1. Deney

FSDP DDP

Bu deneyde FSDP ve DDP paraleleştirme yöntemleri kullanılarak Cifar100 veriseti, ResNet50 modeli ile 4 gpu kullanılarak eğitilmiştir. Eğitim akya-cuda partition içerisinde yapılmıştır ve sonuçlar fsdp_ddp adlı dosyada belirtilmektedir. Bu hiperparametreler tensorboard hparams kısmından indirilmiştir.

512 batch sizeda out of memory hatası alındığı için 512 batch size ile ilgili herhangi bir bilgi bulunmamaktadır.

1 GPU'da eğitim yaklaşık 7 dakika ve her bir epoch yaklaşık 40 saniye sürmüştür.

.out dosyasındaki bir eğitimin 4 GPU'daki training_took bilgileri:

Training took 0:08:01.146711

Running test evaluation...

Training took 0:07:54.791011

Running test evaluation...

Training took 0:08:03.818484

Running test evaluation...

[[Epoch 9: 100%|██████████| 313/313 [00:41<00:00, 7.58it/s, v_num=0, train_loss_step=0.0797, train_acc_step=1.000, val_loss=0.03>

Training took 0:07:55.802053

Running test evaluation...

^MTesting: | | 0/? [00:00<?, ?it/s]^MTesting: 0%| | 0/313 [00:00<?, ?it/s]^MTesting DataLoader 0: 0%| | 0/313 [00:00<?, ?it/s]^MTesting DataLoader 0: 0%| | 1/313 [00>

Test metric	DataLoader 0
test_acc	0.8518000245094299
test_loss	0.5697863698005676

Test results: [{'test_loss': 0.5697863698005676, 'test_acc': 0.8518000245094299}]

DataLoader 0: 99%|▉| 310/313 [00:40<00:00, 7.71it/s]^MTesting DataLoader 0: 99%|▉| 311/313 [00:40<00:00, 7.71it/s]^MTesting DataLoader 0: 100%|█| 312/313 [00:40<00:00, 7.71it/s]^>

Test metric	DataLoader 0
test_acc	0.8518000245094299
test_loss	0.5697863698005676

DataLoader 0: 99%|▉| 310/313 [00:40<00:00, 7.69it/s]^MTesting DataLoader 0: 99%|▉| 311/313 [00:40<00:00, 7.69it/s]^MTesting DataLoader 0: 100%|█| 312/313 [00:40<00:00, 7.69it/s]^>

Test metric	DataLoader 0
test_acc	0.8518000245094299
test_loss	0.5697863698005676

DataLoader 0: 99%|▉| 310/313 [00:39<00:00, 7.78it/s]^MTesting DataLoader 0: 99%|▉| 311/313 [00:39<00:00, 7.78it/s]^MTesting DataLoader 0: 100%|█| 312/313 [00:40<00:00, 7.78it/s]^>

Test metric	DataLoader 0

```
|        test_acc         |    0.8518000245094299     |
|        test_loss        |    0.5697863698005676     |
└─────────────────────────┴───────────────────────────┘
```

Test results: [{'test_loss': 0.5697863698005676, 'test_acc': 0.8518000245094299}]

Test results: [{'test_loss': 0.5697863698005676, 'test_acc': 0.8518000245094299}]

Test results: [{'test_loss': 0.5697863698005676, 'test_acc': 0.8518000245094299}]

--- EXPERIMENT FINISHED ---

GPU bellek bilgisi nvidia-smi komutu ile gözlemlenmiştir.

```
Wed Jun 18 12:39:41 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 565.57.01              Driver Version: 565.57.01      CUDA Version: 12.7     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla V100-SXM2-16GB           On  |   00000000:61:00.0 Off |                    0 |
| N/A   63C    P0            288W /  300W |    4142MiB /  16384MiB |     82%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  Tesla V100-SXM2-16GB           On  |   00000000:62:00.0 Off |                    0 |
| N/A   58C    P0            248W /  300W |    4142MiB /  16384MiB |     88%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   2  Tesla V100-SXM2-16GB           On  |   00000000:89:00.0 Off |                    0 |
| N/A   59C    P0            226W /  300W |    4142MiB /  16384MiB |     86%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   3  Tesla V100-SXM2-16GB           On  |   00000000:8A:00.0 Off |                    0 |
| N/A   62C    P0            201W /  300W |    4142MiB /  16384MiB |     88%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A   3819710      C   ...e3-2024/envs/gpu-2024.0/bin/python3       4138MiB |
|    1   N/A  N/A   3819711      C   ...e3-2024/envs/gpu-2024.0/bin/python3       4138MiB |
|    2   N/A  N/A   3819712      C   ...e3-2024/envs/gpu-2024.0/bin/python3       4138MiB |
```

