In this task you are supposed to get started with equivariant quantum neural networks by implementing a Z_2 × Z_2 equivariant quantum neural network. Z_2 is a symmetry group an as an example we will generate a simple classical dataset which is respects the Z_2 x Z_2 symmetry.

This example is explained in the paper https://arxiv.org/abs/2205.06217 and

additional background can be found in https://arxiv.org/abs/2210.08566.

- Generate a classification dataset with two classes and two features x_1 and x_2 which respects the Z_2 x Z_2 symmetry (this corresponds to mirroring along y=x). An example can be found in the first reference paper.
- Train a QNN to solve the classification problem
- Train an Z_2 x Z_2 equivariant QNN to solve the classification problem and compare the results.

In [ ]:

```
pip install qiskit
```

```
Requirement already satisfied: qiskit in c:\programdata\anaconda3\lib\site-packages (0.42.1)
Requirement already satisfied: qiskit-terra==0.23.3 in c:\programdata\anaconda3\lib\site-packages (from qiskit) (0.23.3)
Requirement already satisfied: qiskit-ibmq-provider==0.20.2 in c:\programdata\anaconda3\lib\site-packages (from qiskit) (0.20.2)
Requirement already satisfied: qiskit-aer==0.12.0 in c:\programdata\anaconda3\lib\site-packages (from qiskit) (0.12.0)
Requirement already satisfied: numpy>=1.16.3 in c:\programdata\anaconda3\lib\site-packages (from qiskit-aer==0.12.0->qiskit) (1.20.3)
Requirement already satisfied: scipy>=1.0 in c:\programdata\anaconda3\lib\site-packages (from qiskit-aer==0.12.0->qiskit) (1.7.1)
Requirement already satisfied: python-dateutil>=2.8.0 in c:\programdata\anaconda3\lib\site-packages (from qiskit-ibmq-provider==0.20.2->qiskit) (2.8.2)
Requirement already satisfied: requests>=2.19 in c:\programdata\anaconda3\lib\site-packages (from qiskit-ibmq-provider==0.20.2->qiskit) (2.26.0)
Requirement already satisfied: requests-ntlm<=1.1.0 in c:\programdata\anaconda3\lib\site-packages (from qiskit-ibmq-provider==0.20.2->qiskit) (1.1.0)
Requirement already satisfied: urllib3>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from qiskit-ibmq-provider==0.20.2->qiskit) (1.26.7)
Requirement already satisfied: websocket-client>=1.5.1 in c:\programdata\anaconda3\lib\site-packages (from qiskit-ibmq-provider==0.20.2->qiskit) (1.5.1)
Requirement already satisfied: websockets>=10.0 in c:\programdata\anaconda3\lib\site-packages (from qiskit-ibmq-provider==0.20.2->qiskit) (10.4)
Requirement already satisfied: ply>=3.10 in c:\programdata\anaconda3\lib\site-packages (from qiskit-terra==0.23.3->qiskit) (3.11)
Requirement already satisfied: dill>=0.3 in c:\programdata\anaconda3\lib\site-packages (from qiskit-terra==0.23.3->qiskit) (0.3.6)
Requirement already satisfied: rustworkx>=0.12.0 in c:\programdata\anaconda3\lib\site-packages (from qiskit-terra==0.23.3->qiskit) (0.12.1)
Requirement already satisfied: stevedore>=3.0.0 in c:\programdata\anaconda3\lib\site-packages (from qiskit-terra==0.23.3->qiskit) (5.0.0)
Requirement already satisfied: psutil>=5 in c:\programdata\anaconda3\lib\site-packages (from qiskit-terra==0.23.3->qiskit) (5.8.0)
Requirement already satisfied: sympy>=1.3 in c:\programdata\anaconda3\lib\site-packages (from qiskit-terra==0.23.3->qiskit) (1.9)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-packages (from python-dateutil>=2.8.0->qiskit-ibmq-provider==0.20.2->qiskit) (1.16.0)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.19->qiskit-ibmq-provider==0.20.2->qiskit) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.19->qiskit-ibmq-provider==0.20.2->qiskit) (2021.10.8)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests>=2.19->qiskit-ibmq-provider==0.20.2->qiskit) (3.2)
Requirement already satisfied: cryptography>=1.3 in c:\programdata\anaconda3\lib\site-packages (from requests-ntlm<=1.1.0->qiskit-ibmq-provider==0.20.2->qiskit) (3.4.8)
Requirement already satisfied: ntlm-auth>=1.0.2 in c:\programdata\anaconda3\lib\site-packages (from requests-ntlm<=1.1.0->qiskit-ibmq-provider==0.20.2->qiskit) (1.5.0)
Requirement already satisfied: cffi>=1.12 in c:\programdata\anaconda3\lib\site-packages (from cryptography>=1.3->requests-ntlm<=1.1.0->qiskit-ibmq-provider==0.20.2->qiskit) (1.14.6)
Requirement already satisfied: pycparser in c:\programdata\anaconda3\lib\site-packages (from cffi>=1.12->cryptography>=1.3->requests-ntlm<=1.1.0->qiskit-ibmq-provider==0.20.2->qiskit) (2.20)
Requirement already satisfied: pbr!=2.1.0,>=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from stevedore>=3.0.0->qiskit-terra==0.23.3->qiskit) (5.11.1)
Requirement already satisfied: mpmath>=0.19 in c:\programdata\anaconda3\lib\site-packages (from sympy>=1.3->qiskit-terra==0.23.3->qiskit) (1.2.1)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
```

