0724-small_hybrid_model_v2

July 25, 2023

[1]:
```
import torch
import torch.nn as nn
import torch.optim as optim
```

[2]:
```
import numpy as np
```

[3]:
```
import sys
sys.path.append('..')

from my_code import functions as f
```

1 Data

[4]:
```
file_path = '../data/energies/Trial/Strings_Energies_4_aa.txt'  # Replace with␣
 ↪the actual path to your 'data.txt' file
string_list, number_list = f.read_data_file(file_path)
score_list = np.array(number_list)/1000
angles_list = np.array([f.string_to_numbers(string) for string in string_list])
```

[5]:
```
X, Y, X_validation, Y_validation = f.create_validating_set(angles_list,␣
 ↪score_list, percentage=0.1)
```

[6]:
```
# Define the dataset
input_data = torch.tensor(X, dtype=torch.float32)
target_data = torch.tensor(Y, dtype=torch.float32).view(-1, 1)

# Define the validation set
input_validation = torch.tensor(X_validation, dtype=torch.float32)
target_validation = torch.tensor(Y_validation, dtype=torch.float32).view(-1, 1)
```

2 Quantum node

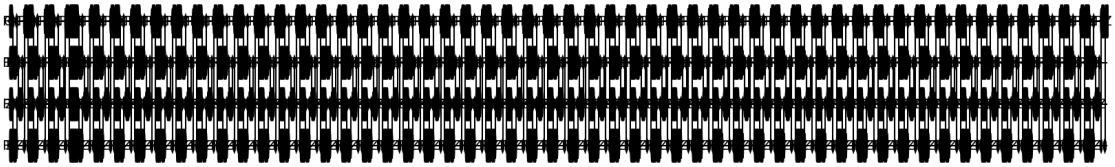
[7]:
```
def qml_RZZ(params, wires):
    """
    RZZ gate.
    """
```

```
    qml.CNOT(wires=wires)
    qml.RZ(params, wires=wires[1])
    qml.CNOT(wires=wires)
```

[8]:
```
import pennylane as qml

n_qubits = 4
n_layers_block = 50
n_layers_embedding = 3
n_layers = n_layers_block + n_layers_embedding
n_params = 5
dev = qml.device("default.qubit", wires=n_qubits)

@qml.qnode(dev)
def qnode(inputs, weights):

    # state preparation (we create an embedding with 3 layers, paper: 2001.
 ↪03622)
    for i in range(n_layers_embedding):

        # angle embedding for each qubit
        qml.AngleEmbedding(inputs, wires=range(n_qubits))

        # ZZ rotation for neighboring qubits
        for x in range(2):
            for j in range(x,n_qubits,2):
                qml_RZZ(weights[i,j,0], wires=[j, (j+1)%n_qubits])

        # rotations for each qubit
        for j in range(n_qubits):
            qml.RY(weights[i,j,1], wires=j)

        # last angle embedding
    qml.AngleEmbedding(inputs, wires=range(n_qubits))

    #layers
    for i in range(n_layers_embedding, n_layers):
        # rotations for each qubit
        for j in range(n_qubits):
            qml.RX(weights[i,j,2], wires=j)
            qml.RZ(weights[i,j,3], wires=j)

        # ZZ rotation for neighboring qubits
        for x in range(2):
            for j in range(x,n_qubits,2):
                qml_RZZ(weights[i,j,4], wires=[j, (j+1)%n_qubits])
```

```
    # measurement
    return [qml.expval(qml.PauliZ(wires=0))]
```

[9]:
```
qml.drawer.use_style("black_white")
fig, ax = qml.draw_mpl(qnode, expansion_strategy="device")([i for i in␣
 ↪range(n_qubits)], np.zeros((n_layers, n_qubits, n_params)))
fig.set_size_inches((16,3))
```



[10]:
```
weight_shapes = {"weights": (n_layers, n_qubits, n_params)}
```

[11]:
```
qlayer = qml.qnn.TorchLayer(qnode, weight_shapes)
```

3 Hybrid model

[12]:
```
input_dim = input_data.size(1)

layers =  [nn.Linear(input_dim*1, input_dim*2), nn.ReLU()]
layers += [nn.Linear(input_dim*2, input_dim*3), nn.ReLU()]
layers += [nn.Linear(input_dim*3, input_dim*3), nn.ReLU()]
layers += [nn.Linear(input_dim*3, input_dim*2), nn.ReLU()]
layers += [nn.Linear(input_dim*2, input_dim*1)]
layers += [qlayer]
Net = nn.Sequential(*layers)
```

