تمرین کامپیوتری دوم درس یادگیری ماشین توزیع شده

پدرام رستمی – ۸۱۰۱۰۰۳۵۳

سوال ۱

الف) مدلی که در این بخش استفاده شده دارای ۴ بلاک است. بلاک‌های اول تا سوم شامل لایه‌های کانولوشنی و پولینگ است و بلاک چهارم شامل لایه‌های خطی برای تولید خروجی نهایی است. در شکل ۱ ساختار این شبکه قابل مشاهده است. این شبکه ساختاری شبیه به مدل VGG دارد و تنها کمی سبک‌تر شده است.

```
ConvNet(
  (block1): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (block2): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (block3): Sequential(
    (0): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (block4): Sequential(
    (0): Linear(in_features=4096, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=32, bias=True)
    (3): ReLU()
    (4): Linear(in_features=32, out_features=10, bias=True)
  )
)
```

شکل ۱- ساختار مدل استفاده شده

برای این بخش، داده‌ها ابتدا دانلود شده و در پوشه‌ی data قرار داده می‌شوند. سپس به کمک تابع load_data که در شکل ۲ گذاشته شده است، داده‌ها load می‌شوند. همچنین هنگام load کردن داده‌ها، آن‌ها normalize می‌شوند. سایز batch هم ۳۲ در نظر گرفته شده است.

```python
def load_data(data_path, download, num_workers, shuffle, batch_size):
    transform = transforms.Compose(
        [transforms.ToTensor(),
         transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]
    )
    trainset = torchvision.datasets.CIFAR10(root=data_path,
                                            train=True,
                                            download=download,
                                            transform=transform)
    trainloader = torch.utils.data.DataLoader(trainset,
                                              batch_size=batch_size,
                                              shuffle=shuffle,
                                              num_workers=num_workers)
    testset = torchvision.datasets.CIFAR10(root=data_path,
                                           train=False,
                                           download=download,
                                           transform=transform)
    testloader = torch.utils.data.DataLoader(testset,
                                             batch_size=batch_size,
                                             shuffle=shuffle,
                                             num_workers=num_workers)
    return trainloader, testloader
```

شکل ۲- تابع load_data

سپس در تابع train مدل آموزش داده می‌شود. این تابع در شکل ۳ قابل مشاهده است. در این تابع برای لاگ از ماژول SummaryWriter استفاده شده است که برای ثبت لاگ به فرمت tensorboard در کتابخانه‌ی پایتورچ است. به کمک این ماژول، در انتهای هر ایپاک مقدار loss داده‌های آموزش و تست و مقدار دقت مدل بر روی داده‌های آموزش و تست ثبت می‌شود. همچنین این مقادیر به همراه مدت زمان اجرای هر ایپاک در انتهای ایپاک چاپ می‌شود.

```python
def train(model, criterion, optimizer, train_loader, test_loader, max_epochs, log_path):
    tb = SummaryWriter(log_path)
    for epoch in range(max_epochs):
        start_time = time.time()
        train_correct_preds = 0
        train_loss = 0
        for batch_num, (x_batch, y_batch) in enumerate(train_loader):
            x_batch = x_batch.cuda(non_blocking=True).float()
            y_batch = y_batch.cuda(non_blocking=True)
            output = model(x_batch)
            optimizer.zero_grad()
            loss = criterion(output, y_batch)
            train_loss += loss.item()
            loss.backward()
            optimizer.step()
            _, pred = torch.max(output.data, 1)
            train_correct_preds += torch.sum(pred == y_batch.data).item()
        train_accuracy = float(train_correct_preds / len(train_loader.dataset))
        train_loss = train_loss / len(train_loader.dataset)
        model.eval()
        test_correct_preds = 0
        test_loss = 0
        for batch_num, (x_batch, y_batch) in enumerate(test_loader):
            x_batch = x_batch.cuda(non_blocking=True).float()
            y_batch = y_batch.cuda(non_blocking=True)
            output = model(x_batch)
            loss = criterion(output, y_batch)
            test_loss += loss.item()
            _, pred = torch.max(output.data, 1)
            test_correct_preds += torch.sum(pred == y_batch.data).item()
        test_accuracy = float(test_correct_preds / len(test_loader.dataset))
        test_loss = test_loss / len(test_loader.dataset)
        end_time = time.time()
        print(f"epoch : {epoch + 1} - train loss : {train_loss:.3f} - train accuracy : {(train_accuracy*100):.3f} - test loss : {test_loss:.3f} - test accuracy : {(test_accuracy*100):.3f} - time : {(end_time
        tb.add_scalar("Train loss", train_loss, epoch + 1)
        tb.add_scalar("Train accuracy", train_accuracy, epoch + 1)
        tb.add_scalar("Test loss", test_loss, epoch + 1)
        tb.add_scalar("Test accuracy", test_accuracy, epoch + 1)
```

شکل ۳- پیاده سازی تابع train

در شکل ۴ main قابل مشاهده است. در این بخش hyperparameter ها مقداردهی شده‌اند. برای آموزش از بهینه‌ساز Adam و از تابع خطای CrossEntropyLoss استفاده شده است. همچنین مدل برای ۲۰ ایپاک آموزش داده شده است و از NVIDIA GeForce RTX 3090 استفاده شده است.

```python
if __name__ == '__main__':
    data_path = "/home/rostami/DMLS/data/"
    download = False
    num_workers = 2
    shuffle = False
    batch_size = 32
    num_classes = 10
    gpu_number = 0
    max_epochs = 20
    log_path = "/home/rostami/DMLS/main/sample"
    optimizer_lr = 5e-4
    optimizer_weight_decay = 5e-4
    print("GPU : ", torch.cuda.get_device_name(gpu_number))
    model = ConvNet(num_classes)
    model.to(gpu_number)
    print(model)
    criterion = nn.CrossEntropyLoss()
    criterion = criterion.cuda(gpu_number)
    optimizer = Adam(model.parameters(), lr=optimizer_lr, weight_decay=optimizer_weight_decay)
    train_loader, test_loader = load_data(data_path, download, num_workers, shuffle, batch_size)
    start_time = time.time()
    train(model, criterion, optimizer, train_loader, test_loader, max_epochs, log_path)
    end_time = time.time()
    print(f"total time : {(end_time - start_time):.3f}")
```

