Supplementary Material S1

**Improving Artificial Neural Network Performance with Zero Initialization.**

Jongwoo Seo and Wuhyun Koh

Code for Zero initialization

use control + F and search # to obtain detail information.

import torch

import csv

import numpy as np

import cv2 as cv

from torch.utils.data import DataLoader

from torch.utils.data import TensorDataset

from torch import nn

device = "cuda" if torch.cuda.is\_available() else "cpu"

print(f"Using {device} device")

print('Device:', device)

print('Current cuda device:', torch.cuda.current\_device())

print('Count of using GPUs:', torch.cuda.device\_count())

print(torch.cuda.current\_device())

temp\_list = list()

r=open ('cifar-10-python/training\_data\_CIFAR.csv', 'r' )

reader=csv.reader(r)

for target in reader:

temp\_list.append(target)

training\_data = np.array(temp\_list).reshape((-1, 3, 32, 32)).astype(float)

print(training\_data.shape)

training\_data = torch.from\_numpy(training\_data).to("cuda")

temp\_list = list()

r=open ('cifar-10-python/test\_data\_CIFAR.csv', 'r' )

reader=csv.reader(r)

for target in reader:

temp\_list.append(target)

test\_data = np.array(temp\_list).reshape((-1, 3, 32, 32)).astype(float)

print(test\_data.shape)

test\_data = torch.from\_numpy(test\_data).to("cuda")

temp\_list = list()

r=open ('cifar-10-python/training\_target\_CIFAR.csv', 'r' )

reader=csv.reader(r)

for target in reader:

if target[0] == '0':

temp\_list.append([1,0,0,0,0,0,0,0,0,0])

continue

if target[0] == '1':

temp\_list.append([0,1,0,0,0,0,0,0,0,0])

continue

if target[0] == '2':

temp\_list.append([0,0,1,0,0,0,0,0,0,0])

continue

if target[0] == '3':

temp\_list.append([0,0,0,1,0,0,0,0,0,0])

continue

if target[0] == '4':

temp\_list.append([0,0,0,0,1,0,0,0,0,0])

continue

if target[0] == '5':

temp\_list.append([0,0,0,0,0,1,0,0,0,0])

continue

if target[0] == '6':

temp\_list.append([0,0,0,0,0,0,1,0,0,0])

continue

if target[0] == '7':

temp\_list.append([0,0,0,0,0,0,0,1,0,0])

continue

if target[0] == '8':

temp\_list.append([0,0,0,0,0,0,0,0,1,0])

continue

if target[0] == '9':

temp\_list.append([0,0,0,0,0,0,0,0,0,1])

continue

training\_target = np.array(temp\_list).astype(float)

print(training\_target.shape)

training\_target = torch.from\_numpy(training\_target).to("cuda")

temp\_list = list()

r=open ('cifar-10-python/test\_target\_CIFAR.csv', 'r' )

reader=csv.reader(r)

for target in reader:

if target[0] == '0':

temp\_list.append([1,0,0,0,0,0,0,0,0,0])

continue

if target[0] == '1':

temp\_list.append([0,1,0,0,0,0,0,0,0,0])

continue

if target[0] == '2':

temp\_list.append([0,0,1,0,0,0,0,0,0,0])

continue

if target[0] == '3':

temp\_list.append([0,0,0,1,0,0,0,0,0,0])

continue

if target[0] == '4':

temp\_list.append([0,0,0,0,1,0,0,0,0,0])

continue

if target[0] == '5':

temp\_list.append([0,0,0,0,0,1,0,0,0,0])

continue

if target[0] == '6':

temp\_list.append([0,0,0,0,0,0,1,0,0,0])

continue

if target[0] == '7':

temp\_list.append([0,0,0,0,0,0,0,1,0,0])

continue

if target[0] == '8':

temp\_list.append([0,0,0,0,0,0,0,0,1,0])

continue

if target[0] == '9':

temp\_list.append([0,0,0,0,0,0,0,0,0,1])

continue

test\_target = np.array(temp\_list).astype(float)

print(test\_target.shape)

test\_target = torch.from\_numpy(test\_target).to("cuda")

"""

# It defends on your configurations regarding dataset files.

load dataset what you want to train

load data. data shapes should be

(50000, 3, 32, 32) training image

(10000, 3, 32, 32) test image

(50000, 10) training target

(10000, 10) test target

"""

batch\_size = 100

epochs = 200

training\_dataset = TensorDataset(training\_data, training\_target)

test\_dataset = TensorDataset(test\_data, test\_target)

training\_dataloader = DataLoader(training\_dataset, batch\_size=batch\_size, shuffle=True)

test\_dataloader = DataLoader(test\_dataset, batch\_size=batch\_size, shuffle=False)

class CNN(nn.Module):

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

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

self.layer\_1 = nn.Conv2d(3,64,3,stride=1,dtype=torch.float64)

self.layer\_2 = nn.MaxPool2d(2)

self.layer\_3 = nn.Conv2d(64,128,3,stride=1,dtype=torch.float64)

self.layer\_4 = nn.MaxPool2d(2)

self.layer\_5 = nn.Conv2d(128,256,3,stride=1,dtype=torch.float64)

self.layer\_6 = nn.MaxPool2d(2)

self.layer\_7 = nn.Linear(1024,10,dtype=torch.float64)

self.lrelu = nn.LeakyReLU()

self.flatten = nn.Flatten()

self.glob\_flag = 0

self.softmax = nn.Softmax(dim=1)

#torch.nn.init.zeros\_(self.layer\_7.weight) #Choose this when you apply zero initialization. if you wan to apply zero initialization to self.layer\_3, set torch.nn.init.zeros\_(self.layer\_3.weight)

def forward(self, x):

x\_1 = self.layer\_2(self.lrelu(self.layer\_1(x)))

x\_1 = self.layer\_4(self.lrelu(self.layer\_3(x\_1)))

x\_1 = self.layer\_6(self.lrelu(self.layer\_5(x\_1)))

x\_1 = self.flatten(x\_1)

x\_1 = self.layer\_7(x\_1)

return x\_1

model = CNN().to("cuda")

print(model)

def train(dataloader, model, loss, optimizer):

loss\_sum=0

for batch, (X, y) in enumerate(dataloader):

pred = model(X)

batch\_loss\_result = loss\_fn(pred, y)

optimizer.zero\_grad()

batch\_loss\_result.backward()

optimizer.step()

loss\_sum+=batch\_loss\_result

loss\_sum=loss\_sum/(batch+1)

def test(dataloader, model, loss):

with torch.no\_grad():

loss\_sum=0

accuracy\_sum=0

for batch, (X, y) in enumerate(dataloader):

pred = model(X)

batch\_loss\_result = loss\_fn(pred, y)

loss\_sum+=batch\_loss\_result

accuracy\_sum+= (torch.argmax(pred, dim=1) == torch.argmax(y,dim=1)).type(torch.float).sum().item()

loss\_sum=loss\_sum/(batch+1)

return accuracy\_sum/10000

summary = list()

for i in range(10):

model = CNN().to("cuda") #if you want to apply new model, declare new object. Do not change the name "model"

loss\_fn = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)

scheduler = torch.optim.lr\_scheduler.StepLR(optimizer, step\_size=20, gamma=0.95)

accuracy = 0

for t in range(epochs):

train(training\_dataloader, model, loss\_fn, optimizer)

accuracy\_temp = test(test\_dataloader, model, loss\_fn)

scheduler.step()

if accuracy\_temp > accuracy:

accuracy = accuracy\_temp

model.glob\_flag+=1

print(accuracy\_temp)

summary.append(accuracy)

print("Done!")

print(summary)