High Energy Physics and Quantum Computing

2024 IonQ 5th week meeting

Weekly goal

[Week 1] Review the reference [1] thoroughly and understand basic idea.

[Week 2~3] Implement the idea using Qiskit and/or **PennyLane**, and reproduce their results for two examples in the paper.

[Week 4] Then consider a more complex and more realistic example, taking electron-positron production at the Large Hadron Collider ($pp \rightarrow e^+e^-$). Effectively, this problem involves four-dimensional integration.

[Week 5] Try to optimize the circuits with different choice of the cost function or variations of quantum circuits.

[Week ?] Study Monte-Carlo sampling with above integration method, and how to implement the importance sampling into a variational quantum circuit.

Integration method

4. Calculating the double Riemann sum:
Multiply the function's value at each representative point and then sum over all intervals for x_2 and x_3 to calculate the double Riemann sum.

$$I(x_1) = \sum_{i=1}^{N} \sum_{j=1}^{N} \boxed{g_{est}(x_1, x_{2i}, x_{3j}, \alpha_0)} \Delta x^2$$

<u>Key point!!!</u>

- Need good approximation to the integrand

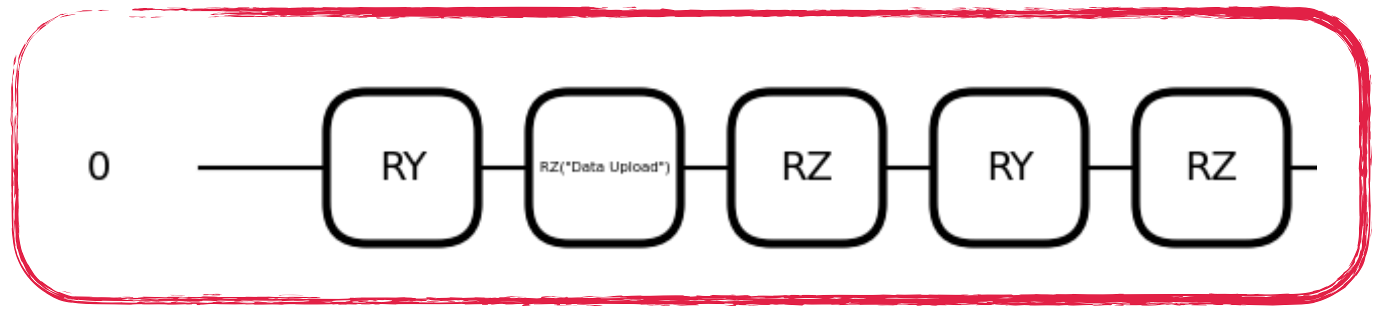
Before solving the realistic example

Considering 1-dim problem

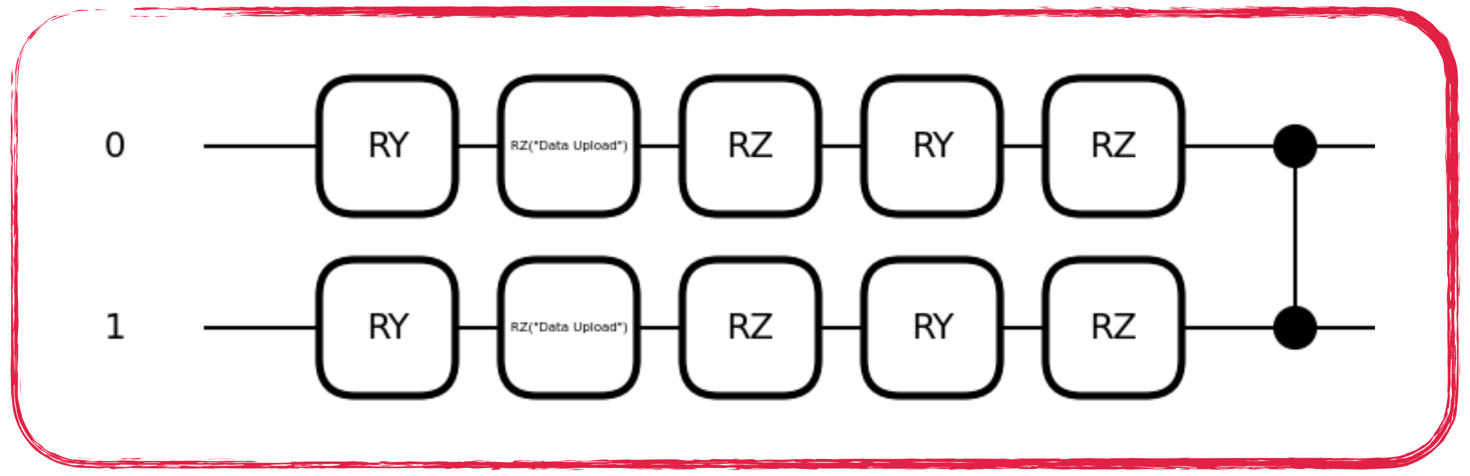
Approximating $f(x) = x$

Training data : 50 points
Learning rate : 0.05

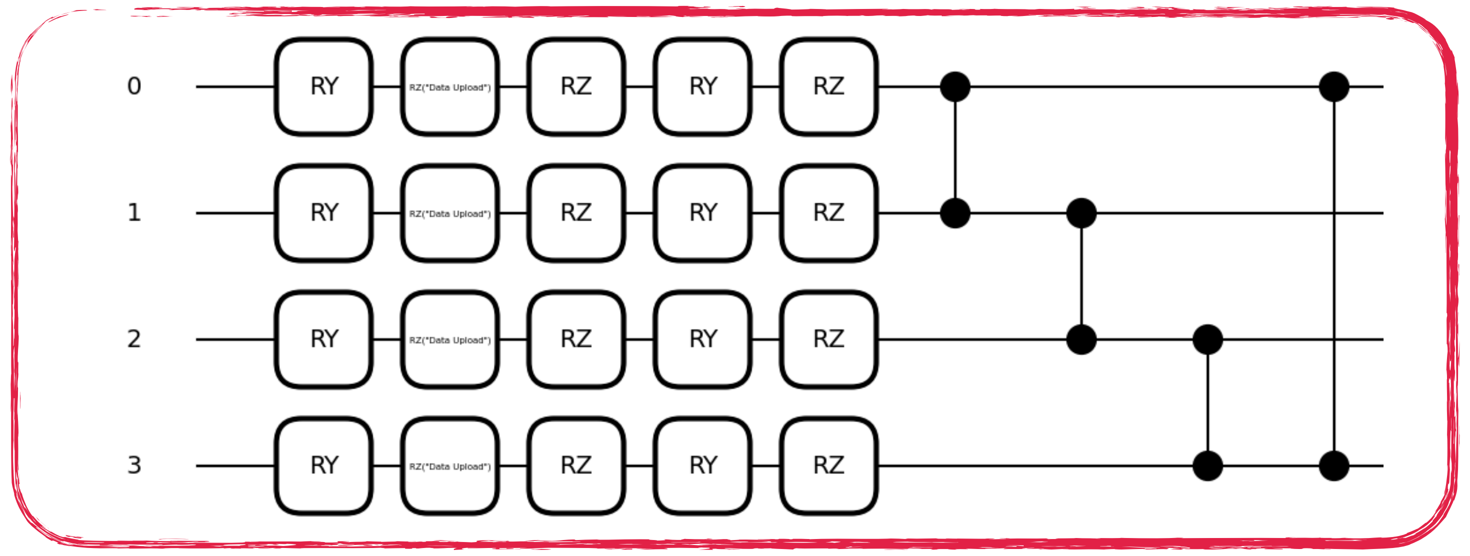
With 8 layers, 41 parameters

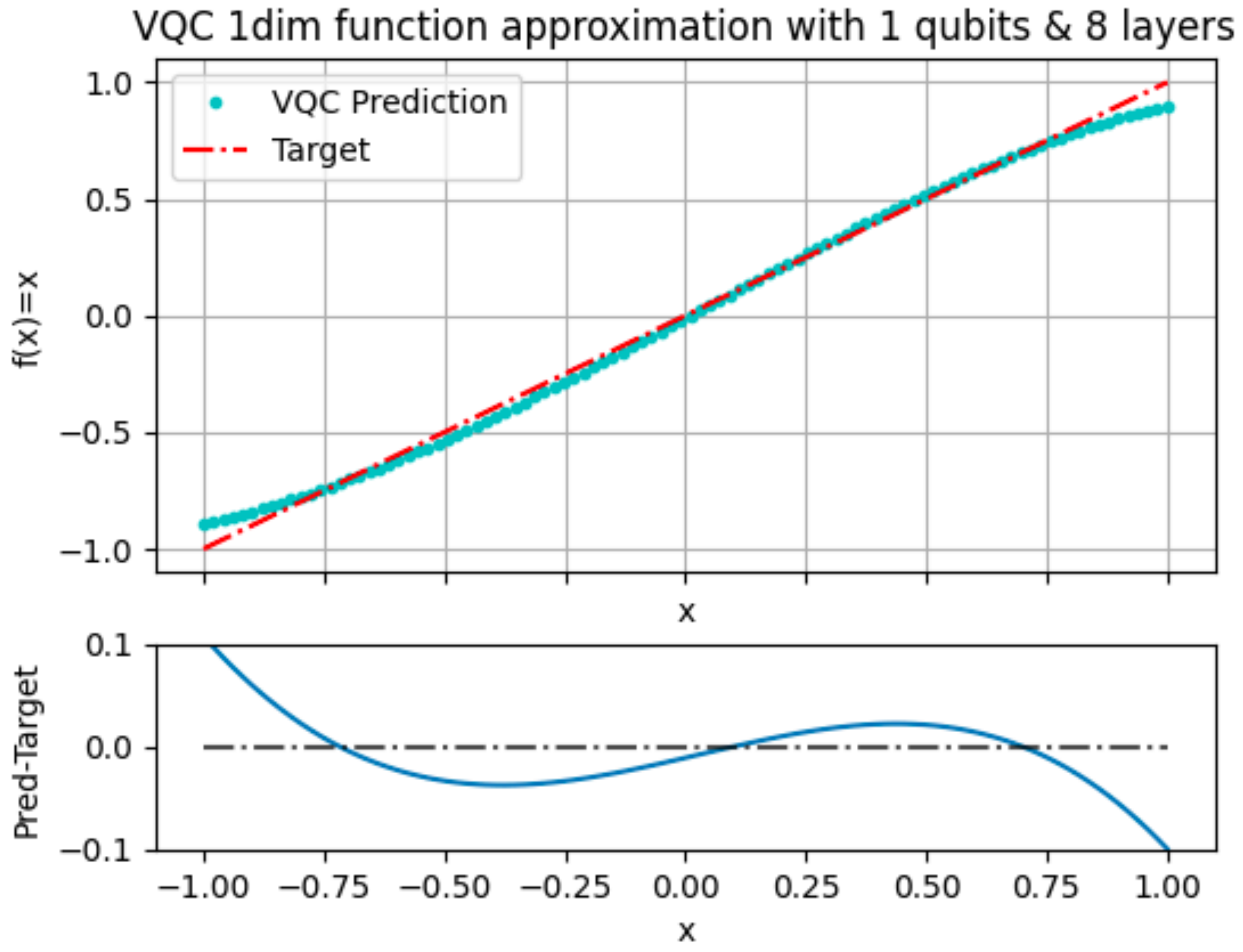


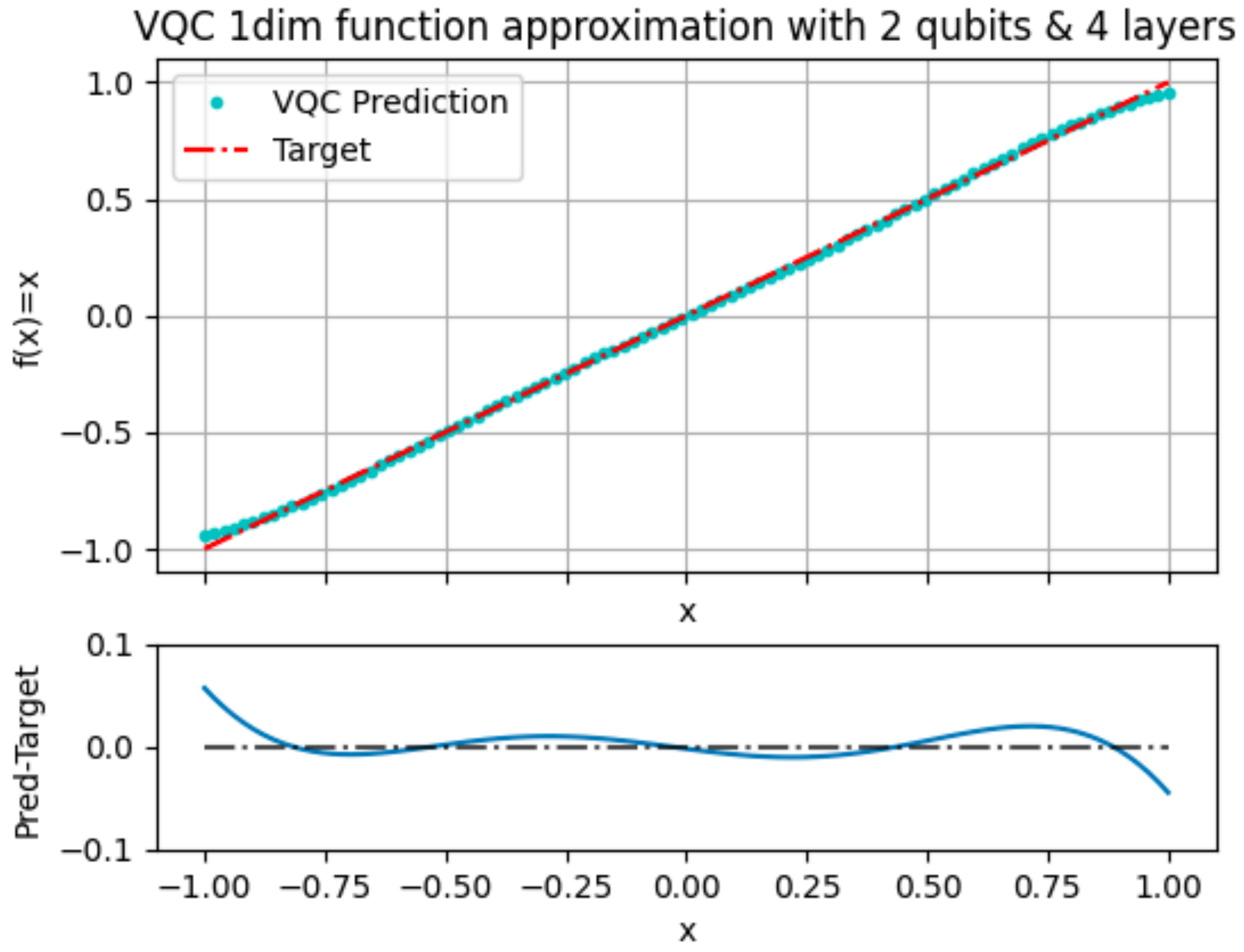
With 4 layers, 42 parameters

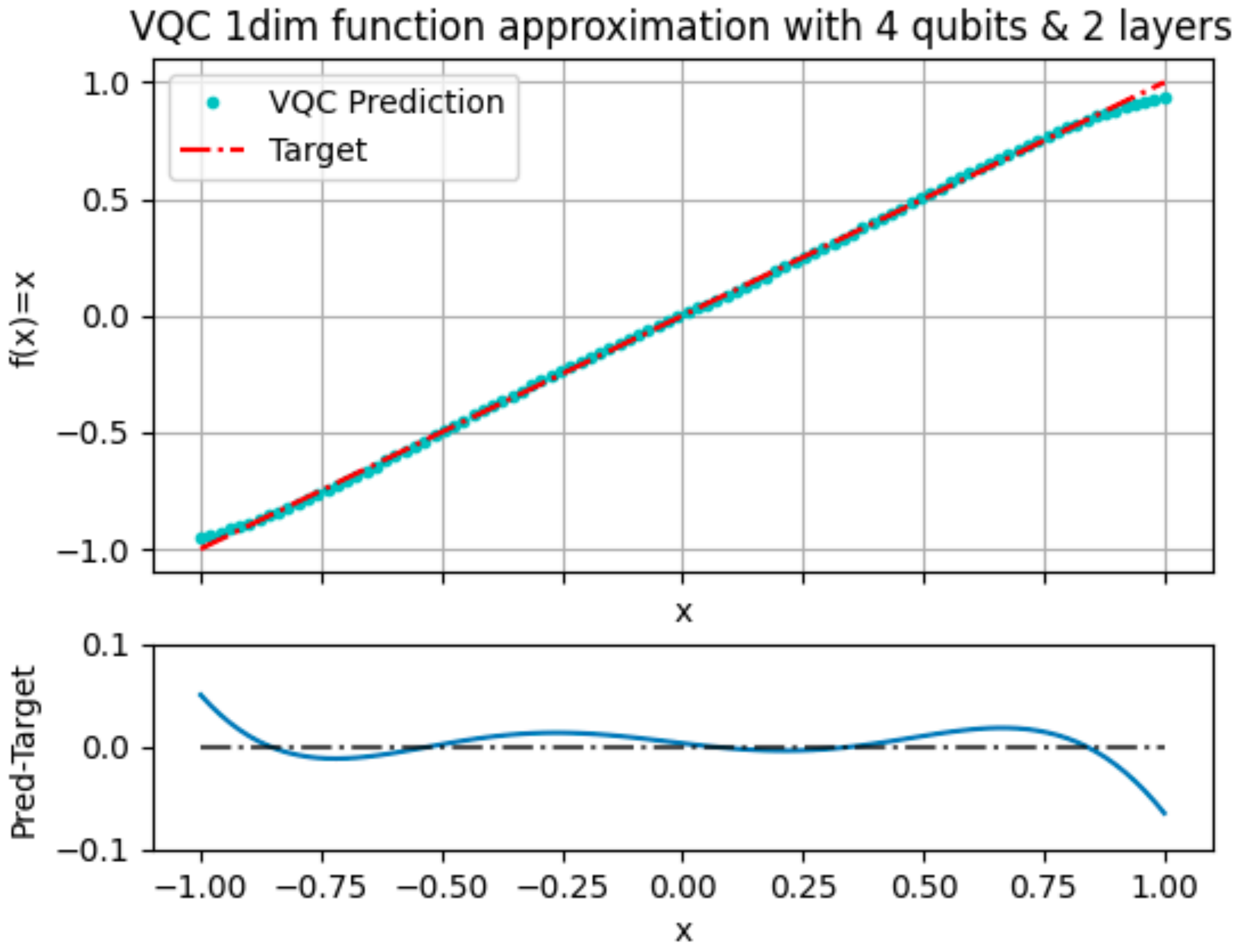


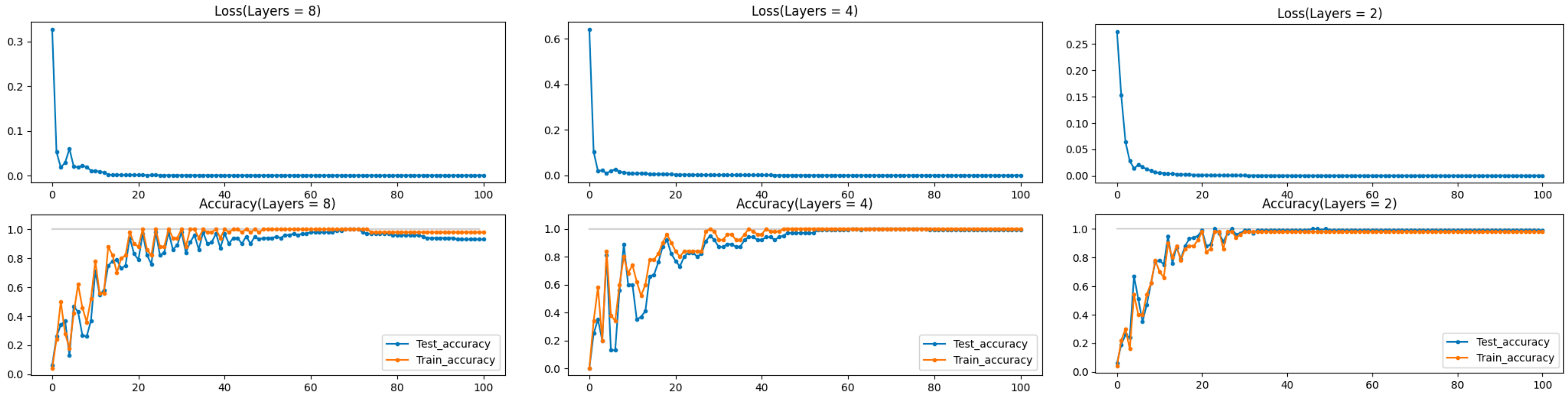
With 2 layers, 44 parameters











Loss(Layers = 8)
Loss(Layers = 4)
Loss(Layers = 2)
Accuracy(Layers = 8)
Accuracy(Layers = 4)
Accuracy(Layers = 2)
Test_accuracy
Train_accuracy

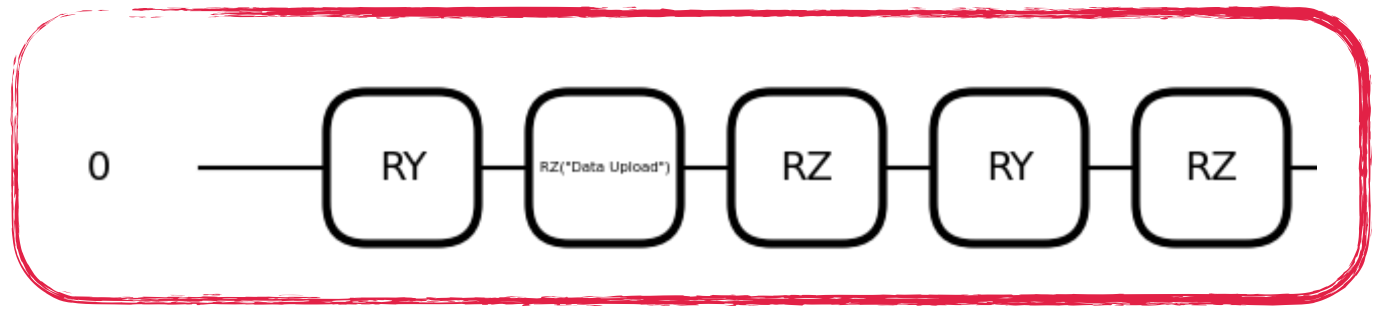
Before solving the realistic example

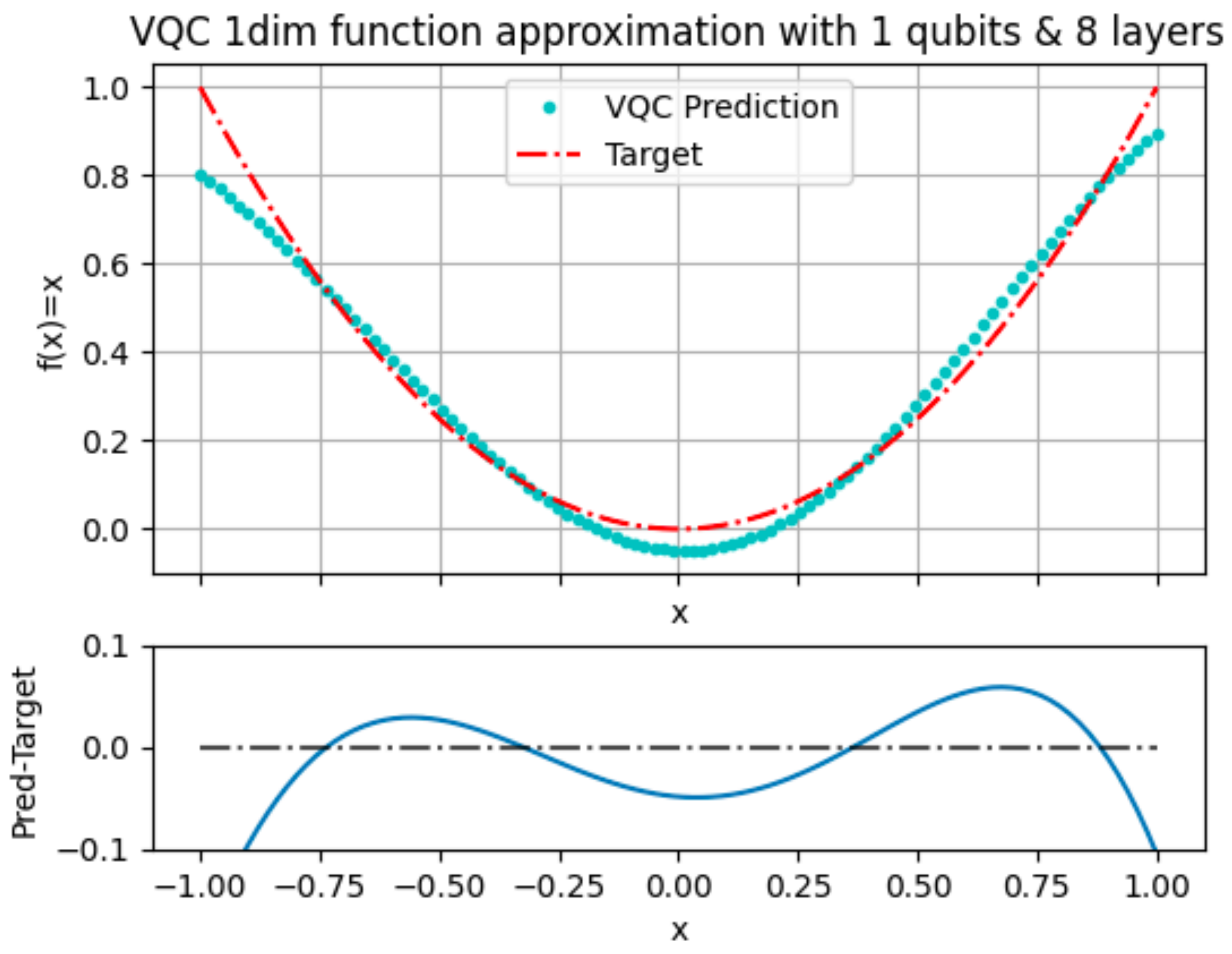
Considering 1-dim problem

Approximating $f(x) = x^2$

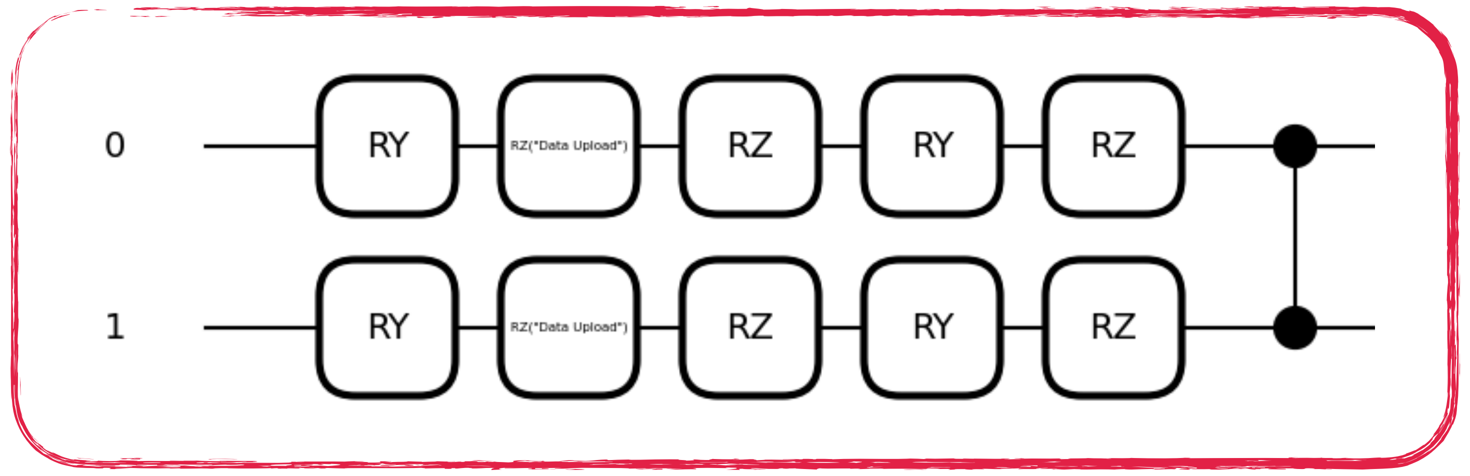
Training data : 50 points
Learning rate : 0.007