شکل ۴- پیاده سازی main

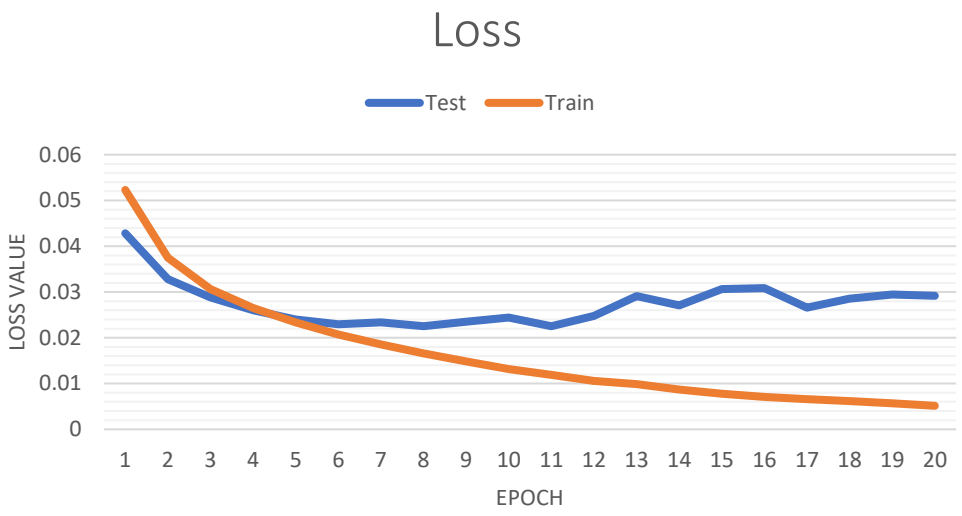
در شکل ۵ لاگ چاپ شده در تابع train قابل مشاهده است. همانطور که قابل مشاهده است، در انتهای ایپاک ۲۰ ام، دقت مدل برای داده‌های آموزش 94.066% و برای داده‌های تست 77.510% است. مقدار خطا و دقت در هر ایپاک و مدت زمان اجرای هر ایپاک هم قابل مشاهده است. همچنین مدت زمان کلی آموزش برابر 160.7 ثانیه است.

```
epoch : 1 - train loss : 0.052 - train accuracy : 37.084 - test loss : 0.043 - test accuracy : 49.920 - time : 8.320 s
epoch : 2 - train loss : 0.038 - train accuracy : 56.660 - test loss : 0.033 - test accuracy : 62.450 - time : 7.848 s
epoch : 3 - train loss : 0.031 - train accuracy : 64.990 - test loss : 0.029 - test accuracy : 67.190 - time : 7.971 s
epoch : 4 - train loss : 0.027 - train accuracy : 69.870 - test loss : 0.026 - test accuracy : 70.540 - time : 7.994 s
epoch : 5 - train loss : 0.023 - train accuracy : 73.622 - test loss : 0.024 - test accuracy : 73.080 - time : 8.021 s
epoch : 6 - train loss : 0.021 - train accuracy : 76.870 - test loss : 0.023 - test accuracy : 74.670 - time : 7.968 s
epoch : 7 - train loss : 0.019 - train accuracy : 79.132 - test loss : 0.023 - test accuracy : 74.170 - time : 7.924 s
epoch : 8 - train loss : 0.017 - train accuracy : 81.348 - test loss : 0.023 - test accuracy : 76.050 - time : 8.269 s
epoch : 9 - train loss : 0.015 - train accuracy : 83.526 - test loss : 0.024 - test accuracy : 75.360 - time : 7.946 s
epoch : 10 - train loss : 0.013 - train accuracy : 85.448 - test loss : 0.024 - test accuracy : 75.370 - time : 7.975 s
epoch : 11 - train loss : 0.012 - train accuracy : 86.752 - test loss : 0.023 - test accuracy : 77.040 - time : 7.842 s
epoch : 12 - train loss : 0.011 - train accuracy : 88.082 - test loss : 0.025 - test accuracy : 76.200 - time : 7.915 s
epoch : 13 - train loss : 0.010 - train accuracy : 88.808 - test loss : 0.029 - test accuracy : 73.670 - time : 7.896 s
epoch : 14 - train loss : 0.009 - train accuracy : 90.084 - test loss : 0.027 - test accuracy : 76.340 - time : 7.885 s
epoch : 15 - train loss : 0.008 - train accuracy : 91.164 - test loss : 0.031 - test accuracy : 74.210 - time : 7.874 s
epoch : 16 - train loss : 0.007 - train accuracy : 91.900 - test loss : 0.031 - test accuracy : 75.780 - time : 8.172 s
epoch : 17 - train loss : 0.007 - train accuracy : 92.518 - test loss : 0.027 - test accuracy : 77.340 - time : 8.120 s
epoch : 18 - train loss : 0.006 - train accuracy : 92.952 - test loss : 0.029 - test accuracy : 76.970 - time : 7.939 s
epoch : 19 - train loss : 0.006 - train accuracy : 93.580 - test loss : 0.029 - test accuracy : 77.840 - time : 7.986 s
epoch : 20 - train loss : 0.005 - train accuracy : 94.066 - test loss : 0.029 - test accuracy : 77.510 - time : 7.916 s
total time : 160.700
```

شکل ۵- لاگ چاپ شده در تابع train

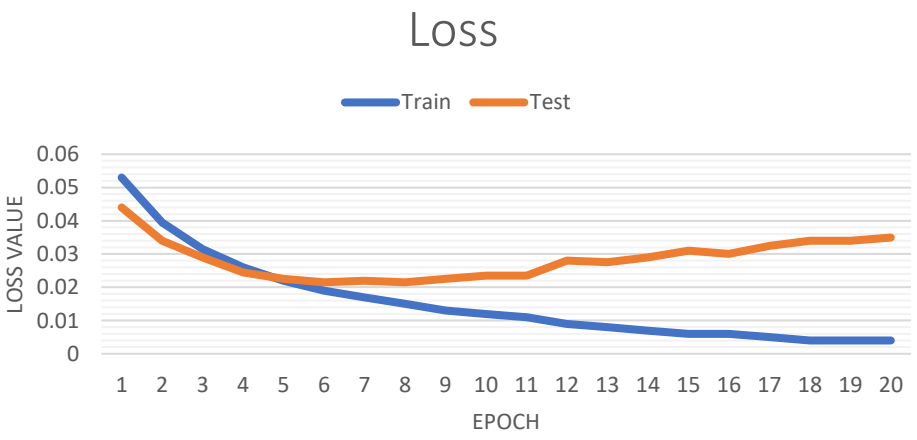
در نمودار ۱ مقدار خطای داده‌های آموزش و تست در ایپاک‌های مختلف قابل مشاهده است.

