#MXNet Distributed

import mxnet as mx

from mxnet import gluon, autograd, init

import horovod.mxnet as hvd

import numpy as np

hvd.init()

context = mx.gpu(hvd.rank()) if mx.context.num\_gpus() > 0 else mx.cpu()

class SimpleModel(gluon.nn.Block):

def \_\_init\_\_(self, \*\*kwargs):

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

self.dense = gluon.nn.Dense(10)

def forward(self, x):

return self.dense(x)

def get\_data\_loader(batch\_size):

data = np.random.rand(100, 10).astype(np.float32)

labels = np.random.rand(100, 10).astype(np.float32)

dataset = mx.gluon.data.ArrayDataset(data, labels)

return mx.gluon.data.DataLoader(dataset, batch\_size=batch\_size)

def train(model, data\_loader, batch\_size):

trainer = gluon.Trainer(model.collect\_params(), 'sgd', {'learning\_rate': 0.01})

loss\_fn = gluon.loss.L2Loss()

for i, (data, label) in enumerate(data\_loader):

data = mx.nd.array(data).as\_in\_context(context)

label = mx.nd.array(label).as\_in\_context(context)

with autograd.record():

output = model(data)

loss = loss\_fn(output, label)

loss.backward()

trainer.step(batch\_size)

hvd.allreduce(model.collect\_params(), op=hvd.Average)

if i % 10 == 0:

print(f"Rank {hvd.rank()}, Loss: {loss.mean().asscalar()}")

def main():

model = SimpleModel()

model.initialize(init.Xavier(), ctx=context)

batch\_size = 32

data\_loader = get\_data\_loader(batch\_size)

train(model, data\_loader, batch\_size)

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

main()