به نام خداوند جان و خرد

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

سامانه‌های یادگیری‌ماشین توزیع شده

تمرین کامپیوتری شماره ۲

نام و نام خانوادگی: علی خرم فر

شماره دانشجویی: ۸۱۰۱۰۲۱۲۹

آذرماه ۱۴۰۳

فهرست مطالب

فهرست اشکال

فهرست جداول

۱_ پاسخ سوال شماره ۱

۱-۱_ الف) تبدیل تصاویر رنگی به سیاه سفید با تابع ساده

در این بخش از تمرین، تصمیم گرفتیم تصاویر رنگی موجود در مجموعه داده STL-10 را به تصاویر سیاه‌وسفید تبدیل کنیم. برای این کار، از فرمولی استفاده کردیم که مقادیر رنگ‌های قرمز، سبز و آبی (R، G و B) را ترکیب می‌کند:

$$\text{Gray} = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B$$

ابتدا، مجموعه داده STL-10 را با استفاده از PyTorch بارگذاری کردیم. از ۲۰۰ تصویر موجود در مجموعه آموزشی به عنوان نمونه استفاده کردیم و برای بهبود مرحله پردازش، اندازه تصاویر را به ۱۲۸ در ۱۲۸ تغییر دادیم. علاوه بر این، مقادیر پیکسل‌ها را نرمال کردیم تا داده‌ها برای مراحل بعدی آماده شوند. سپس، داده‌ها را در یک DataLoader قرار داده شد تا دسترسی به تصاویر آسان‌تر باشد.

برای تبدیل تصاویر، باتوجه به کد ارائه‌شده در نوت بوک تمرین، تابعی به نام rgb2grey_py را تکمیل کردیم که از دو حلقه for برای پیمایش ارتفاع و عرض هر تصویر استفاده می‌کند. در این تابع، مقادیر R، G و B برای هر پیکسل استخراج و مقدار سیاه‌وسفید متناظر محاسبه شد. نتیجه این محاسبات در یک آرایه جدید ذخیره شد که تصویر سیاه‌وسفید نهایی را تشکیل می‌دهد:

```
def rgb2grey_py(image):
    height = image.size(1)
    width = image.size(2)
    gray_image = torch.zeros((height, width))

    for i in range(height):
        for j in range(width):
            r = image[0][i][j]
            g = image[1][i][j]
            b = image[2][i][j]
            gray_value = 0.2989 * r + 0.5870 * g + 0.1140 * b
            gray_image[i][j] = gray_value

    return gray_image
```

این تابع به خوبی عمل کرد و توانستیم تمام ۲۰۰ تصویر را پردازش کنیم. با این حال، سرعت اجرای تابع به دلیل استفاده از حلقه‌های for برای ما زیاد بود. این مسئله باعث شد اهمیت بهینه‌سازی کد در پروژه‌های بزرگ و در دنیای واقعی را درک کنیم. تحلیل بیشتر را در قسمت سوم سوال انجام خواهیم داد.

۱-۲_ ب) تبدیل تصاویر رنگی به سیاه سفید با کد CUDA

کد موجود در نوت بوک ابتدا از یک ساختار پایه شروع شده که شامل تعریف ماکروهایی برای بررسی ورودی‌ها بود تا مطمئن شویم که ورودی‌ها درست بوده و برای جلوگیری از خطاهای ناشی از ورودی‌های

نامعتبر کمک کند. همچنین تابع `cdiv` تعریف شده که ceiling division را انجام می‌دهد تا اندازه گریدها و بلاک‌ها به‌درستی محاسبه شوند.

کد CUDA

در بخش اصلی CUDA، یک kernel به نام rgb_to_grayscale_kernel تکمیل شد که پیکسل‌های تصاویر را به مقادیر سیاه‌وسفید تبدیل می‌کند. این هسته ابتدا شماره پیکسل جاری را با استفاده از ترکیب شماره بلاک و شماره ترد محاسبه می‌کند. سپس، مقدار سیاه‌وسفید برای هر پیکسل بر اساس فرمول گفته شده در فایل تمرین ترکیب رنگ‌ها محاسبه و در آرایه خروجی ذخیره می‌شود.

```
_global__ void rgb_to_grayscale_kernel(unsigned char* x, unsigned char* out, int n)}
  int i = blockIdx.x * blockDim.x + threadIdx.x ;
  if (i < n)}
    out[i] = 0.299f * x[i] + 0.587f * x[i + n] + 0.114f * x[i + 2 * n];
{
{
```

در اینجا هر ترد یک پیکسل را پردازش می‌کند که همان معماری SIMT است که در درس یاد گرفتیم.

برای راحتی استفاده از کد CUDA، تابع rgb_to_grayscale نوشته شد. این تابع ابتدا ورودی را بررسی و تأیید می‌کند، سپس خروجی مناسبی تخصیص می‌دهد و کرنل CUDA را فراخوانی می‌کند کرده و در پایان، تنسور خروجی را برمی‌گرداند.

```
torch::Tensor rgb_to_grayscale(torch::Tensor input)}
  CHECK_INPUT(input) ;
  int h = input.size ;(۱)
  int w = input.size ;(۲)
  int num_pixels = h * w ;

  auto output = torch::empty({h, w}, input.options());

  int threads = 256;

  rgb_to_grayscale_kernel<<<(num_pixels + threads - 1) / threads, threads)<<<
    input.data_ptr<unsigned char>(), output.data_ptr<unsigned char>(), num_pixels;(

  C10_CUDA_KERNEL_LAUNCH_CHECK;()
  return output;
{
```

اجرای کد

در مرحله بعد، تصاویر موجود در مجموعه داده را به GPU منتقل کردیم. برای این کار، ابتدا تصاویر را از بازه یک تا منفی یک به بازه ۰ تا ۲۵۵ تبدیل کردیم. سپس آن‌ها را به نوع داده uint8 تبدیل و به GPU ارسال شد. کرنل CUDA برای تبدیل تصاویر به سیاه‌وسفید فراخوانی شد و نتایج در لیستی ذخیره شدند.

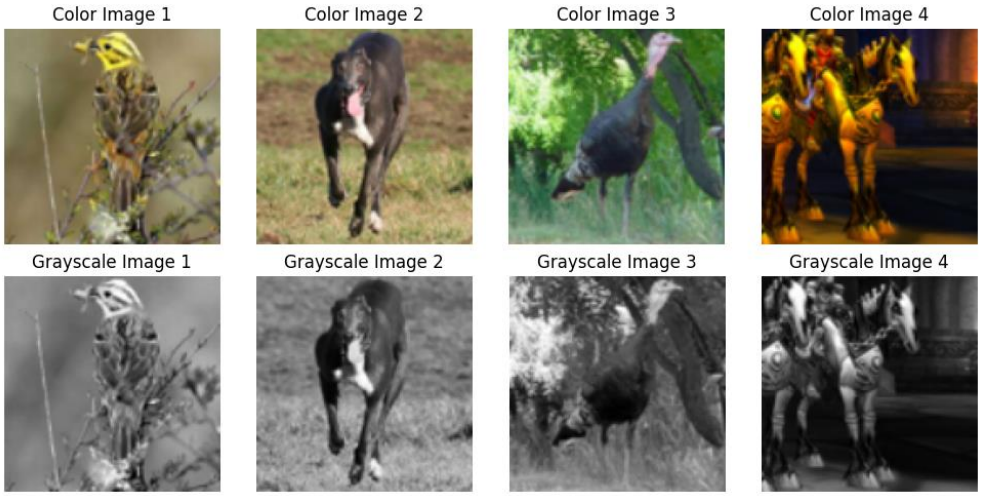
```
start_time_python = time.time()
for i, (image, label) in enumerate(train_dataset_200):
    image = image.to(device)  # Send to the same device as tensor operations

 #    Un-normalize the image from [-1, 1] to[۲۵۵ ,۰]
    image = (image + 1) * 127.5  # Convert from [-1, 1] to[۲۵۵ ,۰]
    image = image.clamp(0, 255)  # Ensure values are in the valid range[۲۵۵ ,۰]

 #    Convert to uint8 and move to GPU (for CUDA kernel)
    image_uint8 = image.to(torch.uint8).contiguous().to(device)

 #    Call the CUDA kernel for grayscale conversion
    gray_image = module.rgb_to_grayscale(image_uint8)  # Pass the uint8 image to the kernel

 #    Append the grayscale image and label
    grayscale_images_part2.append((gray_image, label))

end_time_python = time.time()
CUDA_time = end_time_python - start_time_python
print(f"Time taken for CUDA-based grayscale conversion: {CUDA_time:.4f} seconds")
```

۳-۱_ ج) تحلیل نتایج

در این سوال، هدف ما مقایسه زمان اجرای دو روش تبدیل تصاویر رنگی به سیاه‌وسفید (پایتون و CUDA) و تحلیل نتایج بود. برای این کار، ۲۰۰ تصویر از مجموعه داده STL-10 انتخاب و پردازش شدند. همچنین برای تایید عملکرد هر دو روش، ۴ تصویر رنگی و سیاه‌وسفید متناظر آن‌ها نمایش داده شد.