نمودار 1 - نمودار خطای داده‌های آموزش و تست



ب) در این بخش به کمک DDP در کتابخانه‌ی پایتورچ مدل بر روی هر دو GPU آموزش داده می‌شود. در این بخش پیاده‌سازی صرفا تغییراتی جزئی برای استفاده از DDP نسبت به بخش قبل پیدا می‌کند. در شکل ۶ لاگ چاپ شده در این حالت برای هر کدام از رنک‌ها قابل مشاهده است. همانطور که مشخص است، مدت زمان کلی آموزش 634.106 ثانیه است. همچنین دقت مدل در ایپاک ۲۰ ام برای داده‌های آموزش 95.488% و برای داده‌های تست 75.856% است. در نمودار ۲ هم تغییرات loss برای داده‌های آموزش و تست آورده شده است.

نمودار 2 - نمودار خطای داده‌های آموزش و تست



```
(/opt/anaconda3) rostami@Meshki:~/DMLS/main$ CUDA_VISIBLE_DEVICES='0,1' python distributed.py
rank : 0 - epoch : 1 - train loss : 0.053 - train accuracy : 36.021 - test loss : 0.044 - test accuracy : 47.930 - time : 32.147 s
rank : 1 - epoch : 1 - train loss : 0.053 - train accuracy : 36.033 - test loss : 0.044 - test accuracy : 48.885 - time : 32.169 s
rank : 1 - epoch : 2 - train loss : 0.040 - train accuracy : 53.708 - test loss : 0.034 - test accuracy : 61.087 - time : 31.120 s
rank : 0 - epoch : 2 - train loss : 0.039 - train accuracy : 54.260 - test loss : 0.034 - test accuracy : 61.525 - time : 31.247 s
rank : 0 - epoch : 3 - train loss : 0.031 - train accuracy : 64.634 - test loss : 0.029 - test accuracy : 67.138 - time : 31.270 s
rank : 1 - epoch : 3 - train loss : 0.032 - train accuracy : 63.691 - test loss : 0.029 - test accuracy : 66.899 - time : 31.538 s
rank : 1 - epoch : 4 - train loss : 0.026 - train accuracy : 70.416 - test loss : 0.024 - test accuracy : 72.313 - time : 31.287 s
rank : 0 - epoch : 4 - train loss : 0.026 - train accuracy : 71.028 - test loss : 0.025 - test accuracy : 72.373 - time : 31.521 s
rank : 1 - epoch : 5 - train loss : 0.022 - train accuracy : 75.188 - test loss : 0.022 - test accuracy : 75.318 - time : 31.371 s
rank : 0 - epoch : 5 - train loss : 0.022 - train accuracy : 75.256 - test loss : 0.023 - test accuracy : 74.662 - time : 31.396 s
rank : 0 - epoch : 6 - train loss : 0.019 - train accuracy : 78.485 - test loss : 0.022 - test accuracy : 75.736 - time : 31.108 s
rank : 1 - epoch : 6 - train loss : 0.019 - train accuracy : 78.497 - test loss : 0.021 - test accuracy : 76.393 - time : 31.324 s
rank : 1 - epoch : 7 - train loss : 0.017 - train accuracy : 81.038 - test loss : 0.022 - test accuracy : 76.154 - time : 31.626 s
rank : 0 - epoch : 7 - train loss : 0.017 - train accuracy : 81.158 - test loss : 0.022 - test accuracy : 75.796 - time : 31.788 s
rank : 0 - epoch : 8 - train loss : 0.015 - train accuracy : 83.444 - test loss : 0.022 - test accuracy : 76.115 - time : 31.289 s
rank : 1 - epoch : 8 - train loss : 0.015 - train accuracy : 83.732 - test loss : 0.021 - test accuracy : 77.329 - time : 31.482 s
rank : 0 - epoch : 9 - train loss : 0.013 - train accuracy : 85.118 - test loss : 0.023 - test accuracy : 75.836 - time : 31.342 s
rank : 1 - epoch : 9 - train loss : 0.013 - train accuracy : 85.370 - test loss : 0.022 - test accuracy : 77.209 - time : 31.368 s
rank : 1 - epoch : 10 - train loss : 0.012 - train accuracy : 86.789 - test loss : 0.023 - test accuracy : 76.553 - time : 31.392 s
rank : 0 - epoch : 10 - train loss : 0.012 - train accuracy : 86.549 - test loss : 0.024 - test accuracy : 75.896 - time : 31.720 s
rank : 0 - epoch : 11 - train loss : 0.011 - train accuracy : 88.004 - test loss : 0.024 - test accuracy : 76.692 - time : 31.528 s
rank : 1 - epoch : 11 - train loss : 0.011 - train accuracy : 87.992 - test loss : 0.023 - test accuracy : 77.170 - time : 31.705 s
rank : 0 - epoch : 12 - train loss : 0.009 - train accuracy : 90.082 - test loss : 0.029 - test accuracy : 74.363 - time : 31.362 s
rank : 1 - epoch : 12 - train loss : 0.009 - train accuracy : 90.070 - test loss : 0.027 - test accuracy : 75.199 - time : 31.429 s
rank : 1 - epoch : 13 - train loss : 0.008 - train accuracy : 91.444 - test loss : 0.027 - test accuracy : 76.911 - time : 31.371 s
rank : 0 - epoch : 13 - train loss : 0.008 - train accuracy : 91.364 - test loss : 0.028 - test accuracy : 76.354 - time : 31.561 s
rank : 1 - epoch : 14 - train loss : 0.007 - train accuracy : 91.648 - test loss : 0.028 - test accuracy : 77.468 - time : 31.605 s
rank : 0 - epoch : 14 - train loss : 0.007 - train accuracy : 91.744 - test loss : 0.030 - test accuracy : 76.174 - time : 31.688 s
rank : 0 - epoch : 15 - train loss : 0.006 - train accuracy : 92.719 - test loss : 0.032 - test accuracy : 75.159 - time : 31.116 s
rank : 1 - epoch : 15 - train loss : 0.006 - train accuracy : 92.927 - test loss : 0.030 - test accuracy : 75.617 - time : 31.433 s
rank : 0 - epoch : 16 - train loss : 0.006 - train accuracy : 93.666 - test loss : 0.031 - test accuracy : 76.971 - time : 31.264 s
rank : 1 - epoch : 16 - train loss : 0.006 - train accuracy : 93.207 - test loss : 0.029 - test accuracy : 77.170 - time : 31.206 s
rank : 1 - epoch : 17 - train loss : 0.005 - train accuracy : 93.942 - test loss : 0.031 - test accuracy : 76.612 - time : 31.427 s
rank : 0 - epoch : 17 - train loss : 0.005 - train accuracy : 94.254 - test loss : 0.034 - test accuracy : 75.756 - time : 31.664 s
rank : 1 - epoch : 18 - train loss : 0.004 - train accuracy : 94.857 - test loss : 0.033 - test accuracy : 74.960 - time : 31.620 s
rank : 0 - epoch : 18 - train loss : 0.004 - train accuracy : 94.809 - test loss : 0.035 - test accuracy : 75.657 - time : 31.553 s
rank : 0 - epoch : 19 - train loss : 0.004 - train accuracy : 95.332 - test loss : 0.035 - test accuracy : 76.771 - time : 31.958 s
rank : 1 - epoch : 19 - train loss : 0.004 - train accuracy : 95.528 - test loss : 0.033 - test accuracy : 77.488 - time : 32.163 s
rank : 0 - epoch : 20 - train loss : 0.004 - train accuracy : 95.712 - test loss : 0.036 - test accuracy : 75.318 - time : 31.771 s
rank : 1 - epoch : 20 - train loss : 0.004 - train accuracy : 95.488 - test loss : 0.034 - test accuracy : 75.856 - time : 31.811 s
total time : 634.106
```

