0802-sh_1qm

August 2, 2023

[ ]:
```
initial_path = 'peptide-QML/'
initial_path = '../'
```

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

[ ]:
```
import sys
sys.path.append(initial_path)

from my_code import functions as f
from my_code import pytorch_model as ptm
from my_code import quantum_nodes as qn
from my_code import pytorch_wrappers as pw
```

1 Data

[ ]:
```
file_path = initial_path + 'data/energies/PET/generated/
 ↪bb14_Strings_Energies_10_000_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
vector_list = np.array([f.string_to_vector(string) for string in string_list])␣
 ↪# one hot encoding
```

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

X = X.reshape(X.shape[0], X.shape[1]*X.shape[2]) # flatten
X_validation = X_validation.reshape(X_validation.shape[0], X_validation.
 ↪shape[1]*X_validation.shape[2]) # flatten
```

[ ]:
```
# Define the dataset
input_data = ptm.torch.tensor(X, dtype=ptm.torch.float64)
target_data = ptm.torch.tensor(Y, dtype=ptm.torch.float64).view(-1, 1)

# Define the validation set
input_validation = ptm.torch.tensor(X_validation, dtype=ptm.torch.float64)
```

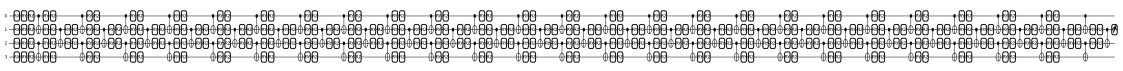
```
target_validation = ptm.torch.tensor(Y_validation, dtype=ptm.torch.float64).
 ↪view(-1, 1)
```

2 Quantum node

[ ]:
```
n_aminoacids = len(string_list[0])
```

[ ]:
```
quantum_layer = qn.circuit(
    n_qubits = n_aminoacids,
    device = "default.qubit.torch",
    device_options = {'shots': None},
    embedding = qn.parts.AngleEmbedding,
    ansatz = qn.parts.Ansatz_11,
    measurement = qn.parts.exp_Z(1),
    n_layers = 25,
    # wrapper_qlayer = pw.QLayer,
    wrapper_qlayer = None,
)
```

[ ]:
```
quantum_layer.draw(size=(50,3))
```



3 Hybrid model

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

n_pre_classical_layers = 4
layers_dim = np.linspace(n_aminoacids, input_dim, 4).astype(int)
```

[ ]:
```
layers = []
for i in range(1, len(layers_dim)):
    layers += [ptm.nn.Linear(layers_dim[-1*i], layers_dim[-1*(i+1)]), ptm.nn.
 ↪ReLU()]
layers += [ptm.nn.Linear(layers_dim[0], layers_dim[0])]
layers += [quantum_layer()]
# layers += [nn.Linear(1, 1)]
# layers += [nn.Linear(2, 4), nn.ReLU()]
# layers += [nn.Linear(4, 1)]
```

[ ]:
```
# Create model and set data:
model = ptm.pytorch_model(layers)
```

```
model.set_data(
    data_X=input_data,
    data_Y=target_data,
    data_X_validation=input_validation,
    data_Y_validation=target_validation
)
```