خروجی کد پایتون

در روش پایتون، به دلیل استفاده از حلقه‌های for برای پردازش تک‌تک پیکسل‌ها، زمان اجرای کد بسیار بالا بود. این امر به‌طور طبیعی به دلیل پردازش ترتیبی است که برای داده‌های تصویری ناکارآمد است.

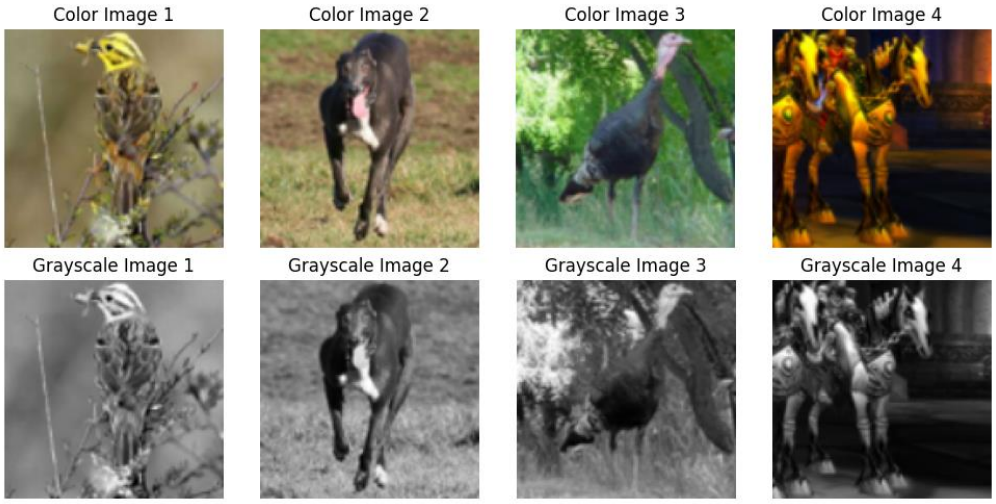


شکل ۱-۱ خروجی تبدیل تصاویر رنگی به سیاه و سفید با کد پایتون

```
Time taken for Python-based grayscale conversion: 205.1622 seconds
```

خروجی کد CUDA

روش CUDA از قدرت GPU برای پردازش هم‌زمان پیکسل‌ها استفاده کرد. در این روش، هر ترد از GPU وظیفه پردازش یک یا چند پیکسل را به عهده دارد که باعث شد فرآیند تبدیل به‌شدت بهینه شود.



شکل ۲-۱ خروجی تبدیل تصاویر رنگی به سیاه و سفید با کد CUDA

```
Time taken for CUDA-based grayscale conversion: 0.1801 seconds
```

مقایسه و تحلیل نتایج

زمان اجرای تبدیل تصاویر به سیاه‌وسفید با استفاده از روش مبتنی بر پایتون حدود ۲۰۵ ثانیه بود. در مقابل، روش مبتنی بر CUDA تنها ۰.۱۸ ثانیه زمان نیاز داشت. این تفاوت زیاد نشان‌دهنده قدرت GPU و بهینه‌سازی کد CUDA در پردازش موازی داده‌هاست، به‌ویژه زمانی که حجم زیادی از داده‌هایی مثل تصاویر نیاز به پردازش دارند. برای اطمینان از صحت عملکرد هر دو روش، چهار تصویر رنگی و سیاه‌وسفید متناظر آن‌ها به‌صورت مقایسه‌ای نمایش داده شد که در شکل‌های قبلی مشاهده کردیم. تصاویر سیاه‌وسفید تولیدشده توسط هر دو روش مشابه بودند، که نشان‌دهنده صحت عملکرد الگوریتم‌هاست.

۲_پاسخ سوال شماره ۲

۲-۱_ الف) اجرا روی یک GPU

در این قسمت هدف ما پیاده‌سازی و آموزش یک شبکه عصبی CNN بر روی یک GPU است

طراحی مدل

مدل ما شامل سه لایه کانولوشنی است که هر لایه کانولوشنی شامل یک عملیات کانولوشن، یک تابع فعال ساز ReLU، و یک عملیات maxpooling است. پس از لایه‌های کانولوشنی، یک لایه fully connected و یک لایه Dropout برای کاهش overfitting اضافه کردیم. در نهایت، لایه خروجی شامل ۱۰ نورون است که هر نورون نماینده یکی از کلاس‌ها است.

```
class SimpleCNN(nn.Module):
  def __init__(self):
    super(SimpleCNN, self).__init()_
#    Convolutional
    self.layer1 = nn.Sequential)
      nn.Conv2d(3, 32, kernel_size=3, padding=1),
      nn.ReLU,()
      nn.MaxPool2d(۲)
(
#    Convolutional
    self.layer2 = nn.Sequential)
      nn.Conv2d(32, 64, kernel_size=3, padding=1),
      nn.ReLU,()
      nn.MaxPool2d(۲)
(
#    Convolutional
    self.layer3 = nn.Sequential)
      nn.Conv2d(64, 128, kernel_size=3, padding=1),
      nn.ReLU,()
      nn.MaxPool2d(۲)
(
#    FC
    self.fc1 = nn.Linear(۵۱۲,۱۶ * ۱۶ * ۱۲۸)
    self.fc2 = nn.Linear (۱۰,۵۱۲)
    self.dropout = nn.Dropout(۰.۲۵)

  def forward(self, x):
    out = self.layer1(x)
    out = self.layer2(out)
    out = self.layer3(out)
    out = out.view(out.size(0), -1)
    out = self.fc1(out)
    out = self.dropout(out)
    out = self.fc2(out)
    return out
```

آموزش مدل

برای آموزش مدل، از تابع خطای CrossEntropy به عنوان معیار بهینه‌سازی استفاده کردیم. بهینه‌ساز Adam با نرخ یادگیری ۰.۰۰۱ استفاده شد و با این هایپرپارامترها آموزش در ۱۰ epoch انجام شد. این

تنظیمات برای کل تمرین ثابت ماند. در این سوال فقط بچ ۳۲ تایی را بررسی خواهیم کرد. همچنین پس از هر epoch، دقت مدل روی مجموعه تست را محاسبه شد.

۲-۲_ ب) اجرای موازی کد روی دو GPU

در این قسمت از سوال با استفاده از PyTorch Distributed Data Parallel (DDP)، کدی نوشتیم که بتواند روی چند GPU اجرا شود. برای این کار، از torch.distributed و کتابخانه torch.multiprocessing استفاده شد.

تابع setup

این تابع آدرس میزبان (MASTER_ADDR)، پورت (MASTER_PORT)، و تنظیمات مربوطه برای هر GPU را مشخص می‌کند.

```
def setup(rank, world_size, master_port, backend, timeout):
  os.environ["MASTER_ADDR"] = 'localhost'
  os.environ["MASTER_PORT"] = master_port
  dist.init_process_group(backend=backend, rank=rank, world_size=world_size, timeout=timeout)
```

تابع انتخاب پورت آزاد:

این تابع تضمین می‌کند که پورت استفاده‌شده قبلاً گرفته نشده باشد.

```
def find_free_port():
  with closing(socket.socket(socket.AF_INET, socket.SOCK_STREAM)) as s:
    s.bind(('', 0))
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    return str(s.getsockname()[1])
```

بارگذاری داده‌ها:

در این بخش، از DistributedSampler استفاده کردیم تا داده‌ها به GPUهای مختلف تقسیم شوند. همچنین، از گزینه‌های pin_memory و persistent_workers برای سرعت بیشتر استفاده شد.

```
def load_data(rank, world_size, batch_size):
  train_sampler = torch.utils.data.distributed.DistributedSampler(train_set, num_replicas=world_size, rank=rank)
  train_loader = torch.utils.data.DataLoader(dataset=train_set,
                    sampler=train_sampler,
                    batch_size=batch_size,
                    shuffle=False,
                    persistent_workers=True,
                    num_workers=1,
                    pin_memory=True(
```

اجرای موازی با mp.spawn:

این خط، تابع train_model را در چند Process جداگانه (یکی برای هر GPU) اجرا می‌کند. پارامتر world_size تعداد GPUها را تعیین می‌کند.

```
mp.spawn(train_model, nprocs=world_size, args=(world_size, master_port, backend, timeout, num_epochs , batch_size), join=True)
```

تغییرات در تابع آموزش:

در تابع train کردن مدل، مدل به DDP تغییر یافت:

```
ddp_model = DDP(model, device_ids=[rank])
```

و داده‌ها و مدل به GPU مربوط به هر رنک تخصیص داده شدند:

```
images, labels = images.to(rank), labels.to(rank)
```

۳-۲_ ج) مقایسه و تحلیل نتایج

مدل اول (Single GPU) فقط روی یک GPU اجرا می‌شد و داده‌ها در یک یکجا پردازش می‌شدند.