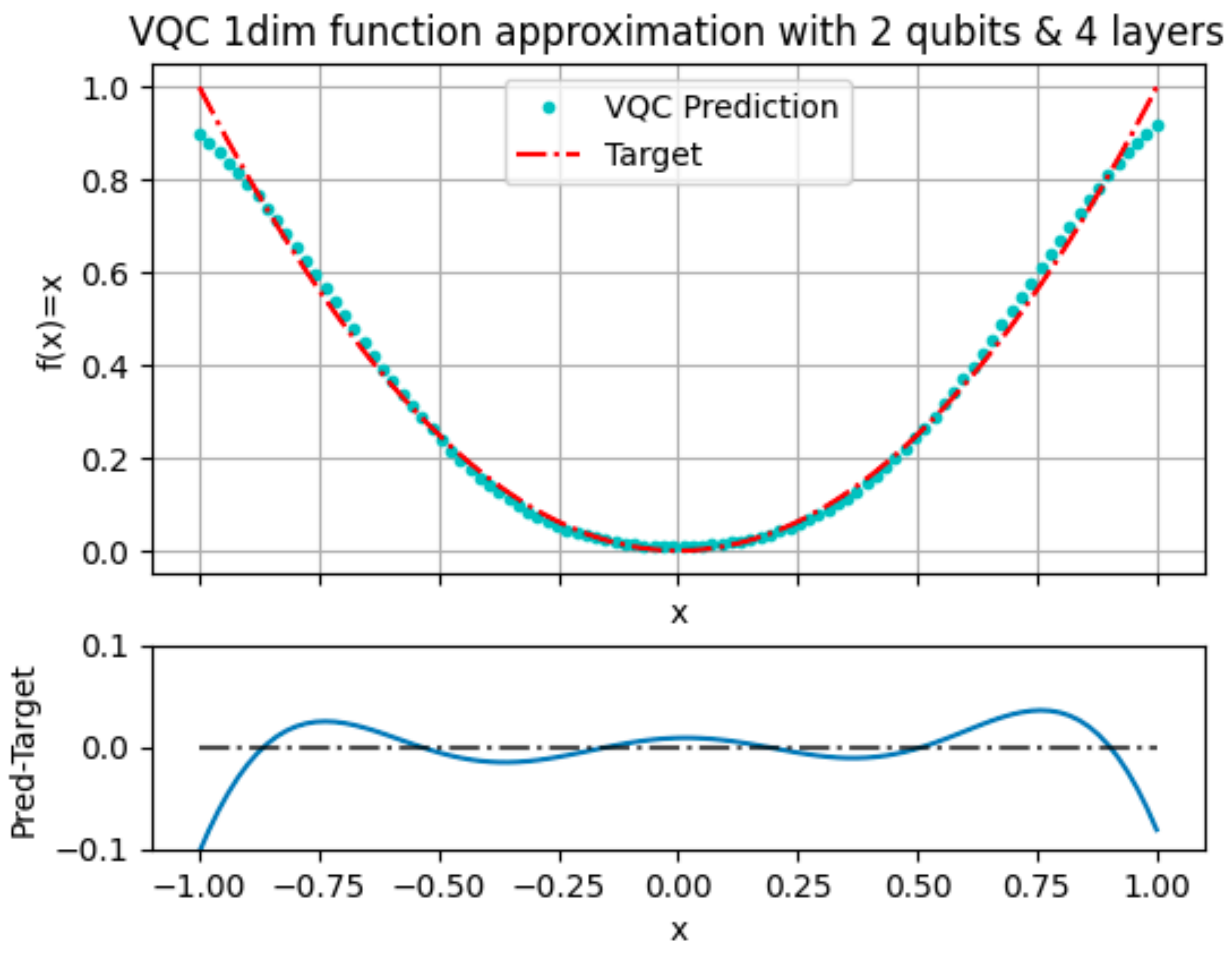
With 8 layers, 41 parameters



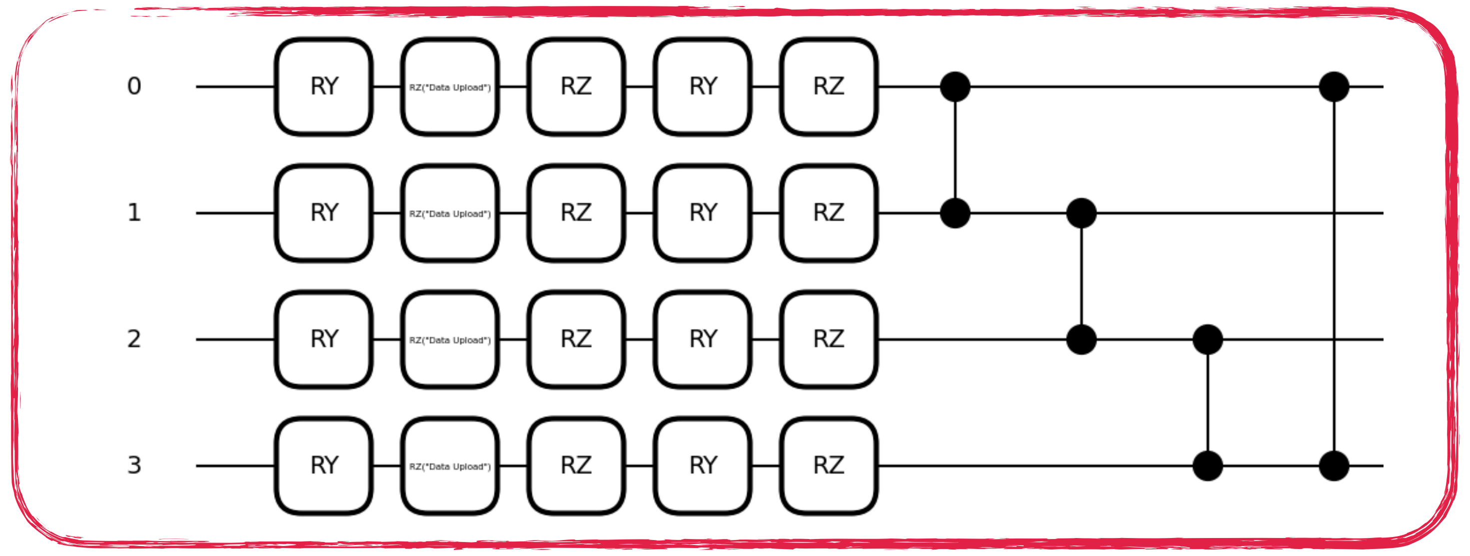


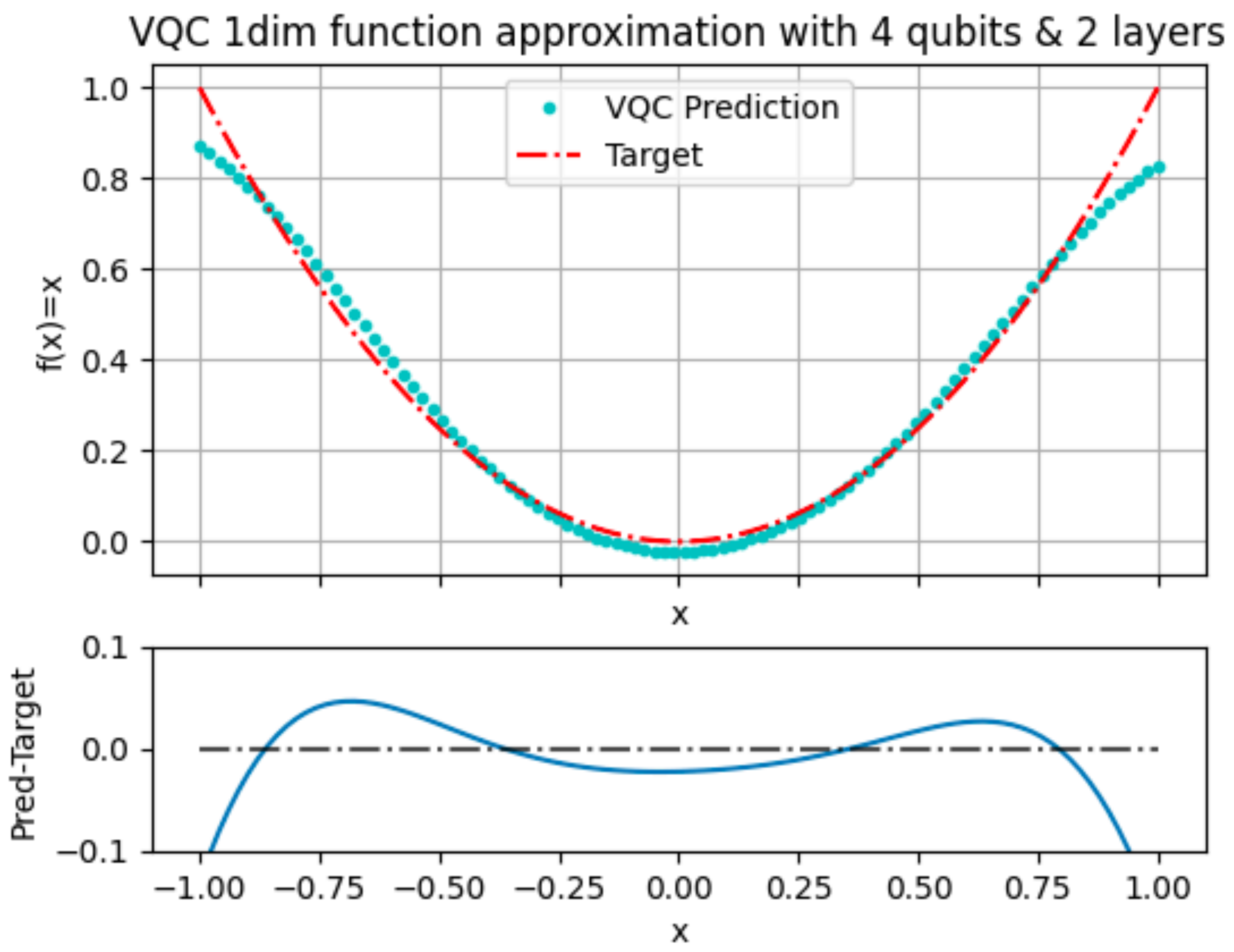
With 4 layers, 42 parameters





With 2 layers, 44 parameters





Predictions with random parameters
Predictions after training
True test data
f(x)
x
Loss(Layers = 8)
Loss(Layers = 4)
Loss(Layers = 2)
Accuracy(Layers = 8)
Accuracy(Layers = 4)
Accuracy(Layers = 2)
Test_accuracy
Train_accuracy