[13]:
```
# Create an instance of the network
model = Net
```

[14]:
```
import time
```

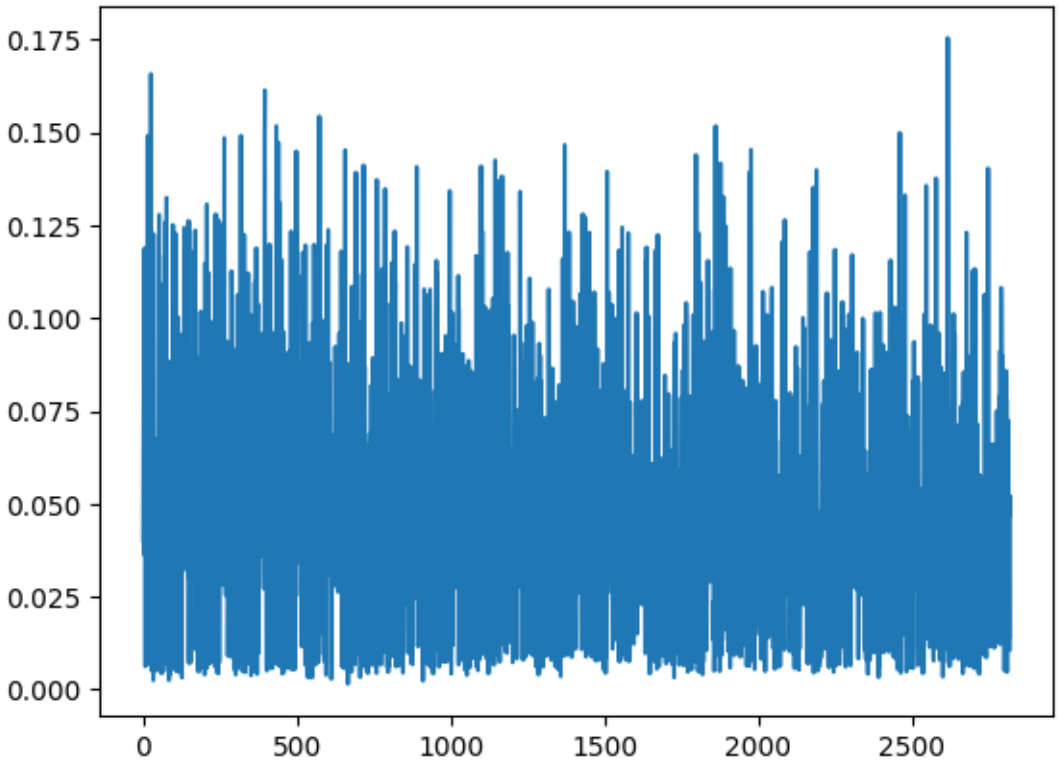
[15]:
```
# time
start_time = time.time()


# Define the loss function and optimizer
criterion = nn.MSELoss()  # Mean Squared Error loss
# optimizer = optim.Adam(model.parameters(), lr=0.001)  # Adam optimizer with␣
 ↪learning rate 0.001
```

```
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Training loop
num_epochs = 10
batch_size = 32

losses = []
losses_epochs = []

for epoch in range(num_epochs):
    # Shuffle the dataset
    indices = torch.randperm(input_data.size(0))
    input_data = input_data[indices]
    target_data = target_data[indices]

    losses_epochs.append(0)

    # Mini-batch training
    for i in range(0, input_data.size(0), batch_size):
        inputs = input_data[i:i+batch_size]
        targets = target_data[i:i+batch_size]

        # Forward pass
        outputs = model(inputs)

        # Compute the loss
        loss = criterion(outputs, targets)

        # Backward pass and optimization
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        # Store the loss
        losses.append(loss.item())

        print('- Epoch [{}/{}], i: [{}/{}], Loss: {:.4f}'.format(epoch+1,␣
 ↪num_epochs, i, input_data.size(0), loss.item()), end='\r')

        # add to the epoch loss
        losses_epochs[-1] += loss.item() / (input_data.size(0) / batch_size)

    # time
    # Compute elapsed time and remaining time
    elapsed_time = time.time() - start_time
    avg_time_per_epoch = elapsed_time / (epoch + 1)
    remaining_epochs = num_epochs - (epoch + 1)
```