مدل دوم (Multi-GPU) داده‌ها بین GPUهای مختلف تقسیم شدند و پردازش‌ها به صورت موازی انجام شد که باعث کاهش زمان کل آموزش شد.

نتایج حالت الف

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier.py
Using device: cuda
Epoch [1/10], Train Loss: 1.8551, Train Accuracy: 31.28%, Test Loss: 1.5276, Test Accuracy: 42.91%
Epoch [2/10], Train Loss: 1.4067, Train Accuracy: 47.30%, Test Loss: 1.4228, Test Accuracy: 47.08%
Epoch [3/10], Train Loss: 1.2000, Train Accuracy: 56.08%, Test Loss: 1.4209, Test Accuracy: 48.40%
Epoch [4/10], Train Loss: 1.0011, Train Accuracy: 62.46%, Test Loss: 1.3434, Test Accuracy: 53.80%
Epoch [5/10], Train Loss: 0.8044, Train Accuracy: 71.62%, Test Loss: 1.3800, Test Accuracy: 54.10%
Epoch [6/10], Train Loss: 0.5519, Train Accuracy: 80.72%, Test Loss: 1.7594, Test Accuracy: 50.77%
Epoch [7/10], Train Loss: 0.3980, Train Accuracy: 85.64%, Test Loss: 2.0004, Test Accuracy: 51.66%
Epoch [8/10], Train Loss: 0.2635, Train Accuracy: 90.60%, Test Loss: 2.2445, Test Accuracy: 52.15%
Epoch [9/10], Train Loss: 0.1823, Train Accuracy: 94.06%, Test Loss: 2.7230, Test Accuracy: 51.91%
Epoch [10/10], Train Loss: 0.1840, Train Accuracy: 93.70%, Test Loss: 2.9123, Test Accuracy: 50.60%
CUDA Memory Usage: 497.94 MB
Test Accuracy: 50.60%
Total Time: 179.19 seconds
Batch size:32
```

مدل پس از ۱۰ ایپاک به دقت **۵۰.۶۰** درصد روی دیتاست تست رسید. باتوجه به تغییر دقت و لاس مربوط به داده آموزش میتوانیم حدس بزنیم که بیش‌برازش رخ داده است.

زمان کل اجرای آموزش **۱۷۹.۱۹** ثانیه بود که به دلیل استفاده از یک GPU، این زمان نسبتاً زیاد است.

همچنین میزان حافظه مصرف‌شده توسط GPU برابر با **۴۹۷.۹۴** مگابایت بود.

شکل ۱-۲ نمودار عملکرد GPU روی یک GPU و اندازه Batch ۳۲

همان‌طور که در تصویربالا مشاهده می‌کنیم، GPU به‌طور متناوب مشغول پردازش بوده و الگوی استفاده از منابع نسبتاً ثابت بود.

نتایج حالت ب)

```
Rank 0, Epoch [1/10], Train Loss: 1.8657, Train Accuracy: 33.08%, Test Loss: 1.6614, Test Accuracy: 38.40%
Rank 1, Epoch [1/10], Train Loss: 1.8684, Train Accuracy: 31.64%, Test Loss: 1.6642, Test Accuracy: 39.30%
Rank 0, Epoch [2/10], Train Loss: 1.4045, Train Accuracy: 47.88%, Test Loss: 1.3573, Test Accuracy: 49.95%
Rank 1, Epoch [2/10], Train Loss: 1.4020, Train Accuracy: 47.00%, Test Loss: 1.3795, Test Accuracy: 50.17%
Rank 0, Epoch [3/10], Train Loss: 1.1723, Train Accuracy: 57.32%, Test Loss: 1.2485, Test Accuracy: 54.48%
Rank 1, Epoch [3/10], Train Loss: 1.1446, Train Accuracy: 57.36%, Test Loss: 1.2763, Test Accuracy: 54.75%
Rank 0, Epoch [4/10], Train Loss: 0.9372, Train Accuracy: 65.92%, Test Loss: 1.2684, Test Accuracy: 54.98%
Rank 1, Epoch [4/10], Train Loss: 0.9211, Train Accuracy: 66.88%, Test Loss: 1.3156, Test Accuracy: 54.08%
Rank 0, Epoch [5/10], Train Loss: 0.7468, Train Accuracy: 72.48%, Test Loss: 1.4082, Test Accuracy: 54.35%
Rank 1, Epoch [5/10], Train Loss: 0.7399, Train Accuracy: 72.76%, Test Loss: 1.4573, Test Accuracy: 52.88%
Rank 0, Epoch [6/10], Train Loss: 0.6346, Train Accuracy: 76.84%, Test Loss: 1.4727, Test Accuracy: 54.23%
Rank 1, Epoch [6/10], Train Loss: 0.6264, Train Accuracy: 76.92%, Test Loss: 1.5111, Test Accuracy: 53.80%
Rank 1, Epoch [7/10], Train Loss: 0.5399, Train Accuracy: 81.20%, Test Loss: 1.8620, Test Accuracy: 48.88%
```

```
Rank 0, Epoch [7/10], Train Loss: 0.5535, Train Accuracy: 79.96%, Test Loss: 1.8475, Test
Accuracy: 49.67%
Rank 0, Epoch [8/10], Train Loss: 0.4492, Train Accuracy: 83.72%, Test Loss: 2.1819, Test
Accuracy: 49.33%
Rank 1, Epoch [8/10], Train Loss: 0.4539, Train Accuracy: 83.36%, Test Loss: 2.2056, Test
Accuracy: 48.33%
Rank 0, Epoch [9/10], Train Loss: 0.3702, Train Accuracy: 86.48%, Test Loss: 2.2545, Test
Accuracy: 51.27%
Rank 1, Epoch [9/10], Train Loss: 0.3747, Train Accuracy: 86.48%, Test Loss: 2.2977, Test
Accuracy: 51.95%
Rank 1, Epoch [10/10], Train Loss: 0.2981, Train Accuracy: 89.40%, Test Loss: 2.6745, Test
Accuracy: 49.25%
Rank 1, Training Time: 52.41 seconds
Rank 1, CUDA Memory Usage: 562.32 MB
Rank 1, Test Accuracy: 49.25%
Rank 0, Epoch [10/10], Train Loss: 0.2847, Train Accuracy: 89.48%, Test Loss: 2.5550, Test
Accuracy: 51.52%
Rank 0, Training Time: 52.41 seconds
Rank 0, CUDA Memory Usage: 562.32 MB
Rank 0, Test Accuracy: 51.52%
Total time: 63.88 seconds
Batch size:32
Backend:nccl
```

دقت نهایی رنک صفر = **۵۱.۵۲**

دقت نهایی رنک یک = **۴۹.۲۵**

تفاوت جزئی بین دقت دو GPU مشاهده شد، که می‌تواند به نحوه تقسیم داده‌ها بین GPUها و تصادفی‌بودن فرآیند آموزش مربوط باشد.

زمان کل اجرای آموزش **۶۳.۸۸** ثانیه بود. این کاهش زمان نشان‌دهنده کارایی بالای استفاده از چند GPU است که هر GPU بخشی از داده‌ها را پردازش کرده و آموزش به صورت موازی انجام میشود.

حافظه مصرفی هر GPU برابر با **۵۶۲.۳۲** مگابایت بود. این افزایش در حافظه می‌تواند به دلیل ارتباط بین GPUها و نگهداری داده‌های اضافی مرتبط با آموزش موازی باشد.

شکل ۲-۲ نمودار عملکرد GPU روی دو GPU موازی و اندازه Batch ۳۲

تصویر بالا نشان می‌دهد که هر دو GPU به‌طور همزمان مشغول پردازش هستند و زمان بیکاری GPUها کاهش یافته که این نشان‌دهنده استفاده بهتر از منابع سخت‌افزاری است.

تحلیل و مقایسه نتایج

استفاده از چند GPU در این تمرین توانست زمان کل آموزش را به میزان قابل‌توجهی کاهش دهد، اما دقت مدل تغییر خاصی نداشت و زمان کل را از **۱۷۹** ثانیه در حالت تکGPU به **۶۴** ثانیه کاهش داد. این بهبود نشان می‌دهد که پردازش موازی چگونه می‌تواند کارایی را افزایش دهد. مصرف حافظه نیز در حالت چند GPU بیشتر بود، که این افزایش به دلیل سربار ارتباطاتی بین GPUها طبیعی به نظر می‌رسد. همچنین نمودار عملکرد نشان داد که در حالت چند GPU، منابع سخت‌افزاری به طور مداوم‌تر و بهتر درگیر شدند، در حالی که در حالت تکGPU استفاده از منابع در مقاطعی متوقف می‌شد. در مجموع، استفاده از چند GPU در این تمرین توانست زمان اجرا را به‌خوبی کاهش دهد، اما برای بهبود دقت، ممکن است نیاز به تنظیمات بیشتری داشته باشیم که در ادامه تمرین بعضی آن‌ها را بررسی می‌کنیم.

۳_ پاسخ سوال شماره ۳

۳-۱_ آموزش با پارامترهای مختلف

آموزش با یک GPU و Batch Size برابر با ۱۶

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier.py
Using device: cuda
Epoch [1/10], Train Loss: 1.7972, Train Accuracy: 32.76%, Test Loss: 1.6142, Test Accuracy: 39.33%
Epoch [2/10], Train Loss: 1.4443, Train Accuracy: 46.04%, Test Loss: 1.4174, Test Accuracy: 47.42%
Epoch [3/10], Train Loss: 1.2257, Train Accuracy: 54.70%, Test Loss: 1.2778, Test Accuracy: 53.90%
Epoch [4/10], Train Loss: 0.9847, Train Accuracy: 63.32%, Test Loss: 1.2962, Test Accuracy: 54.56%
Epoch [5/10], Train Loss: 0.7019, Train Accuracy: 74.60%, Test Loss: 1.6162, Test Accuracy: 52.09%
Epoch [6/10], Train Loss: 0.4554, Train Accuracy: 83.66%, Test Loss: 1.7293, Test Accuracy: 52.64%
Epoch [7/10], Train Loss: 0.2757, Train Accuracy: 90.54%, Test Loss: 2.2019, Test Accuracy: 52.36%
Epoch [8/10], Train Loss: 0.1922, Train Accuracy: 93.18%, Test Loss: 2.4601, Test Accuracy: 53.69%
Epoch [9/10], Train Loss: 0.1671, Train Accuracy: 94.26%, Test Loss: 2.7448, Test Accuracy: 52.85%
Epoch [10/10], Train Loss: 0.1693, Train Accuracy: 94.32%, Test Loss: 3.0999, Test Accuracy: 50.23%
CUDA Memory Usage: 387.82 MB
Test Accuracy: 50.23%
Total Time: 185.15 seconds
```