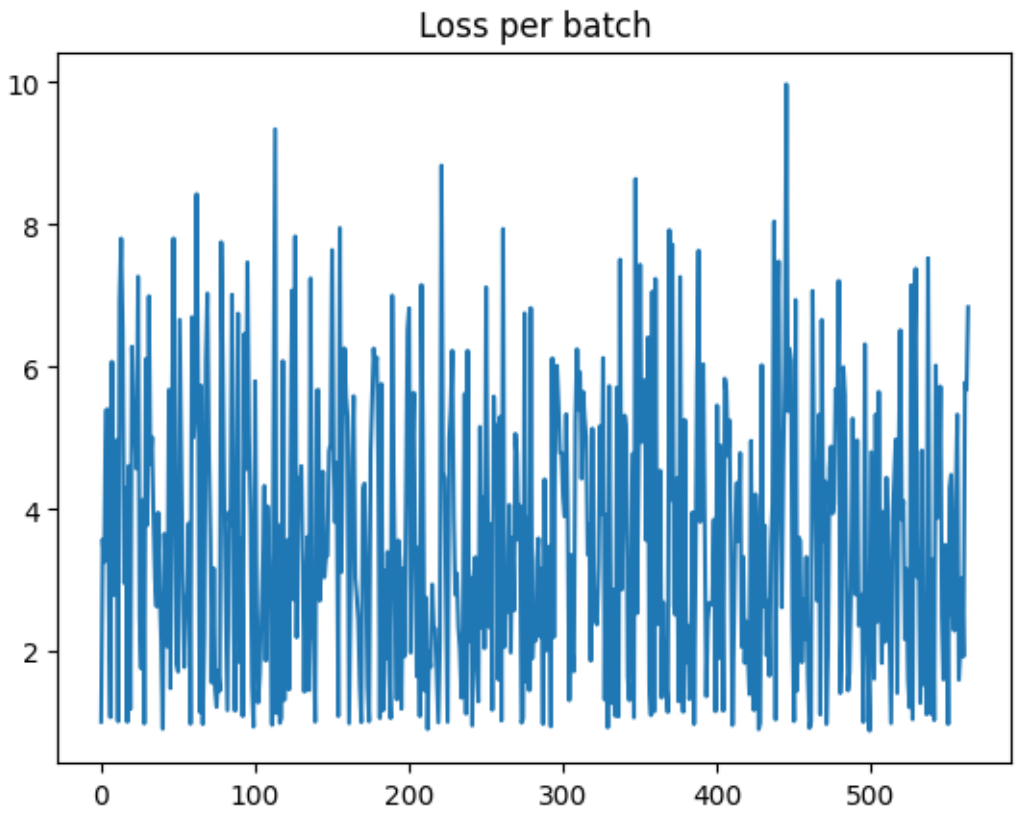
[ ]:
```
print(model(input_data[0]).item())
print(model(input_data[1]).item())
```

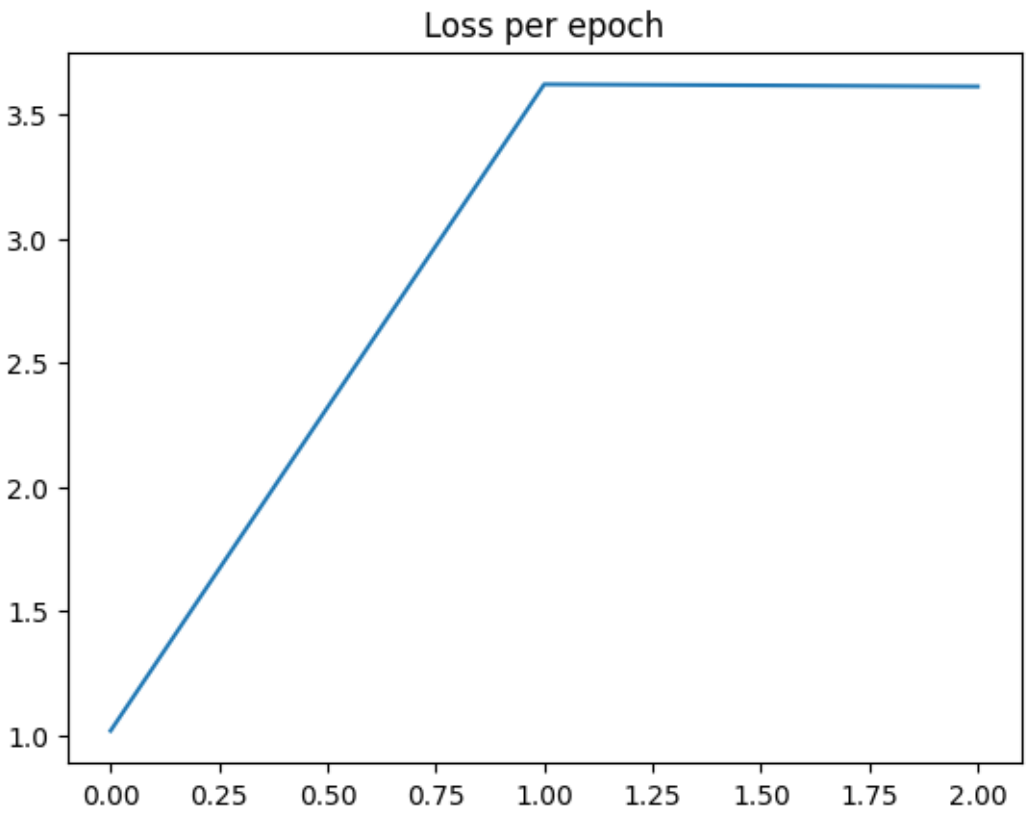
[ ]:
```
tensor([-0.0065], grad_fn=<CatBackward0>)
```