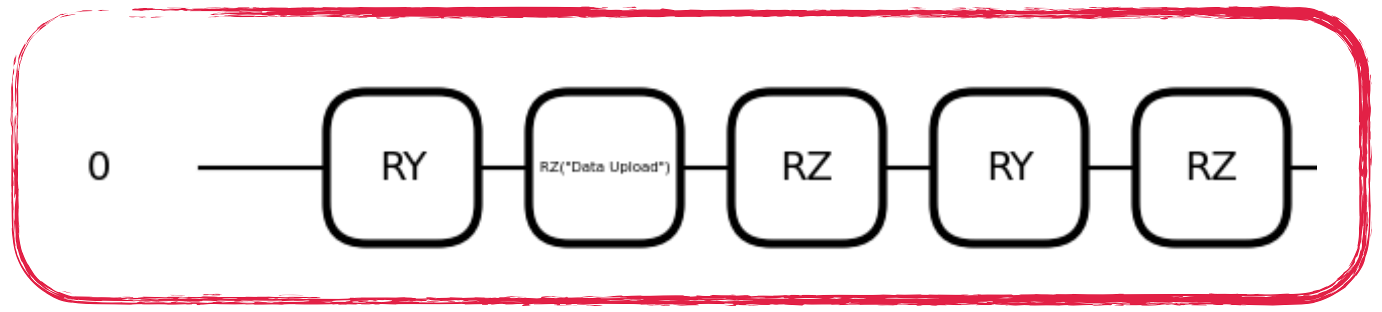
Before solving the realistic example

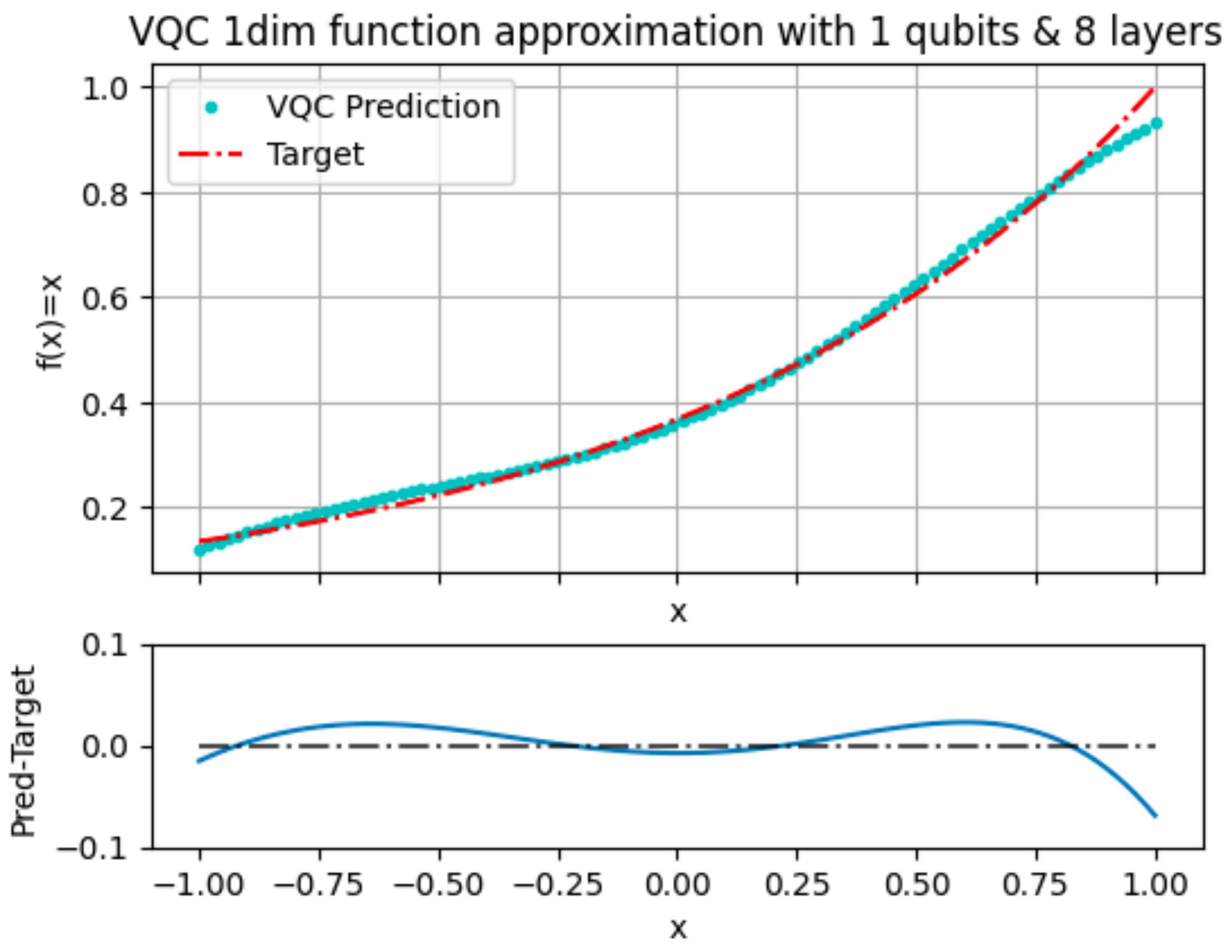
Considering 1-dim problem

Approximating $f(x) = e^{x-1}$

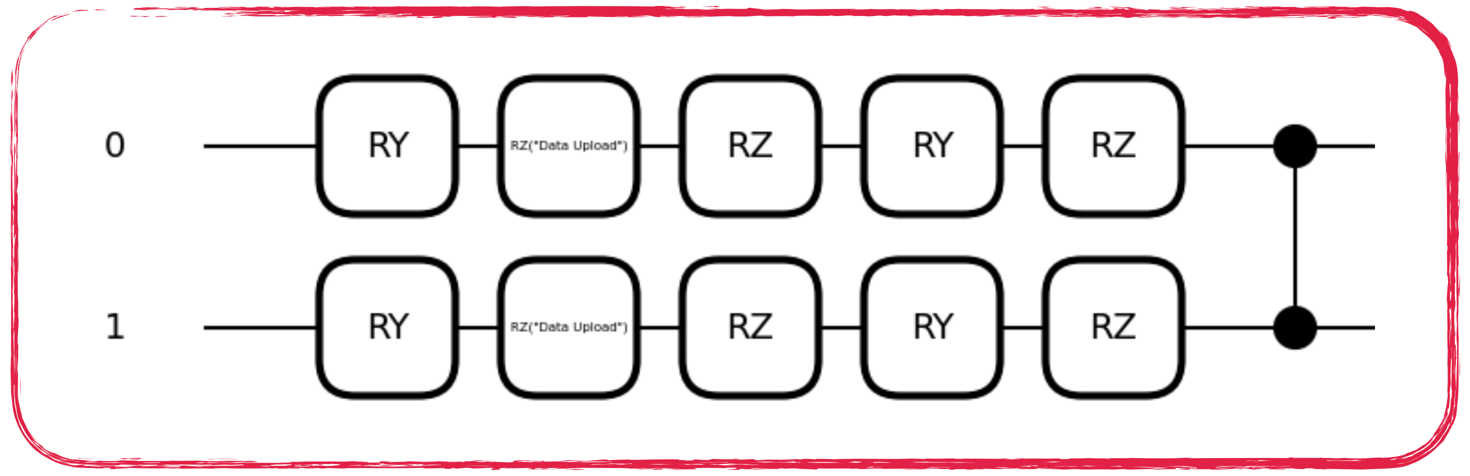
Training data : 50 points
Learning rate : 0.007

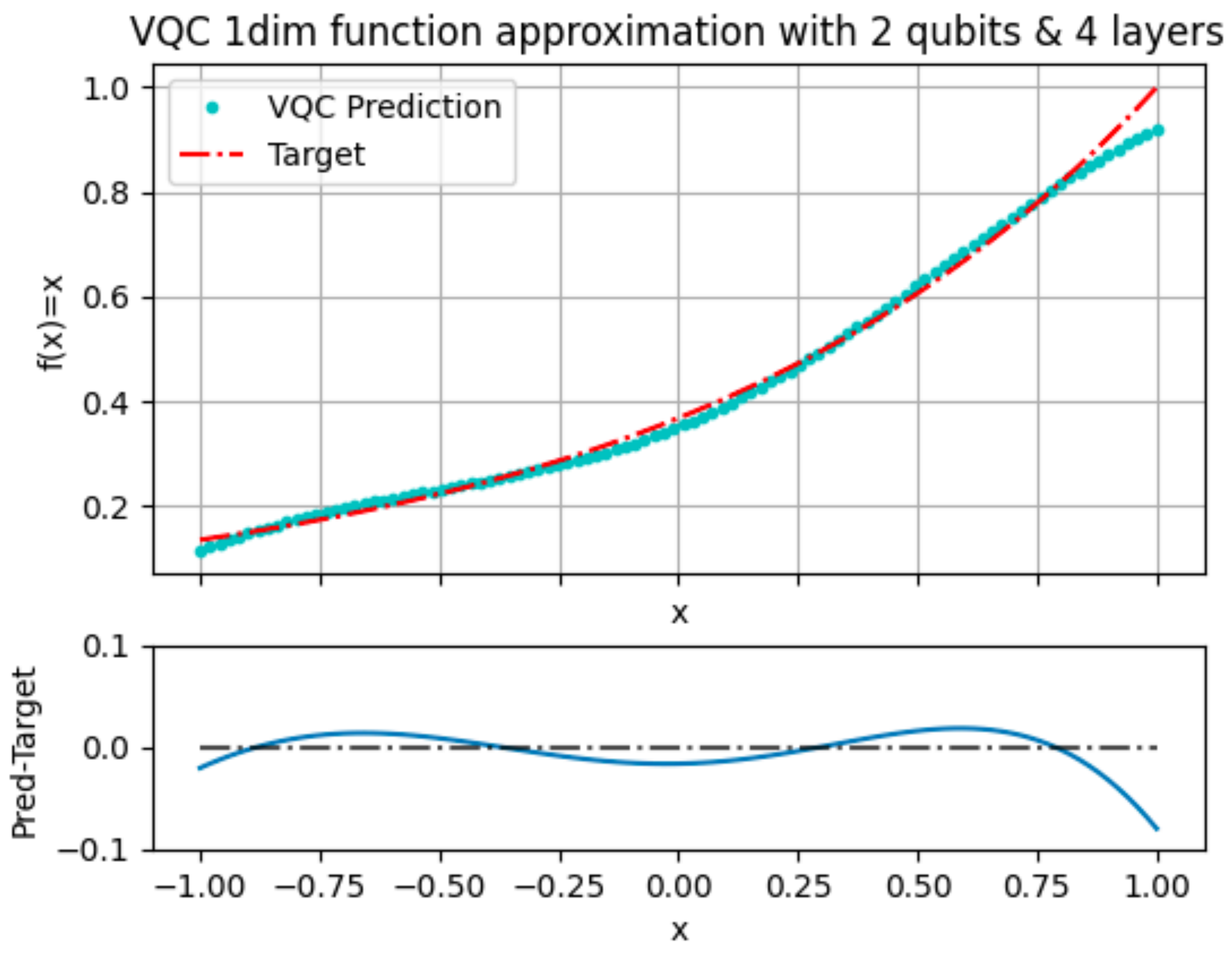
With 8 layers, 41 parameters



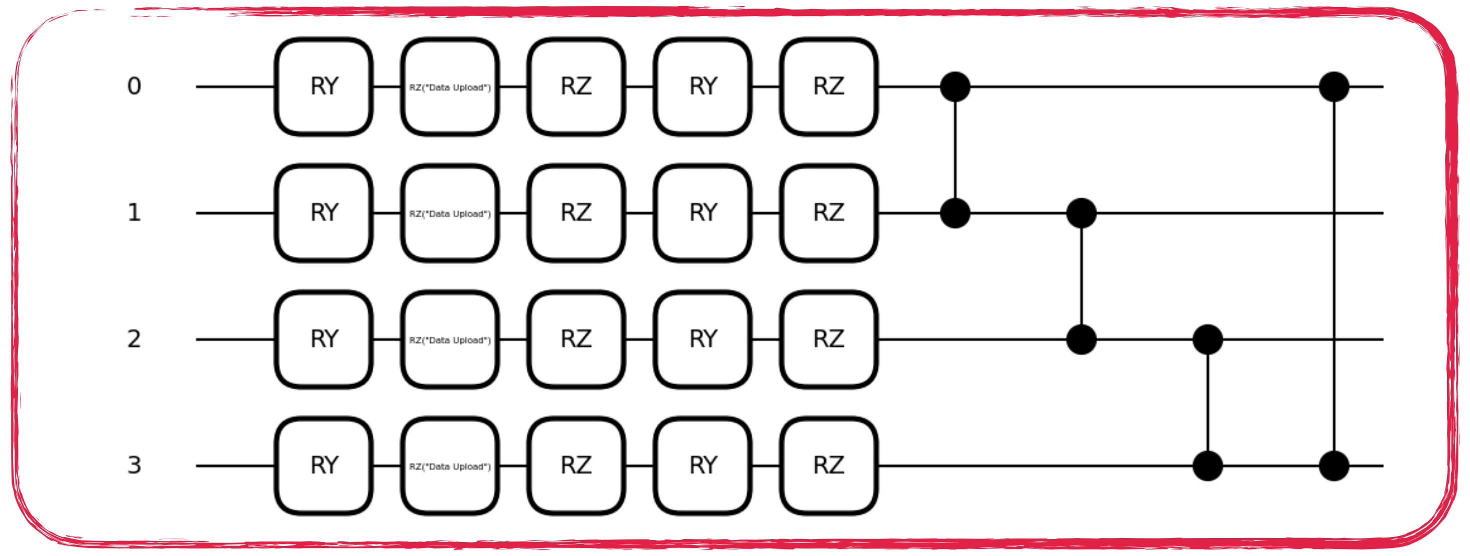


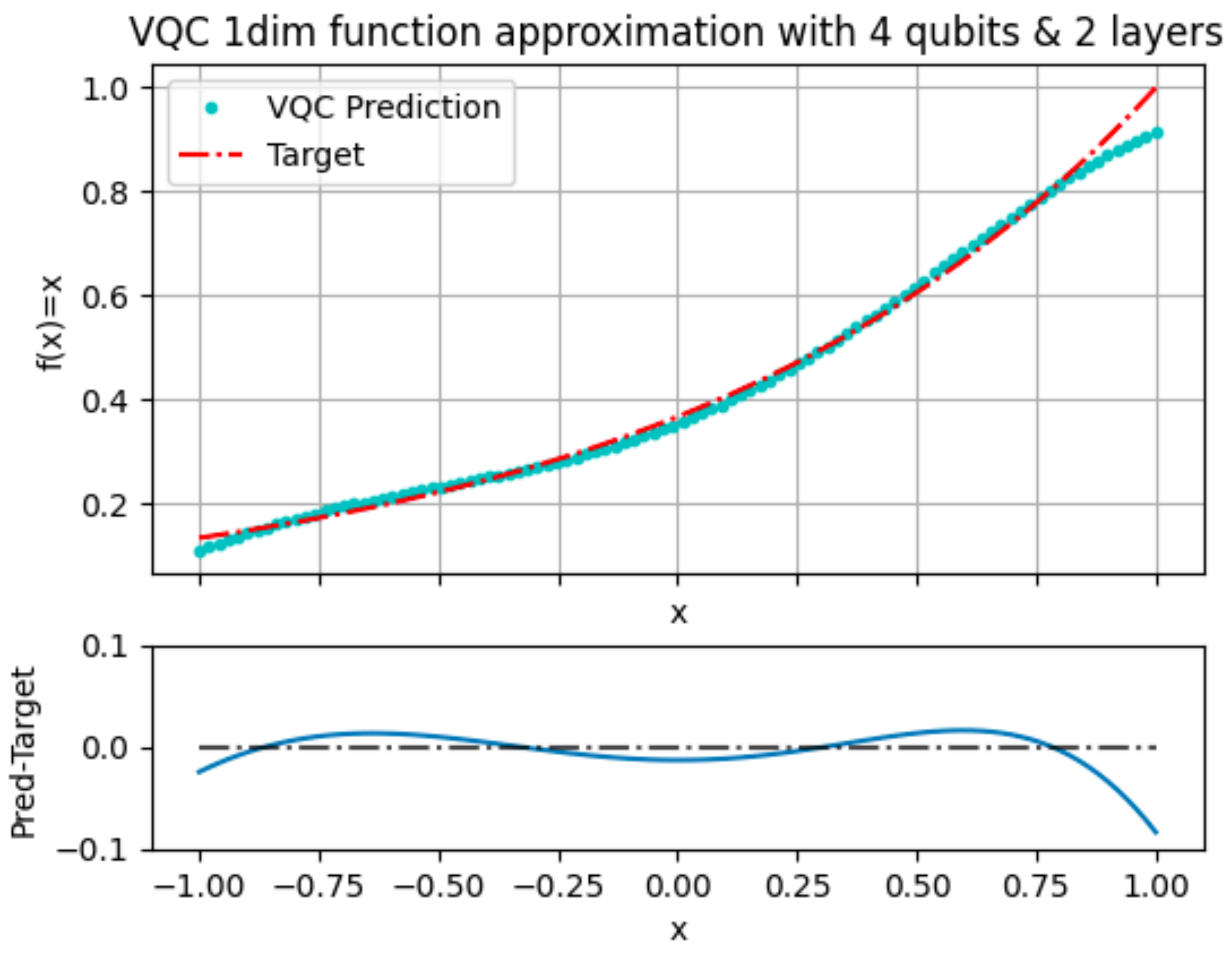
With 4 layers, 42 parameters





With 2 layers, 44 parameters





Predictions with random parameters
Predictions after training
True test data
Predictions with random parameters
Predictions after training
True test data
Predictions with random parameters
Predictions after training
True test data
Loss(Layers = 8)
Loss(Layers = 4)
Loss(Layers = 2)
Accuracy(Layers = 8)
Accuracy(Layers = 4)
Accuracy(Layers = 2)
Test_accuracy
Train_accuracy
Test_accuracy
Train_accuracy
Test_accuracy
Train_accuracy