در این آزمایش، از یک GPU با سایز batch برابر با ۱۶ استفاده کردیم. نتایج نشان داد که زمان کل آموزش برابر با **۱۸۵.۱۵** ثانیه بود و مصرف حافظه GPU به ۳۸۷.۸۲ مگابایت رسید. دقت نهایی مدل بر روی مجموعه تست، **۵۰.۲۳** درصد بود.

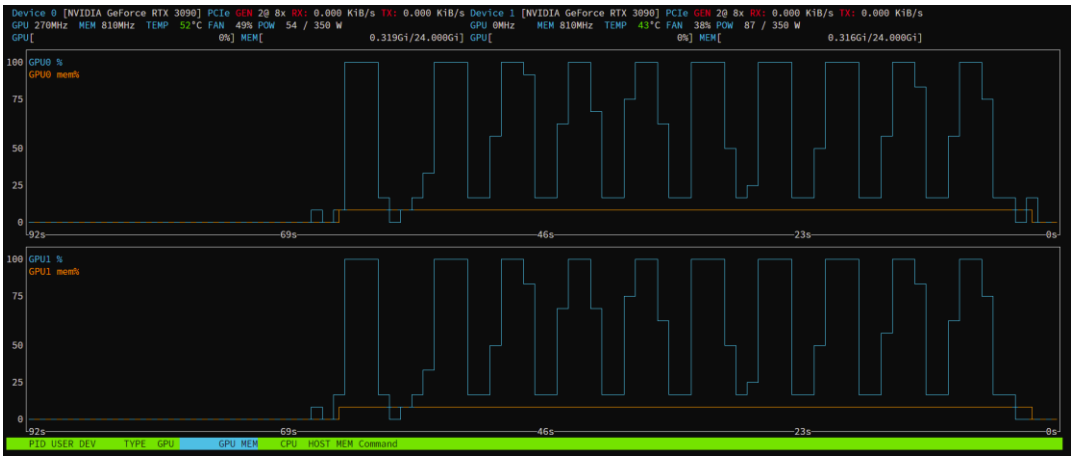
شکل ۱-۳ وضعیت GPU در حالت آموزش با یک GPU و Batch Size برابر با ۱۶

آموزش با دو GPU و Batch Size برابر با ۱۶

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier mp.py
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Rank 0, Epoch [1/10], Train Loss: 1.7865, Train Accuracy: 35.32%, Test Loss: 1.4631, Test
Accuracy: 44.38%
Rank 1, Epoch [1/10], Train Loss: 1.7903, Train Accuracy: 33.32%, Test Loss: 1.4814, Test
Accuracy: 44.25%
Rank 0, Epoch [2/10], Train Loss: 1.3762, Train Accuracy: 50.04%, Test Loss: 1.3537, Test
Accuracy: 50.20%
Rank 1, Epoch [2/10], Train Loss: 1.3672, Train Accuracy: 48.96%, Test Loss: 1.4000, Test
Accuracy: 49.15%
Rank 0, Epoch [3/10], Train Loss: 1.1482, Train Accuracy: 58.60%, Test Loss: 1.4230, Test
Accuracy: 50.75%
Rank 1, Epoch [3/10], Train Loss: 1.1386, Train Accuracy: 57.48%, Test Loss: 1.4498, Test
Accuracy: 50.10%
Rank 0, Epoch [4/10], Train Loss: 0.9361, Train Accuracy: 66.36%, Test Loss: 1.4819, Test
Accuracy: 51.73%
Rank 1, Epoch [4/10], Train Loss: 0.9527, Train Accuracy: 65.04%, Test Loss: 1.5218, Test
Accuracy: 50.67%
Rank 1, Epoch [5/10], Train Loss: 0.7861, Train Accuracy: 72.52%, Test Loss: 1.8548, Test
Accuracy: 47.52%
Rank 0, Epoch [5/10], Train Loss: 0.7575, Train Accuracy: 72.72%, Test Loss: 1.8221, Test
Accuracy: 46.40%
Rank 1, Epoch [6/10], Train Loss: 0.6279, Train Accuracy: 77.76%, Test Loss: 1.8962, Test
Accuracy: 48.90%
Rank 0, Epoch [6/10], Train Loss: 0.6492, Train Accuracy: 77.04%, Test Loss: 1.8620, Test
Accuracy: 47.77%
Rank 1, Epoch [7/10], Train Loss: 0.4990, Train Accuracy: 81.68%, Test Loss: 2.1748, Test
Accuracy: 48.62%
Rank 0, Epoch [7/10], Train Loss: 0.4865, Train Accuracy: 82.88%, Test Loss: 2.1743, Test
Accuracy: 48.80%
Rank 0, Epoch [8/10], Train Loss: 0.3855, Train Accuracy: 85.80%, Test Loss: 2.2216, Test
Accuracy: 51.08%
Rank 1, Epoch [8/10], Train Loss: 0.3949, Train Accuracy: 85.96%, Test Loss: 2.2614, Test
Accuracy: 50.15%
Rank 1, Epoch [9/10], Train Loss: 0.3204, Train Accuracy: 88.68%, Test Loss: 2.5688, Test
Accuracy: 51.25%
Rank 0, Epoch [9/10], Train Loss: 0.3156, Train Accuracy: 88.88%, Test Loss: 2.4882, Test
Accuracy: 51.05%
Rank 0, Epoch [10/10], Train Loss: 0.2507, Train Accuracy: 91.20%, Test Loss: 2.9409, Test
Accuracy: 49.40%
Rank 1, Epoch [10/10], Train Loss: 0.2552, Train Accuracy: 91.36%, Test Loss: 2.9721, Test
Accuracy: 49.15%
```

