Preliminary Development on HEP Data Analysis Using Quantum Computing based on IBM Qiskit (progress report)

Wen Guan, Shaojun Sun, Alex Wang, Sau Lan Wu, Chen Zhou
University of Wisconsin-Madison
and
Federico Carminati
Chief Innovation Officer, CERN Openlab

November 5-6, 2018
Quantum computing for HEP workshop

Machine learning and quantum computing

- Machine Learning has become one of the most popular and powerful techniques and tools for HEP data analysis
- **Machine Learning: This is the field that gives computers "the ability to learn without explicitly programming them".**
- Issues raised by ML
  - Heavy CPU time is needed to train complex models
    - With the size of more data, the training time increases very quickly
  - May lead to local optimization, instead of global optimization
- Quantum computing
  - **Can speed up certain types of problems effectively**
  - **It is possible that quantum computing can find a different, and perhaps better, way to achieve global optimization.**

Ref: "Global Optimization Inspired by Quantum Physics", 10.1007/978-3-642-38703-6_41

Our program with IBM Qiskit

Our Goal:

Perform High Energy Physics analysis with Quantum computing

Our preliminary program can be divided into two Parts with the Environment of IBM Qiskit:

Part 1. Evaluation of the time consumption of IBM Qiskit backends.

Part 2. Employing SVM Quantum Kernel (QSVM) method for High Energy Physics (HEP) analysis, for example ttH ($H \rightarrow \gamma\gamma$), Higgs coupling to two top quarks analysis.