شکل ۶- لاگ چاپ شده در تابع train

ج) همانطور که مشخص است، دقت نهایی مدل در شرایط Data Parallel کمی کمتر شده است. در حالی که بهترین دقت مدل‌ها برای داده‌های تست تفاوت بسیار ناچیزی با همدیگر دارد (در الف برابر 77.840% و در ب برابر 77.488% است). از آنجایی که این تفاوت بسیار ناچیز است، قابل صرفنظر کردن است و می‌توان نتیجه گرفت که در هر دو حالت مدل به دقت یکسانی می‌رسد.

در حالی که مدت زمان اجرای حالت موازی بسیار بیشتر از حالت تک GPU شده است. در حالت استفاده از هر دو GPU هر ایپاک حدودا ۳۱ ثانیه طول کشیده است (طبق شکل ۶) در حالی که در حالت تک GPU (NVIDIA GeForce RTX 3090) هر ایپاک حدودا ۸ ثانیه طول کشیده است (طبق شکل ۵). علت اصلی این اتفاق این است که سایز batch در هر دو مدل ۳۲ فرض شده است. در DDP در ابتدا به هر GPU یک batch داده می‌شود و زمانی که GPU ها محاسباتشان را انجام دادند، گرادیان‌ها میانگین گرفته می‌شوند. زمانی که سایز batch ها کوچک باشد، مدت زمان به اشتراک گذاری گرادیان‌ها نسبت به مدت زمان محاسبات بیشتر می‌شود. در حالی که اگر batch ها بزرگ باشند این اتفاق نمی‌افتد. به همین منظور کدهای بخش **الف** و **ب** با سایز batch برابر با ۲۵۶ هم اجرا شدند که در شکل‌های ۷ و ۸ لاگ‌های چاپ شده‌ی هر کدام قابل نمایش است. با مقایسه‌ی زمان هر ایپاک در شکل‌های ۶ و ۸ متوجه می‌شویم که با افزایش سایز batch مدت زمان اجرای هر

ایپاک از ۳۲ ثانیه به تقریبا ۱۰ ثانیه رسیده است در حالی که این کاهش زمان در حالت استفاده از یک GPU کمتر بوده است.

```
epoch : 1 - train loss : 0.007 - train accuracy : 30.390 - test loss : 0.006 - test accuracy : 42.300 - time : 9.212 s
epoch : 2 - train loss : 0.006 - train accuracy : 48.000 - test loss : 0.005 - test accuracy : 53.160 - time : 7.499 s
epoch : 3 - train loss : 0.005 - train accuracy : 56.082 - test loss : 0.005 - test accuracy : 58.640 - time : 7.477 s
epoch : 4 - train loss : 0.004 - train accuracy : 61.786 - test loss : 0.004 - test accuracy : 62.870 - time : 7.500 s
epoch : 5 - train loss : 0.004 - train accuracy : 65.990 - test loss : 0.004 - test accuracy : 65.860 - time : 7.551 s
epoch : 6 - train loss : 0.003 - train accuracy : 69.508 - test loss : 0.004 - test accuracy : 67.680 - time : 7.519 s
epoch : 7 - train loss : 0.003 - train accuracy : 71.932 - test loss : 0.003 - test accuracy : 69.730 - time : 7.554 s
epoch : 8 - train loss : 0.003 - train accuracy : 74.438 - test loss : 0.003 - test accuracy : 71.270 - time : 7.493 s
epoch : 9 - train loss : 0.003 - train accuracy : 76.314 - test loss : 0.003 - test accuracy : 72.130 - time : 7.471 s
epoch : 10 - train loss : 0.002 - train accuracy : 78.106 - test loss : 0.003 - test accuracy : 72.920 - time : 7.504 s
epoch : 11 - train loss : 0.002 - train accuracy : 80.274 - test loss : 0.003 - test accuracy : 73.000 - time : 7.535 s
epoch : 12 - train loss : 0.002 - train accuracy : 81.274 - test loss : 0.003 - test accuracy : 73.420 - time : 7.541 s
epoch : 13 - train loss : 0.002 - train accuracy : 82.594 - test loss : 0.004 - test accuracy : 72.700 - time : 7.630 s
epoch : 14 - train loss : 0.002 - train accuracy : 83.576 - test loss : 0.004 - test accuracy : 71.800 - time : 7.529 s
epoch : 15 - train loss : 0.002 - train accuracy : 85.532 - test loss : 0.004 - test accuracy : 71.790 - time : 7.510 s
epoch : 16 - train loss : 0.002 - train accuracy : 86.166 - test loss : 0.004 - test accuracy : 71.520 - time : 7.530 s
epoch : 17 - train loss : 0.001 - train accuracy : 88.044 - test loss : 0.004 - test accuracy : 70.770 - time : 7.530 s
epoch : 18 - train loss : 0.001 - train accuracy : 88.850 - test loss : 0.004 - test accuracy : 71.950 - time : 7.543 s
epoch : 19 - train loss : 0.001 - train accuracy : 89.360 - test loss : 0.004 - test accuracy : 72.610 - time : 7.508 s
epoch : 20 - train loss : 0.001 - train accuracy : 91.142 - test loss : 0.005 - test accuracy : 71.870 - time : 7.538 s
total time : 153.137
```

