply the $RX(\theta)$ rotation based on the angle θ obtained from the 0 to π scaling.

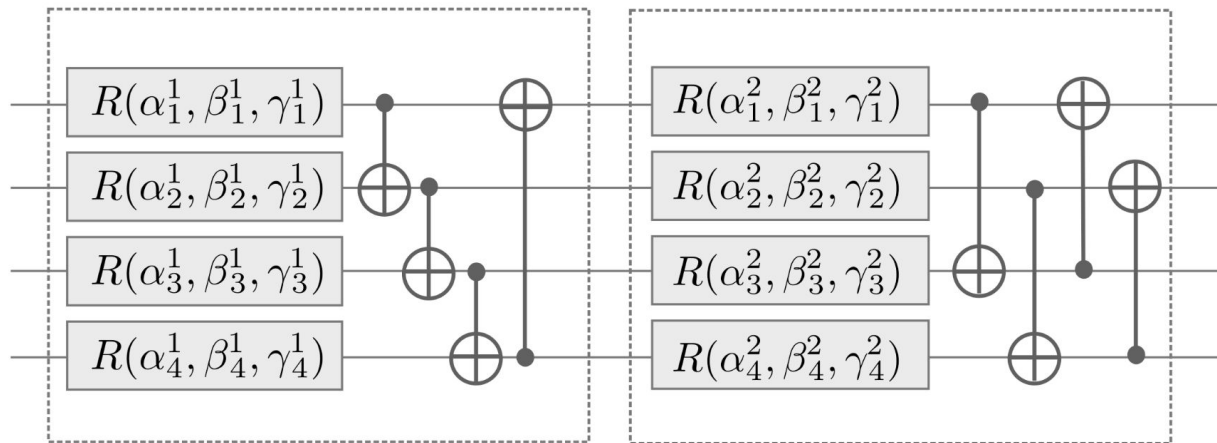
Thus, our data is encoded into “qubit form”



Processing: Strongly Entangling Layers

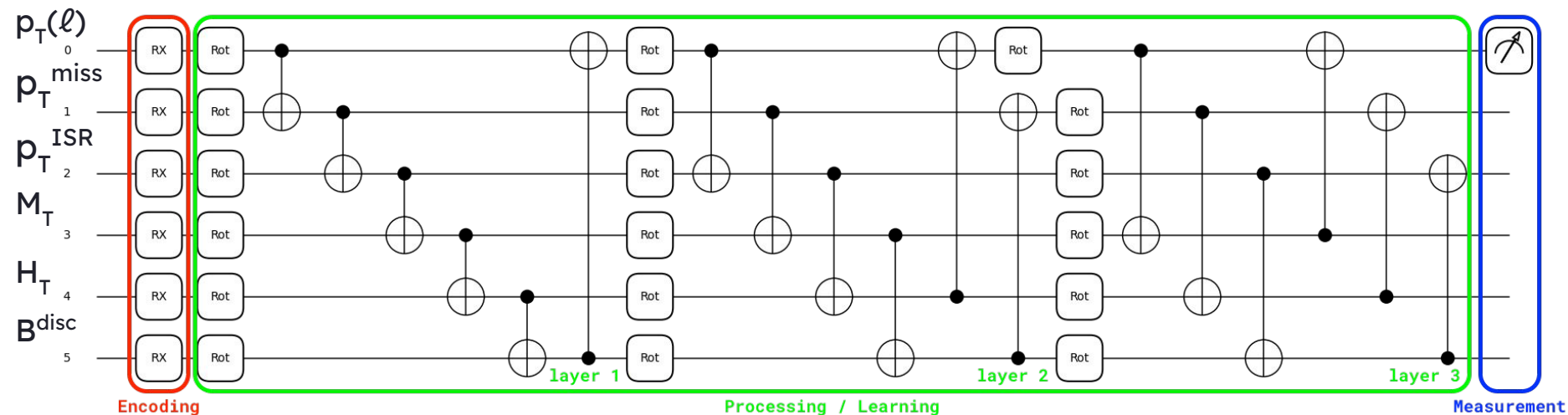
Layers consisting of single qubit rotations and entanglers, inspired by the circuit-centric classifier design [arXiv:1804.00633](https://arxiv.org/abs/1804.00633)

We entangle our **n qubits** across **m layers**: α_n^m , β_n^m and γ_n^m are the parameters we will tune until we have a model that is capable of classifying signal and background we call them weights, similar to NNs.