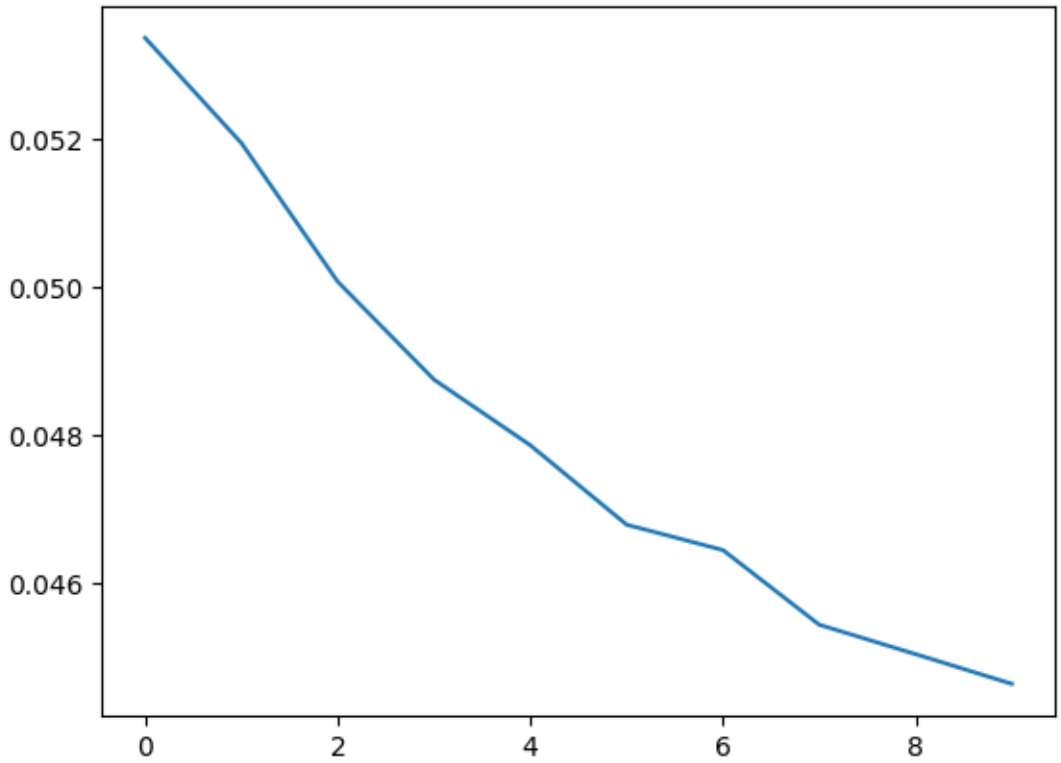
```
    estimated_remaining_time = avg_time_per_epoch * remaining_epochs

    # Convert remaining time to hours, minutes, and seconds for better␣
↪readability
    hours, remainder = divmod(estimated_remaining_time, 3600)
    minutes, seconds = divmod(remainder, 60)

    # Print the loss and remaining time for this epoch
    print('Epoch [{}/{}], Loss: {:.4f}, Time remaining: ~{}h {}m {:.0f}s'.
↪format(
          epoch+1, num_epochs, losses_epochs[-1], hours, minutes, seconds))
```

```
Epoch [1/10], Loss: 0.0534, Time remaining: ~0.0h 55.0m 11s
Epoch [2/10], Loss: 0.0519, Time remaining: ~0.0h 52.0m 24s
Epoch [3/10], Loss: 0.0501, Time remaining: ~0.0h 46.0m 55s
Epoch [4/10], Loss: 0.0487, Time remaining: ~0.0h 40.0m 34s
Epoch [5/10], Loss: 0.0479, Time remaining: ~0.0h 34.0m 3s
Epoch [6/10], Loss: 0.0468, Time remaining: ~0.0h 26.0m 57s
Epoch [7/10], Loss: 0.0464, Time remaining: ~0.0h 20.0m 2s
Epoch [8/10], Loss: 0.0454, Time remaining: ~0.0h 13.0m 17s
Epoch [9/10], Loss: 0.0450, Time remaining: ~0.0h 6.0m 37s
Epoch [10/10], Loss: 0.0446, Time remaining: ~0.0h 0.0m 0s
```