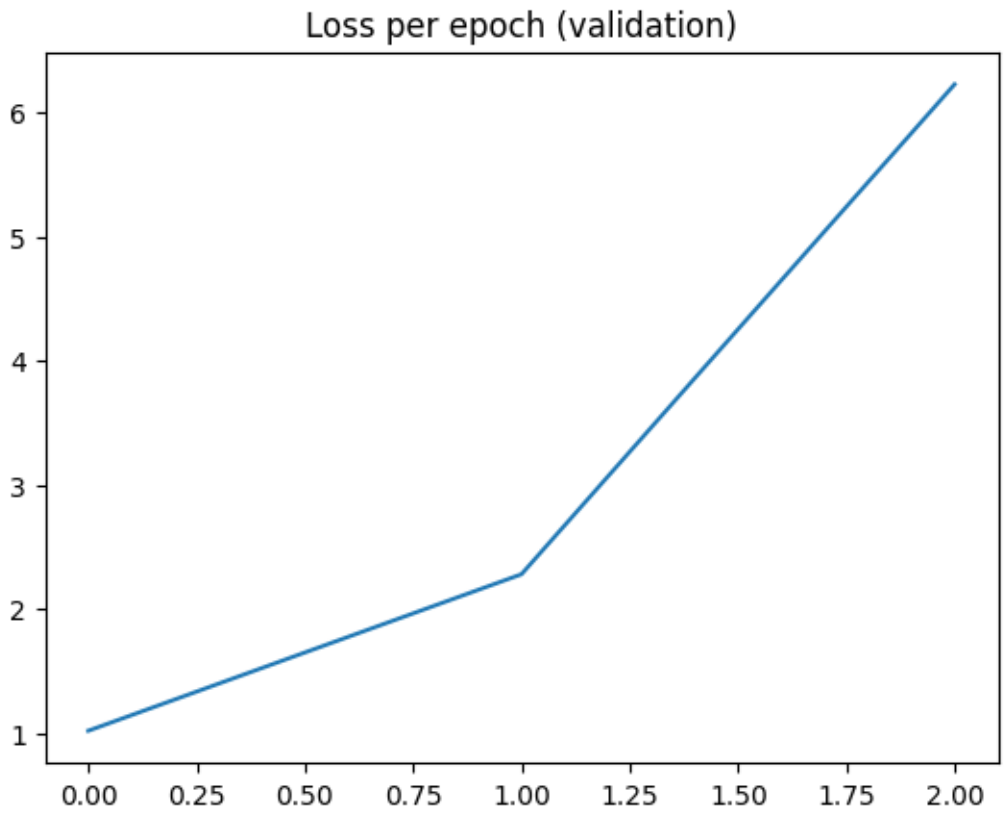
[ ]:
```
# train the model
model.train(
    num_epochs=2,
    batch_size = 32,
)
```

```
Epoch [0/2], Loss: 1.0165, Loss validation: 1.0164
        Validation string,      i: 0;   prediction: 0.2922,     target: 0.1608,
loss: 0.8172
        Validation string,      i: 1;   prediction: 0.2919,     target: 0.1417,
loss: 1.0595
        Validation string,      i: 2;   prediction: 0.2922,     target: 0.1656,
loss: 0.7651
Epoch [1/2], Loss: 3.6225, Loss validation: 2.2778, Time remaining: ~0.0h 2.0m
3s
        Validation string,      i: 0;   prediction: 0.8043,     target: 0.1608,
loss: 4.0026
        Validation string,      i: 1;   prediction: 0.8043,     target: 0.1417,
loss: 4.6752
        Validation string,      i: 2;   prediction: 0.8043,     target: 0.1656,
loss: 3.8584
Epoch [2/2], Loss: 3.6137, Loss validation: 6.2268, Time remaining: ~0.0h 0.0m
0s
```

[ ]:
```
# plot the losses of the trainig loop
model.plot_losses()
```


Loss per batch
10
8
6
4
2
0
100
200
300
400
500


Loss per epoch
3.5
3.0
2.5
2.0
1.5
1.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