Training our Quantum Circuit

pred.



- We are using 3 strongly entangling layers
- The weights are randomly initialized
- Train+validation is split into train and validation (80/20%)
- We are not using PCA (but you can try it in the code!)
- 81 batches of 8 events
- We train for 5 epochs
- And using Adagrad optimizer to update the model weights

Loss function

For a realistic representation of the physics of signal and background, the chosen cost function is the Weighted Mean Squares Error (WMSE):

$$WMSE = \frac{1}{N \times \sum_{i=1}^N w_i} \sum_{i=1}^N (y_i - \hat{y}_i)^2 \times w_i$$

where N is the total number of events, y_i is the real label and \hat{y}_i the model prediction

This model takes **~3 minutes** to train on a modern laptop.

Metrics of the VQC

We can see that our quantum circuit trained, learned and didn't overfit

Area under the ROC

train: 89.6% | test: 88.8%

Accuracy:

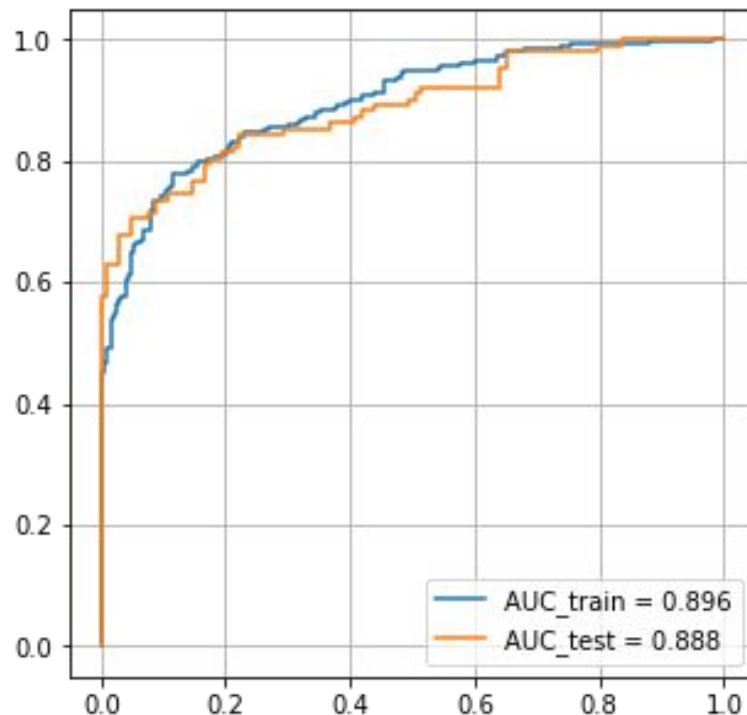
train: 73.3 % | val: 78.7 % | test: 74.1 %

Pro:

the model learns with a few data samples

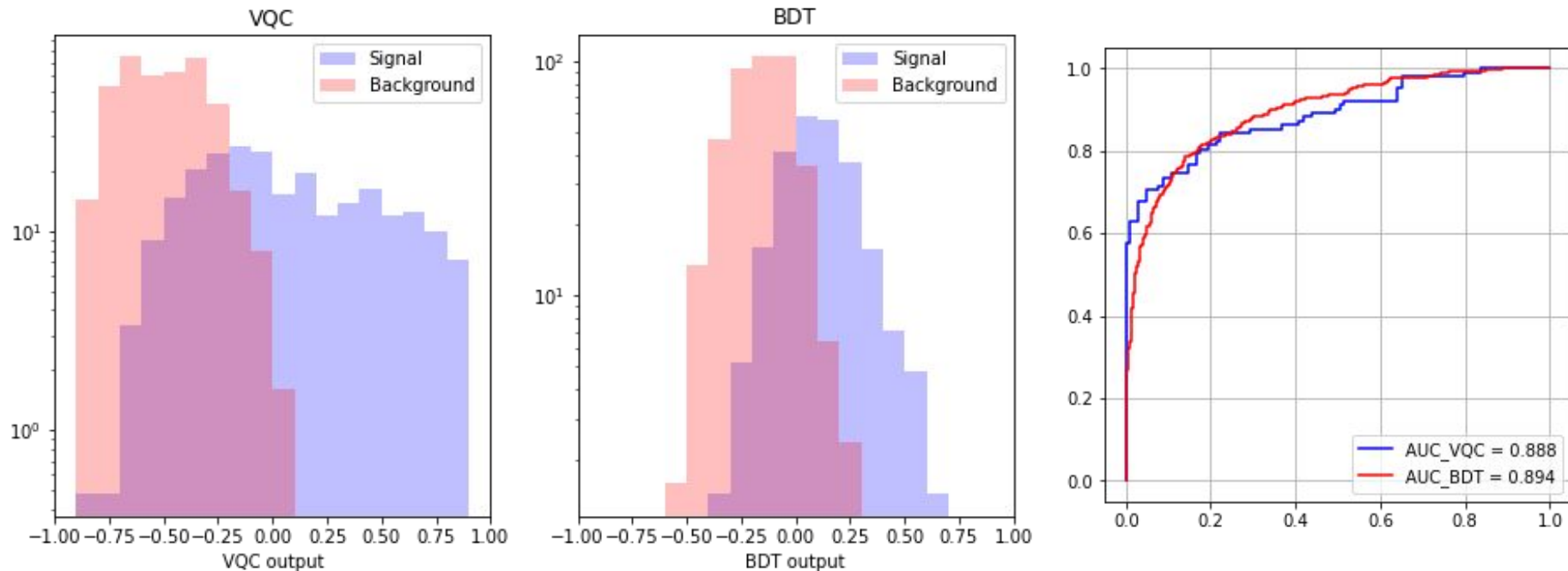
Con:

we are limited by simulating quantum computing in a personal laptop



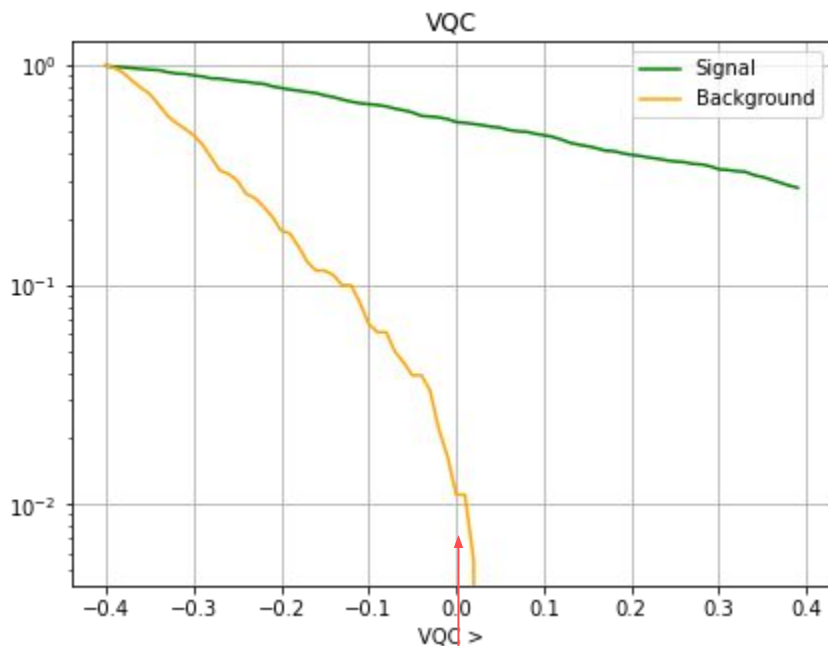
Comparing Quantum VS Classical: VQC VS BDT