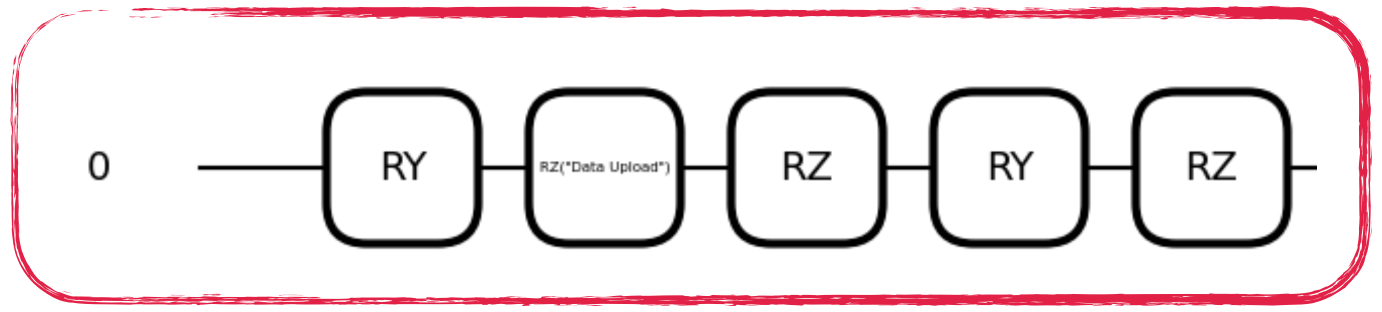
Before solving the realistic example

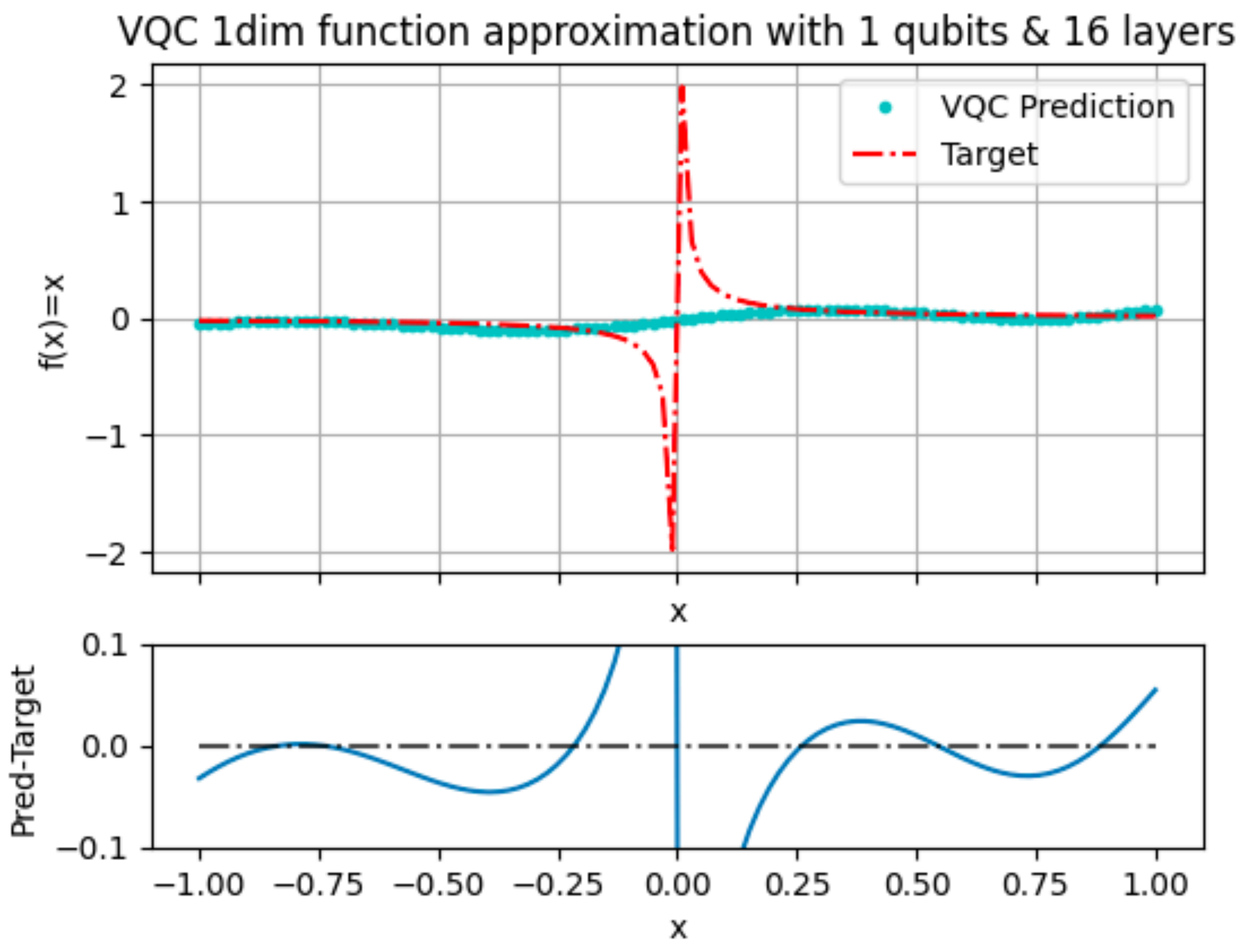
Considering 1-dim problem

Approximating $f(x) = 1/50x$

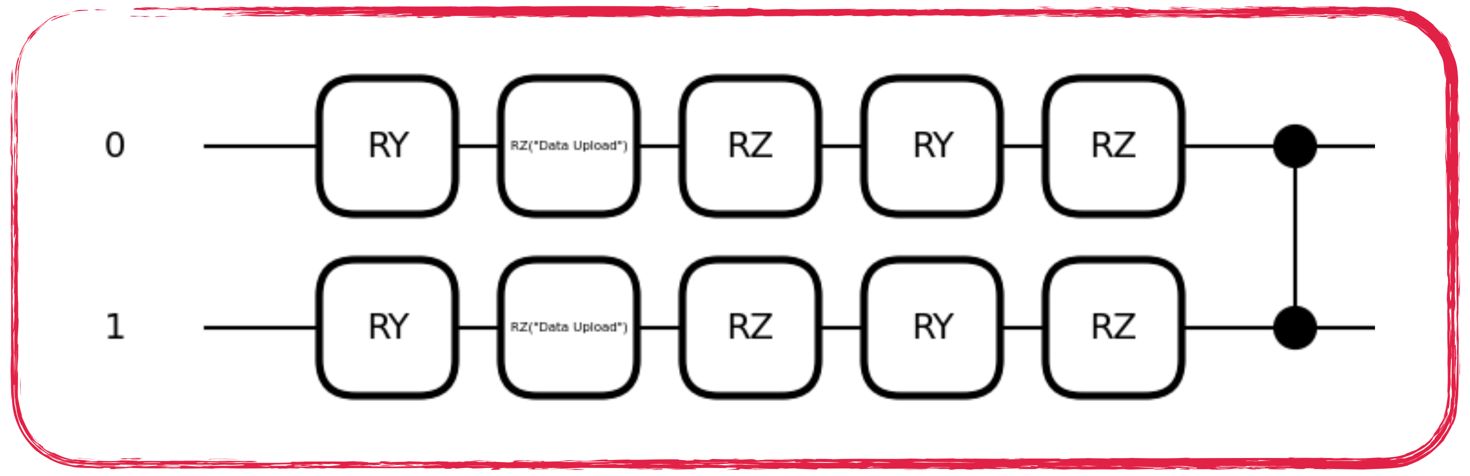
Training data : 50 points
Learning rate : 0.007