[ ]:
```
#save model
name_notebook = "0802-sh_1qm.ipynb"

version = model.save_state_dict(name_notebook=name_notebook,␣
  ↪initial_path=initial_path)
```

```
Model saved as ../Notebooks/models/0802/0802-sh_1qm_0.pth
```

[ ]:
```
# push changes to git
!cd peptide-QML && git add . && git commit -m "data trained model" && git push
```

```
The system cannot find the path specified.
```

[ ]:
```
#load model
model.load_state_dict(name_notebook=name_notebook, version=version,␣
  ↪initial_path=initial_path)
```

```
Model loaded from ../Notebooks/models/0802/0802-sh_1qm_0.pth
```

[ ]:
```
# print validation
model.print_validation()
```

```
i: 0,    target: 0.161,          output: 0.804,          loss: 4.003
```

```
i: 1,    target: 0.138,         output: 0.804,         loss: 4.826
i: 2,    target: -0.071,        output: 0.804,         loss: 12.318
i: 3,    target: 0.200,         output: 0.804,         loss: 3.025
i: 4,    target: -0.031,        output: 0.804,         loss: 26.808
i: 5,    target: 0.196,         output: 0.804,         loss: 3.096
i: 6,    target: -0.076,        output: 0.804,         loss: 11.602
i: 7,    target: -0.033,        output: 0.804,         loss: 25.254
i: 8,    target: -0.088,        output: 0.804,         loss: 10.092
i: 9,    target: -0.058,        output: 0.804,         loss: 14.826
i: 10,   target: 0.142,         output: 0.804,         loss: 4.675
i: 11,   target: 0.029,         output: 0.804,         loss: 26.669
i: 12,   target: 0.017,         output: 0.804,         loss: 46.614
i: 13,   target: 0.193,         output: 0.804,         loss: 3.176
i: 14,   target: -0.076,        output: 0.804,         loss: 11.603
i: 15,   target: 0.196,         output: 0.804,         loss: 3.096
i: 16,   target: -0.080,        output: 0.804,         loss: 11.096
i: 17,   target: -0.075,        output: 0.804,         loss: 11.656
i: 18,   target: 0.073,         output: 0.804,         loss: 10.069
i: 19,   target: 0.020,         output: 0.804,         loss: 38.928
i: 20,   target: 0.166,         output: 0.804,         loss: 3.858
i: 21,   target: 0.048,         output: 0.804,         loss: 15.741
i: 22,   target: 0.187,         output: 0.804,         loss: 3.296
i: 23,   target: -0.084,        output: 0.804,         loss: 10.627
i: 24,   target: -0.067,        output: 0.804,         loss: 12.998
i: 25,   target: -0.073,        output: 0.804,         loss: 11.956
i: 26,   target: -0.070,        output: 0.804,         loss: 12.450
i: 27,   target: 0.146,         output: 0.804,         loss: 4.521
i: 28,   target: 0.120,         output: 0.804,         loss: 5.676
i: 29,   target: 0.001,         output: 0.804,         loss: 618.677
i: 30,   target: -0.018,        output: 0.804,         loss: 44.537
i: 31,   target: -0.094,        output: 0.804,         loss: 9.560
i: 32,   target: -0.082,        output: 0.804,         loss: 10.756
i: 33,   target: 0.299,         output: 0.804,         loss: 1.686
i: 34,   target: 0.127,         output: 0.804,         loss: 5.341
i: 35,   target: -0.112,        output: 0.804,         loss: 8.165
i: 36,   target: -0.063,        output: 0.804,         loss: 13.822
i: 37,   target: -0.066,        output: 0.804,         loss: 13.223
i: 38,   target: -0.041,        output: 0.804,         loss: 20.558
i: 39,   target: -0.082,        output: 0.804,         loss: 10.805
i: 40,   target: 0.127,         output: 0.804,         loss: 5.355
i: 41,   target: -0.046,        output: 0.804,         loss: 18.520
i: 42,   target: -0.073,        output: 0.804,         loss: 11.948
i: 43,   target: -0.025,        output: 0.804,         loss: 32.832
i: 44,   target: -0.087,        output: 0.804,         loss: 10.281
i: 45,   target: 0.437,         output: 0.804,         loss: 0.839
i: 46,   target: -0.017,        output: 0.804,         loss: 47.539
i: 47,   target: -0.082,        output: 0.804,         loss: 10.847
i: 48,   target: -0.071,        output: 0.804,         loss: 12.307
```

```
i: 49,   target: 0.046,         output: 0.804,         loss: 16.305
i: 50,   target: -0.061,        output: 0.804,         loss: 14.207
i: 51,   target: 0.246,         output: 0.804,         loss: 2.269
i: 52,   target: 0.064,         output: 0.804,         loss: 11.552