Signal and Background Separation

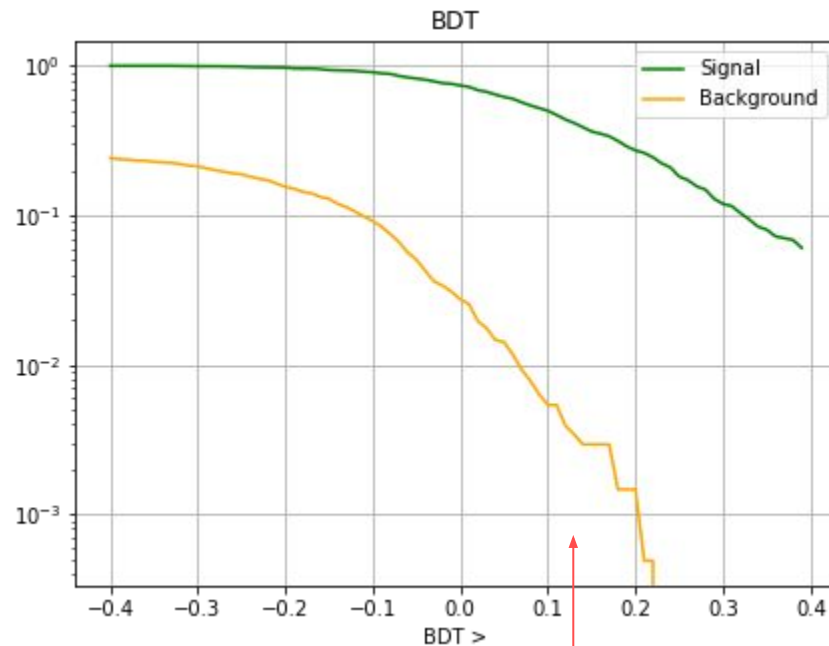


The Boosted Decision Tree (BDT) shown is the one used in my PhD analysis which is being used as the classification tool and was training using the complete dataset.

Signal and Background efficiencies



Fluctuations due to
lack of stats.



Fluctuations due to
lack of stats.

To evaluate the impact in the analysis of using a VQC for classification we use a Figure of Merit (FOM) and model scores before there's a lack of stats.

Figure of Merit

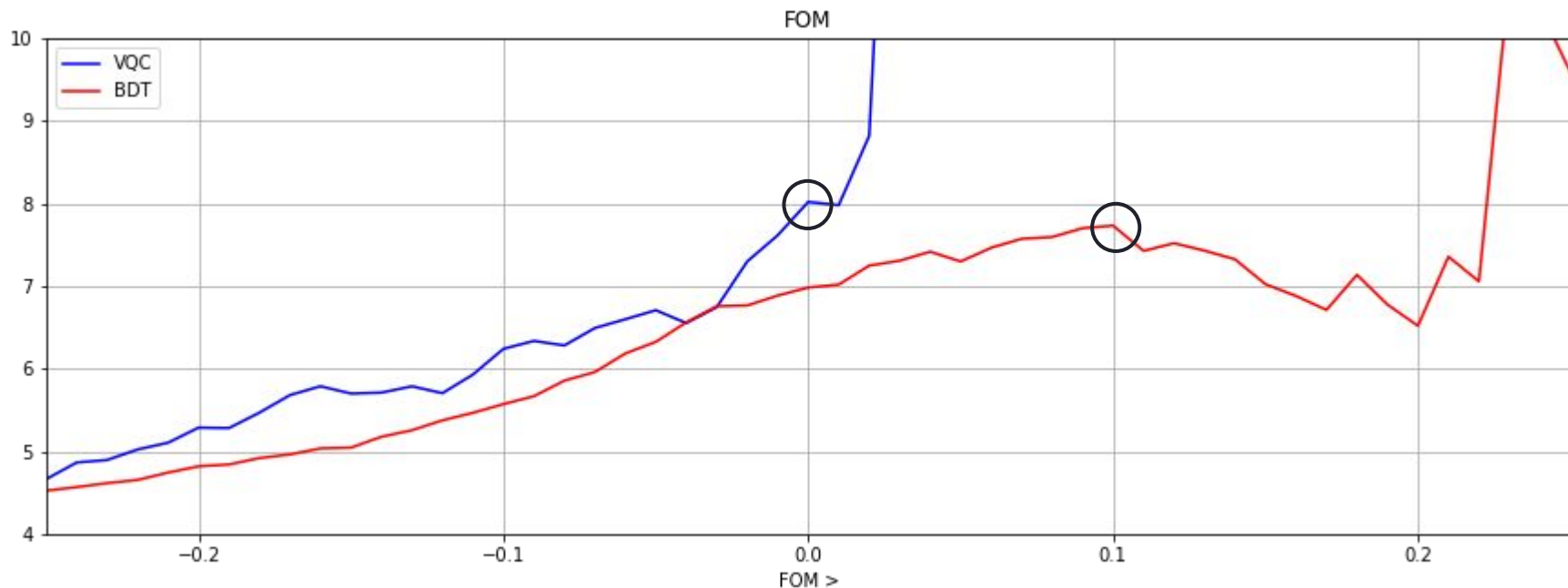
Until now, it's not obvious which algorithm performs the best and the overall impact of using it in the search for supersymmetry.