[16]:
```
#plot the loss
import matplotlib.pyplot as plt
plt.plot(losses)
plt.show()
```



[17]:
```
#plot the loss
import matplotlib.pyplot as plt
plt.plot(losses_epochs)
plt.show()
```



[18]:
```
avg_loss = 0
for x, (i, t) in enumerate(zip((input_validation), target_validation)):
    loss = criterion(model(i), t)
    avg_loss += loss/len(target_validation)
    print('i: {}, target: {:.3f}, output: {:.3f}, loss: {:.3f}'.format(x, t.
 ↪item(), model(i).item(), loss))
    print('')

print('Average loss: {:.3f}'.format(avg_loss))
```

```
i: 0, target: 0.052, output: 0.169, loss: 0.014

i: 1, target: -0.066, output: -0.080, loss: 0.000

i: 2, target: 0.955, output: 0.100, loss: 0.732

i: 3, target: 0.014, output: 0.089, loss: 0.006

i: 4, target: -0.004, output: 0.173, loss: 0.031

i: 5, target: -0.125, output: -0.118, loss: 0.000
```

```
i: 6, target: -0.149, output: -0.121, loss: 0.001
i: 7, target: -0.058, output: -0.050, loss: 0.000
i: 8, target: -0.071, output: 0.102, loss: 0.030
i: 9, target: -0.040, output: -0.077, loss: 0.001
i: 10, target: -0.192, output: -0.123, loss: 0.005
i: 11, target: -0.030, output: -0.109, loss: 0.006
i: 12, target: -0.280, output: -0.123, loss: 0.025
i: 13, target: 0.922, output: 0.003, loss: 0.845
i: 14, target: -0.060, output: -0.122, loss: 0.004
i: 15, target: -0.181, output: -0.110, loss: 0.005
i: 16, target: -0.176, output: 0.083, loss: 0.067
i: 17, target: -0.060, output: 0.256, loss: 0.100
i: 18, target: -0.120, output: 0.001, loss: 0.015
i: 19, target: -0.111, output: -0.095, loss: 0.000
i: 20, target: 0.885, output: 0.110, loss: 0.601
i: 21, target: -0.053, output: -0.075, loss: 0.001
i: 22, target: -0.086, output: -0.080, loss: 0.000
i: 23, target: -0.006, output: 0.122, loss: 0.016
i: 24, target: -0.115, output: 0.193, loss: 0.095
i: 25, target: -0.154, output: -0.098, loss: 0.003
i: 26, target: -0.160, output: -0.110, loss: 0.003
i: 27, target: 0.961, output: 0.064, loss: 0.804
i: 28, target: 0.089, output: 0.057, loss: 0.001
i: 29, target: -0.112, output: -0.099, loss: 0.000
```

```
i: 30, target: -0.134, output: -0.133, loss: 0.000
i: 31, target: -0.035, output: -0.098, loss: 0.004
i: 32, target: 0.047, output: 0.131, loss: 0.007
i: 33, target: -0.046, output: 0.016, loss: 0.004
i: 34, target: -0.168, output: -0.061, loss: 0.011
i: 35, target: 0.046, output: 0.201, loss: 0.024
i: 36, target: -0.134, output: -0.122, loss: 0.000
i: 37, target: -0.143, output: -0.114, loss: 0.001
i: 38, target: -0.043, output: -0.099, loss: 0.003
i: 39, target: -0.062, output: -0.070, loss: 0.000
i: 40, target: -0.063, output: -0.054, loss: 0.000
i: 41, target: -0.060, output: -0.117, loss: 0.003
i: 42, target: -0.143, output: 0.048, loss: 0.036
i: 43, target: 0.049, output: 0.184, loss: 0.018
i: 44, target: 0.891, output: 0.077, loss: 0.662
i: 45, target: -0.060, output: -0.134, loss: 0.005
i: 46, target: -0.123, output: 0.047, loss: 0.029
i: 47, target: -0.139, output: -0.129, loss: 0.000
i: 48, target: -0.130, output: 0.138, loss: 0.072
i: 49, target: -0.048, output: -0.128, loss: 0.006
i: 50, target: -0.037, output: -0.116, loss: 0.006
i: 51, target: -0.085, output: 0.053, loss: 0.019
i: 52, target: -0.009, output: 0.169, loss: 0.032
i: 53, target: -0.100, output: -0.104, loss: 0.000
```