1. Deney sonuçlar:

Genel olarak bakıldığında, en yüksek doğruluklar SGD optimizer'ı ile elde edilmiştir. Özellikle SGD ile CosineAnnealing scheduler'ı bir arada kullanıldığında, hem FSDP hem de DDP stratejilerinde doğruluk

oranları %85'e kadar çıkmıştır. Bu yapılandırmaların kayıp değerleri de oldukça düşüktür. SGD, özellikle batch size 32 ile kullanıldığında modelin genelleme başarısı dikkat çekici şekilde artmıştır. Öte yandan, öğrenme oranı 0.01 olan konfigürasyonlar genellikle daha başarılı sonuçlar vermiştir, ancak bu sadece SGD için geçerli değildir. Adam optimizer'ı için daha düşük bir öğrenme oranı olan 0.001 ile yapılan eğitimler daha stabil ve başarılı olmuştur. Adam ile birlikte Cosine veya StepLR scheduler'larının kullanılması, doğruluk oranını %82–83 bandına taşımıştır. Bununla birlikte, öğrenme oranı 0.01 olan Adam konfigürasyonları modelin aşırı öğrenmesine ve doğruluk oranının düşmesine neden olmuştur.

RMSprop optimizer'ı ise bu deneylerde en düşük başarıyı göstermiştir. Çoğu kombinasyonda doğruluk oranı %60'ın altında kalmış, bazı durumlarda %40'lara kadar gerilemiştir. Özellikle batch size 128 ve öğrenme oranı 0.01 olan durumlarda hem FSDP hem de DDP ile çok yüksek kayıplar ve düşük doğruluklar gözlemlenmiştir. Bu durum, RMSprop'un ResNet50 ve CIFAR-100 gibi karmaşık bir yapı ve veri seti üzerinde yeterli performansı sağlayamadığını göstermektedir.

Paralelleştirme stratejileri arasında belirgin bir fark olmamakla birlikte, bazı durumlarda FSDP'nin az da olsa daha yüksek doğruluk verdiği gözlemlenmiştir. FSDP'nin özellikle 32 batch size ile çalışırken daha etkili olduğu; buna karşın DDP'nin 128 batch size kullanıldığında daha stabil sonuçlar sunduğu görülmüştür. Ayrıca DDP ile SGD kullanılarak yapılan deneylerde, doğruluk oranı %85'e kadar ulaşmıştır ki bu FSDP'ye çok yakındır. Bu da her iki stratejinin farklı senaryolarda avantaj sunduğunu ortaya koymaktadır.

Scheduler'lar açısından değerlendirildiğinde, CosineAnnealingLR neredeyse tüm deneylerde en yüksek doğrulukları sağlamıştır. Özellikle SGD ve Adam ile kullanıldığında modelin doğruluk oranını önemli ölçüde artırmıştır. StepLR de oldukça dengeli sonuçlar vermiş; ancak sabit learning rate ile yapılan eğitimler genellikle daha düşük doğruluklara sahip olmuştur. Bu durum, zamanla azalan bir öğrenme oranının modelin genel başarısını artırdığını göstermektedir.

Son olarak batch size parametresinin etkisi incelendiğinde, 32 batch size kullanılan deneylerin genellikle daha başarılı olduğu görülmüştür. 128 batch size ile yapılan eğitimlerde, özellikle Adam ve RMSprop kullanıldığında doğruluk oranları düşmüş; SGD ile ise etkilenme daha az olmuştur. Bu durum, 32'lik batch'lerin özellikle daha düşük öğrenme oranı ile birlikte, modelin daha yavaş ve istikrarlı öğrenmesini sağladığını göstermektedir.Ayrıca 512 batch size'da out of memory hatası görüldüğü için 512 batch size ile ilgili bir sonuç bulunmamaktadır.

Tüm bu sonuçlar bir arada değerlendirildiğinde, SGD optimizer'ı ve CosineAnnealingLR scheduler'ı ile 32 batch size kullanımı, hem FSDP hem de DDP altında en başarılı strateji olarak öne çıkmaktadır. Ancak sistem kaynaklarının durumuna göre DDP büyük batch'lerle daha uygun olabilirken, FSDP küçük batch'lerde daha fazla avantaj sunmaktadır. RMSprop ise bu bağlamda önerilmeyen bir seçenek olarak değerlendirilebilir.

2. Deney

FSDP1 ve FSDP2

1.dendeyde kullanılan CIFAR100 veriseti ve ResNet50 modeli kullanılmıştır. FSDP1 ve FSDP2 paralelleştirme yöntemleri için barbun-cuda partitionu kullanılmıştır. Eğitim 2 GPU kullanılarak gerçekleştirilmiştir. Metrikler fsdp1_fsdp2 adlı dosyada gösterilmiştir.

Her bir GPU'daki eğitim yaklaşık 23 dakika sürmüştür.

.out dosyasındaki süre çıktısı:
Training took 0:24:53.252506

Running test evaluation...

[[Epoch 9: 100%|██████████| 625/625 [02:23<00:00, 4.37it/s, v_num=132,
train_loss_st>

Training took 0:24:57.817380

Running test evaluation...