i: 53,   target: -0.075,        output: 0.804,         loss: 11.710
i: 54,   target: -0.079,        output: 0.804,         loss: 11.221
i: 55,   target: -0.079,        output: 0.804,         loss: 11.230
i: 56,   target: 0.001,         output: 0.804,         loss: 1143.156
i: 57,   target: 0.081,         output: 0.804,         loss: 8.924
i: 58,   target: -0.101,        output: 0.804,         loss: 8.990
i: 59,   target: -0.048,        output: 0.804,         loss: 17.749
i: 60,   target: -0.070,        output: 0.804,         loss: 12.557
i: 61,   target: -0.079,        output: 0.804,         loss: 11.237
i: 62,   target: 0.139,         output: 0.804,         loss: 4.773
i: 63,   target: 0.082,         output: 0.804,         loss: 8.789
i: 64,   target: -0.031,        output: 0.804,         loss: 26.621
i: 65,   target: -0.088,        output: 0.804,         loss: 10.177
i: 66,   target: 0.042,         output: 0.804,         loss: 18.086
i: 67,   target: -0.087,        output: 0.804,         loss: 10.220
i: 68,   target: -0.064,        output: 0.804,         loss: 13.478
i: 69,   target: -0.071,        output: 0.804,         loss: 12.260
i: 70,   target: 0.021,         output: 0.804,         loss: 38.057
i: 71,   target: 0.096,         output: 0.804,         loss: 7.417
i: 72,   target: 0.049,         output: 0.804,         loss: 15.577
i: 73,   target: 0.195,         output: 0.804,         loss: 3.126
i: 74,   target: -0.072,        output: 0.804,         loss: 12.210
i: 75,   target: -0.051,        output: 0.804,         loss: 16.753
i: 76,   target: 0.124,         output: 0.804,         loss: 5.509
i: 77,   target: 0.189,         output: 0.804,         loss: 3.247
i: 78,   target: 0.213,         output: 0.804,         loss: 2.769
i: 79,   target: 0.042,         output: 0.804,         loss: 18.270
i: 80,   target: -0.038,        output: 0.804,         loss: 22.172
i: 81,   target: -0.080,        output: 0.804,         loss: 11.010
i: 82,   target: -0.086,        output: 0.804,         loss: 10.338
i: 83,   target: 0.141,         output: 0.804,         loss: 4.721
i: 84,   target: 0.199,         output: 0.804,         loss: 3.050
i: 85,   target: -0.076,        output: 0.804,         loss: 11.563
i: 86,   target: -0.065,        output: 0.804,         loss: 13.434
i: 87,   target: 0.178,         output: 0.804,         loss: 3.520
i: 88,   target: 0.092,         output: 0.804,         loss: 7.762
i: 89,   target: -0.039,        output: 0.804,         loss: 21.647
i: 90,   target: -0.083,        output: 0.804,         loss: 10.633
i: 91,   target: 0.165,         output: 0.804,         loss: 3.874
i: 92,   target: -0.059,        output: 0.804,         loss: 14.659
i: 93,   target: -0.054,        output: 0.804,         loss: 16.033
i: 94,   target: 0.157,         output: 0.804,         loss: 4.130
i: 95,   target: 0.164,         output: 0.804,         loss: 3.913
i: 96,   target: 0.218,         output: 0.804,         loss: 2.685
```

```
i: 97,   target: 0.283,          output: 0.804,          loss: 1.845
i: 98,   target: -0.040,         output: 0.804,          loss: 21.019
i: 99,   target: 0.144,          output: 0.804,          loss: 4.569
i: 100,          target: -0.073,         output: 0.804,          loss: 12.041
i: 101,          target: -0.083,         output: 0.804,          loss: 10.715
i: 102,          target: -0.075,         output: 0.804,          loss: 11.706
i: 103,          target: 0.032,          output: 0.804,          loss: 24.002
i: 104,          target: -0.067,         output: 0.804,          loss: 13.001
i: 105,          target: -0.049,         output: 0.804,          loss: 17.496
i: 106,          target: -0.079,         output: 0.804,          loss: 11.187
i: 107,          target: 0.188,          output: 0.804,          loss: 3.288
i: 108,          target: 0.025,          output: 0.804,          loss: 31.489
i: 109,          target: 0.035,          output: 0.804,          loss: 21.723
i: 110,          target: 0.025,          output: 0.804,          loss: 31.201
i: 111,          target: -0.053,         output: 0.804,          loss: 16.276
i: 112,          target: -0.059,         output: 0.804,          loss: 14.558