```
i: 54, target: 0.020, output: -0.108, loss: 0.016
i: 55, target: -0.036, output: -0.138, loss: 0.010
i: 56, target: -0.103, output: 0.089, loss: 0.037
i: 57, target: -0.086, output: -0.122, loss: 0.001
i: 58, target: -0.186, output: -0.091, loss: 0.009
i: 59, target: -0.061, output: -0.043, loss: 0.000
i: 60, target: -0.139, output: -0.131, loss: 0.000
i: 61, target: 0.066, output: 0.104, loss: 0.001
i: 62, target: -0.079, output: -0.117, loss: 0.001
i: 63, target: -0.065, output: -0.076, loss: 0.000
i: 64, target: -0.156, output: -0.076, loss: 0.006
i: 65, target: -0.120, output: -0.052, loss: 0.005
i: 66, target: -0.092, output: -0.147, loss: 0.003
i: 67, target: -0.127, output: 0.117, loss: 0.059
i: 68, target: -0.064, output: -0.105, loss: 0.002
i: 69, target: -0.145, output: -0.125, loss: 0.000
i: 70, target: -0.138, output: -0.138, loss: 0.000
i: 71, target: 0.044, output: 0.182, loss: 0.019
i: 72, target: -0.061, output: -0.120, loss: 0.003
i: 73, target: -0.185, output: -0.117, loss: 0.005
i: 74, target: -0.095, output: -0.130, loss: 0.001
i: 75, target: -0.078, output: -0.061, loss: 0.000
i: 76, target: -0.188, output: -0.140, loss: 0.002
i: 77, target: -0.154, output: 0.134, loss: 0.083
```

```
i: 78, target: -0.115, output: 0.179, loss: 0.086
i: 79, target: -0.070, output: -0.126, loss: 0.003
i: 80, target: -0.001, output: 0.095, loss: 0.009
i: 81, target: -0.059, output: -0.120, loss: 0.004
i: 82, target: -0.108, output: -0.141, loss: 0.001
i: 83, target: -0.079, output: 0.141, loss: 0.049
i: 84, target: -0.103, output: -0.131, loss: 0.001
i: 85, target: -0.076, output: -0.133, loss: 0.003
i: 86, target: -0.130, output: -0.126, loss: 0.000
i: 87, target: -0.074, output: -0.126, loss: 0.003
i: 88, target: -0.057, output: -0.086, loss: 0.001
i: 89, target: -0.054, output: -0.132, loss: 0.006
i: 90, target: -0.105, output: -0.114, loss: 0.000
i: 91, target: -0.059, output: -0.153, loss: 0.009
i: 92, target: 0.861, output: 0.117, loss: 0.553
i: 93, target: -0.175, output: -0.128, loss: 0.002
i: 94, target: -0.171, output: -0.123, loss: 0.002
i: 95, target: -0.046, output: -0.131, loss: 0.007
i: 96, target: -0.098, output: -0.098, loss: 0.000
i: 97, target: -0.128, output: -0.081, loss: 0.002
i: 98, target: -0.152, output: -0.085, loss: 0.005
i: 99, target: -0.178, output: -0.129, loss: 0.002
i: 100, target: -0.061, output: 0.171, loss: 0.054
i: 101, target: -0.127, output: -0.123, loss: 0.000
```

```
i: 102, target: -0.053, output: -0.076, loss: 0.001
i: 103, target: -0.143, output: -0.126, loss: 0.000
i: 104, target: -0.100, output: -0.110, loss: 0.000
i: 105, target: -0.107, output: -0.105, loss: 0.000
i: 106, target: -0.189, output: -0.092, loss: 0.009
i: 107, target: -0.046, output: -0.125, loss: 0.006
i: 108, target: -0.080, output: -0.028, loss: 0.003
i: 109, target: -0.150, output: -0.137, loss: 0.000
i: 110, target: -0.104, output: -0.094, loss: 0.000
i: 111, target: -0.149, output: 0.117, loss: 0.071
i: 112, target: -0.055, output: -0.103, loss: 0.002
i: 113, target: -0.066, output: 0.102, loss: 0.028
i: 114, target: -0.038, output: -0.047, loss: 0.000
i: 115, target: 0.952, output: 0.094, loss: 0.737
i: 116, target: -0.053, output: -0.145, loss: 0.008
i: 117, target: -0.073, output: -0.135, loss: 0.004