شکل ۷- لاگ چاپ شده برای بخش الف با سایز batch برابر با ۲۵۶

```
(/opt/anaconda3) rostami@Meshki:~/DMLS/main$ CUDA_VISIBLE_DEVICES="0,1" python distributed.py
rank : 0 - epoch : 1 - train loss : 0.008 - train accuracy : 25.207 - test loss : 0.007 - test accuracy : 33.184 - time : 12.134 s
rank : 1 - epoch : 1 - train loss : 0.008 - train accuracy : 25.012 - test loss : 0.007 - test accuracy : 33.965 - time : 12.181 s
rank : 0 - epoch : 2 - train loss : 0.006 - train accuracy : 39.465 - test loss : 0.006 - test accuracy : 42.188 - time : 10.500 s
rank : 1 - epoch : 2 - train loss : 0.006 - train accuracy : 39.278 - test loss : 0.006 - test accuracy : 43.398 - time : 10.499 s
rank : 0 - epoch : 3 - train loss : 0.005 - train accuracy : 48.330 - test loss : 0.005 - test accuracy : 50.859 - time : 10.364 s
rank : 1 - epoch : 3 - train loss : 0.006 - train accuracy : 48.015 - test loss : 0.005 - test accuracy : 52.051 - time : 10.341 s
rank : 0 - epoch : 4 - train loss : 0.005 - train accuracy : 54.568 - test loss : 0.005 - test accuracy : 54.785 - time : 10.408 s
rank : 1 - epoch : 4 - train loss : 0.005 - train accuracy : 54.249 - test loss : 0.005 - test accuracy : 55.488 - time : 10.535 s
rank : 0 - epoch : 5 - train loss : 0.004 - train accuracy : 58.717 - test loss : 0.004 - test accuracy : 58.184 - time : 10.620 s
rank : 1 - epoch : 5 - train loss : 0.005 - train accuracy : 58.606 - test loss : 0.004 - test accuracy : 58.613 - time : 10.498 s
rank : 0 - epoch : 6 - train loss : 0.004 - train accuracy : 61.994 - test loss : 0.004 - test accuracy : 60.371 - time : 10.344 s
rank : 1 - epoch : 6 - train loss : 0.004 - train accuracy : 61.806 - test loss : 0.004 - test accuracy : 61.406 - time : 10.329 s
rank : 0 - epoch : 7 - train loss : 0.004 - train accuracy : 65.119 - test loss : 0.004 - test accuracy : 63.359 - time : 10.390 s
rank : 1 - epoch : 7 - train loss : 0.004 - train accuracy : 64.489 - test loss : 0.004 - test accuracy : 63.379 - time : 10.431 s
rank : 1 - epoch : 8 - train loss : 0.004 - train accuracy : 67.267 - test loss : 0.004 - test accuracy : 65.508 - time : 10.322 s
rank : 0 - epoch : 8 - train loss : 0.003 - train accuracy : 67.897 - test loss : 0.004 - test accuracy : 64.688 - time : 10.407 s
rank : 1 - epoch : 9 - train loss : 0.003 - train accuracy : 69.487 - test loss : 0.003 - test accuracy : 68.262 - time : 10.394 s
rank : 0 - epoch : 9 - train loss : 0.003 - train accuracy : 70.097 - test loss : 0.003 - test accuracy : 66.934 - time : 10.444 s
rank : 1 - epoch : 10 - train loss : 0.003 - train accuracy : 71.716 - test loss : 0.003 - test accuracy : 68.848 - time : 10.577 s
rank : 0 - epoch : 10 - train loss : 0.003 - train accuracy : 72.134 - test loss : 0.003 - test accuracy : 68.066 - time : 10.550 s
rank : 0 - epoch : 11 - train loss : 0.003 - train accuracy : 74.011 - test loss : 0.003 - test accuracy : 68.613 - time : 10.316 s
rank : 1 - epoch : 11 - train loss : 0.003 - train accuracy : 73.298 - test loss : 0.003 - test accuracy : 69.668 - time : 10.419 s
rank : 1 - epoch : 12 - train loss : 0.003 - train accuracy : 74.438 - test loss : 0.003 - test accuracy : 70.273 - time : 10.348 s
rank : 0 - epoch : 12 - train loss : 0.003 - train accuracy : 75.167 - test loss : 0.003 - test accuracy : 69.531 - time : 10.396 s
rank : 0 - epoch : 13 - train loss : 0.003 - train accuracy : 76.339 - test loss : 0.003 - test accuracy : 69.043 - time : 10.467 s
rank : 1 - epoch : 13 - train loss : 0.003 - train accuracy : 75.618 - test loss : 0.003 - test accuracy : 69.434 - time : 10.511 s
rank : 1 - epoch : 14 - train loss : 0.003 - train accuracy : 76.877 - test loss : 0.003 - test accuracy : 68.926 - time : 10.215 s
rank : 0 - epoch : 14 - train loss : 0.002 - train accuracy : 77.714 - test loss : 0.003 - test accuracy : 68.398 - time : 10.294 s
rank : 1 - epoch : 15 - train loss : 0.002 - train accuracy : 78.679 - test loss : 0.003 - test accuracy : 72.129 - time : 10.550 s
rank : 0 - epoch : 15 - train loss : 0.002 - train accuracy : 78.878 - test loss : 0.003 - test accuracy : 71.797 - time : 10.590 s
rank : 1 - epoch : 16 - train loss : 0.002 - train accuracy : 79.863 - test loss : 0.003 - test accuracy : 72.070 - time : 10.351 s
rank : 0 - epoch : 16 - train loss : 0.002 - train accuracy : 79.939 - test loss : 0.003 - test accuracy : 71.230 - time : 10.318 s
rank : 0 - epoch : 17 - train loss : 0.002 - train accuracy : 80.660 - test loss : 0.003 - test accuracy : 70.449 - time : 10.279 s
rank : 1 - epoch : 17 - train loss : 0.002 - train accuracy : 80.309 - test loss : 0.003 - test accuracy : 70.371 - time : 10.345 s
rank : 1 - epoch : 18 - train loss : 0.002 - train accuracy : 81.908 - test loss : 0.003 - test accuracy : 71.504 - time : 10.415 s
rank : 0 - epoch : 18 - train loss : 0.002 - train accuracy : 82.223 - test loss : 0.003 - test accuracy : 71.660 - time : 10.477 s
rank : 1 - epoch : 19 - train loss : 0.002 - train accuracy : 83.219 - test loss : 0.003 - test accuracy : 73.359 - time : 10.436 s
rank : 0 - epoch : 19 - train loss : 0.002 - train accuracy : 83.430 - test loss : 0.003 - test accuracy : 72.441 - time : 10.433 s
rank : 0 - epoch : 20 - train loss : 0.002 - train accuracy : 83.279 - test loss : 0.003 - test accuracy : 72.383 - time : 10.385 s
rank : 1 - epoch : 20 - train loss : 0.002 - train accuracy : 83.279 - test loss : 0.003 - test accuracy : 73.438 - time : 10.473 s
total time : 214.393
```

شکل ۸ - لاگ چاپ شده برای بخش ب با سایز batch ۲۵۶

سوال ۲

الف) با توجه به نمودارهای ۱ و ۲ می‌توان متوجه شد که مدل‌ها بیش برازش شده‌اند. زیرا در هر دو نمودار، از ایپاکی به بعد، مقدار loss آموزش همچنان در حال کاهش است در حالی که مقدار loss تست نه تنها کاهشی نمی‌یابد بلکه بیشتر هم می‌شود. در نتیجه overfitting اتفاق افتاده است.