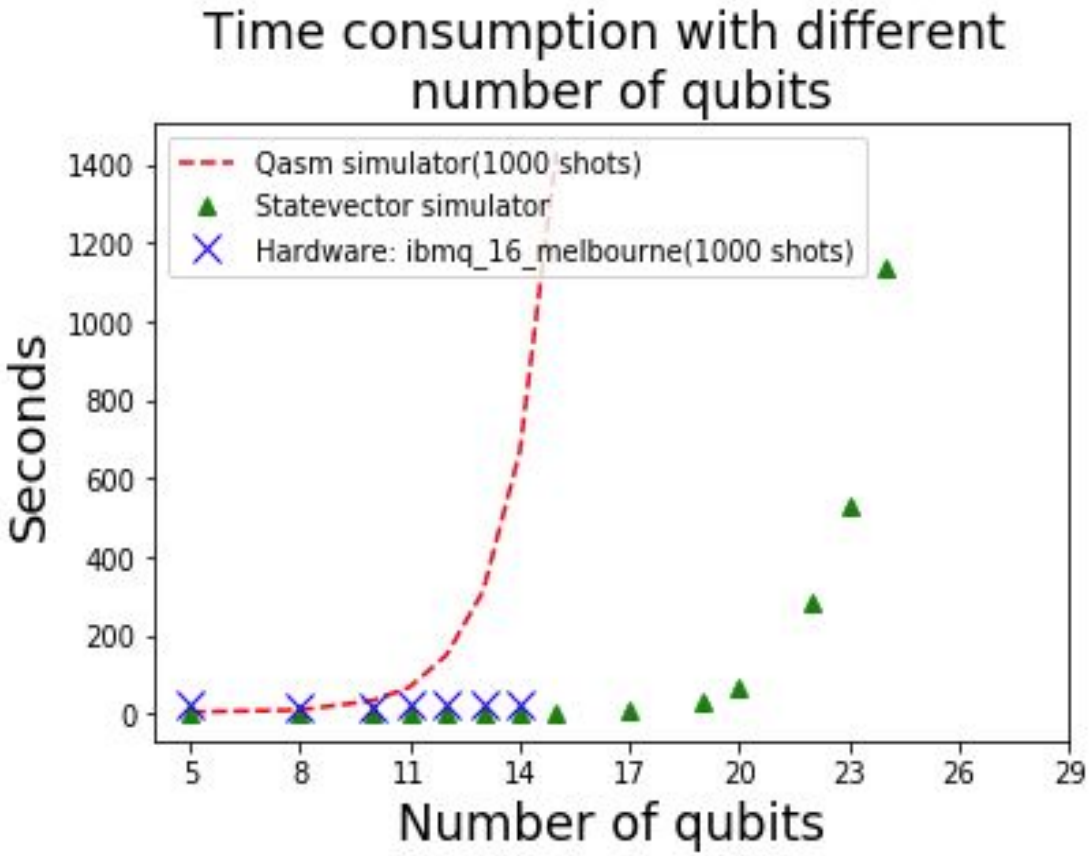
*** SVM = Support Vector Machine**

Part 1: Evaluation of Time Consumption IBM Qiskit backends

- IBM Qiskit supports several different backends, here we evaluate two simulators and one IBM Q hardware
  - **Qasm simulator**: quantum assembly language simulator
    - Expected to be more similar to hardware
  - **Statevector simulator**
    - Expected to be faster
  - IBM Q hardware: **Ibmq_16_melbourne,** which supports only 14 qubits

Ref: Ryan LaRose, “Overview and Comparison of Gate Level Quantum Software Platform”, 2018

Part 1: Time Consumption of backends



- Hardware has a limited number of qubits; need to test with more qubits

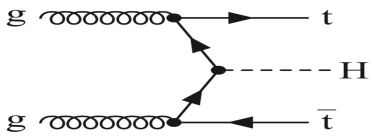
- Test time consumption on different backends with different numbers of qubits to calculate an inner product of two vectors
  - Time consumption increases exponentially on simulators
    - Statevector is faster
  - With present available qubits, time consumption on hardware remains constant.

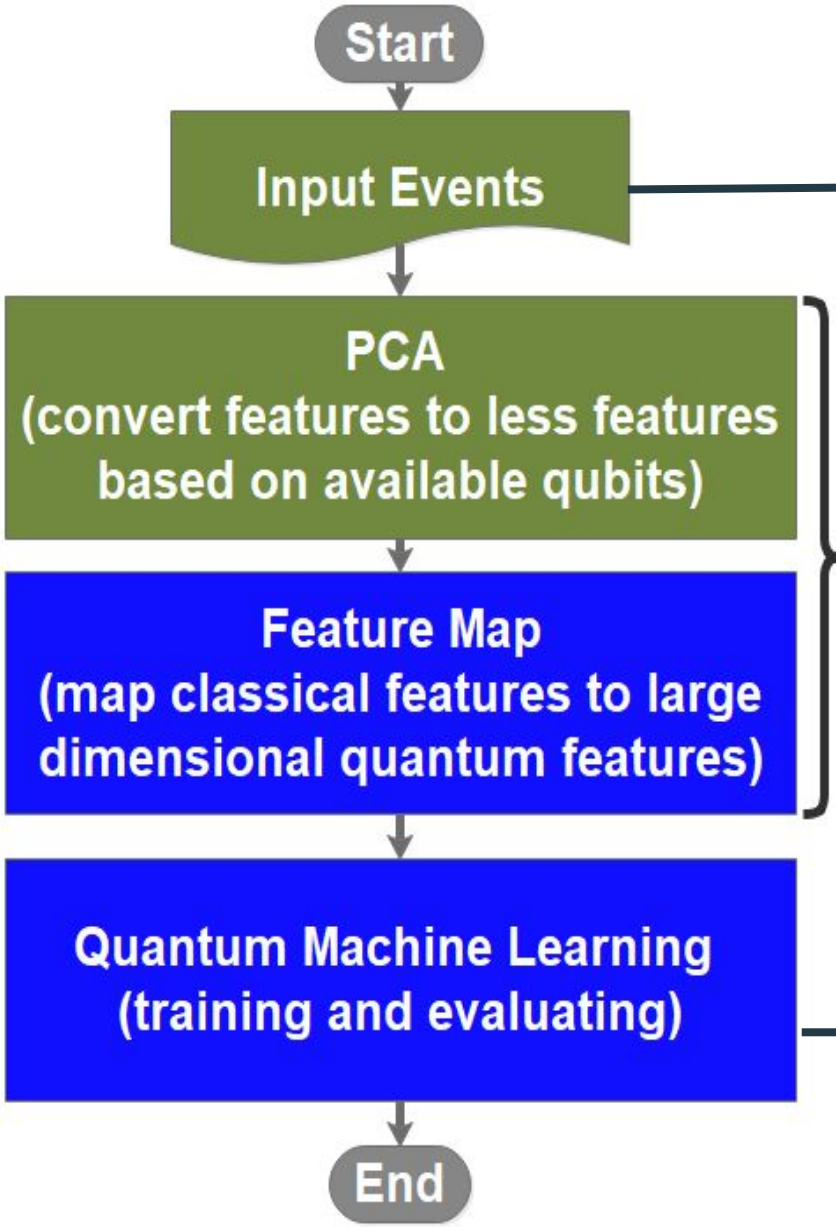
Part 2: Employing Quantum ML for HEP analysis