i: 118, target: -0.096, output: -0.110, loss: 0.000
i: 119, target: -0.099, output: -0.092, loss: 0.000
i: 120, target: 0.975, output: 0.156, loss: 0.671
i: 121, target: -0.105, output: -0.140, loss: 0.001
i: 122, target: -0.105, output: -0.107, loss: 0.000
i: 123, target: -0.124, output: -0.090, loss: 0.001
i: 124, target: -0.083, output: -0.012, loss: 0.005
i: 125, target: -0.091, output: 0.070, loss: 0.026
```

```
i: 126, target: -0.093, output: -0.147, loss: 0.003
i: 127, target: -0.115, output: 0.137, loss: 0.063
i: 128, target: -0.149, output: -0.118, loss: 0.001
i: 129, target: -0.090, output: -0.073, loss: 0.000
i: 130, target: -0.170, output: -0.120, loss: 0.003
i: 131, target: -0.120, output: -0.125, loss: 0.000
i: 132, target: -0.195, output: -0.110, loss: 0.007
i: 133, target: -0.054, output: -0.046, loss: 0.000
i: 134, target: -0.138, output: -0.103, loss: 0.001
i: 135, target: -0.103, output: -0.066, loss: 0.001
i: 136, target: -0.070, output: -0.111, loss: 0.002
i: 137, target: -0.014, output: 0.136, loss: 0.022
i: 138, target: -0.152, output: -0.110, loss: 0.002
i: 139, target: -0.160, output: -0.000, loss: 0.025
i: 140, target: -0.105, output: -0.019, loss: 0.007
i: 141, target: -0.110, output: 0.135, loss: 0.060
i: 142, target: -0.138, output: -0.096, loss: 0.002
i: 143, target: 0.971, output: 0.112, loss: 0.737
i: 144, target: -0.065, output: -0.074, loss: 0.000
i: 145, target: -0.053, output: 0.122, loss: 0.030
i: 146, target: -0.101, output: -0.096, loss: 0.000
i: 147, target: -0.164, output: 0.174, loss: 0.115
i: 148, target: -0.124, output: -0.119, loss: 0.000
i: 149, target: -0.119, output: -0.127, loss: 0.000
```

```
i: 150, target: -0.026, output: -0.141, loss: 0.013
i: 151, target: -0.106, output: -0.116, loss: 0.000
i: 152, target: 0.915, output: 0.106, loss: 0.655
i: 153, target: -0.141, output: -0.130, loss: 0.000
i: 154, target: -0.067, output: -0.046, loss: 0.000
i: 155, target: -0.133, output: 0.004, loss: 0.019
i: 156, target: -0.087, output: 0.072, loss: 0.025
i: 157, target: -0.093, output: -0.114, loss: 0.000
i: 158, target: -0.150, output: 0.037, loss: 0.035
i: 159, target: -0.095, output: 0.169, loss: 0.070
i: 160, target: -0.075, output: -0.130, loss: 0.003
i: 161, target: -0.056, output: -0.130, loss: 0.006
i: 162, target: -0.030, output: -0.120, loss: 0.008
i: 163, target: -0.082, output: -0.126, loss: 0.002
i: 164, target: -0.012, output: -0.095, loss: 0.007
i: 165, target: -0.141, output: -0.090, loss: 0.003
i: 166, target: -0.125, output: -0.046, loss: 0.006
i: 167, target: -0.133, output: -0.026, loss: 0.011
i: 168, target: -0.131, output: 0.079, loss: 0.044
i: 169, target: -0.192, output: -0.110, loss: 0.007
i: 170, target: 0.985, output: 0.045, loss: 0.885
i: 171, target: -0.215, output: -0.111, loss: 0.011
i: 172, target: 0.940, output: 0.124, loss: 0.665
i: 173, target: 0.864, output: 0.090, loss: 0.599
```

```
i: 174, target: -0.140, output: -0.130, loss: 0.000
i: 175, target: -0.058, output: -0.106, loss: 0.002
i: 176, target: -0.078, output: -0.115, loss: 0.001
i: 177, target: -0.005, output: -0.143, loss: 0.019
i: 178, target: -0.118, output: -0.099, loss: 0.000
i: 179, target: -0.087, output: -0.112, loss: 0.001
i: 180, target: -0.196, output: -0.119, loss: 0.006