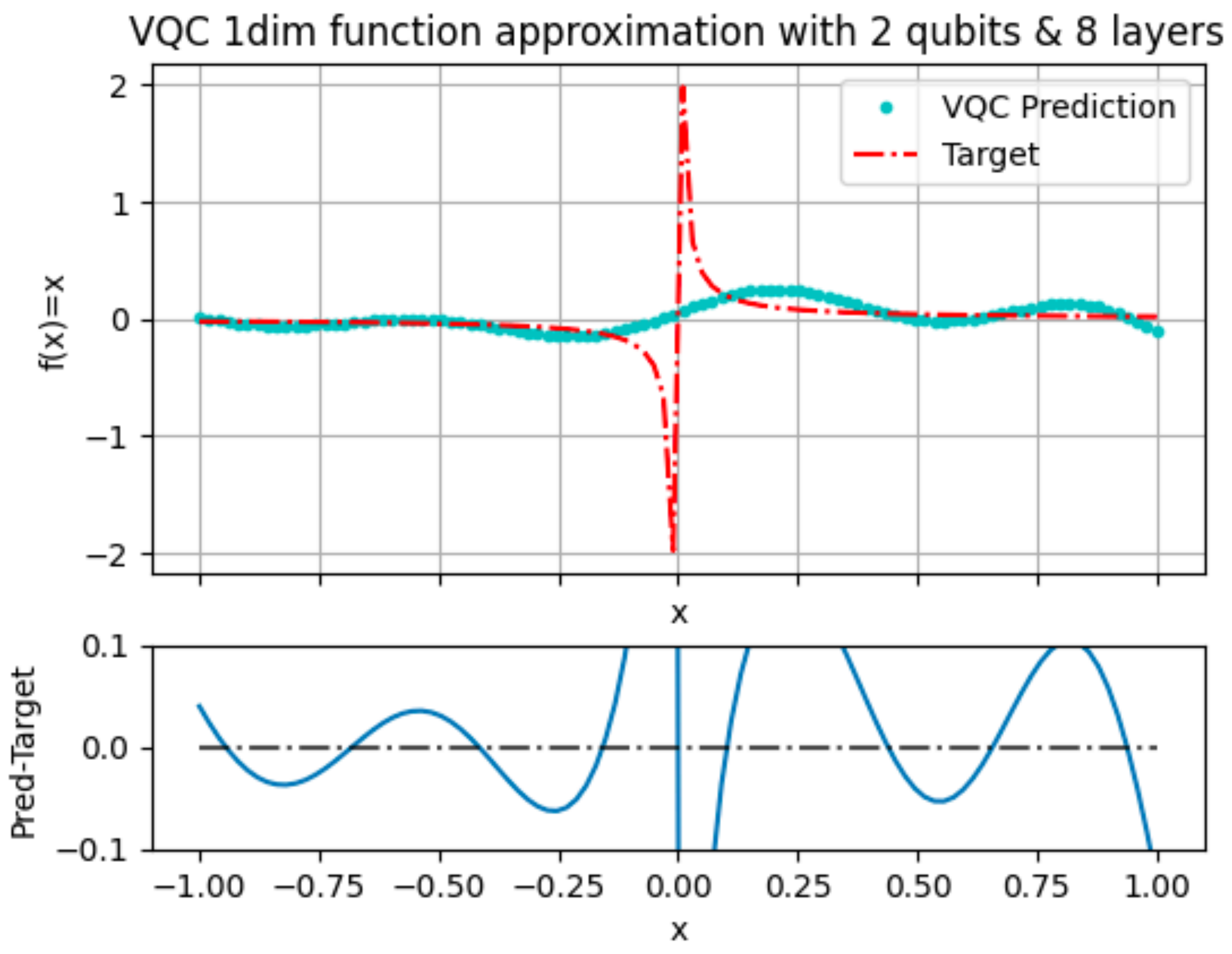
With 16 layers, 81 parameters



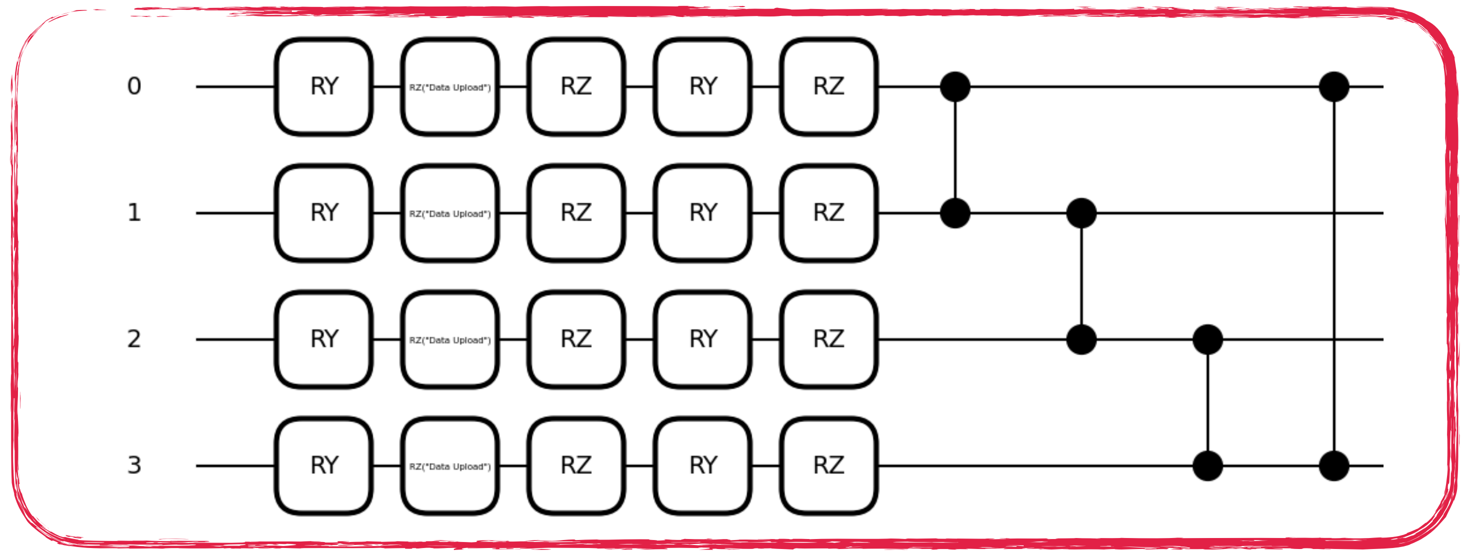


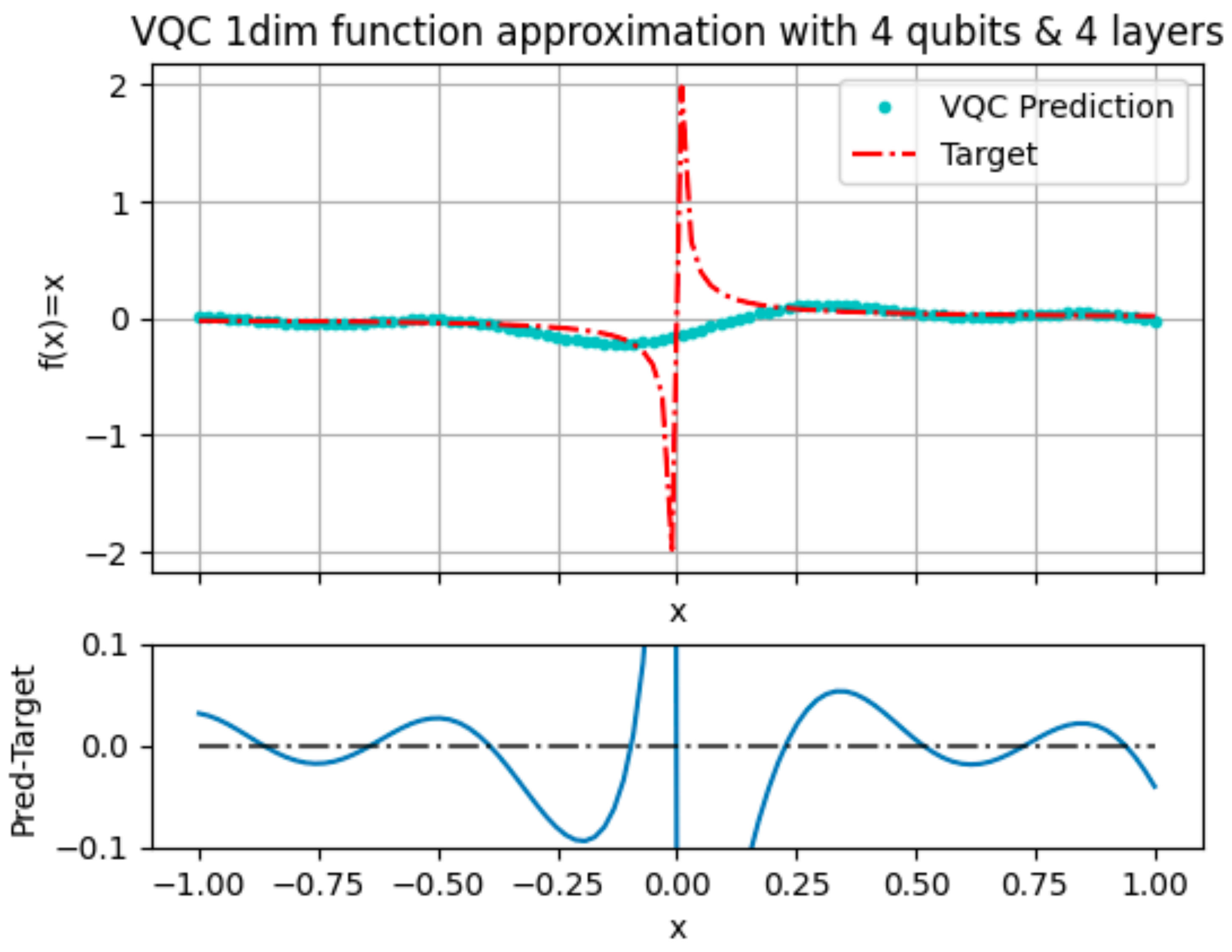
With 8 layers, 82 parameters





With 4 layers, 84 parameters



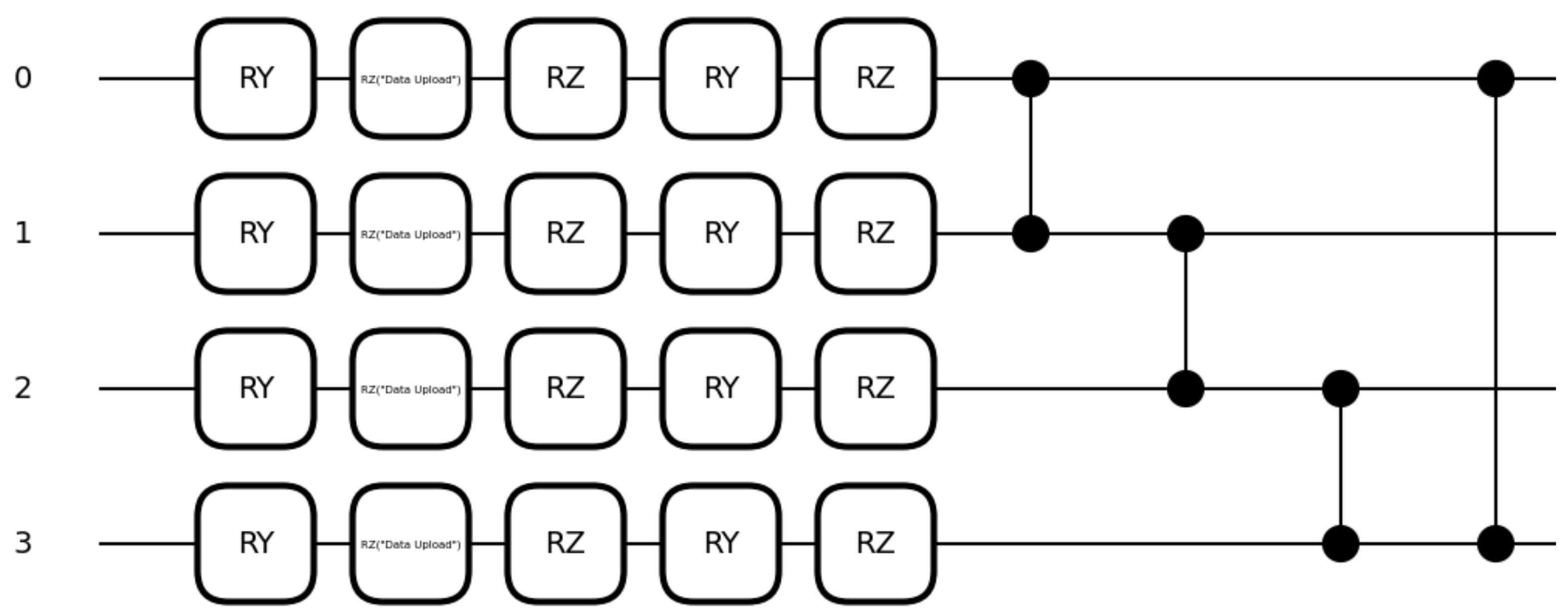



Predictions with random parameters
Predictions after training
True test data
Loss(Layers = 16)
Accuracy(Layers = 16)
Loss(Layers = 8)
Accuracy(Layers = 8)
Loss(Layers = 4)
Accuracy(Layers = 4)
Test_accuracy
Train_accuracy

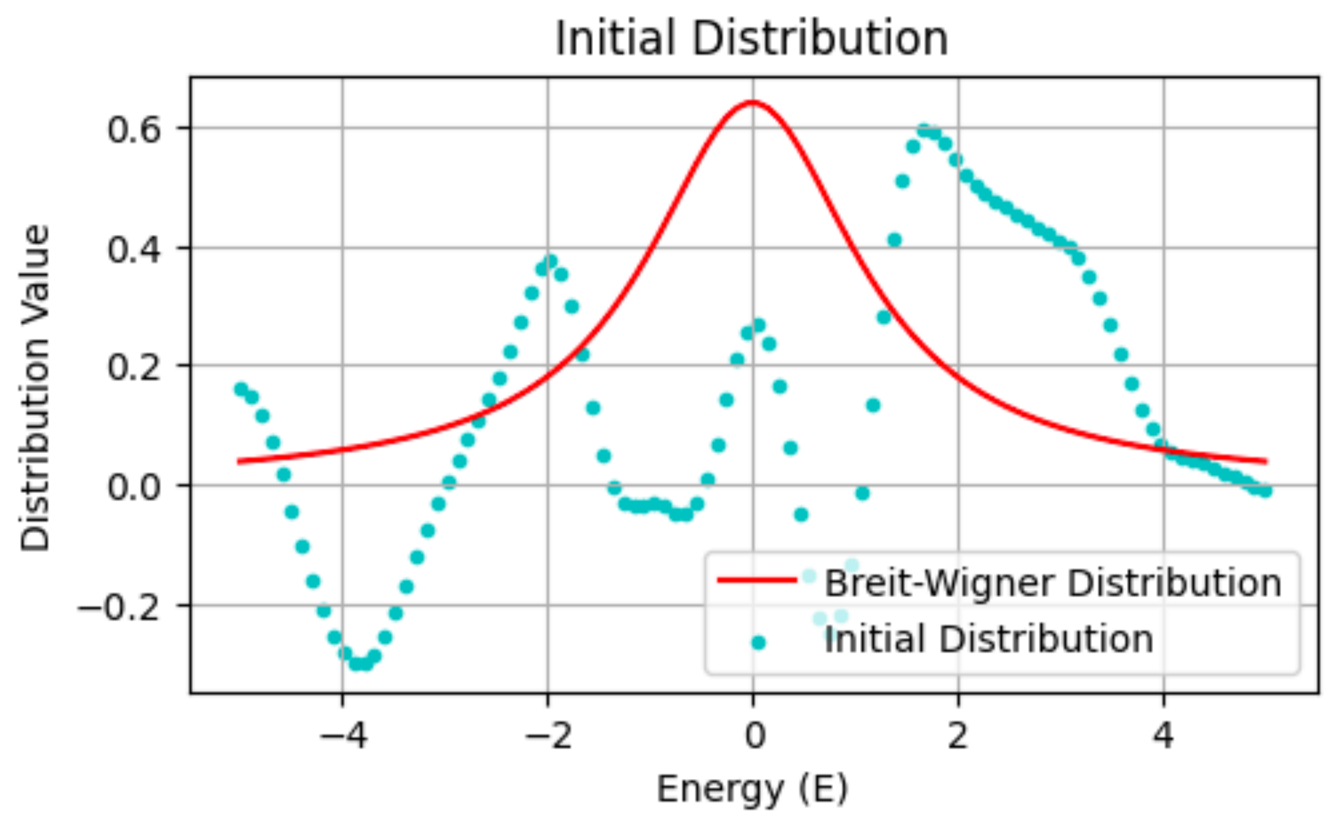
Solving a simple 1-dim realistic example