^MTesting: | | 0/? [00:00<?, ?it/s]^MTesting: 0%| | 0/313 [00:00<?, ?it/s]^MTesting
DataLoader 0:   0%| | 0/313 [00:00<?, ?it/s]^MTestin>

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃       Test metric         ┃       DataLoader 0        ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
│        test_acc           │    0.8500000238418579     │
│        test_loss          │    0.6088020205497742     │
└───────────────────────────┴───────────────────────────┘
```

Test results: [{'test_loss': 0.6088020205497742, 'test_acc': 0.8500000238418579}]

^MTesting DataLoader 0: 96%|██████████| 299/313 [00:30<00:01, 9.75it/s]^MTesting
DataLoader 0: 96%|██████████| 300/313 [00:30<00:01, 9.77it/s]^MTesting
DataLoade>

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━┓
┃       Test metric         ┃       DataLoader 0        ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━┩
```

| test_acc | 0.8500000238418579 |

| test_loss | 0.6088020205497742 |

└──────────────────────────┴──────────────────────────┘

Test results: [{'test_loss': 0.6088020205497742, 'test_acc': 0.8500000238418579}]

NVDIA-SMI GPU Bilgisi

```
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 565.57.01              Driver Version: 565.57.01      CUDA Version: 12.7     |
|-----------------------------------------+------------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id          Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |           Memory-Usage | GPU-Util  Compute M. |
|                                         |                        |               MIG M. |
|=========================================+========================+======================|
|   0  Tesla P100-PCIE-16GB           On  |   00000000:3B:00.0 Off |                    0 |
| N/A   57C    P0            184W /  250W |    3741MiB /  16384MiB |     77%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+
|   1  Tesla P100-PCIE-16GB           On  |   00000000:D8:00.0 Off |                    0 |
| N/A   61C    P0            173W /  250W |    3705MiB /  16384MiB |     90%      Default |
|                                         |                        |                  N/A |
+-----------------------------------------+------------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|    0   N/A  N/A    126748      C   ...e3-2024/envs/gpu-2024.0/bin/python3       3738MiB |
|    1   N/A  N/A    126749      C   ...e3-2024/envs/gpu-2024.0/bin/python3       3702MiB |
+-----------------------------------------------------------------------------------------+
```

2. Deney Sonuçlar

Genel olarak en istikrarlı ve yüksek doğruluk sağlayan yapılandırmalar SGD optimizer'ı ve CosineAnnealingLR veya StepLR scheduler kombinasyonları olmuştur. Bu yapılandırmalar özellikle 32 batch size ile kullanıldığında, hem fsdp1 hem de fsdp2 altında %85'e varan doğruluk sonuçları sağlamıştır. Cosine scheduler, SGD ile birlikte en iyi sonucu vererek %85.28 doğruluk elde edilmesini sağlamıştır. Bu sonuçlar, küçük batch size ve zamanla azalan öğrenme oranının SGD ile oldukça uyumlu çalıştığını göstermektedir.

Buna karşılık, Adam optimizer'ı için başarı oranları daha dalgalı olmuştur. 0.001 öğrenme oranı kullanıldığında doğruluk %80'in üzerine çıkabilirken, 0.01 değerinde doğruluklar önemli ölçüde düşmüştür. Özellikle sabit öğrenme oranı ile birlikte kullanıldığında Adam optimizer'ı ile elde edilen doğruluk %50'nin altına gerilemiştir. Bu da, Adam'ın yüksek öğrenme oranlarında aşırı sapma gösterdiğini ve scheduler ile desteklenmesi gerektiğini ortaya koymaktadır.

RMSprop optimizer'ı tüm deneyler içinde en düşük doğruluk değerlerine sahip olmuştur. Öğrenme oranı düşürülse dahi (0.001), doğruluk genellikle %60-70 aralığında kalmış ve özellikle 0.01 oranında doğruluk

%50'nin altına inmiştir. Bu sonuçlar, RMSprop'un bu veri seti ve model yapısı için uygun bir tercih olmadığını göstermektedir.

Paralelleştirme stratejileri açısından fsdp1 ve fsdp2 sonuçları birbirine oldukça yakın seyretmiştir. Ancak bazı durumlarda fsdp2, örneğin SGD + Cosine kombinasyonunda daha yüksek doğruluk (%85.13) sağlarken, fsdp1 biraz daha düşük (%85.28) sonuçlar vermiştir. Ancak bu fark oldukça sınırlıdır ve genel olarak iki strateji arasında ciddi bir performans farkı gözlemlenmemiştir.

Sonuç olarak, 2 GPU ile fsdp1 ve fsdp2 stratejilerinin kullanıldığı deneylerde en başarılı kombinasyonlar SGD + CosineAnnealingLR ve SGD + StepLR yapılandırmaları olmuştur. Öğrenme oranının 0.001 yerine 0.01 olarak seçildiği durumlar genellikle daha başarılı sonuçlar getirmiştir. Adam ve RMSprop gibi alternatif optimizer'lar ise sadece belirli düşük öğrenme oranı ve scheduler kombinasyonlarında rekabetçi olabilmiştir. Bu da, dağıtık eğitimde hiperparametre seçimlerinin başarıyı doğrudan etkilediğini ve özellikle SGD gibi klasik optimizasyon yöntemlerinin iyi ayarlandığında hâlâ güçlü performans sunduğunu göstermektedir.