برای جلوگیری از این اتفاق می‌توان از لایه‌های Dropout یا Batch Normalization استفاده کرد. همچنین می‌توان با کم کردن تعداد ایپاک‌ها یا استفاده از early stopping قبل از اینکه بیش برازش اتفاق بیفتد، آموزش مدل را متوقف کرد.

ب) در روش mixed precision به علت ذخیره‌ی مدل با فرمت fp16 ، سایز مدل کاهش یافته و در نتیجه مدل‌های بزرگتری در حافظه جا می‌شوند. همچنین در این روش، سرعت آموزش بیشتر از حالت استفاده از روش fp32 است.

علت اینکه نمی‌توان همواره از fp16 استفاده کرد این است که ممکن است برخی اوقات تعدادی از وزن‌های بسیار بزرگ یا بسیار کوچک بشوند و در این موارد fp16 نتواند آن‌ها را نشان بدهد. در حالی که در روش mixed precision یک کپی از وزن‌ها به روش fp32 ذخیره می‌شود و این وزن‌ها قبل از شروع محاسبات به فرمت fp16 تبدیل شده و سپس محاسبات بر روی آن انجام می‌شود. پس از اتمام محاسبات از گرادیان‌های به دست آمده برای آپدیت وزن‌های کپی که فرمت fp32 داشتند، استفاده می‌شود.

برای بررسی تاثیر mixed precision، کد بخش ب سوال ۱ را با سایز batch برابر با ۲۵۶ اجرا کرده و سپس با افزودن بخش‌های مربوط به mixed precision (طبق راهنمای پایتورچ) آن را با ۱۰ ایپاک اجرا می‌کنیم. همچنین در ایپاک ۵ ام حافظه‌ی اشغال شده را در هر دو حالت بررسی می‌کنیم.

در شکل ۹ لاگ در حالتی که از mixed precision استفاده شده است، قابل مشاهده است. همانطور که مشخص است، در این حالت آموزش هر ایپاک حدودا ۹.۵ ثانیه طول کشیده است. در شکل ۱۰ لاگ در حالت عادی قابل مشاهده است. علت افزایش زمان اجرای ایپاک‌ها بعد از ایپاک ۴، اجرا شدن فرایندی دیگر است. در نتیجه برای محاسبه‌ی زمان اجرای هر ایپاک تنها به ۴ ایپاک اول اکتفا می‌کنیم. در این حالت آموزش هر ایپاک حدودا ۱۰.۵ ثانیه طول کشیده است. در نتیجه حالت mixed precision توانسته است حدودا ۱۰٪ از زمان آموزش را کاهش دهد. همچنین دقت مدل در حالت mixed precision تنها 0.4% کمتر از حالت عادی است. حافظه‌ی استفاده شده هم در هر دو حالت تقریبا با همدیگر برابر است.

```
(/opt/anaconda3) rostami@Meshki:~/DMLS/main$ CUDA_VISIBLE_DEVICES="0,1" python distributed_fp16.py
rank : 0 - epoch : 1 - train loss : 0.008 - train accuracy : 24.577 - test loss : 0.007 - test accuracy : 33.418 - time : 11.307 s
rank : 1 - epoch : 1 - train loss : 0.008 - train accuracy : 24.554 - test loss : 0.007 - test accuracy : 34.648 - time : 11.364 s
rank : 0 - epoch : 2 - train loss : 0.006 - train accuracy : 39.541 - test loss : 0.006 - test accuracy : 43.496 - time : 9.508 s
rank : 1 - epoch : 2 - train loss : 0.006 - train accuracy : 39.816 - test loss : 0.006 - test accuracy : 43.184 - time : 9.468 s
rank : 0 - epoch : 3 - train loss : 0.006 - train accuracy : 47.465 - test loss : 0.005 - test accuracy : 48.320 - time : 9.561 s
rank : 1 - epoch : 3 - train loss : 0.006 - train accuracy : 47.624 - test loss : 0.005 - test accuracy : 48.477 - time : 9.550 s
rank : 0 - epoch : 4 - train loss : 0.005 - train accuracy : 53.085 - test loss : 0.005 - test accuracy : 52.578 - time : 9.528 s
rank : 1 - epoch : 4 - train loss : 0.005 - train accuracy : 52.930 - test loss : 0.005 - test accuracy : 52.852 - time : 9.526 s
rank : 1 - epoch : 5 - train loss : 0.005 - train accuracy : 57.773 - test loss : 0.005 - test accuracy : 56.934 - time : 9.514 s
Memory usage:  243785728
rank : 0 - epoch : 5 - train loss : 0.005 - train accuracy : 57.840 - test loss : 0.005 - test accuracy : 56.133 - time : 9.571 s
Memory usage:  243916800
rank : 1 - epoch : 6 - train loss : 0.004 - train accuracy : 61.950 - test loss : 0.004 - test accuracy : 59.609 - time : 9.538 s
rank : 0 - epoch : 6 - train loss : 0.004 - train accuracy : 62.237 - test loss : 0.004 - test accuracy : 59.570 - time : 9.530 s
rank : 1 - epoch : 7 - train loss : 0.004 - train accuracy : 64.764 - test loss : 0.004 - test accuracy : 61.602 - time : 9.488 s
rank : 0 - epoch : 7 - train loss : 0.004 - train accuracy : 64.672 - test loss : 0.004 - test accuracy : 61.406 - time : 9.565 s
rank : 1 - epoch : 8 - train loss : 0.004 - train accuracy : 67.467 - test loss : 0.004 - test accuracy : 63.535 - time : 9.620 s
rank : 0 - epoch : 8 - train loss : 0.004 - train accuracy : 67.769 - test loss : 0.004 - test accuracy : 63.301 - time : 9.558 s
rank : 1 - epoch : 9 - train loss : 0.003 - train accuracy : 69.567 - test loss : 0.004 - test accuracy : 63.730 - time : 9.446 s
rank : 0 - epoch : 9 - train loss : 0.003 - train accuracy : 70.029 - test loss : 0.004 - test accuracy : 63.555 - time : 9.493 s
rank : 1 - epoch : 10 - train loss : 0.003 - train accuracy : 71.496 - test loss : 0.004 - test accuracy : 65.039 - time : 9.534 s
rank : 0 - epoch : 10 - train loss : 0.003 - train accuracy : 71.999 - test loss : 0.004 - test accuracy : 65.215 - time : 9.480 s
```

