#PyTorch Distributed (DDP)

import torch

import torch. Distributed as dist

import torchon as nn

import torch. Optima as optima

from torch.nn.parallel import DistributedDataParallel as DDP

import torch.multiprocessing as mp

import os

def init\_process(rank, size, fn, backend='nccl'):

os.environ['MASTER\_ADDR'] = 'localhost'

os.environ['MASTER\_PORT'] = '12355'

dist.init\_process\_group(backend, rank=rank, world\_size=size)

fn(rank, size)

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)

def run(rank, size):

model = SimpleModel().to(rank)

model = DDP(model, device\_ids=[rank])

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

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

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"Rank {rank}, Loss: {loss.item()}")

def main():

size = 2 # Number of processes (nodes or GPUs)

mp.spawn(init\_process, args=(size, run), nprocs=size, join=True)

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

main()