```
Rank 0, Training Time: 68.86 seconds
Rank 1, Training Time: 68.86 seconds
Rank 0, CUDA Memory Usage: 452.20 MB
Rank 1, CUDA Memory Usage: 452.20 MB
Rank 0, Test Accuracy: 49.40%
Rank 1, Test Accuracy: 49.15%
Total time: 80.60 seconds
Batch size:16
Backend:nccl
```

با افزایش تعداد GPUها به دو و نگه‌داشتن سایز batch در مقدار ۱۶، توانستیم زمان آموزش را به **۸۰.۶۰** ثانیه کاهش دهیم. مصرف حافظه هر GPU به **۴۵۲.۲۰** مگابایت افزایش یافت و دقت نهایی مدل برابر با **۴۹.۴۰** در Rank 0 و **۴۹.۱۵** در Rank 1 بود.



شکل ۲-۳ وضعیت GPU در حالت آموزش با دو GPU و Batch Size برابر با ۱۶

آموزش با یک GPU و Batch Size برابر با ۳۲

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier.py
Using device: cuda
Epoch [1/10], Train Loss: 1.8551, Train Accuracy: 31.28%, Test Loss: 1.5276, Test Accuracy:
42.91%
Epoch [2/10], Train Loss: 1.4067, Train Accuracy: 47.30%, Test Loss: 1.4228, Test Accuracy:
47.08%
Epoch [3/10], Train Loss: 1.2000, Train Accuracy: 56.08%, Test Loss: 1.4209, Test Accuracy:
48.40%
Epoch [4/10], Train Loss: 1.0011, Train Accuracy: 62.46%, Test Loss: 1.3434, Test Accuracy:
53.80%
Epoch [5/10], Train Loss: 0.8044, Train Accuracy: 71.62%, Test Loss: 1.3800, Test Accuracy:
54.10%
Epoch [6/10], Train Loss: 0.5519, Train Accuracy: 80.72%, Test Loss: 1.7594, Test Accuracy:
50.77%
Epoch [7/10], Train Loss: 0.3980, Train Accuracy: 85.64%, Test Loss: 2.0004, Test Accuracy:
51.66%
Epoch [8/10], Train Loss: 0.2635, Train Accuracy: 90.60%, Test Loss: 2.2445, Test Accuracy:
52.15%
Epoch [9/10], Train Loss: 0.1823, Train Accuracy: 94.06%, Test Loss: 2.7230, Test Accuracy:
51.91%
Epoch [10/10], Train Loss: 0.1840, Train Accuracy: 93.70%, Test Loss: 2.9123, Test Accuracy:
50.60%
CUDA Memory Usage: 497.94 MB
Test Accuracy: 50.60%
Total Time: 179.19 seconds
Batch size:32
```

در این حالت، زمان آموزش برابر با **۱۷۹.۱۹** ثانیه بود و مصرف حافظه GPU به **۴۹۷.۹۴** مگابایت رسید. دقت نهایی مدل **۵۰.۶۰ درصد** بود که کمی بهتر از حالت batch برابر با ۱۶ است.

شکل ۳-۳ وضعیت GPU در حالت آموزش با یک GPU و Batch Size برابر با ۳۲

آموزش با دو GPU و Batch Size برابر با ۳۲

```
Rank 0, Epoch [1/10], Train Loss: 1.8657, Train Accuracy: 33.08%, Test Loss: 1.6614, Test
Accuracy: 38.40%
Rank 1, Epoch [1/10], Train Loss: 1.8684, Train Accuracy: 31.64%, Test Loss: 1.6642, Test
Accuracy: 39.30%
Rank 0, Epoch [2/10], Train Loss: 1.4045, Train Accuracy: 47.88%, Test Loss: 1.3573, Test
Accuracy: 49.95%
Rank 1, Epoch [2/10], Train Loss: 1.4020, Train Accuracy: 47.00%, Test Loss: 1.3795, Test
Accuracy: 50.17%
Rank 0, Epoch [3/10], Train Loss: 1.1723, Train Accuracy: 57.32%, Test Loss: 1.2485, Test
Accuracy: 54.48%
Rank 1, Epoch [3/10], Train Loss: 1.1446, Train Accuracy: 57.36%, Test Loss: 1.2763, Test
Accuracy: 54.75%
Rank 0, Epoch [4/10], Train Loss: 0.9372, Train Accuracy: 65.92%, Test Loss: 1.2684, Test
Accuracy: 54.98%
Rank 1, Epoch [4/10], Train Loss: 0.9211, Train Accuracy: 66.88%, Test Loss: 1.3156, Test
Accuracy: 54.08%
Rank 0, Epoch [5/10], Train Loss: 0.7468, Train Accuracy: 72.48%, Test Loss: 1.4082, Test
Accuracy: 54.35%
Rank 1, Epoch [5/10], Train Loss: 0.7399, Train Accuracy: 72.76%, Test Loss: 1.4573, Test
Accuracy: 52.88%
Rank 0, Epoch [6/10], Train Loss: 0.6346, Train Accuracy: 76.84%, Test Loss: 1.4727, Test
Accuracy: 54.23%
Rank 1, Epoch [6/10], Train Loss: 0.6264, Train Accuracy: 76.92%, Test Loss: 1.5111, Test
Accuracy: 53.80%
Rank 1, Epoch [7/10], Train Loss: 0.5399, Train Accuracy: 81.20%, Test Loss: 1.8620, Test
Accuracy: 48.88%
Rank 0, Epoch [7/10], Train Loss: 0.5535, Train Accuracy: 79.96%, Test Loss: 1.8475, Test
Accuracy: 49.67%
Rank 0, Epoch [8/10], Train Loss: 0.4492, Train Accuracy: 83.72%, Test Loss: 2.1819, Test
Accuracy: 49.33%
Rank 1, Epoch [8/10], Train Loss: 0.4539, Train Accuracy: 83.36%, Test Loss: 2.2056, Test
Accuracy: 48.33%
Rank 0, Epoch [9/10], Train Loss: 0.3702, Train Accuracy: 86.48%, Test Loss: 2.2545, Test
Accuracy: 51.27%
Rank 1, Epoch [9/10], Train Loss: 0.3747, Train Accuracy: 86.48%, Test Loss: 2.2977, Test
Accuracy: 51.95%
Rank 1, Epoch [10/10], Train Loss: 0.2981, Train Accuracy: 89.40%, Test Loss: 2.6745, Test
Accuracy: 49.25%
Rank 1, Training Time: 52.41 seconds
Rank 1, CUDA Memory Usage: 562.32 MB
Rank 1, Test Accuracy: 49.25%
Rank 0, Epoch [10/10], Train Loss: 0.2847, Train Accuracy: 89.48%, Test Loss: 2.5550, Test
Accuracy: 51.52%
```

```
Rank 0, Training Time: 52.41 seconds
Rank 0, CUDA Memory Usage: 562.32 MB
Rank 0, Test Accuracy: 51.52%
Total time: 63.88 seconds
Batch size:32
Backend:nccl
```

با استفاده از دو GPU و سایز batch برابر با ۳۲، زمان کل آموزش به **۶۳.۸۸** ثانیه کاهش یافت و مصرف حافظه هر GPU به **۵۶۲.۳۲** مگابایت رسید. دقت نهایی مدل برای Rank 0 برابر با **۵۱.۵۲** و برای Rank 1 برابر با **۴۹.۲۵** بود و زمان آموزش به‌طور چشمگیری کاهش پیدا کرد در حالی که دقت در سطح قابل قبولی باقی ماند.