We need to use a function that takes as inputs the **signal** and **background** yield predicted by placing a selection on the output of our machine learning algorithm. In experimental particle physics we call this function a Figure of Merit (FOM):

$$FOM = \sqrt{2((S + B)\ln(\frac{(S+B) \cdot (B+\sigma_B^2)}{B^2 + (S+B) \cdot \sigma_B^2}) - \frac{B^2}{\sigma_B^2} \ln(1 + \frac{\sigma_B^2 \cdot S}{B \cdot (B+\sigma_B^2)}))}$$

We want to see how the FOM evolves as a function of the ML output for both methods and compare them.

FOM as a function of the model output



In general, the VQC scores slightly higher in FOM compared to the BDT and achieves a higher maximum before we start to run out of stats. Nonetheless, it's too soon to conclude that the model will give a substantial improvement once it's used in a real environment. But it's promising!

Conclusion and next steps

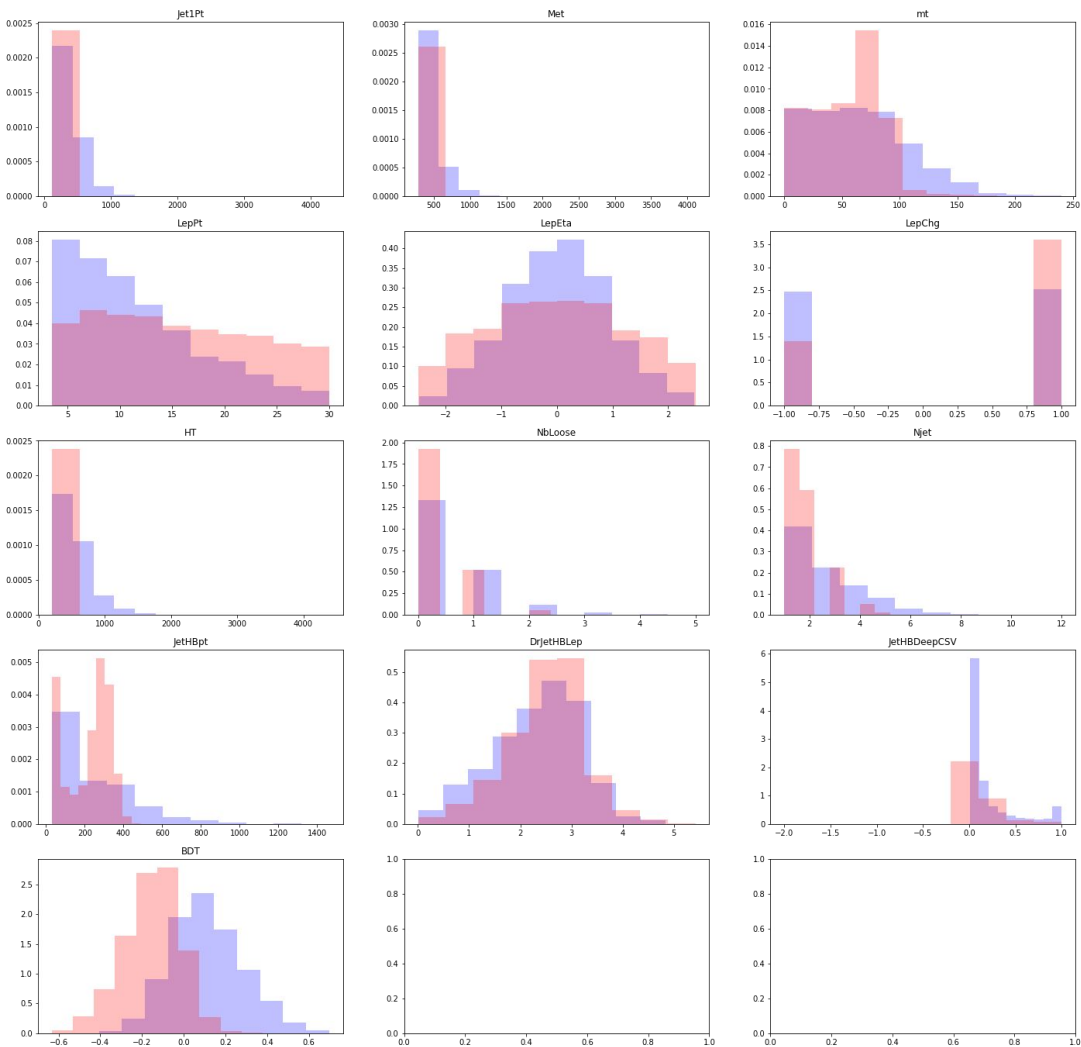
Conclusion:

- We've understood, developed and tested a quantum machine learning algorithm in the field of HEP
- VQCs can learn from a small number of samples but the algorithm is limited to be run on a personal computer
- Nonetheless, these results are promising enough to motivate the same study in a real quantum computer

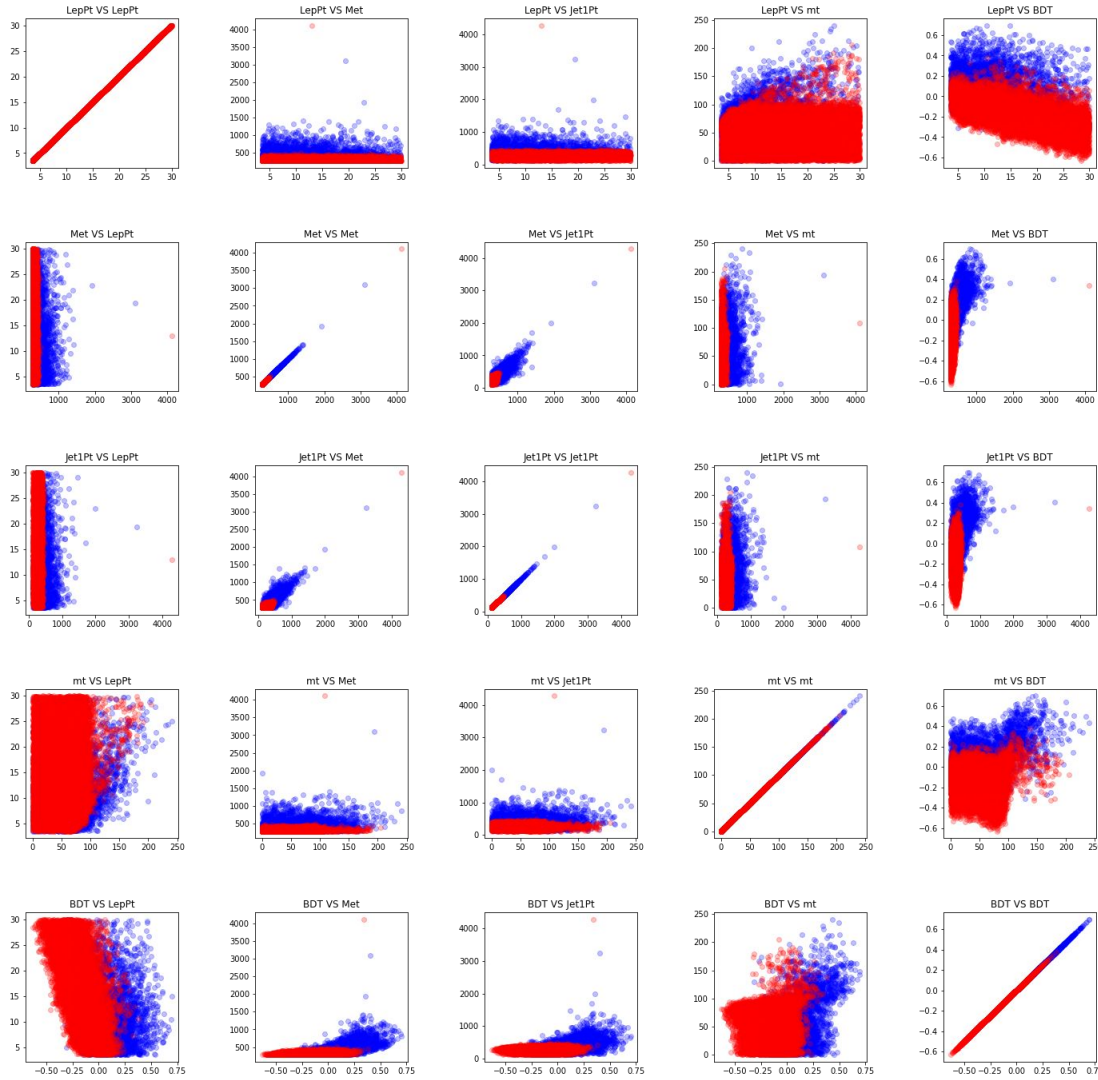
Next steps:

- Contact CERN QTI via LIP-CMS to get access to a real quantum computer in research applications.
IBM, XANADU or IONQ have such programs and have worked with CERN
- Publish the resulting benchmarks

Backup



Discriminant Variables



2D plots