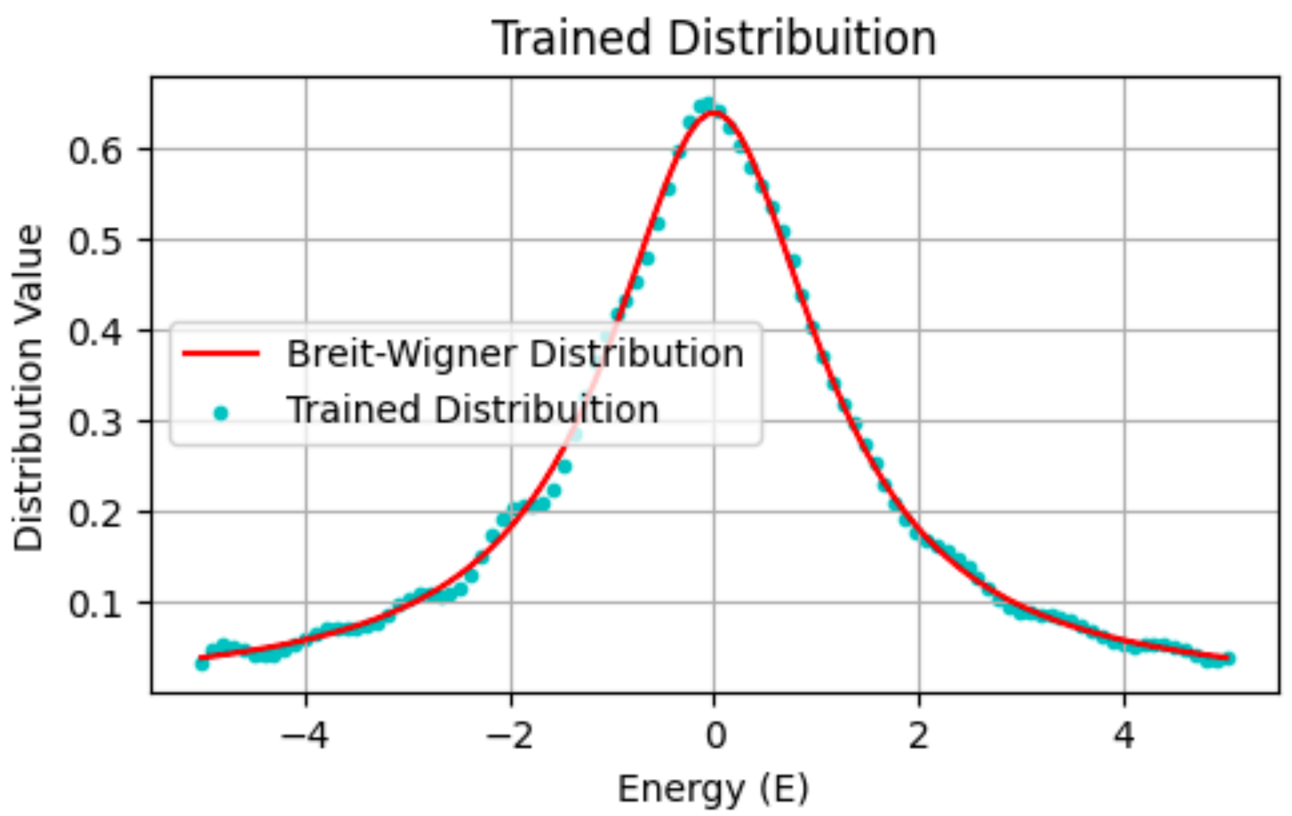
<u>'Breit-Wigner Equation'</u>

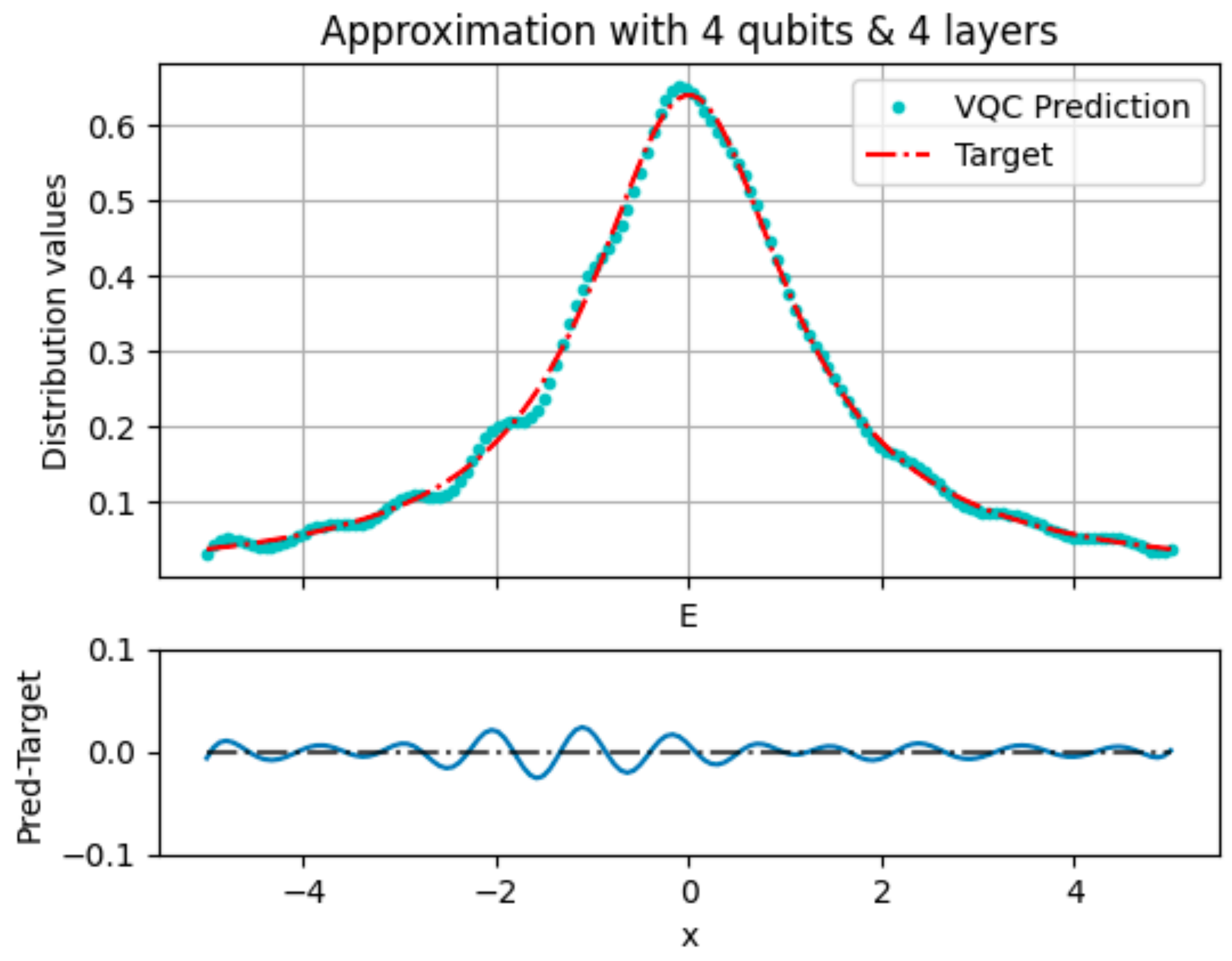
$$\sigma = \frac{1}{(E - E_0)^2 + (\Gamma/2)^2} \quad \text{for } E_0 = 0 \ \& \ \Gamma = 2.5$$



$\times$ 4 layers





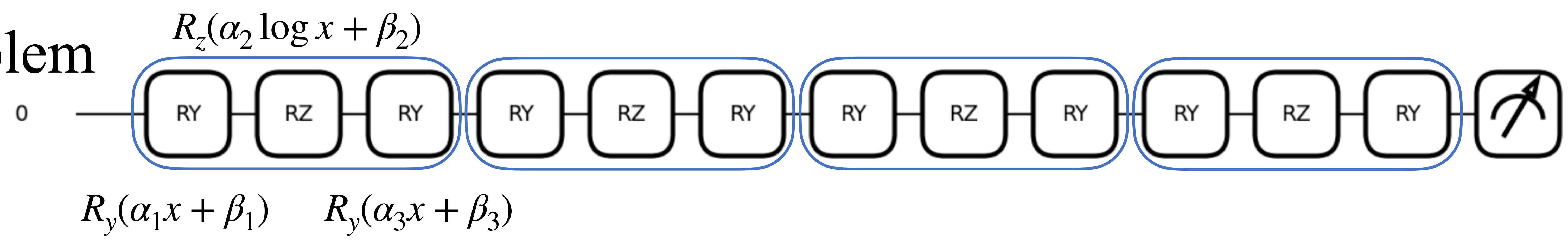


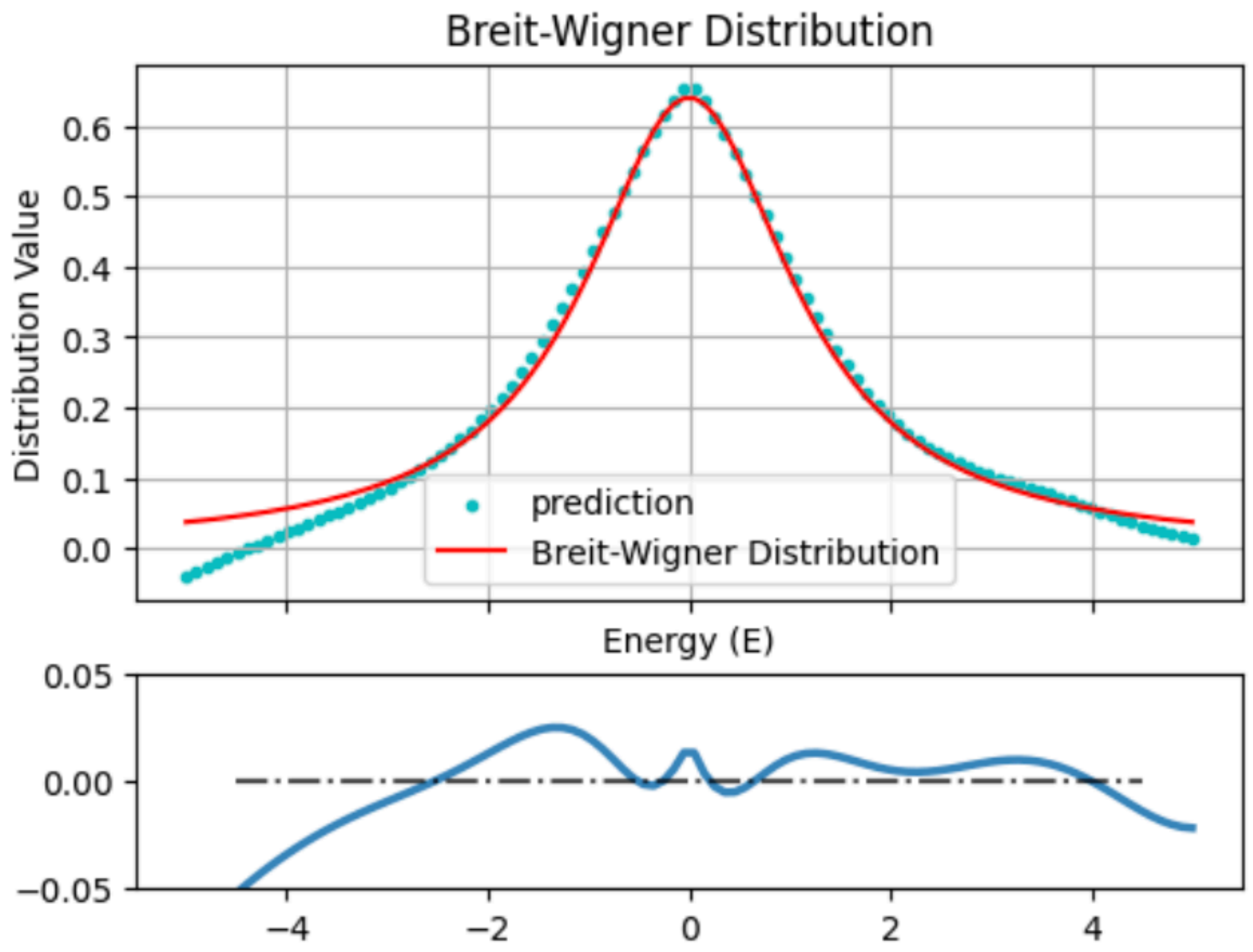
Solving a simple 1-dim realistic example

‘Breit-Wigner Equation’

$$\sigma = \frac{1}{(E - E_0)^2 + (\Gamma/2)^2} \quad \text{for } E_0 = 0 \;\&\; \Gamma = 2.5$$

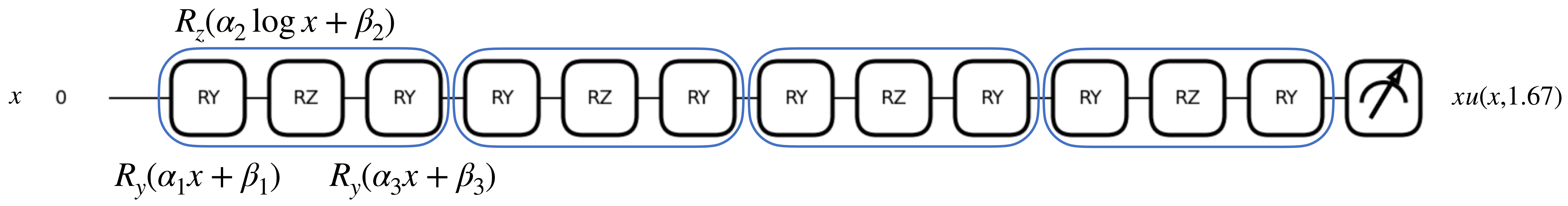
Ansatz using for PDF problem from the paper

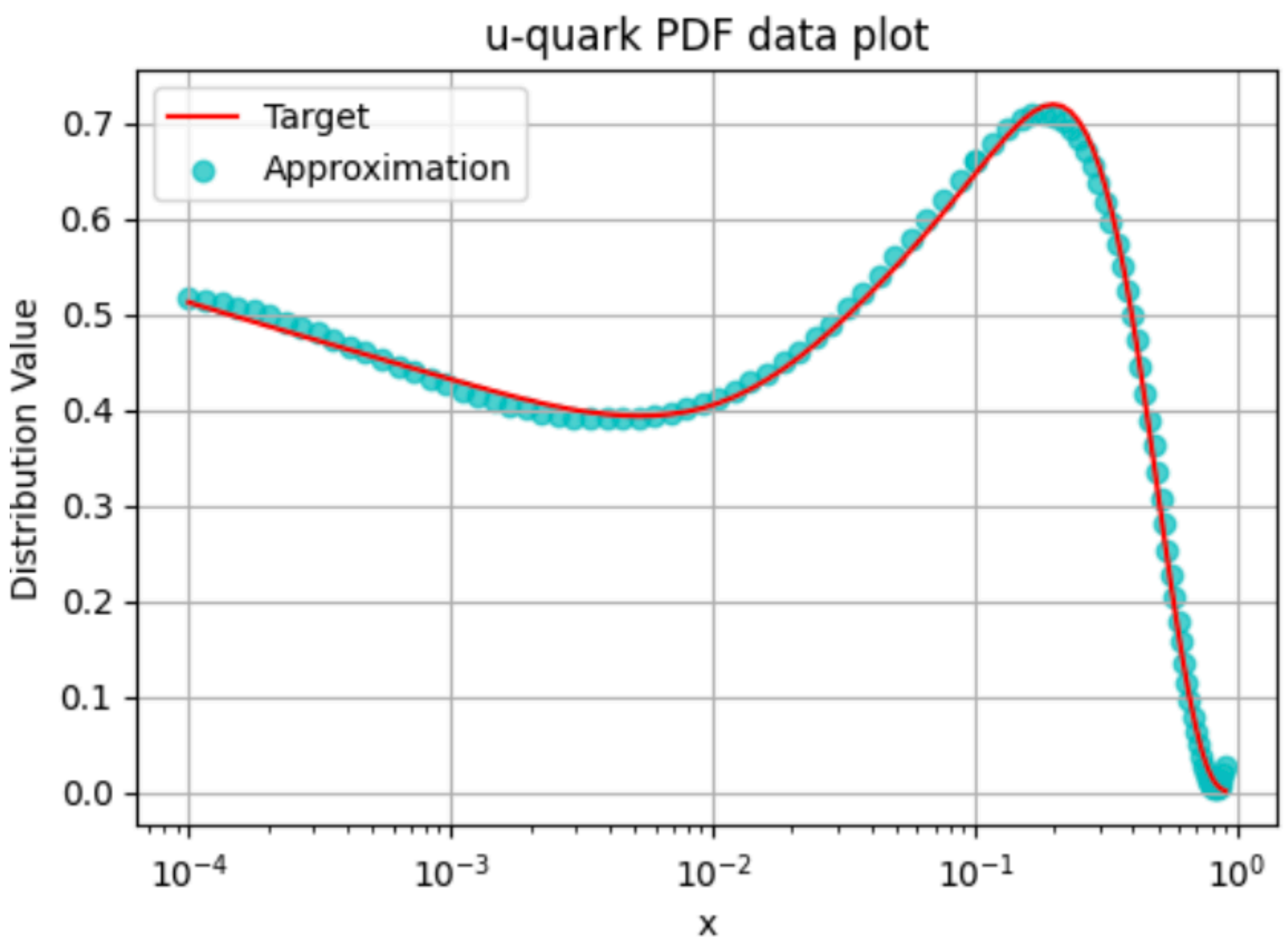




Reproducing the PDF problem in paper

"u quark PDF $xu(x, Q^2)$ with simple 1D case $Q = 1.67$"





Message we get

1. For some similar numbers of parameters, we could use more qubits to use less layers which means that we can design a quantum circuit with lower circuit depths.
2. With knowing some properties of the data set, we can re-upload the datas with some manipulation.