شکل ۹- لاگ مدل در حالت mixed precision با batch ۲۵۶

```
(/opt/anaconda3) rostami@Meshki:~/DMLS/main$ CUDA_VISIBLE_DEVICES="0,1" python distributed.py
rank : 1 - epoch : 1 - train loss : 0.008 - train accuracy : 25.474 - test loss : 0.006 - test accuracy : 37.422 - time : 12.219 s
rank : 0 - epoch : 1 - train loss : 0.008 - train accuracy : 25.195 - test loss : 0.007 - test accuracy : 36.426 - time : 12.226 s
rank : 0 - epoch : 2 - train loss : 0.006 - train accuracy : 41.845 - test loss : 0.006 - test accuracy : 43.965 - time : 10.496 s
rank : 1 - epoch : 2 - train loss : 0.006 - train accuracy : 41.777 - test loss : 0.006 - test accuracy : 45.625 - time : 10.570 s
rank : 0 - epoch : 3 - train loss : 0.005 - train accuracy : 47.935 - test loss : 0.005 - test accuracy : 49.824 - time : 10.583 s
rank : 1 - epoch : 3 - train loss : 0.006 - train accuracy : 47.656 - test loss : 0.005 - test accuracy : 50.859 - time : 10.551 s
rank : 1 - epoch : 4 - train loss : 0.005 - train accuracy : 52.216 - test loss : 0.005 - test accuracy : 55.312 - time : 10.475 s
Memory usage:  241261568
rank : 0 - epoch : 4 - train loss : 0.005 - train accuracy : 53.033 - test loss : 0.005 - test accuracy : 54.043 - time : 10.529 s
Memory usage:  242211840
rank : 0 - epoch : 5 - train loss : 0.005 - train accuracy : 57.685 - test loss : 0.004 - test accuracy : 57.383 - time : 14.159 s
rank : 1 - epoch : 5 - train loss : 0.005 - train accuracy : 56.673 - test loss : 0.004 - test accuracy : 57.949 - time : 14.385 s
rank : 0 - epoch : 6 - train loss : 0.004 - train accuracy : 60.842 - test loss : 0.004 - test accuracy : 58.516 - time : 21.729 s
rank : 1 - epoch : 6 - train loss : 0.004 - train accuracy : 60.132 - test loss : 0.004 - test accuracy : 59.609 - time : 22.011 s
rank : 1 - epoch : 7 - train loss : 0.004 - train accuracy : 63.477 - test loss : 0.004 - test accuracy : 61.953 - time : 21.970 s
rank : 0 - epoch : 7 - train loss : 0.004 - train accuracy : 64.258 - test loss : 0.004 - test accuracy : 61.035 - time : 22.704 s
rank : 0 - epoch : 8 - train loss : 0.004 - train accuracy : 66.817 - test loss : 0.004 - test accuracy : 62.539 - time : 21.898 s
rank : 1 - epoch : 8 - train loss : 0.004 - train accuracy : 66.327 - test loss : 0.004 - test accuracy : 63.047 - time : 22.403 s
rank : 1 - epoch : 9 - train loss : 0.003 - train accuracy : 68.443 - test loss : 0.004 - test accuracy : 64.551 - time : 22.029 s
rank : 0 - epoch : 9 - train loss : 0.003 - train accuracy : 69.033 - test loss : 0.004 - test accuracy : 64.082 - time : 22.610 s
rank : 1 - epoch : 10 - train loss : 0.003 - train accuracy : 70.205 - test loss : 0.004 - test accuracy : 66.074 - time : 22.309 s
rank : 0 - epoch : 10 - train loss : 0.003 - train accuracy : 70.950 - test loss : 0.004 - test accuracy : 65.664 - time : 22.745 s
```

شکل ۱۰- لاگ مدل در حالت عادی با batch ۲۵۶

ج)

۱. استفاده از regularization های L1 و L2 : افزودن ترم‌های L1 یا L2 به تابع خطا یکی از راه‌های جلوگیری از بیش برازش و تعمیم پذیری بهتر مدل است. این ترم‌ها با کاهش مقادیر وزن‌های مدل، باعث می‌شود که مدل تعمیم پذیری بهتری داشته باشد.

۲. استفاده از Dropout: با استفاده از این لایه، در هر iteration تعدادی از نورون‌ها حذف می‌شوند (وزنشان صفر می‌شود) و همین کار باعث می‌شود تا عملکرد مدل تنها وابسته به نورون‌هایی خاص نباشد و همه‌ی مدل‌ها در آموزش نقش داشته باشند.

۳. روش label smoothing: این روش در مسائل classification کاربرد مناسبی دارد. در این مسائل معمولا در لایه‌ی آخر از softmax استفاده می‌شود که باعث می‌شود احتمال انتخاب کلاسی که بیشترین مقدار را دارد،

۱ و بقیه را ۰ در نظر بگیرد. در روش label smoothing با افزودن مقداری نویز به کلاس‌هایی که ۰ هستند و کم کردن این مقدار از کلاسی که ۱ است، باعث می‌شود یادگیری مدل تعمیم پذیرتر باشد.

۴. استفاده از early stopping: در این روش تعداد ایپاک‌ها محدود می‌شود. در این روش زمانی که دیگر پیشرفتی در مقدار خطا یا دقت مدل بر روی داده‌های valid ایجاد نشود، آموزش مدل متوقف می‌شود.

۵. استفاده از لایه‌ی batch normalization: نحوه‌ی کار این لایه بدین صورت است که میانگین خروجی‌ها را نزدیک به صفر و انحراف از معیارشان را ۱ در نظر می‌گیرد. به این ترتیب تمام خروجی‌ها در یک بازه هستند و نرمال شده اند.