i: 113,          target: -0.060,         output: 0.804,          loss: 14.445
i: 114,          target: -0.051,         output: 0.804,          loss: 16.644
i: 115,          target: -0.068,         output: 0.804,          loss: 12.782
i: 116,          target: -0.075,         output: 0.804,          loss: 11.697
i: 117,          target: -0.075,         output: 0.804,          loss: 11.775
i: 118,          target: -0.064,         output: 0.804,          loss: 13.646
i: 119,          target: -0.039,         output: 0.804,          loss: 21.871
i: 120,          target: 0.156,          output: 0.804,          loss: 4.168
i: 121,          target: -0.074,         output: 0.804,          loss: 11.893
i: 122,          target: -0.087,         output: 0.804,          loss: 10.209
i: 123,          target: -0.074,         output: 0.804,          loss: 11.824
i: 124,          target: -0.056,         output: 0.804,          loss: 15.362
i: 125,          target: 0.146,          output: 0.804,          loss: 4.516
i: 126,          target: 0.069,          output: 0.804,          loss: 10.588
i: 127,          target: -0.053,         output: 0.804,          loss: 16.177
i: 128,          target: -0.074,         output: 0.804,          loss: 11.918
i: 129,          target: -0.031,         output: 0.804,          loss: 27.336
i: 130,          target: -0.040,         output: 0.804,          loss: 21.225
i: 131,          target: 0.024,          output: 0.804,          loss: 32.567
i: 132,          target: 0.029,          output: 0.804,          loss: 26.584
i: 133,          target: 0.210,          output: 0.804,          loss: 2.836
i: 134,          target: 0.200,          output: 0.804,          loss: 3.026
i: 135,          target: 0.315,          output: 0.804,          loss: 1.552
i: 136,          target: 0.017,          output: 0.804,          loss: 45.985
i: 137,          target: 0.220,          output: 0.804,          loss: 2.657
i: 138,          target: -0.102,         output: 0.804,          loss: 8.895
i: 139,          target: 0.093,          output: 0.804,          loss: 7.653
i: 140,          target: 0.051,          output: 0.804,          loss: 14.733
i: 141,          target: -0.060,         output: 0.804,          loss: 14.498
i: 142,          target: -0.018,         output: 0.804,          loss: 45.395
i: 143,          target: 0.045,          output: 0.804,          loss: 17.049
i: 144,          target: -0.075,         output: 0.804,          loss: 11.718
```

```
i: 145,          target: -0.056,          output: 0.804,          loss: 15.370
i: 146,          target: 0.298,           output: 0.804,          loss: 1.703
i: 147,          target: -0.067,          output: 0.804,          loss: 13.007
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[21], line 2
      1 # print validation
----> 2 model.print_validation()

File d:\Raul\OneDrive - Cornell University\Code\peptide-QML\Notebooks\..
 ↪\my_code\pytorch_model.py:177, in pytorch_model.print_validation(self)
    175 avg_loss = 0
    176 for x, (i, t) in enumerate(zip((self.data_X_validation), self.
 ↪data_Y_validation)):
--> 177     outputs = self.model(i)
    178     loss = self.loss_function(outputs, t)
    179     avg_loss += loss/len(self.data_Y_validation)

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\torch\nn\modules\module.
 ↪py:1501, in Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\torch\nn\modules\container.py
 ↪217, in Sequential.forward(self, input)
    215 def forward(self, input):
    216     for module in self:
--> 217         input = module(input)
    218     return input

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\torch\nn\modules\module.
 ↪py:1501, in Module._call_impl(self, *args, **kwargs)
   1496 # If we don't have any hooks, we want to skip the rest of the logic in
   1497 # this function, and just call forward.
   1498 if not (self._backward_hooks or self._backward_pre_hooks or self.
 ↪_forward_hooks or self._forward_pre_hooks
   1499         or _global_backward_pre_hooks or _global_backward_hooks
   1500         or _global_forward_hooks or _global_forward_pre_hooks):
-> 1501     return forward_call(*args, **kwargs)
```

```
   1502 # Do not call functions when jit is used
   1503 full_backward_hooks, non_full_backward_hooks = [], []