- **Employing SVM Quantum kernel for HEP analysis**
  - **For example, ttH ($H \rightarrow \gamma\gamma$), Higgs coupling to two top quarks analysis**
  - **Exploring different feature map methods**
  - **Training and evaluating quantum ML methods with different numbers of qubits and events**

*** SVM = Support Vector Machine**

Part 2: Our Workflow for Quantum Machine Learning process



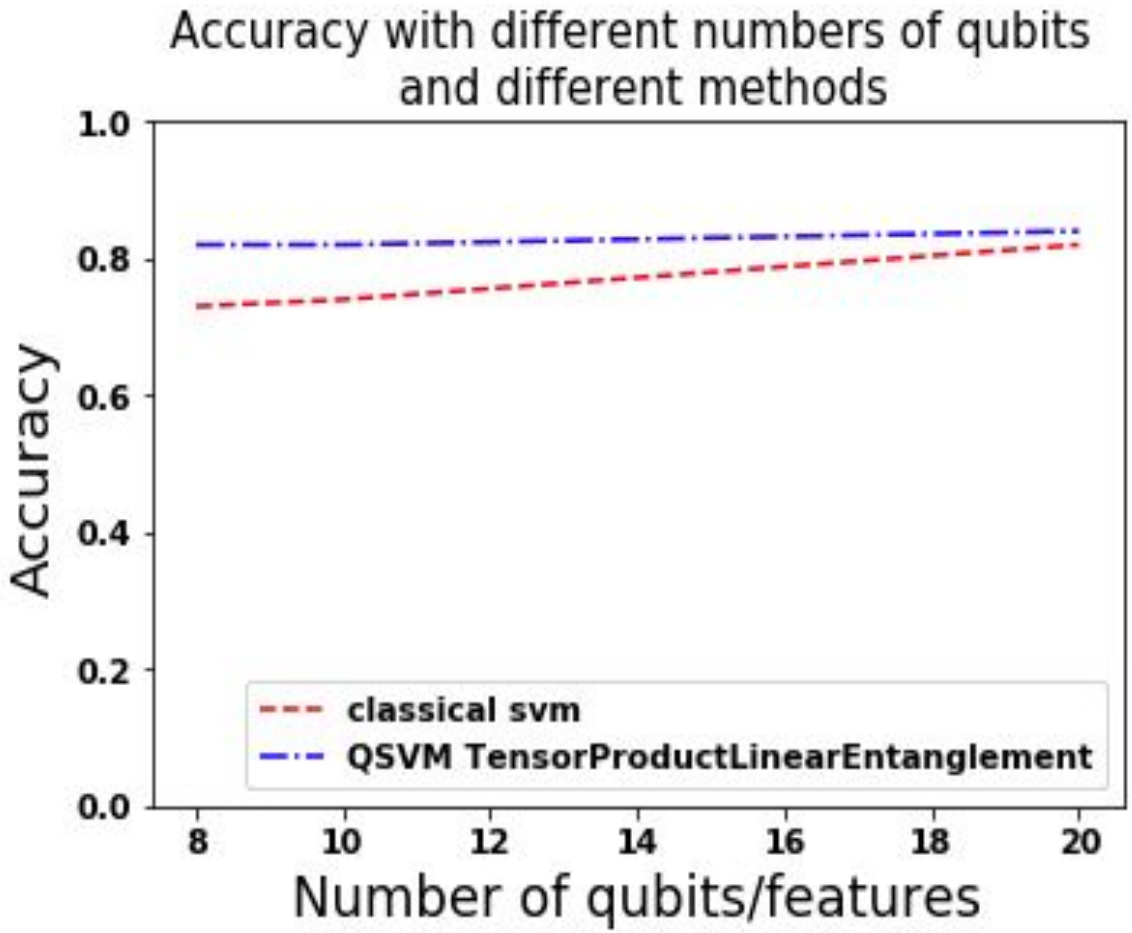


ttH: number of features(variables) = 45

- FeatureMap: Each feature(variable) of input event is encoded in the amplitude of one separate qubit, but we have much more features for an event than available qubits
- (Number of qubits = 8, 10, 20 for example)
- PCA: Principal Component Analysis method is used to convert/combine features to less features to be able to be encoded into quantum system.