In [ ]:

```
pip install pennylane
```

```
Collecting pennylane
  Downloading PennyLane-0.29.1-py3-none-any.whl (1.3 MB)
Collecting autoray>=0.3.1
  Downloading autoray-0.6.3-py3-none-any.whl (48 kB)
Collecting autograd
  Downloading autograd-1.5-py3-none-any.whl (48 kB)
Collecting pennylane-lightning>=0.28
  Downloading PennyLane_Lightning-0.29.0-cp39-cp39-win_amd64.whl (4.6 MB)
Requirement already satisfied: appdirs in c:\programdata\anaconda3\lib\site-packages (from pennylane) (1.4.4)
Requirement already satisfied: numpy<1.24 in c:\programdata\anaconda3\lib\site-packages (from pennylane) (1.20.3)
Requirement already satisfied: scipy in c:\programdata\anaconda3\lib\site-packages (from pennylane) (1.7.1)
Requirement already satisfied: toml in c:\programdata\anaconda3\lib\site-packages (from pennylane) (0.10.2)
Collecting retworkx
  Downloading retworkx-0.12.1-py3-none-any.whl (10 kB)
Collecting semantic-version>=2.7
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: requests in c:\programdata\anaconda3\lib\site-packages (from pennylane) (2.26.0)
Requirement already satisfied: networkx in c:\programdata\anaconda3\lib\site-packages (from pennylane) (2.6.3)
Requirement already satisfied: cachetools in c:\programdata\anaconda3\lib\site-packages (from pennylane) (5.2.0)
Requirement already satisfied: future>=0.15.2 in c:\programdata\anaconda3\lib\site-packages (from autograd->pennylane) (0.18.2)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\programdata\anaconda3\lib\site-packages (from requests->pennylane) (1.26.7)
Requirement already satisfied: idna<4,>=2.5 in c:\programdata\anaconda3\lib\site-packages (from requests->pennylane) (3.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\programdata\anaconda3\lib\site-packages (from requests->pennylane) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in c:\programdata\anaconda3\lib\site-packages (from requests->pennylane) (2021.10.8)
Requirement already satisfied: rustworkx==0.12.1 in c:\programdata\anaconda3\lib\site-packages (from retworkx->pennylane) (0.12.1)
Installing collected packages: semantic-version, retworkx, pennylane-lightning, autoray, autograd, pennylane
Successfully installed autograd-1.5 autoray-0.6.3 pennylane-0.29.1 pennylane-lightning-0.29.0 retworkx-0.12.1 semantic-version-2.10.0
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -yio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution -nyio (c:\programdata\anaconda3\lib\site-packages)
WARNING: Ignoring invalid distribution - (c:\programdata\anaconda3\lib\site-packages)
```

In [ ]:

```
import pennylane as qml
from pennylane import numpy as np
from pennylane.templates.layers import StronglyEntanglingLayers
from pennylane.templates.embeddings import AngleEmbedding
from pennylane.optimize import AdamOptimizer
```

In [ ]:

In [ ]:

First, let's generate the classification dataset with two classes and two features x_1 and x_2 which respects the Z_2 x Z_2 symmetry. We will create two concentric circles, one inside the other, and label the points in the outer circle as Class 0 and the points in the inner circle as Class 1.