در این بخش برای بهبود تعمیم پذیری مدل از لایه‌ی batch normalization در بلاک‌های کانولوشنی، از لایه‌ی dropout در بلاک خطی و انتهایی و همچنین از early stopping با مقدار patience برابر ۲ برای خطای داده‌های تست استفاده شده است. همانطور که در شکل ۱۱ قابل مشاهده است، این تغییرات باعث شده است تا مدل تنها ۶ ایپاک آموزش ببیند. همچنین در شکل ۱۲ تغییرات نمودارهای loss و دقت برای داده‌های تست قابل مشاهده است و نشان می‌دهد که زمانی که مدل در حال نوسان است، متوقف شده است.

```
(/opt/anaconda3) rostami@Meshki:~/DMLS/main$ CUDA_VISIBLE_DEVICES="0,1" python Regularized_distributed_256.py
rank : 0 - epoch : 1 - train loss : 0.006 - train accuracy : 47.800 - test loss : 0.004 - test accuracy : 60.645 - time : 10.811 s
rank : 1 - epoch : 1 - train loss : 0.006 - train accuracy : 46.819 - test loss : 0.004 - test accuracy : 60.840 - time : 10.805 s
rank : 0 - epoch : 2 - train loss : 0.004 - train accuracy : 60.288 - test loss : 0.004 - test accuracy : 66.152 - time : 9.869 s
rank : 1 - epoch : 2 - train loss : 0.005 - train accuracy : 59.419 - test loss : 0.003 - test accuracy : 67.090 - time : 9.868 s
rank : 0 - epoch : 3 - train loss : 0.003 - train accuracy : 74.083 - test loss : 0.003 - test accuracy : 71.289 - time : 9.937 s
rank : 1 - epoch : 3 - train loss : 0.003 - train accuracy : 73.386 - test loss : 0.003 - test accuracy : 72.207 - time : 9.935 s
rank : 0 - epoch : 4 - train loss : 0.002 - train accuracy : 79.129 - test loss : 0.003 - test accuracy : 71.367 - time : 9.964 s
rank : 1 - epoch : 4 - train loss : 0.002 - train accuracy : 78.747 - test loss : 0.003 - test accuracy : 71.660 - time : 9.965 s
rank : 0 - epoch : 5 - train loss : 0.002 - train accuracy : 82.322 - test loss : 0.003 - test accuracy : 73.965 - time : 10.762 s
rank : 1 - epoch : 5 - train loss : 0.002 - train accuracy : 82.059 - test loss : 0.003 - test accuracy : 73.594 - time : 10.761 s
rank : 0 - epoch : 6 - train loss : 0.002 - train accuracy : 85.722 - test loss : 0.003 - test accuracy : 73.809 - time : 10.075 s
rank : 1 - epoch : 6 - train loss : 0.002 - train accuracy : 85.475 - test loss : 0.003 - test accuracy : 73.965 - time : 10.076 s
total time : 65.113
```

شکل ۱۱- لاگ مدل در حال regularized

شکل ۱۲- نمودارهای loss و دقت در حالتی که مدل regularized شده است

د) backend این متد می‌تواند ۳ مقدار gloo، mpi و nccl را داشته باشد. هر کدام از این backend ها می‌تواند متدهایی مشخص برای ارتباط با دیگر نودها را اجرا کند. در حال حاضر با توجه به جدولی که توسط پایتورچ منتشر شده (شکل ۱۳)، gloo و mpi بهترین backend ها برای آموزش توزیع شده روی چند CPU و nccl بهترین backend برای آموزش بر روی چندین GPU است.

در شکل ۱۴ لاگ مدل در حالتی که از nccl برای backend استفاده شده، قابل مشاهده است. شکل ۱۱ آموزش مدل در شرایطی کاملا مشابه با gloo backend را نشان می‌دهد. دقت مدل در این دو حالت با هم تقریبا برابر شده است. در حالی که زمان آموزش برای nccl حدودا 4% کمتر از gloo برای این مدل است که نشان دهنده‌ی این است که nccl برتری کمی نسبت به gloo دارد.

Backend	gloo		mpi		nccl	
Device	CPU	GPU	CPU	GPU	CPU	GPU
send	✓	✗	✓	?	✗	✓
recv	✓	✗	✓	?	✗	✓
broadcast	✓	✓	✓	?	✗	✓
all_reduce	✓	✓	✓	?	✗	✓
reduce	✓	✗	✓	?	✗	✓
all_gather	✓	✗	✓	?	✗	✓
gather	✓	✗	✓	?	✗	✓
scatter	✓	✗	✓	?	✗	✗
reduce_scatter	✗	✗	✗	✗	✗	✓
all_to_all	✗	✗	✓	?	✗	✓
barrier	✓	✗	✓	?	✗	✓

شکل ۱۳- جدول قابلیت‌های backend های مختلف

```
(/opt/anaconda3) rostami@Meshki:~/DMLS/main$ CUDA_VISIBLE_DEVICES="0,1" python Regularized_distributed_256_NCCL.py
rank : 0 - epoch : 1 - train loss : 0.006 - train accuracy : 45.843 - test loss : 0.004 - test accuracy : 58.145 - time : 10.914 s
rank : 1 - epoch : 1 - train loss : 0.006 - train accuracy : 45.863 - test loss : 0.004 - test accuracy : 59.121 - time : 10.997 s
rank : 0 - epoch : 2 - train loss : 0.004 - train accuracy : 64.995 - test loss : 0.003 - test accuracy : 67.051 - time : 9.378 s
rank : 1 - epoch : 2 - train loss : 0.004 - train accuracy : 64.094 - test loss : 0.003 - test accuracy : 68.398 - time : 9.386 s
rank : 0 - epoch : 3 - train loss : 0.003 - train accuracy : 75.466 - test loss : 0.003 - test accuracy : 71.172 - time : 9.466 s
rank : 1 - epoch : 3 - train loss : 0.003 - train accuracy : 74.641 - test loss : 0.003 - test accuracy : 71.738 - time : 9.408 s
rank : 0 - epoch : 4 - train loss : 0.002 - train accuracy : 80.114 - test loss : 0.003 - test accuracy : 71.973 - time : 9.344 s
rank : 1 - epoch : 4 - train loss : 0.002 - train accuracy : 79.632 - test loss : 0.003 - test accuracy : 72.012 - time : 9.419 s
rank : 0 - epoch : 5 - train loss : 0.002 - train accuracy : 83.550 - test loss : 0.003 - test accuracy : 73.184 - time : 9.395 s
rank : 1 - epoch : 5 - train loss : 0.002 - train accuracy : 83.386 - test loss : 0.003 - test accuracy : 73.027 - time : 9.436 s
rank : 0 - epoch : 6 - train loss : 0.001 - train accuracy : 86.775 - test loss : 0.003 - test accuracy : 74.512 - time : 9.398 s
rank : 1 - epoch : 6 - train loss : 0.001 - train accuracy : 86.767 - test loss : 0.003 - test accuracy : 73.984 - time : 9.403 s
total time : 63.224
```

شکل ۱۴- لاگ مدل در شرایطی که از nccl به عنوان backend استفاده شده است