Questions to discuss

1. Despite of getting a good approximation, we need some correction method for the tail…
2. From the inverse function approximation example, there is some hardness for approximation… Is there some method we could consider for the singularity?
3. For Riemann Integration, we need to run the quantum circuit too many times…

1-Dim Integration Problem Process

$$F(x,\theta) = \int f(x,\theta)dx$$

VQC Model

Parameters (xdata, thetas)

$$\partial_x F(x,\theta)$$

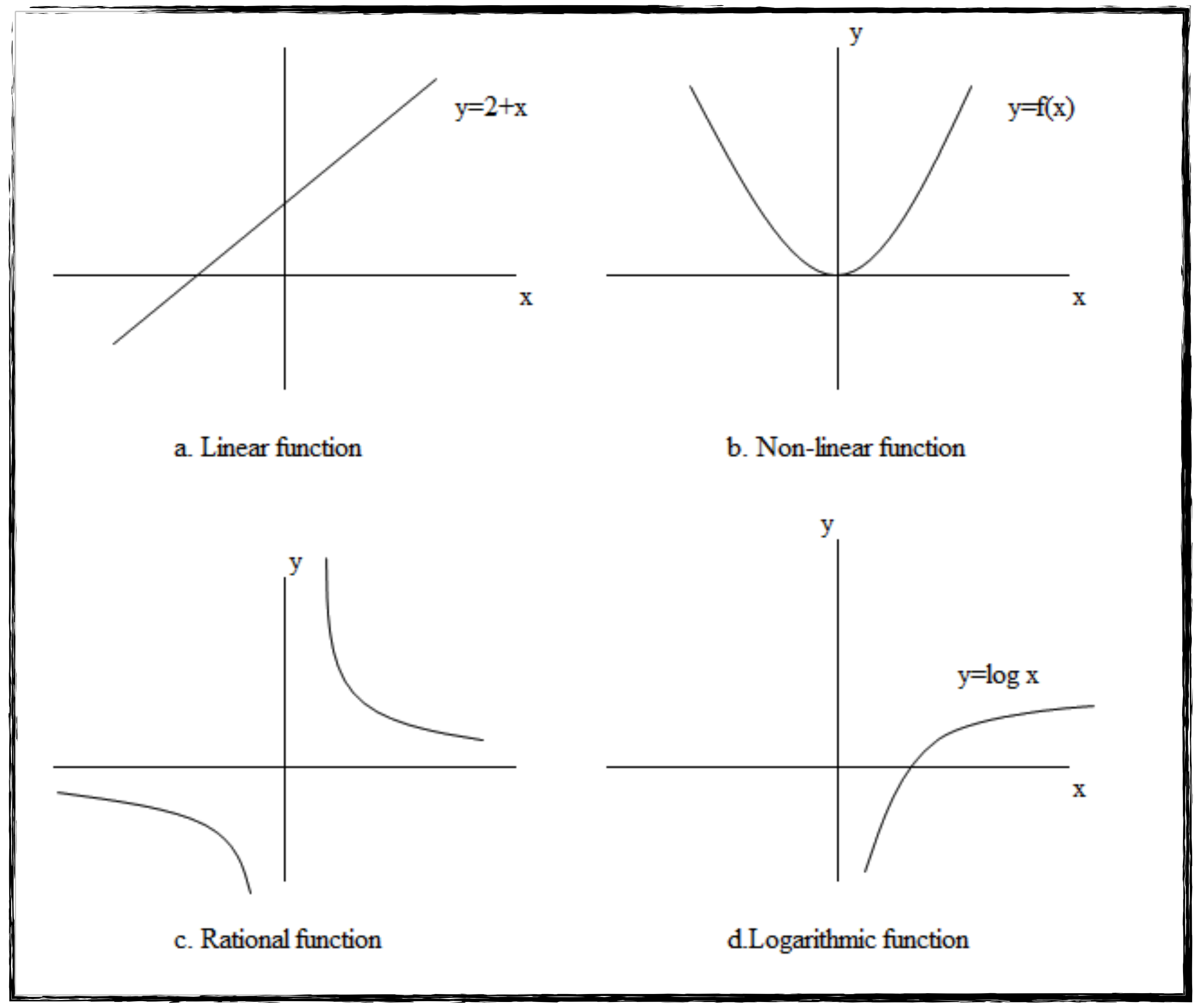
VQC Model
Gradient depend on x

Parameters (xdata, thetas)

Applying PSR depend on x

$\sim$

Target integrand $f(x,\theta)$



Optimizing parameters(θ) to approximate the function

$$F(x,\theta_{opt})$$

Opt. VQC Model

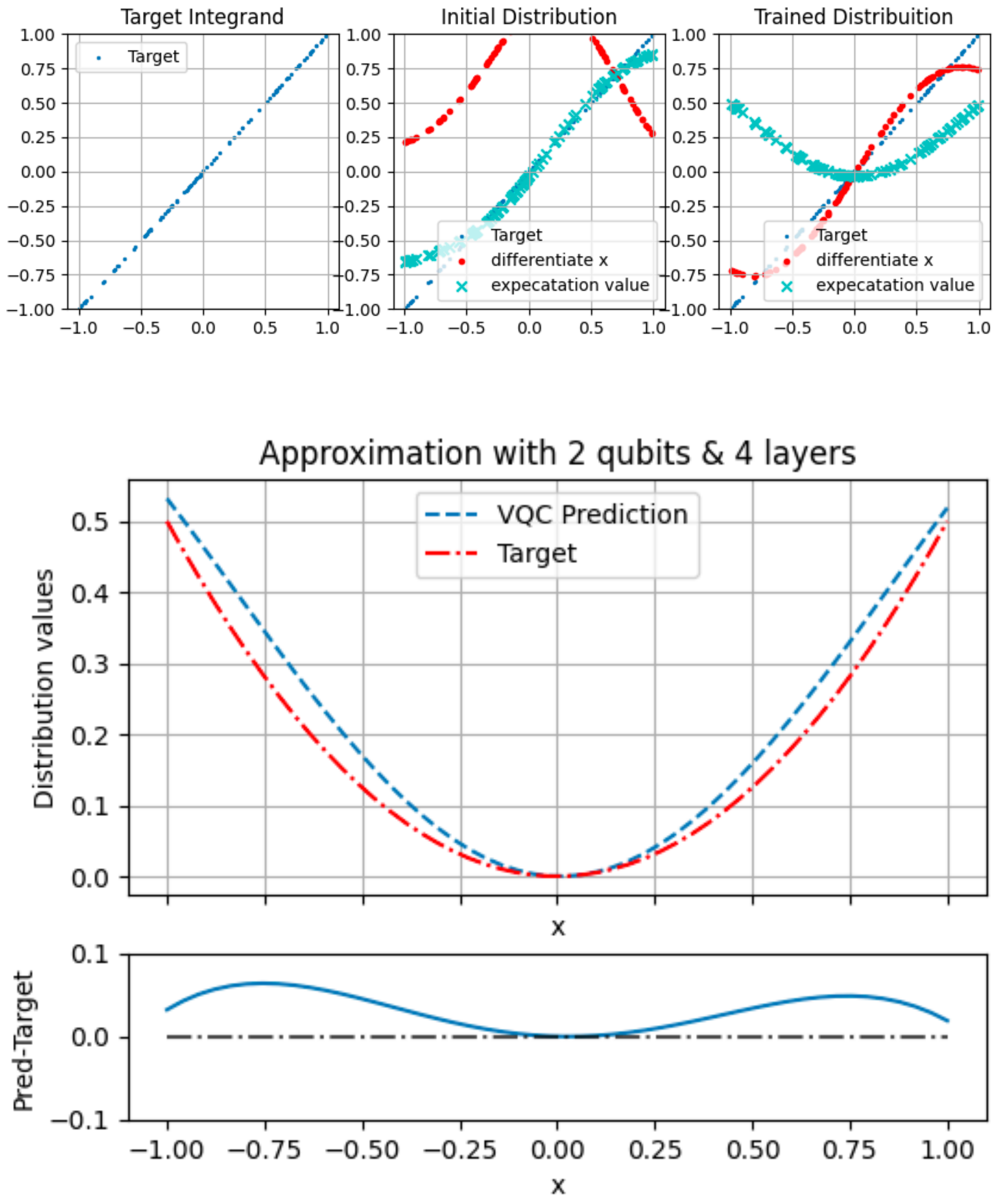
<u>Running **two times** for getting the integral</u>

Integral = $F(x_{upper\ bound},\theta_{opt}) - F(x_{lower\ bound},\theta_{opt})$

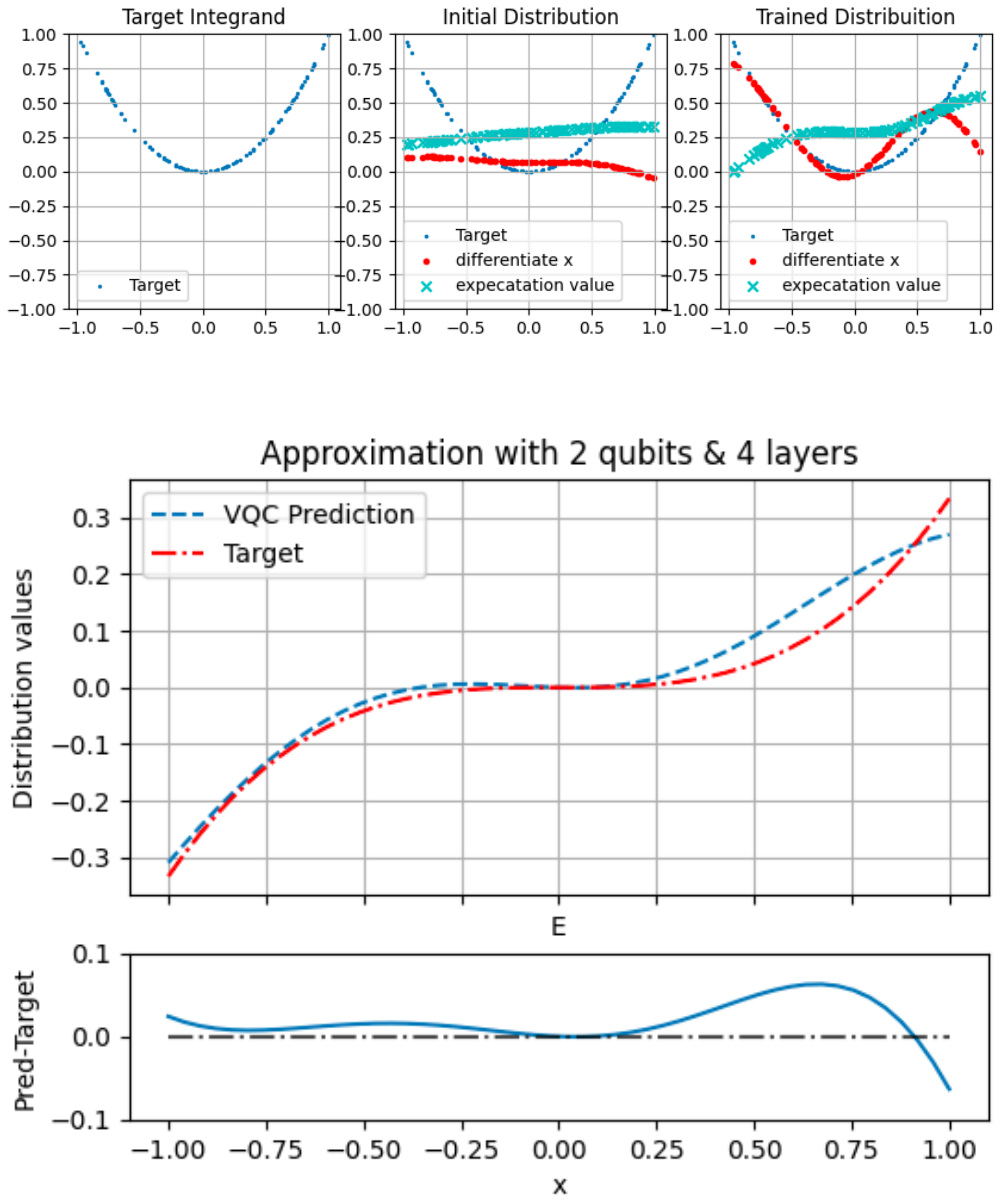
Maybe the paper(arXiv:2308.05657)'s method

1-Dim Integration Problem Results

$$\int_0^x x' dx'$$



$$\int_0^x x'^2 dx'$$



$$\int_0^x e^{x'-1} dx'$$