i: 181, target: -0.144, output: -0.120, loss: 0.001
i: 182, target: -0.171, output: -0.101, loss: 0.005
i: 183, target: -0.086, output: -0.118, loss: 0.001
i: 184, target: -0.123, output: -0.133, loss: 0.000
i: 185, target: 0.024, output: 0.148, loss: 0.015
i: 186, target: -0.089, output: -0.130, loss: 0.002
i: 187, target: -0.034, output: 0.013, loss: 0.002
i: 188, target: -0.051, output: -0.049, loss: 0.000
i: 189, target: -0.047, output: -0.123, loss: 0.006
i: 190, target: -0.134, output: -0.140, loss: 0.000
i: 191, target: -0.079, output: -0.090, loss: 0.000
i: 192, target: -0.171, output: -0.112, loss: 0.004
i: 193, target: -0.113, output: -0.062, loss: 0.003
i: 194, target: -0.134, output: -0.099, loss: 0.001
i: 195, target: -0.047, output: -0.091, loss: 0.002
i: 196, target: -0.196, output: -0.122, loss: 0.005
i: 197, target: -0.091, output: -0.124, loss: 0.001
```

```
i: 198, target: -0.138, output: -0.115, loss: 0.001
i: 199, target: -0.144, output: -0.132, loss: 0.000
i: 200, target: -0.230, output: -0.121, loss: 0.012
i: 201, target: -0.169, output: -0.126, loss: 0.002
i: 202, target: -0.212, output: -0.119, loss: 0.009
i: 203, target: -0.087, output: -0.125, loss: 0.001
i: 204, target: -0.172, output: -0.137, loss: 0.001
i: 205, target: -0.148, output: -0.127, loss: 0.000
i: 206, target: -0.091, output: -0.038, loss: 0.003
i: 207, target: -0.097, output: -0.122, loss: 0.001
i: 208, target: -0.123, output: -0.112, loss: 0.000
i: 209, target: -0.085, output: -0.109, loss: 0.001
i: 210, target: -0.062, output: -0.059, loss: 0.000
i: 211, target: -0.110, output: -0.099, loss: 0.000
i: 212, target: 0.931, output: 0.070, loss: 0.742
i: 213, target: 0.950, output: 0.056, loss: 0.800
i: 214, target: -0.045, output: -0.125, loss: 0.006
i: 215, target: -0.062, output: -0.115, loss: 0.003
i: 216, target: -0.013, output: 0.078, loss: 0.008
i: 217, target: -0.071, output: -0.094, loss: 0.001
i: 218, target: -0.091, output: -0.131, loss: 0.002
i: 219, target: -0.063, output: 0.065, loss: 0.016
i: 220, target: -0.090, output: -0.129, loss: 0.002
i: 221, target: -0.059, output: -0.147, loss: 0.008
```

```
i: 222, target: -0.094, output: -0.108, loss: 0.000
```

4 Save the Notebook as a PDF

[ ]:
```python
# SAVE THE NOTEBOOK

from IPython.display import Javascript

# Define the function to save the notebook
def save_notebook():
    display(Javascript('IPython.notebook.save_notebook()'))

# Call the save_notebook function to save the notebook
save_notebook()
```

[ ]:
```python
import subprocess
import os

name_notebook = "0724-small_hybrid_model_v2.ipynb"

output_filename = "results/"+ name_notebook[:4] +"/" + name_notebook[:-6] + "_0.
 ↪pdf"

#check if the output file already exists
while os.path.exists(output_filename):
    print("The file {} already exists".format(output_filename))
    output_filename = output_filename[:-5] + str(int(output_filename[-5]) + 1)␣
 ↪+ ".pdf"
    print("Trying to save the file as {}".format(output_filename))


subprocess.run(["jupyter", "nbconvert", "--to", "pdf", "--output",␣
 ↪output_filename, name_notebook])
```

[ ]:
```
CompletedProcess(args=['jupyter', 'nbconvert', '--to', 'pdf', '--output',
'results/0724/0724-small_hybrid_model_0.pdf', '0724-small_hybrid_model.ipynb'],
returncode=0)
```