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\qnn\torch.py:
 ↪408, in TorchLayer.forward(self, inputs)
    405     results = torch.stack(reconstructor)
    406 else:
    407     # calculate the forward pass as usual
--> 408     results = self._evaluate_qnode(inputs)
    410 # reshape to the correct number of batch dims
    411 if has_batch_dim:

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\qnn\torch.py:
 ↪429, in TorchLayer._evaluate_qnode(self, x)
    417 """Evaluates the QNode for a single input datapoint.
    418
    419 Args:
   (…)
    423     tensor: output datapoint
    424 """
    425 kwargs = {
    426     **{self.input_arg: x},
    427     **{arg: weight.to(x) for arg, weight in self.qnode_weights.items()}
    428 }
--> 429 res = self.qnode(**kwargs)
    431 if isinstance(res, torch.Tensor):
    432     return res.type(x.dtype)

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\qnode.py:950,␣
 ↪in QNode.__call__(self, *args, **kwargs)
    948     self.execute_kwargs.pop("mode")
    949 # pylint: disable=unexpected-keyword-arg
--> 950 res = qml.execute(
    951     [self.tape],
    952     device=self.device,
    953     gradient_fn=self.gradient_fn,
    954     interface=self.interface,
    955     gradient_kwargs=self.gradient_kwargs,
    956     override_shots=override_shots,
    957     **self.execute_kwargs,
    958 )
    960 res = res[0]
    962 # convert result to the interface in case the qfunc has no parameters
```

```
File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\interfaces\executio.
 ↪py:511, in execute(tapes, device, gradient_fn, interface, grad_on_execution,␣
 ↪gradient_kwargs, cache, cachesize, max_diff, override_shots, expand_fn,␣
 ↪max_expansion, device_batch_transform)
    503     # use qml.interfaces so that mocker can spy on it during testing
    504     cached_execute_fn = qml.interfaces.cache_execute(
    505         batch_execute,
    506         cache,
   (…)
    509         pass_kwargs=new_device_interface,
    510     )
--> 511     results = cached_execute_fn(tapes, execution_config=config)
    512     return batch_fn(results)
    514 # the default execution function is batch_execute
    515 # use qml.interfaces so that mocker can spy on it during testing

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\interfaces\executio.
 ↪py:287, in cache_execute.<locals>.wrapper(tapes, **kwargs)
    282         return (res, []) if return_tuple else res
    284 else:
    285     # execute all unique tapes that do not exist in the cache
    286     # convert to list as new device interface returns a tuple
--> 287     res = list(fn(execution_tapes.values(), **kwargs))
    289 final_res = []
    291 for i, tape in enumerate(tapes):

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\interfaces\executio.
 ↪py:210, in cache_execute.<locals>.fn(tapes, **kwargs)
    208 def fn(tapes: Sequence[QuantumTape], **kwargs):  # pylint:␣
 ↪disable=function-redefined
    209     tapes = [expand_fn(tape) for tape in tapes]
--> 210     return original_fn(tapes, **kwargs)

File d:\Raul\Programs\envs\PennyLane\lib\contextlib.py:79, in ContextDecorator.
 ↪__call__.<locals>.inner(*args, **kwds)
     76 @wraps(func)
     77 def inner(*args, **kwds):
     78     with self._recreate_cm():
---> 79         return func(*args, **kwds)

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\_qubit_device.
 ↪py:603, in QubitDevice.batch_execute(self, circuits)
    598 for circuit in circuits:
    599     # we need to reset the device here, else it will
    600     # not start the next computation in the zero state
    601     self.reset()
```

```
--> 603     res = self.execute(circuit)
    604     results.append(res)
    606 if self.tracker.active:

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\devices\default_qub t_torch.
 ↪py:232, in DefaultQubitTorch.execute(self, circuit, **kwargs)
    224         if params_cuda_device != specified_device_cuda:
    225             warnings.warn(
    226                 f"Torch device {self._torch_device} specified "
    227                 "upon PennyLane device creation does not match the "
    228                 "Torch device of the gate parameters; "