Support Vector Machine (SVM) quantum kernel, for example

Part 2: Accuracy with QSVM for ttH HEP analysis



- **QSVM analysis is Simulated with IBM Qiskit statevector simulator**
- **QSVM Tensor Product feature map with Linear Entanglement gives a slight better accuracy over classical SVM method**
  - **Entanglement encodes relationships between features.**

$$Accuracy = \frac{\text{Number of correct prediction}}{\text{Total number of predictions}}$$

Part 2: Accuracy with QSVM for ttH HEP analysis

- **Problem with this preliminary study**
  - Our input data has 45 features (variables) per event.
  - With PCA to convert to less features(8, 10 or 20), we are losing a lot of event information because of limited number of qubits.
  - Training statistics is very poor(200 training events and 100 testing events for current study).

Part 2: Employing QSVM for ttH HEP analysis

Current problems :

1. Hardware: IBM Q hardware has a payload size limitation, and therefore, we cannot process enough events on the hardware machine.
2. Simulator: We don't have enough computing resources to run the full training with IBM Q simulators. We only run with limited number of events (200 training events and 100 testing events for current study) and a limited number of qubits.
   a. The number of kernels is $O(m^2)$, where m is the number of events.
   b. With more qubits, CPU time and memory consumption increase exponentially $O(2^n)$, where n is the number of qubits.
   c. After some basic circuit simulation tests, we found a huge amount of CPU time is required for a full training .

Next Steps

- **Distribute the training & testing to a cluster of computers or HPC when using simulators.**
- **Feature map**
  - **The way to convert classical information to quantum system plays an important role on the performance of quantum ML**
  - **Will work to explore more feature map methods and algorithms**
- **Quantum ML algorithms**
  - **Different quantum ML algorithms may get different performance**
  - **Currently we only evaluated QSVM.Kernel and we are working on another QSVM method (QSVM.Variantional).**
  - **We will also look to explore more Quantum machine learning methods**

Limitations for near future

- **Hardware**
  - Limited number of qubits and limited access
- **Simulators**
  - CPU time and memory consumption increase exponentially with the number of qubits
    - 17GB (GigaByte) memory for 30 qubits; 34GB for 31 qubits
    - With a full entanglement feature map to train 200 events with 8 qubits, 47GB memory consumed
- **Algorithm complexity**
  - For SVM method, the number of "kernels" to be calculated is $O(m^2)$, where m is the number of events. But for HEP, frequently we have a lot of events.

Summary

Referring to Part 1 of this presentation:

- **Using IBM Qiskit, we have successfully evaluated the time consumption on IBM hardware and simulators as a function of number of qubits.**

Summary

Referring to Part 2 of this presentation:

- Using IBM Qiskit simulator, we have employed SVM Quantum Kernel method for ttH High Energy Physics analysis. We have measured the accuracy of the result as a function of qubits.
- Again, the accuracy is limited by the number of qubits and the number of events. With the simulator, using more than 20 or 30 qubits will run into severe problem with memory and CPU time.

Our goal:

Perform High Energy Physics analysis using Quantum Computing. We shall take one LHC physics analysis as an example to eventually succeed in performing the analysis with Quantum computing.

IBM, Google, ……, please give us more qubits and more access time! We can make progress fast.

BACKUP SLIDES

Quantum measurement

- Quantum state is a superposition which contains the probabilities of possible positions.
- When the final state is measured, they will only be found in one of the possible positions.
  - The quantum state 'collapses' to a classical state as a result of making the measurement.
- "No-cloning theorem"
  - Impossible to create an identical copy of an arbitrary unknown quantum state.
- To obtain the probability of a possible position, some number of shots are needed.