شکل ۴-۳ وضعیت GPU در حالت آموزش با دو GPU و Batch Size برابر با ۳۲

آموزش با یک GPU و Batch Size برابر با ۶۴

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier.py
Using device: cuda
Epoch [1/10], Train Loss: 1.9070, Train Accuracy: 31.28%, Test Loss: 1.5881, Test Accuracy:
40.08%
Epoch [2/10], Train Loss: 1.4673, Train Accuracy: 46.20%, Test Loss: 1.4012, Test Accuracy:
48.60%
Epoch [3/10], Train Loss: 1.2361, Train Accuracy: 54.78%, Test Loss: 1.3064, Test Accuracy:
52.84%
Epoch [4/10], Train Loss: 1.0511, Train Accuracy: 61.26%, Test Loss: 1.3168, Test Accuracy:
53.12%
Epoch [5/10], Train Loss: 0.8516, Train Accuracy: 69.04%, Test Loss: 1.3736, Test Accuracy:
54.51%
Epoch [6/10], Train Loss: 0.6529, Train Accuracy: 76.74%, Test Loss: 1.5741, Test Accuracy:
51.04%
Epoch [7/10], Train Loss: 0.4792, Train Accuracy: 82.70%, Test Loss: 1.7526, Test Accuracy:
54.12%
Epoch [8/10], Train Loss: 0.3196, Train Accuracy: 88.84%, Test Loss: 1.9375, Test Accuracy:
54.98%
Epoch [9/10], Train Loss: 0.2226, Train Accuracy: 92.40%, Test Loss: 2.2189, Test Accuracy:
54.09%
Epoch [10/10], Train Loss: 0.1326, Train Accuracy: 95.66%, Test Loss: 2.5879, Test Accuracy:
54.05%
CUDA Memory Usage: 719.94 MB
Test Accuracy: 54.05%
Total Time: 168.66 seconds
Batch size:64
```

در این آزمایش، زمان آموزش به **۱۶۸.۶۶** ثانیه کاهش یافت و مصرف حافظه GPU به **۷۱۹.۹۴** مگابایت رسید. دقت نهایی مدل **۵۴.۰۵** درصد بود که نشان می‌دهد افزایش سایز batch به بهبود دقت کمک میکند، اما مصرف حافظه GPU نیز افزایش می‌یابد.

شکل ۵-۳ وضعیت GPU در حالت آموزش با یک GPU و Batch Size برابر با ۶۴

آموزش با دو GPU و Batch Size برابر با ۶۴

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier_mp.py
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Rank 1, Epoch [1/10], Train Loss: 2.1372, Train Accuracy: 26.08%, Test Loss: 1.9422, Test
Accuracy: 30.48%
Rank 0, Epoch [1/10], Train Loss: 2.1157, Train Accuracy: 25.72%, Test Loss: 1.9612, Test
Accuracy: 28.55%
Rank 0, Epoch [2/10], Train Loss: 1.6063, Train Accuracy: 39.20%, Test Loss: 1.5318, Test
Accuracy: 43.83%
Rank 1, Epoch [2/10], Train Loss: 1.6002, Train Accuracy: 40.48%, Test Loss: 1.5209, Test
Accuracy: 44.48%
Rank 0, Epoch [3/10], Train Loss: 1.3970, Train Accuracy: 47.96%, Test Loss: 1.4491, Test
Accuracy: 46.83%
Rank 1, Epoch [3/10], Train Loss: 1.3929, Train Accuracy: 47.20%, Test Loss: 1.4502, Test
Accuracy: 47.77%
Rank 0, Epoch [4/10], Train Loss: 1.2734, Train Accuracy: 52.08%, Test Loss: 1.3707, Test
Accuracy: 50.77%
Rank 1, Epoch [4/10], Train Loss: 1.2535, Train Accuracy: 53.72%, Test Loss: 1.3829, Test
Accuracy: 50.20%
Rank 0, Epoch [5/10], Train Loss: 1.1502, Train Accuracy: 56.68%, Test Loss: 1.3478, Test
Accuracy: 52.58%
Rank 1, Epoch [5/10], Train Loss: 1.1281, Train Accuracy: 58.52%, Test Loss: 1.3919, Test
Accuracy: 50.98%
Rank 0, Epoch [6/10], Train Loss: 1.0444, Train Accuracy: 59.92%, Test Loss: 1.3650, Test
Accuracy: 51.42%
Rank 1, Epoch [6/10], Train Loss: 1.0133, Train Accuracy: 62.12%, Test Loss: 1.4299, Test
Accuracy: 49.80%
Rank 0, Epoch [7/10], Train Loss: 0.9361, Train Accuracy: 65.84%, Test Loss: 1.4810, Test
Accuracy: 49.70%
Rank 1, Epoch [7/10], Train Loss: 0.9167, Train Accuracy: 66.64%, Test Loss: 1.5265, Test
Accuracy: 49.23%
Rank 0, Epoch [8/10], Train Loss: 0.8290, Train Accuracy: 68.92%, Test Loss: 1.5012, Test
Accuracy: 50.45%
Rank 1, Epoch [8/10], Train Loss: 0.8192, Train Accuracy: 70.32%, Test Loss: 1.5236, Test
Accuracy: 51.08%
```

```
Rank 1, Epoch [9/10], Train Loss: 0.7012, Train Accuracy: 74.28%, Test Loss: 1.7544, Test
Accuracy: 48.62%
Rank 0, Epoch [9/10], Train Loss: 0.7007, Train Accuracy: 74.44%, Test Loss: 1.7244, Test
Accuracy: 48.77%
Rank 0, Epoch [10/10], Train Loss: 0.6425, Train Accuracy: 76.00%, Test Loss: 1.6434, Test
Accuracy: 51.00%
Rank 0, Training Time: 48.78 seconds
Rank 0, CUDA Memory Usage: 784.32 MB
Rank 0, Test Accuracy: 51.00%
Rank 1, Epoch [10/10], Train Loss: 0.6163, Train Accuracy: 76.44%, Test Loss: 1.6990, Test
Accuracy: 50.62%
Rank 1, Training Time: 48.81 seconds
Rank 1, CUDA Memory Usage: 784.32 MB
Rank 1, Test Accuracy: 50.62%
Total time: 60.64 seconds
Batch size:64
Backend:nccl
```

استفاده از دو GPU و سایز batch برابر با **۶۴**، زمان کل آموزش را به **۶۰.۶۴** ثانیه کاهش داد. مصرف حافظه هر GPU به **۷۸۴.۳۲** مگابایت رسید و دقت نهایی مدل برای Rank 0 برابر با **۵۱** و برای Rank 1 برابر با **۵۰.۶۲** درصد بود.

شکل ۶-۳ وضعیت GPU در حالت آموزش با دو GPU و Batch Size برابر با ۶۴

آموزش با یک GPU و Batch Size برابر با ۱۲۸

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier.py
Using device: cuda
Epoch [1/10], Train Loss: 2.1208, Train Accuracy: 25.82%, Test Loss: 1.7230, Test Accuracy:
34.30%
Epoch [2/10], Train Loss: 1.6277, Train Accuracy: 39.78%, Test Loss: 1.5845, Test Accuracy:
36.70%
Epoch [3/10], Train Loss: 1.4506, Train Accuracy: 46.14%, Test Loss: 1.4282, Test Accuracy:
47.12%
Epoch [4/10], Train Loss: 1.2876, Train Accuracy: 52.36%, Test Loss: 1.3776, Test Accuracy:
48.70%
Epoch [5/10], Train Loss: 1.1932, Train Accuracy: 57.48%, Test Loss: 1.4919, Test Accuracy:
46.27%
Epoch [6/10], Train Loss: 1.1188, Train Accuracy: 59.92%, Test Loss: 1.3227, Test Accuracy:
52.73%
Epoch [7/10], Train Loss: 1.0134, Train Accuracy: 63.28%, Test Loss: 1.3235, Test Accuracy:
53.25%
Epoch [8/10], Train Loss: 0.9098, Train Accuracy: 67.72%, Test Loss: 1.4061, Test Accuracy:
52.34%
Epoch [9/10], Train Loss: 0.8521, Train Accuracy: 69.50%, Test Loss: 1.4735, Test Accuracy:
52.21%
```

```
Epoch [10/10], Train Loss: 0.7903, Train Accuracy: 70.64%, Test Loss: 1.3666, Test Accuracy:
55.84%
CUDA Memory Usage: 1163.95 MB
Test Accuracy: 55.84%
Total Time: 156.75 seconds
Batch size:128
```

در این حالت، زمان آموزش برابر با **۱۵۶.۷۵** ثانیه بود و مصرف حافظه GPU به **۱۱۶۳.۹۵** مگابایت رسید. دقت نهایی مدل **۵۵.۸۴** درصد بود. اگرچه سایز batch بزرگ می‌تواند دقت مدل را افزایش دهد، اما حافظه زیادی بیشتری دارد.

شکل ۷-۳ وضعیت GPU در حالت آموزش با یک GPU و Batch Size برابر با ۱۲۸

آموزش با دو GPU و Batch Size برابر با ۱۲۸

```
Rank 1, Epoch [1/10], Train Loss: 2.3671, Train Accuracy: 22.24%, Test Loss: 1.8646, Test
Accuracy: 33.50%
Rank 0, Epoch [1/10], Train Loss: 2.4089, Train Accuracy: 22.92%, Test Loss: 1.8620, Test
Accuracy: 32.90%
Rank 1, Epoch [2/10], Train Loss: 1.7266, Train Accuracy: 37.76%, Test Loss: 1.6422, Test
Accuracy: 41.45%
Rank 0, Epoch [2/10], Train Loss: 1.7227, Train Accuracy: 36.20%, Test Loss: 1.6260, Test
Accuracy: 40.85%
Rank 1, Epoch [3/10], Train Loss: 1.5006, Train Accuracy: 45.28%, Test Loss: 1.4568, Test
Accuracy: 47.23%
Rank 0, Epoch [3/10], Train Loss: 1.5097, Train Accuracy: 43.96%, Test Loss: 1.4279, Test
Accuracy: 48.02%
Rank 1, Epoch [4/10], Train Loss: 1.3391, Train Accuracy: 51.44%, Test Loss: 1.3816, Test
Accuracy: 50.50%
Rank 0, Epoch [4/10], Train Loss: 1.3460, Train Accuracy: 50.48%, Test Loss: 1.3549, Test
Accuracy: 49.90%
Rank 1, Epoch [5/10], Train Loss: 1.2136, Train Accuracy: 55.12%, Test Loss: 1.3314, Test
Accuracy: 52.85%
Rank 0, Epoch [5/10], Train Loss: 1.2244, Train Accuracy: 55.28%, Test Loss: 1.3024, Test
Accuracy: 52.95%
Rank 1, Epoch [6/10], Train Loss: 1.1006, Train Accuracy: 59.72%, Test Loss: 1.3092, Test
Accuracy: 54.15%
Rank 0, Epoch [6/10], Train Loss: 1.1041, Train Accuracy: 60.12%, Test Loss: 1.2631, Test
Accuracy: 54.65%
Rank 1, Epoch [7/10], Train Loss: 0.9931, Train Accuracy: 63.64%, Test Loss: 1.3295, Test
Accuracy: 53.30%
Rank 0, Epoch [7/10], Train Loss: 0.9805, Train Accuracy: 65.64%, Test Loss: 1.2808, Test
Accuracy: 54.35%
Rank 1, Epoch [8/10], Train Loss: 0.8670, Train Accuracy: 67.96%, Test Loss: 1.3620, Test
Accuracy: 54.50%
Rank 0, Epoch [8/10], Train Loss: 0.8528, Train Accuracy: 69.84%, Test Loss: 1.3045, Test
Accuracy: 55.48%
Rank 1, Epoch [9/10], Train Loss: 0.7497, Train Accuracy: 72.52%, Test Loss: 1.4334, Test
Accuracy: 53.05%
```