    229                 f"{self._torch_device} will be used."
    230             )
--> 232 return super().execute(circuit, **kwargs)

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\_qubit_device.
 ↪py:320, in QubitDevice.execute(self, circuit, **kwargs)
    317 self.check_validity(circuit.operations, circuit.observables)
    319 # apply all circuit operations
--> 320 self.apply(circuit.operations, rotations=self.
 ↪_get_diagonalizing_gates(circuit), **kwargs)
    322 # generate computational basis samples
    323 if self.shots is not None or circuit.is_sampled:

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\devices\default_qub t.
 ↪py:293, in DefaultQubit.apply(self, operations, rotations, **kwargs)
    291         self._state = self._apply_parametrized_evolution(self._state,␣
 ↪operation)
    292     else:
--> 293         self._state = self._apply_operation(self._state, operation)
    295 # store the pre-rotated state
    296 self._pre_rotated_state = self._state

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\devices\default_qub t.
 ↪py:333, in DefaultQubit._apply_operation(self, state, operation)
    330     axes = [ax + shift for ax in self.wires.indices(wires)]
    331     return self._apply_ops[operation.name](state, axes)
--> 333 matrix = self._asarray(self._get_unitary_matrix(operation), dtype=self.
 ↪C_DTYPE)
    335 if operation in diagonal_in_z_basis:
    336     return self._apply_diagonal_unitary(state, matrix, wires)

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\devices\default_qub t_torch.
 ↪py:304, in DefaultQubitTorch._get_unitary_matrix(self, unitary)
    302 if unitary in diagonal_in_z_basis:
```

```
    303     return self._asarray(unitary.eigvals(), dtype=self.C_DTYPE)
--> 304 return self._asarray(unitary.matrix(), dtype=self.C_DTYPE)

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\operation.py:
 ↪748, in Operator.matrix(self, wire_order)
    728 def matrix(self, wire_order=None):
    729     r"""Representation of the operator as a matrix in the computational
 ↪basis.
    730
    731     If ``wire_order`` is provided, the numerical representation␣
 ↪considers the position of the
   (…)
    746         tensor_like: matrix representation
    747     """
--> 748     canonical_matrix = self.compute_matrix(*self.parameters, **self.
 ↪hyperparameters)
    750     if wire_order is None or self.wires == Wires(wire_order):
    751         return canonical_matrix

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\ops\qubit\parametri:_ops_single_
 ↪py:211, in RY.compute_matrix(theta)
    209 c = (1 + 0j) * c
    210 s = (1 + 0j) * s
--> 211 return qml.math.stack([stack_last([c, -s]), stack_last([s, c])], axis=-1)

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\math\multi_dispatch
 ↪py:151, in multi_dispatch.<locals>.decorator.<locals>.wrapper(*args, **kwargs
    148 interface = interface or get_interface(*dispatch_args)
    149 kwargs["like"] = interface
--> 151 return fn(*args, **kwargs)

File d:
 ↪\Raul\Programs\envs\PennyLane\lib\site-packages\pennylane\math\multi_dispatch
 ↪py:488, in stack(values, axis, like)
    459 """Stack a sequence of tensors along the specified axis.
    460
    461 .. warning::
   (…)
    485         [5.00e+00, 8.00e+00, 1.01e+02]], dtype=float32)>
    486 """
    487 values = np.coerce(values, like=like)
--> 488 return np.stack(values, axis=axis, like=like)

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\autoray\autoray.py:79, i␣
 ↪do(fn, like, *args, **kwargs)
     30 """Do function named ``fn`` on ``(*args, **kwargs)``, peforming single
```

```
     31 dispatch to retrieve ``fn`` based on whichever library defines the clas
 ↪of
     32 the ``args[0]``, or the ``like`` keyword argument if specified.
   (…)
     76     <tf.Tensor: id=91, shape=(3, 3), dtype=float32>
     77 """
     78 backend = choose_backend(fn, *args, like=like, **kwargs)
---> 79 return get_lib_fn(backend, fn)(*args, **kwargs)

File d:\Raul\Programs\envs\PennyLane\lib\site-packages\autoray\autoray.py:1185,
 ↪in translate_wrapper.<locals>.translated_function(*args, **kwargs)
   1182 for key, value in translation.items():
   1183     new_kwargs[value[0]] = value[1]
-> 1185 return fn(**new_kwargs)

KeyboardInterrupt:
```