import torch

import torch.nn as nn

# Fully connected neural network with one hidden layer

class RNN(nn.Module):

def \_\_init\_\_(self, input\_size, hidden\_size):

super(RNN, self).\_\_init\_\_()

self.rnn = nn.RNN(input\_size, hidden\_size, num\_layers=2, batch\_first=True)

# Uncomment and use GRU or LSTM if needed

# self.gru = nn.GRU(input\_size, hidden\_size, num\_layers=2, batch\_first=True)

# self.lstm = nn.LSTM(input\_size, hidden\_size, num\_layers=2, batch\_first=True)

self.fc = nn.Linear(hidden\_size, 1)

def forward(self, x):

\_, hidden = self.rnn(x)

out = hidden[-1, :, :]

out = self.fc(out)

return out

# Example usage:

input\_size = 64 # Size of the input feature vector

hidden\_size = 32 # Size of the hidden state

model = RNN(input\_size, hidden\_size)

print(model)

# Create a random tensor with shape (batch\_size, sequence\_length, input\_size)

# For example, batch\_size = 5, sequence\_length = 3, input\_size = 64

input\_tensor = torch.randn(5, 3, input\_size)

# Forward pass through the model

output = model(input\_tensor)

print(output)