```
Rank 0, Epoch [9/10], Train Loss: 0.7488, Train Accuracy: 73.72%, Test Loss: 1.3742, Test
Accuracy: 53.98%
Rank 1, Epoch [10/10], Train Loss: 0.6659, Train Accuracy: 75.92%, Test Loss: 1.4572, Test
Accuracy: 54.20%
Rank 1, Training Time: 46.34 seconds
Rank 1, CUDA Memory Usage: 1228.33 MB
Rank 1, Test Accuracy: 54.20%
Rank 0, Epoch [10/10], Train Loss: 0.6814, Train Accuracy: 75.52%, Test Loss: 1.3848, Test
Accuracy: 56.45%
Rank 0, Training Time: 46.36 seconds
Rank 0, CUDA Memory Usage: 1228.33 MB
Rank 0, Test Accuracy: 56.45%
Total time: 57.87 seconds
Batch size:128
Backend:nccl
```

در نهایت، با استفاده از دو GPU و سایز batch برابر با **۱۲۸**، زمان کل آموزش به **۵۷.۸۷** ثانیه کاهش یافت. مصرف حافظه هر GPU به **۱۲۲۸.۳۳ مگابایت** رسید و دقت نهایی مدل برای **Rank 0** برابر با **۵۶.۴۵** و برای Rank 1 برابر با **۵۴.۲۰ درصد** بود.

شکل ۸-۳ وضعیت GPU در حالت آموزش با دو GPU و Batch Size برابر با ۱۲۸

۲-۳_ نتیجه‌گیری کلی

Batch اندازه	GPUتعداد	زمان آموزش کل	حافظه مصرفی به مگابایت	دقت نهایی
16	1 GPU	185.15	387.82	50.23
16	2 GPUs	80.60	452.20	49.40
32	1 GPU	179.19	497.94	50.60
32	2 GPUs	63.88	562.32	51.52
64	1 GPU	168.66	719.94	54.05
64	2 GPUs	60.64	784.32	51.00
128	1 GPU	156.75	1163.95	55.84
128	2 GPUs	57.87	1228.33	56.45

جدول ۱ جدول نتایج تغییر سایز Batch و نوع آموزش

در این آزمایش‌ها نتیجه گرفتیم که تغییر سایز batch و تعداد GPU تأثیر زیادی بر زمان آموزش، مصرف حافظه و دقت مدل دارد. با افزایش سایز batch، زمان آموزش کاهش یافت، چون که GPU در هر پردازش داده‌های بیشتری را دریافت کرد اما مصرف حافظه GPU نیز افزایش پیدا کرد. دقت مدل با افزایش سایز batch، به ویژه در سایزهای بزرگ‌تر مثل ۱۲۸، بهبود پیدا کرد، هرچند این افزایش با هزینه بیشتری در منابع سخت‌افزاری همراه بود.

استفاده از دو GPU باعث کاهش قابل‌توجه زمان آموزش شد، اما مصرف حافظه بیشتری نیاز داشت و دقت مدل در برخی موارد کمی کاهش یافت. این مسئله احتمالاً به هماهنگی بین GPUها و نحوه تقسیم داده‌ها مرتبط است. به طور کلی، ما مشاهده کردیم که انتخاب سایز مناسب batch و تعداد GPU به تعادل بین منابع سخت‌افزاری و نیازهای دقت بستگی دارد. این نتایج به ما کمک کرد تا بهتر درک کنیم چگونه پارامترهای مختلف بر عملکرد مدل تأثیر دارند و بهترین تنظیمات را برای نیازهای خاص پروژه انتخاب کنیم.

نمودارهای وضعیت GPU ها نیز نشان می‌دهند که استفاده از چند GPU می‌تواند بهره‌وری را به طور قابل توجهی افزایش دهد، اما بهبود هماهنگی و تنظیم سایز Batch برای جلوگیری از مصرف غیرضروری حافظه و کاهش نوسانات ضروری است.

اندازه Batch	رنک GPU	زمان آموزش کل	حافظه مصرفی به مگابایت	دقت نهایی
16	Rank 0	80.60	452.20	49.40
16	Rank 1	80.60	452.20	49.15
32	Rank 0	63.88	562.32	51.52
32	Rank 1	63.88	562.32	50.17
64	Rank 0	60.64	784.32	51.00
64	Rank 1	60.64	784.32	50.62
128	Rank 0	57.87	1228.33	56.45
128	Rank 1	57.87	1228.33	54.20

جدول ۲ جدول نتایج بر اساس هر GPU در حالت استفاده موازی از دو GPU

در این جدول مشاهده می‌کنیم که در حالت موازی‌سازی دو GPU، هر دو GPU تقریباً زمان آموزش یکسانی را دارند. حافظه مصرفی هر دو GPU نیز برای یک سایز batch مشابه است، اما دقت نهایی در GPUهای مختلف اندکی متفاوت بود که ممکن است به دلیل توزیع متفاوت داده‌ها در حالت موازی است.

۴_پاسخ سوال شماره ۴

نتایج مربوط به اندازه بچ ۳۲ و استفاده از GLOO

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier_mp.py
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Rank 1, Epoch [1/10], Train Loss: 1.9535, Train Accuracy: 29.44%, Test Loss: 1.6881, Test
Accuracy: 36.42%
Rank 0, Epoch [1/10], Train Loss: 1.9639, Train Accuracy: 29.64%, Test Loss: 1.6840, Test
Accuracy: 35.15%
Rank 0, Epoch [2/10], Train Loss: 1.4862, Train Accuracy: 45.32%, Test Loss: 1.4134, Test
Accuracy: 47.55%
Rank 1, Epoch [2/10], Train Loss: 1.4598, Train Accuracy: 45.04%, Test Loss: 1.4157, Test
Accuracy: 48.30%
Rank 1, Epoch [3/10], Train Loss: 1.2307, Train Accuracy: 54.48%, Test Loss: 1.3427, Test
Accuracy: 52.05%
Rank 0, Epoch [3/10], Train Loss: 1.2606, Train Accuracy: 53.52%, Test Loss: 1.3205, Test
Accuracy: 51.40%
Rank 1, Epoch [4/10], Train Loss: 1.0255, Train Accuracy: 62.32%, Test Loss: 1.3580, Test
Accuracy: 53.38%
Rank 0, Epoch [4/10], Train Loss: 1.0431, Train Accuracy: 61.96%, Test Loss: 1.3342, Test
Accuracy: 53.17%
Rank 1, Epoch [5/10], Train Loss: 0.8285, Train Accuracy: 69.08%, Test Loss: 1.4234, Test
Accuracy: 52.95%
Rank 0, Epoch [5/10], Train Loss: 0.8551, Train Accuracy: 69.12%, Test Loss: 1.3970, Test
Accuracy: 53.70%
Rank 1, Epoch [6/10], Train Loss: 0.6601, Train Accuracy: 76.68%, Test Loss: 1.6011, Test
Accuracy: 51.55%
Rank 0, Epoch [6/10], Train Loss: 0.7024, Train Accuracy: 73.76%, Test Loss: 1.5636, Test
Accuracy: 52.95%
Rank 0, Epoch [7/10], Train Loss: 0.5695, Train Accuracy: 79.36%, Test Loss: 1.9020, Test
Accuracy: 50.00%
Rank 1, Epoch [7/10], Train Loss: 0.5435, Train Accuracy: 79.20%, Test Loss: 1.9292, Test
Accuracy: 50.30%
Rank 1, Epoch [8/10], Train Loss: 0.4676, Train Accuracy: 82.80%, Test Loss: 2.3504, Test
Accuracy: 45.58%
Rank 0, Epoch [8/10], Train Loss: 0.4629, Train Accuracy: 83.64%, Test Loss: 2.2849, Test
Accuracy: 45.98%
Rank 1, Epoch [9/10], Train Loss: 0.3661, Train Accuracy: 86.84%, Test Loss: 2.3183, Test
Accuracy: 47.77%
Rank 0, Epoch [9/10], Train Loss: 0.3697, Train Accuracy: 86.80%, Test Loss: 2.2480, Test
Accuracy: 48.25%
Rank 1, Epoch [10/10], Train Loss: 0.2836, Train Accuracy: 90.16%, Test Loss: 2.4967, Test
Accuracy: 48.17%
Rank 1, Training Time: 76.35 seconds
Rank 1, CUDA Memory Usage: 562.32 MB
Rank 1, Test Accuracy: 48.17%
Rank 0, Epoch [10/10], Train Loss: 0.3012, Train Accuracy: 89.20%, Test Loss: 2.3172, Test
Accuracy: 50.48%
Rank 0, Training Time: 76.39 seconds
Rank 0, CUDA Memory Usage: 562.32 MB
Rank 0, Test Accuracy: 50.48%
Total time: 87.32 seconds
Batch size:32
Backend:gloo
```