Hardware Information

- **Hardware status currently**
  - **Classical computer:**
    - **3~4 GHz**
    - **Millions of circuits with many cores, GPU can have thousands of cores**
  - **Quantum computer**
    - **200 ns per operation**
    - **5M Hz**
    - **Not many parallel channels or threads**
    - https://quantumcomputing.stackexchange.com/questions/2402/how-many-operations-can-a-quantum-computer-perform-per-second

How to use quantum computer

- How to use quantum computers
  a. Convert classical features to be able to be processed to quantum computers
     - Feature map
  b. Using quantum algorithms to process the data
     - Algorithms developed based on quantum computers, such as Quantum Support Vector Machine, Quantum annealing, Grover Search and so on

Tensor product feature map

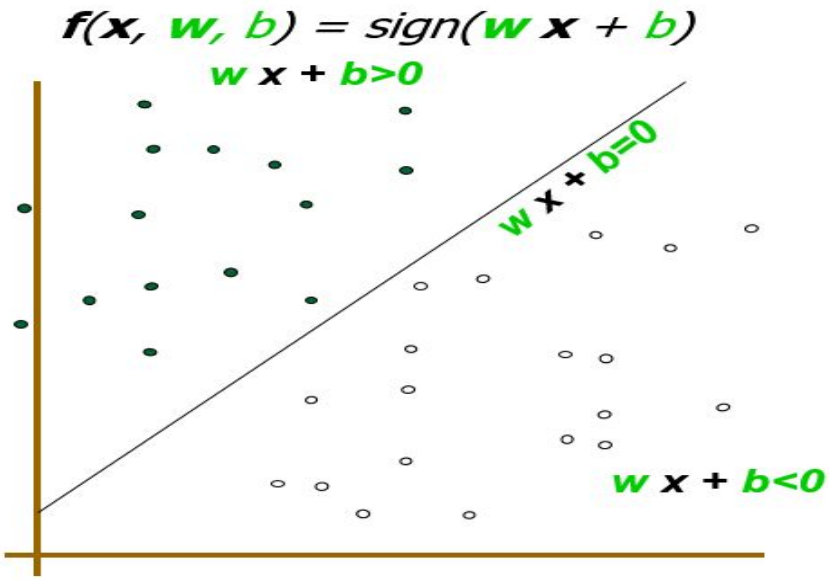
- Quantum feature map: Map bit info non-linearly to quantum 'feature Hilbert space'
  - Tensor product encoding
    - Each feature(variable) of input event is encoded in the amplitude of one separate qubit
    - All features of one event is the tensor product of corresponding qubits
  - Entanglement between features
    - Without entanglement
    - Between next one feature(linear entanglement)
    - Between all of the next features(full entanglement)

Other feature map methods

- Basic encoding
  - One bit maps to one qubit
  - For example, two bits "01" maps to two qubits "$|01\rangle$"
- Amplitude encoding
  - N classical features maps to $\log_2 N$ qubits
  - $X = (x_0, \ldots, x_{N-1})$, $N=2^n$
  - $|\varphi_x\rangle = \Sigma\ X_i * |i\rangle$ (qubit "$|i\rangle$" is the i'th computational basis state)
  - Looking whether it's possible and how to do it

Support Vector Machine

- **Support Vector Machine (SVM)**
  - a supervised ML that draws a decision boundary between two classes to classify data points
  - Originally it's constructed as a linear classifier
  - Maximize the distance from the line or hyperplane to the nearest data point on each side

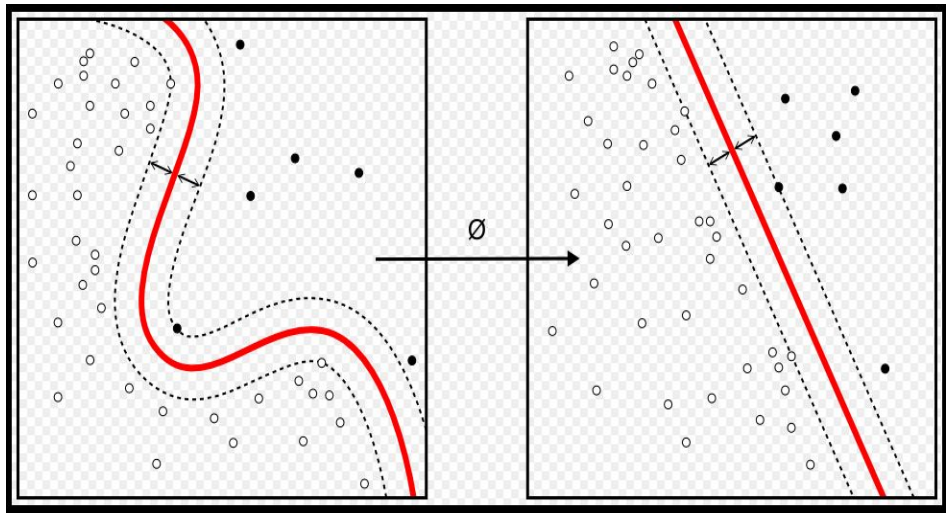


Ref: Support Vector Machine and Its Application(Mingyue Tan, 2004)