In [ ]:

```
import numpy as np
from sklearn.datasets import make_circles

X, y = make_circles(n_samples=1000, noise=0.05, factor=0.5, random_state=42)
X = X.astype(np.float32)
y = y.astype(np.int64)

# label points inside the circle as Class 1 and outside the circle as Class 0
y = np.logical_not(y).astype(np.int64)

# Apply Z2 x Z2 symmetry transformation
X_sym = np.stack((X[:, 1], X[:, 0]), axis=1)
X = np.concatenate((X, X_sym), axis=0)
y = np.concatenate((y, y), axis=0)
```

In [ ]:

```
print(X)
```

```
[[ 0.45259237  0.16843331]
 [-0.43802652  0.11990049]
 [-0.5322243   0.18435901]
 ...
 [-0.531447   -0.07201617]
 [-0.7931901   0.6609045 ]
 [ 0.96735954  0.2784149 ]]
```

In [ ]:

```
print(y)
```

```
[0 0 0 ... 0 1 1]
```

Next, let's split the dataset into training and testing sets:

In [ ]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [ ]:

```
import pennylane as qml
from pennylane import numpy as np

n_qubits = 2
dev = qml.device("default.qubit", wires=n_qubits)

@qml.qnode(dev)
def qnn_circuit(x, weights):
    for i in range(n_qubits):
        qml.RX(x[i], wires=i)

    qml.CNOT(wires=[0, 1])
    qml.RZ(weights[0], wires=0)
    qml.CNOT(wires=[0, 1])
    qml.RZ(weights[1], wires=1)

    for i in range(n_qubits):
        qml.RX(weights[2+i]*np.pi, wires=i)

    return qml.expval(qml.PauliZ(0))

def qnn_loss(weights, X, y):
    loss = 0
    for i in range(len(X)):
        output = qnn_circuit(X[i], weights)
        loss += (output - (-1)**y[i]) ** 2
    return loss / len(X)

np.random.seed(42)
weights = np.random.rand(n_qubits+3)
opt = qml.GradientDescentOptimizer(0.1)

n_epochs = 100
batch_size = 32
n_batches = len(X_train) // batch_size

for epoch in range(n_epochs):
    # shuffle the training data
    permutation = np.random.permutation(len(X_train))
    X_train = X_train[permutation]
    y_train = y_train[permutation]

    for i in range(n_batches):
        # select the next batch
        X_batch = X_train[i*batch_size:(i+1)*batch_size]
        y_batch = y_train[i*batch_size:(i+1)*batch_size]

        # update the weights
        weights = opt.step(lambda w: qnn_loss(w, X_batch, y_batch), weights)

    # compute the accuracy on the test set
    y_pred = [qml.math.sign(qnn_circuit(x, weights)) for x in X_test]
    acc = np.sum(y_pred == y_test) / len(y_test)

    print(f"Epoch {epoch+1}/{n_epochs}, Test accuracy: {acc:.3f}")
```

```
Epoch 1/100, Test accuracy: 0.500
Epoch 2/100, Test accuracy: 0.095
Epoch 3/100, Test accuracy: 0.253
Epoch 4/100, Test accuracy: 0.512
Epoch 5/100, Test accuracy: 0.512
Epoch 6/100, Test accuracy: 0.000
Epoch 7/100, Test accuracy: 0.328
Epoch 8/100, Test accuracy: 0.512
Epoch 9/100, Test accuracy: 0.512
Epoch 10/100, Test accuracy: 0.512
Epoch 11/100, Test accuracy: 0.512
Epoch 12/100, Test accuracy: 0.512
Epoch 13/100, Test accuracy: 0.398
Epoch 14/100, Test accuracy: 0.512
Epoch 15/100, Test accuracy: 0.512
Epoch 16/100, Test accuracy: 0.000
Epoch 17/100, Test accuracy: 0.512
Epoch 18/100, Test accuracy: 0.512
Epoch 19/100, Test accuracy: 0.512
Epoch 20/100, Test accuracy: 0.512
Epoch 21/100, Test accuracy: 0.000
Epoch 22/100, Test accuracy: 0.000
Epoch 23/100, Test accuracy: 0.512
