Profiling Training Pipeline

nanoTabPFN

- NanoTabPFN experiments
- Hardware: NVIDIA A100-SXM4-80GB

Baseline: train.py

```
with h5py.File(filename, "r") as f:
    for _ in range(num_steps):
        x = torch.from_numpy(f["X"][ptr:end])
        y = torch.from_numpy(f["y"][ptr:end])
        yield dict(
            x=x.to(device),
            y=y.to(device),
        )
```

Profiling

Profiling Setup

```
from torch.profiler import profile, ProfilerActivity

with profile(
    activities=[ProfilerActivity.CPU, ProfilerActivity.CUDA],
    record_shapes=True,
    profile_memory=True,
) as prof:
    model, history = train(model, prior, lr=4e-3)

print(prof.key_averages().table(sort_by="cuda_time_total"))
```

Baseline Profile (100 steps, batch=6, 5000 rows)

Metric	Value
Total time	68.75s
Steps/sec	1.5
ms/step	687.51
Self CPU time total	57.415s
Self CUDA time total	50.156s

Baseline: Top CPU Operations

Operation	Self CPU	% CPU
cudaMemcpyAsync	44,084ms	76.78%
cudaMalloc	7,081ms	12.33%
cudaLaunchKernel	645ms	1.12%
aten::bmm	180ms	0.31%

Baseline: Top CUDA Operations

Operation	Self CUDA	% CUDA
aten::bmm	32,509ms	64.82%
ampere_sgemm_128x128_nn	13,783ms	27.48%
ampere_sgemm_128x128_nt	13,706ms	27.33%
aten::_softmax_backward_data	3,880ms	7.73%
aten::mul	3,397ms	6.77%

Baseline: Memory Transfers

Operation	CPU Time	Calls
aten::to	44,240ms	7,362
aten::_to_copy	44,231ms	462
aten::copy_	44,352ms	12,824
cudaMemcpyAsync	44,084ms	1,024

Optimizations

train_optimized.py

1. Pinned memory
2. Non-blocking transfers
3. CUDA streams for overlap
4. Double buffering (prefetch to VRAM)
5. GPU Direct Storage (kvikio)

Pinned Memory + Non-blocking

```
# Before
x = torch.from_numpy(x_np).to(device)

# After
x = torch.from_numpy(x_np).pin_memory().to(device, non_blocking=True)
```

CUDA Streams + Double Buffering

```
self.transfer_stream = torch.cuda.Stream()

vram_buffer = [self._load_to_vram(f) for _ in range(prefetch)]

for step in range(num_steps):
    batch = vram_buffer.pop(0)

    with torch.cuda.stream(self.transfer_stream):
        next_batch = self._load_to_vram(f)
    vram_buffer.append(next_batch)

    torch.cuda.current_stream().wait_stream(self.transfer_stream)
    yield batch
```

GPU Direct Storage (kvikio)

```
import kvikio
import cupy as cp

x_gpu = cp.empty((batch_size, max_seq, num_features), dtype=cp.float32)
x_gpu[:] = cp.asarray(f["X"][ptr:end])
x = torch.as_tensor(x_gpu, device=self.device)
```

Results

Optimized Profile (100 steps, batch=6, 5000 rows, GDS)

Metric	Baseline	Optimized
Total time	68.75s	45.30s
Steps/sec	1.5	2.2
ms/step	687.51	453.00
Self CPU time total	57.415s	49.992s
Self CUDA time total	50.156s	30.691s

Optimized: Top CPU Operations

Operation	Self CPU	% CPU
Command Buffer Full	23,450ms	46.91%
cudaLaunchKernel	10,733ms	21.47%
cudaMalloc	5,607ms	11.22%
cuLaunchKernel	3,249ms	6.50%

Optimized: Top CUDA Operations

Operation	Self CUDA	% CUDA
aten::bmm	13,850ms	45.13%
aten::_softmax_backward_data	4,160ms	13.55%
Command Buffer Full	3,975ms	12.95%
cutlass_80_tensorop_s1688gemm	3,723ms	12.13%
aten::mul	3,650ms	11.89%

Memory Transfer Comparison

Operation	Baseline	Optimized
aten::to	44,240ms	109ms
cudaMemcpyAsync	44,084ms	268ms
aten::copy_	44,352ms	4,425ms

CUDA Memory Summary

Metric	Baseline	Optimized
Peak Usage	72,327 MiB	72,326 MiB
Total Allocated	11,394 GiB	12,864 GiB
cudaMalloc retries	5	5