شکل ۱-۴ وضعیت GPU در حالت آموزش با دو GPU و Batch Size برابر با ۳۲ و استفاده از GLOO

نتایج مربوط به اندازه بچ ۱۲۸ و استفاده از GLOO

```
(pytorch) khorramfar@Sabalan1:~/CA2$ python Classifier_mp.py
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Using device: cuda
Rank 1, Epoch [1/10], Train Loss: 2.3185, Train Accuracy: 19.84%, Test Loss: 1.9204, Test
Accuracy: 32.17%
Rank 0, Epoch [1/10], Train Loss: 2.3544, Train Accuracy: 20.16%, Test Loss: 1.9070, Test
Accuracy: 33.65%
Rank 1, Epoch [2/10], Train Loss: 1.7754, Train Accuracy: 36.60%, Test Loss: 1.6879, Test
Accuracy: 39.75%
Rank 0, Epoch [2/10], Train Loss: 1.7793, Train Accuracy: 34.40%, Test Loss: 1.6704, Test
Accuracy: 39.20%
Rank 1, Epoch [3/10], Train Loss: 1.5741, Train Accuracy: 42.96%, Test Loss: 1.5128, Test
Accuracy: 45.85%
Rank 0, Epoch [3/10], Train Loss: 1.5561, Train Accuracy: 42.48%, Test Loss: 1.4961, Test
Accuracy: 45.73%
Rank 1, Epoch [4/10], Train Loss: 1.3704, Train Accuracy: 50.60%, Test Loss: 1.4396, Test
Accuracy: 48.58%
Rank 0, Epoch [4/10], Train Loss: 1.3940, Train Accuracy: 49.52%, Test Loss: 1.4266, Test
Accuracy: 47.95%
Rank 1, Epoch [5/10], Train Loss: 1.2463, Train Accuracy: 54.24%, Test Loss: 1.3669, Test
Accuracy: 51.70%
Rank 0, Epoch [5/10], Train Loss: 1.2712, Train Accuracy: 53.68%, Test Loss: 1.3461, Test
Accuracy: 50.55%
Rank 1, Epoch [6/10], Train Loss: 1.1162, Train Accuracy: 60.44%, Test Loss: 1.3409, Test
Accuracy: 52.83%
Rank 0, Epoch [6/10], Train Loss: 1.1365, Train Accuracy: 59.04%, Test Loss: 1.3236, Test
Accuracy: 52.10%
Rank 1, Epoch [7/10], Train Loss: 1.0147, Train Accuracy: 64.36%, Test Loss: 1.3807, Test
Accuracy: 52.02%
Rank 0, Epoch [7/10], Train Loss: 1.0413, Train Accuracy: 61.36%, Test Loss: 1.3696, Test
Accuracy: 51.20%
Rank 1, Epoch [8/10], Train Loss: 0.9270, Train Accuracy: 67.20%, Test Loss: 1.3462, Test
Accuracy: 54.12%
Rank 0, Epoch [8/10], Train Loss: 0.9245, Train Accuracy: 66.80%, Test Loss: 1.3260, Test
Accuracy: 54.50%
Rank 1, Epoch [9/10], Train Loss: 0.8049, Train Accuracy: 71.60%, Test Loss: 1.4123, Test
Accuracy: 53.48%
Rank 0, Epoch [9/10], Train Loss: 0.8007, Train Accuracy: 71.16%, Test Loss: 1.3884, Test
Accuracy: 54.12%
Rank 1, Epoch [10/10], Train Loss: 0.7278, Train Accuracy: 74.20%, Test Loss: 1.5572, Test
Accuracy: 51.12%
Rank 1, Training Time: 50.75 seconds
Rank 1, CUDA Memory Usage: 1228.33 MB
Rank 1, Test Accuracy: 51.12%
```

```
Rank 0, Epoch [10/10], Train Loss: 0.7378, Train Accuracy: 73.20%, Test Loss: 1.5272, Test
Accuracy: 51.35%
Rank 0, Training Time: 50.96 seconds
Rank 0, CUDA Memory Usage: 1228.33 MB
Rank 0, Test Accuracy: 51.35%
Total time: 61.56 seconds
Batch size:128
Backend:gloo
```

شکل ۲-۴ وضعیت GPU در حالت آموزش با دو GPU و Batch Size برابر با ۱۲۸ و استفاده از GLOO

در این بخش، دو Backend gloo و nccl را برای دو سایز batch مختلف ۳۲ و ۱۲۸ مقایسه کردیم.

موارد مربوط به NCCL در قسمت قبل آورده شده و در اینجا تکرار نمی‌کنیم. نتایج نشان داد که استفاده از nccl زمان آموزش را نسبت به gloo کاهش داد.

به عنوان مثال، برای batch size برابر با ۳۲، زمان آموزش با nccl به **۶۳.۸۸** ثانیه رسید، در حالی که gloo به **۸۷.۳۲** ثانیه نیاز داشت. این کاهش زمان، به دلیل بهینه‌تر بودن nccl در مدیریت ارتباطات بین GPUها است و توسعه توسط انویدیا است.

برای batch size برابر با **۱۲۸** نیز نتایج مشابهی داریم، به طوری که nccl زمان آموزش را به **۵۷.۸۷** ثانیه کاهش داد، در حالی که gloo زمان بیشتری برابر با **۶۱.۵۶** ثانیه مصرف کرد.

دقت نهایی Rank 1	دقت نهایی Rank 0	حافظه مصرفی	زمان آموزش	Backend	اندازه Batch
48.17	50.48	562.32	87.32	gloo	**32**
50.17	51.52	562.32	63.88	nccl	**32**
51.12	51.35	1228.33	61.56	gloo	**128**
54.20	56.45	1228.33	57.87	nccl	**128**

جدول ۳ جدول نتایج مقایسه Backendهای gloo و nccl در PyTorch DDP

از نظر مصرف حافظه GPU، در هر دو سایز batch و برای هر دو Backend، مصرف حافظه GPU برای هر GPU مشابه بود. در مجموع، ما مشاهده کردیم که استفاده از nccl در مقایسه با gloo، زمان آموزش را کاهش داده، دقت مدل بهتر شده و از منابع GPU به طور بهینه‌تری استفاده کرده است.

۵_استفاده از چت‌بات‌ها و ابزارها در تمرین

۵-۱_ پرامپت‌های اصلی استفاده شده در چت‌بات ChatGPT

کمک برای طراحی شبکه عصبی ساده

write a simple neural network in PyTorch for image classification with three convolutional layers.

البته تغییرات زیادی انجام شد و همچنین تعداد نورون‌ها و دراپ‌اوت تنظیم شد. برای حل مشکلات و ارورهای حلقه آموزش نیز از چت‌بات استفاده شد.

حل مشکل کد کودا

کل کودایی که نوشته بودم تصاویر را سیاه و سفید می‌کرد ولی ایندکس‌دهی اشتباه بود و تصویر خروجی ۳ در ۳ بود و ۹ تصویر کوچک داشتیم. و کد اولیه Compile نمی‌شد.

This CUDA code for grayscale image conversion is not working. Fix it.

ساخت جداول نهایی

I have GPU training results with batch sizes and accuracys

make a table from following results:

I want another table based on GPU rank.

پیاده‌سازی دیتالودر در حالت توزیع‌شده

How can I set up and choose best parameters in distributed data loaders in PyTorch to train on multiple GPUs

ادامه پرامپت‌ها بیشتر مربوط به تلاش بیشتر برای نشان دادن مشکل به چت‌بات بود.

استفاده از پاکنویس

برای حل مشکلات نگارشی و رعایت نیم‌فاصله از افزونه پاکنویس در Microsoft Word استفاده شد.