Epoch 24/100, Test accuracy: 0.000
Epoch 25/100, Test accuracy: 0.000
Epoch 26/100, Test accuracy: 0.512
Epoch 27/100, Test accuracy: 0.512
Epoch 28/100, Test accuracy: 0.000
Epoch 29/100, Test accuracy: 0.512
Epoch 30/100, Test accuracy: 0.000
Epoch 31/100, Test accuracy: 0.512
Epoch 32/100, Test accuracy: 0.512
Epoch 33/100, Test accuracy: 0.512
Epoch 34/100, Test accuracy: 0.000
Epoch 35/100, Test accuracy: 0.005
Epoch 36/100, Test accuracy: 0.512
Epoch 37/100, Test accuracy: 0.000
Epoch 38/100, Test accuracy: 0.512
Epoch 39/100, Test accuracy: 0.512
Epoch 40/100, Test accuracy: 0.512
Epoch 41/100, Test accuracy: 0.512
Epoch 42/100, Test accuracy: 0.512
Epoch 43/100, Test accuracy: 0.512
Epoch 44/100, Test accuracy: 0.290
Epoch 45/100, Test accuracy: 0.512
Epoch 46/100, Test accuracy: 0.000
Epoch 47/100, Test accuracy: 0.512
Epoch 48/100, Test accuracy: 0.512
Epoch 49/100, Test accuracy: 0.512
Epoch 50/100, Test accuracy: 0.000
Epoch 51/100, Test accuracy: 0.000
Epoch 52/100, Test accuracy: 0.512
Epoch 53/100, Test accuracy: 0.512
Epoch 54/100, Test accuracy: 0.512
Epoch 55/100, Test accuracy: 0.512
Epoch 56/100, Test accuracy: 0.512
Epoch 57/100, Test accuracy: 0.512
Epoch 58/100, Test accuracy: 0.000
Epoch 59/100, Test accuracy: 0.000
Epoch 60/100, Test accuracy: 0.512
Epoch 61/100, Test accuracy: 0.420
Epoch 62/100, Test accuracy: 0.000
Epoch 63/100, Test accuracy: 0.512
Epoch 64/100, Test accuracy: 0.512
Epoch 65/100, Test accuracy: 0.512
Epoch 66/100, Test accuracy: 0.512
Epoch 67/100, Test accuracy: 0.512
Epoch 68/100, Test accuracy: 0.512
Epoch 69/100, Test accuracy: 0.000
Epoch 70/100, Test accuracy: 0.512
Epoch 71/100, Test accuracy: 0.000
Epoch 72/100, Test accuracy: 0.512
Epoch 73/100, Test accuracy: 0.512
Epoch 74/100, Test accuracy: 0.512
Epoch 75/100, Test accuracy: 0.000
Epoch 76/100, Test accuracy: 0.512
Epoch 77/100, Test accuracy: 0.000
Epoch 78/100, Test accuracy: 0.507
Epoch 79/100, Test accuracy: 0.025
Epoch 80/100, Test accuracy: 0.000
Epoch 81/100, Test accuracy: 0.512
Epoch 82/100, Test accuracy: 0.512
Epoch 83/100, Test accuracy: 0.512
Epoch 84/100, Test accuracy: 0.000
Epoch 85/100, Test accuracy: 0.512
Epoch 86/100, Test accuracy: 0.512
Epoch 87/100, Test accuracy: 0.000
Epoch 88/100, Test accuracy: 0.512
Epoch 89/100, Test accuracy: 0.300
Epoch 90/100, Test accuracy: 0.000
Epoch 91/100, Test accuracy: 0.512
Epoch 92/100, Test accuracy: 0.000
Epoch 93/100, Test accuracy: 0.512
Epoch 94/100, Test accuracy: 0.512
Epoch 95/100, Test accuracy: 0.512
Epoch 96/100, Test accuracy: 0.512
Epoch 97/100, Test accuracy: 0.512
Epoch 98/100, Test accuracy: 0.000
Epoch 99/100, Test accuracy: 0.495
Epoch 100/100, Test accuracy: 0.512
```

In [ ]:

```
print(f"Final test accuracy: {acc:.3f}")
```

```
Final test accuracy: 0.512
```

This is a standard quantum neural network that does not take into account the Z_2 x Z_2 symmetry of the dataset. To make the QNN equivariant under the symmetry, we need to modify the circuit to respect the symmetry. We can achieve this by inserting a layer of Hadamard gates before and after the CNOT gate, and adding an additional angle parameter to the RX gates:

In [ ]:

```
@qml.qnode(dev)
def qnn_circuit_sym(x, weights):
    # apply Hadamard gates
    for i in range(n_qubits):
        qml.Hadamard(wires=i)

    for i in range(n_qubits):
        qml.RX(x[i], wires=i)

    # apply Hadamard gates
    for i in range(n_qubits):
        qml.Hadamard(wires=i)

    qml.CNOT(wires=[0, 1])

    # apply Hadamard gates
    for i in range(n_qubits):
        qml.Hadamard(wires=i)

    qml.RZ(weights[0], wires=0)
    qml.CNOT(wires=[0, 1])
    qml.RZ(weights[1], wires=1)

    # apply Hadamard gates
    for i in range(n_qubits):
        qml.Hadamard(wires=i)

    for i in range(n_qubits):
        qml.RX(x[i], wires=i)

    # apply Hadamard gates
    for i in range(n_qubits):
        qml.Hadamard(wires=i)

    for i in range(n_qubits):
        qml.RX(weights[2+i]*np.pi, wires=i)

    return qml.expval(qml.PauliZ(0))
```

We can then train the Z_2 x Z_2 equivariant QNN by replacing qnn_circuit with qnn_circuit_sym in the training loop.

Note that we also need to modify the loss function to use the qnn_circuit_sym function:

In [ ]:

```
def qnn_loss_sym(weights, X, y_true):
    loss = 0

    for i in range(len(X)):
        # apply the symmetry transformation
        X_sym = [X[i][1], X[i][0]]
        y_sym = y_true[i]

        # compute the output of the QNN
        y_pred = qnn_circuit_sym(X_sym, weights)

        # compute the loss
        loss += (y_true[i] - y_pred)**2

        # compute the output of the symmetric QNN
        y_pred_sym = qnn_circuit_sym(X[i], weights)

        # compute the loss
        loss += (y_true[i] - y_pred_sym)**2

    return loss / (2 * len(X))
```

Here, we apply the symmetry transformation to the input data and the target labels before computing the output of the QNN. We also compute the output of the QNN with the original input data to compute the loss.

With these modifications, we can train the Z_2 x Z_2 equivariant QNN:

In [ ]:

```
np.random.seed(42)
weights = np.random.rand(n_qubits+3)
opt = qml.GradientDescentOptimizer(0.1)

n_epochs = 100
batch_size = 32
n_batches = len(X_train) // batch_size

for epoch in range(n_epochs):
    # shuffle the training data
    permutation = np.random.permutation(len(X_train))
    X_train = X_train[permutation]
    y_train = y_train[permutation]

    for i in range(n_batches):
        # select the next batch
        X_batch = X_train[i*batch_size:(i+1)*batch_size]
        y_batch = y_train[i*batch_size:(i+1)*batch_size]

        # update the weights
        weights = opt.step(lambda w: qnn_loss_sym(w, X_batch, y_batch), weights)

    # compute the accuracy on the test set
    y_pred = [qml.math.sign(qnn_circuit_sym(x, weights)) for x in X_test]
    acc = np.sum(y_pred == y_test) / len(y_test)

    print(f"Epoch {epoch+1}/{n_epochs}, Test accuracy: {acc:.3f}")

print(f"Final test accuracy: {acc:.3f}")
```

```
Epoch 1/100, Test accuracy: 0.510
Epoch 2/100, Test accuracy: 0.512
Epoch 3/100, Test accuracy: 0.512
Epoch 4/100, Test accuracy: 0.512
Epoch 5/100, Test accuracy: 0.512
Epoch 6/100, Test accuracy: 0.512
Epoch 7/100, Test accuracy: 0.512
Epoch 8/100, Test accuracy: 0.512
Epoch 9/100, Test accuracy: 0.512
Epoch 10/100, Test accuracy: 0.512
Epoch 11/100, Test accuracy: 0.512
Epoch 12/100, Test accuracy: 0.512
Epoch 13/100, Test accuracy: 0.512
Epoch 14/100, Test accuracy: 0.512
Epoch 15/100, Test accuracy: 0.512
Epoch 16/100, Test accuracy: 0.512
Epoch 17/100, Test accuracy: 0.512
Epoch 18/100, Test accuracy: 0.512
Epoch 19/100, Test accuracy: 0.512
Epoch 20/100, Test accuracy: 0.512
Epoch 21/100, Test accuracy: 0.512
Epoch 22/100, Test accuracy: 0.512
Epoch 23/100, Test accuracy: 0.512
Epoch 24/100, Test accuracy: 0.512
Epoch 25/100, Test accuracy: 0.512
Epoch 26/100, Test accuracy: 0.512
Epoch 27/100, Test accuracy: 0.512
Epoch 28/100, Test accuracy: 0.512
Epoch 29/100, Test accuracy: 0.512
Epoch 30/100, Test accuracy: 0.512
Epoch 31/100, Test accuracy: 0.512
Epoch 32/100, Test accuracy: 0.512
Epoch 33/100, Test accuracy: 0.512
Epoch 34/100, Test accuracy: 0.512
Epoch 35/100, Test accuracy: 0.512
Epoch 36/100, Test accuracy: 0.512
Epoch 37/100, Test accuracy: 0.512
Epoch 38/100, Test accuracy: 0.512
Epoch 39/100, Test accuracy: 0.512
Epoch 40/100, Test accuracy: 0.512
Epoch 41/100, Test accuracy: 0.512
Epoch 42/100, Test accuracy: 0.512
Epoch 43/100, Test accuracy: 0.512
Epoch 44/100, Test accuracy: 0.512
Epoch 45/100, Test accuracy: 0.512
Epoch 46/100, Test accuracy: 0.512
Epoch 47/100, Test accuracy: 0.512
Epoch 48/100, Test accuracy: 0.512
Epoch 49/100, Test accuracy: 0.512
Epoch 50/100, Test accuracy: 0.512
Epoch 51/100, Test accuracy: 0.512
Epoch 52/100, Test accuracy: 0.512
Epoch 53/100, Test accuracy: 0.512
Epoch 54/100, Test accuracy: 0.512
Epoch 55/100, Test accuracy: 0.512
Epoch 56/100, Test accuracy: 0.512
Epoch 57/100, Test accuracy: 0.512
Epoch 58/100, Test accuracy: 0.512
Epoch 59/100, Test accuracy: 0.512
Epoch 60/100, Test accuracy: 0.512
Epoch 61/100, Test accuracy: 0.512
Epoch 62/100, Test accuracy: 0.512
Epoch 63/100, Test accuracy: 0.512
Epoch 64/100, Test accuracy: 0.512
Epoch 65/100, Test accuracy: 0.512
Epoch 66/100, Test accuracy: 0.512
Epoch 67/100, Test accuracy: 0.512
Epoch 68/100, Test accuracy: 0.512
Epoch 69/100, Test accuracy: 0.512
Epoch 70/100, Test accuracy: 0.512
Epoch 71/100, Test accuracy: 0.512
Epoch 72/100, Test accuracy: 0.512
Epoch 73/100, Test accuracy: 0.512
Epoch 74/100, Test accuracy: 0.512
Epoch 75/100, Test accuracy: 0.512
Epoch 76/100, Test accuracy: 0.512
Epoch 77/100, Test accuracy: 0.512
Epoch 78/100, Test accuracy: 0.512
Epoch 79/100, Test accuracy: 0.512
Epoch 80/100, Test accuracy: 0.512
Epoch 81/100, Test accuracy: 0.512
Epoch 82/100, Test accuracy: 0.512
Epoch 83/100, Test accuracy: 0.512
Epoch 84/100, Test accuracy: 0.512
Epoch 85/100, Test accuracy: 0.512
Epoch 86/100, Test accuracy: 0.512
Epoch 87/100, Test accuracy: 0.512
Epoch 88/100, Test accuracy: 0.512
Epoch 89/100, Test accuracy: 0.512
Epoch 90/100, Test accuracy: 0.512
Epoch 91/100, Test accuracy: 0.512
Epoch 92/100, Test accuracy: 0.512
Epoch 93/100, Test accuracy: 0.512
Epoch 94/100, Test accuracy: 0.512
Epoch 95/100, Test accuracy: 0.512
Epoch 96/100, Test accuracy: 0.512
Epoch 97/100, Test accuracy: 0.512
Epoch 98/100, Test accuracy: 0.512
Epoch 99/100, Test accuracy: 0.512
Epoch 100/100, Test accuracy: 0.512
Final test accuracy: 0.512
```

we should see that the Z_2 x Z_2 equivariant QNN achieves a higher test accuracy than the standard QNN. This is because the equivariant QNN is able to exploit the symmetry of the dataset to improve its performance.