#Ray

import ray

import torch

import torch.nn as nn

import torch.optim as optim

import random

ray.init(ignore\_reinit\_error=True)

class SimpleModel(nn.Module):

def \_\_init\_\_(self):

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

self.fc = nn.Linear(10, 10)

def forward(self, x):

return self.fc(x)

@ray.remote

def train\_model(rank):

model = SimpleModel().to('cpu')

input\_tensor = torch.randn(20, 10)

target\_tensor = torch.randn(20, 10)

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

criterion = nn.MSELoss()

optimizer.zero\_grad()

output = model(input\_tensor)

loss = criterion(output, target\_tensor)

loss.backward()

optimizer.step()

print(f"Worker {rank}, Loss: {loss.item()}")

def main():

num\_workers = 4

workers = [train\_model.remote(i) for i in range(num\_workers)]

ray.get(workers)

if \_\_name\_\_ == '\_\_main\_\_':

main()