Ref: Support vector machine(Wikipedia)

SVM kernel function

- Kernel function
  - Often the sets of data points are not linearly separable
  - Map data points to a much higher dimensional space which presumably making the separation easier



Ref: Support vector machine(Wikipedia)

- Performance depends on different kernel functions
- Limitation to successful solutions when feature space becomes large
- Computationally expensive to estimate the kernel

Quantum SVM

- Quantum SVM
  - Take advantage of the large dimensionality of quantum Hilbert space
    - Non-linearly maps input data into a very large dimensional feature Hilbert space
    - Exploiting an exponentially large quantum state space
  - Take advantage of the quantum speedup
  - Estimate the kernel and optimize the classifier

Backup for Backends evaluation(1)

- Simulator tested on a 6 Core 3.7GHz machine
- It includes the software setup time, so for statevector simulator, values between 15 qubits are not exactly the time used by simulating
- For ibmq hardware, the time is the value returned from backend result
- For all these tests, it's using FirstOrderExpansion(Tensor product without entanglement) featuremap; with SecondOrderExpansion(Tensor product with entanglement) feature map to encode more info, the simulator will be even much more slower

Backup for Backends evaluation(2)

- Statevector simulator is using a very different way other than quantum hardware. It's based on state vector. It doesn't need measurement and more than 1 shot.
- Qasm simulator and ibmq_16_melbourne are more similar, good to compare them

qubits	5	8	10	11	12	13	14	15	17	19	20	22	23	24
Qasm simulator (1000 shots)	5.0s	10.2s	34s	69s	150s	313s	668s	1431s			1385 m			
Statevector simulator (1shot)	3.5s	4.0s	4.4s	4.3s	4.3s	4.7s	4.6s	6.3s	11s	34s	65s	287s	528s	1139 s
Ibmq_16_melbourne(1000 shots)	18s	19s	23s	18s	22s	19s	19s	Not suppo rt						

Employ QSVM for HEP analysis

- Note:
  - For classical SVM, no qubits, it means N attributes per event
  - Our input data has 45 attributes. PCA(Principal Component Analysis) method is used to convert 45 attributes to 8, 10 or 20 qubits or attributes. This operation loses a lot of information. So we more qubits, we should get even better result.
  - For QSVM, the results are simulated with Statevector simulator
  - FirstOrderExpansion(Tensor product without entanglement) featuremap
  - SecondOrderExpansion(Tensor product with entanglement) feature map: still waiting results

- classical svm
- Quantum: SecondOrderExpansion featuremap with linear entanglement, depth=2(default)

	8 qubits/attributes	10 qubits/attributes	20 qubits/attributes
Train:200, test:100	0.73,0.82	0.74,0.82	0.82,0.84

Backup: ibmq_16_melbourne

- Why not finish some QSVM training on it?
  - To finish a training, the number of "kernels" to be calculated is $O(m^2)$, where m is the number of events
  - IBMQ system is just a test bed, it has a payload size limitation; So a training will be split to many many small jobs
  - Submitting a job to IBMQ system, the queue time sometimes/frequently can be many hours
    - Current backend submits jobs to IBMQ system one after another
    - No idea whether one user can queue a lot of jobs there, being a good user I didn't test
  - The total time to finish a training with enough data will be very very long if using IBMQ hardware

Hardware errors and precision

- Error is one part needing evaluating.
- Precision is another part needing to check
  - Input: Converting classical info to quantum info
    - Easy to convert 0, 1, 0.5 and so on to quantum system
    - What about 0.000005? Will only 0 be converted to quantum system?
  - Quantum hardware has error correction solutions to correct errors?
    - But quantum is not fully clonable
    - What's the precision of these error correction solution
  - What's the precision with more and more gate operations
    - More operations can increase the errors
  - Output: Measurement precision
    - When measuring a quantum will collapse to a classical state, so many shots are needed than we get the probability by counting different states
    - 1000 shots can get precision no better than 0.001