Benchmarking Notes

1 benchmarking_script

a

Written in `benchmark.py`.

b

Commands:

```
# small
python benchmark.py --d-model 768 --d-ff 3072 --num-layers 12 --num-heads 12

# medium
python benchmark.py --d-model 1024 --d-ff 4096 --num-layers 24 --num-heads 16

# large
python benchmark.py --d-model 1280 --d-ff 5120 --num-layers 36 --num-heads 20

# xl
python benchmark.py --d-model 1600 --d-ff 6400 --num-layers 48 --num-heads 25

# 2.7B
python benchmark.py --d-model 2560 --d-ff 10240 --num-layers 32 --num-heads 32
```

Results:

Table 1: Benchmark results for small and medium models.

Model	Pass	Warmup	Min (s)	Max (s)	Avg (s)	Std (s)
Small	Forward	0	0.030826	0.437965	0.072297	0.121894
Small	Backward	0	0.065635	0.470007	0.106601	0.121139
Small	Forward	1	0.030985	0.032706	0.031458	0.000598
Small	Backward	1	0.066029	0.072107	0.067038	0.001763
Medium	Forward	0	0.095544	0.489508	0.136340	0.117727
Medium	Backward	0	0.199060	0.608251	0.241945	0.122108
Medium	Forward	1	0.095837	0.099985	0.097735	0.001412
Medium	Backward	1	0.199726	0.203641	0.202035	0.001391

```
Cannot do rest due to memory limitations (8GB).
```

c

Minor increase in measured time with 0 warmup steps. This happens because some optimizations are done based on the first pass, so warming up lets the correct cache/shapes be known in advance for the next passes.

2 nsys_profile

a

Total time is roughly 40 ms for all context sizes (did small model only due to memory constraints) but our measured time in Python keeps increasing due to device sync overhead.

b

`ampere_sgemm_128x64_nn` takes up the most time in both forward and backward passes. It is called 52 times in the forward pass.

c

Forward:

```
Time    Total Time    Instances    Avg         Med         Min         Max         StdDev    Name
5.6%    2.167 ms      94           23.053 \textmus   22.688 \textmus   21.920 \textmus   30.304 \
    textmus   1.260 \textmus  void at::native::elementwise_kernel<(int)128, (int)2, void at::native
    ::gpu_kernel_impl_nocast<...>>
```

Backward:

```
Time     Total Time    Instances    Avg         Med         Min         Max         StdDev    Name
13.7%    7.872 ms      11           715.670 \textmus  597.540 \textmus  592.869 \textmus  1.914 ms
       397.501 \textmus void cutlass::Kernel2<cutlass_80_simt_sgemm_128x64_8x5_nt_align1>(T1::
    Params)
10.5%    6.012 ms      68           88.410 \textmus   36.176 \textmus   1.152 \textmus    274.627 \
    textmus  85.005 \textmus  void at::native::vectorized_elementwise_kernel<(int)4, ...>
```

d

Optimizer takes up a huge chunk of time but overall, kernel contribution remains the same.

e

Matrix multiplication takes approximately 762 μs while computing softmax takes approximately 800 μs. The matrix multiplication has much more FLOPs than softmax.

3 mixed_precision_accumulation

```
ans.
tensor(10.0001)
tensor(9.9531, dtype=torch.float16)
tensor(10.0021)
tensor(10.0021)
```

Accumulating in FP32 lets us retain a more accurate result when adding floats of lower precision, regardless of whether we upscale the lower precision float or not.

4 benchmarking_mixed_precision

a

- Model parameters: FP16
- Output of first feedforward layer: FP32
- Output of layer norm: FP16
- Predicted logits: FP16
- Loss: FP32
- Gradients: FP16

b

(Empty / not provided.)