Alejandro Lucena
2/21/2026

Experiment 2: MLP Report

Introduction and Methodology

This report evaluates the performance of two Multi-Layer Perceptron (MLP) inference implementations: a “baseline” version utilizing separate kernels for GEMM, bias addition, and activation, and an “activation_fused” version that combines bias and activation into a single kernel. The study analyzes GFLOP/s throughput across varying layer widths, batch sizes, and activation functions (ReLU and GELU) to quantify the impact of kernel fusion.

First, the platform details. All tests were conducted on an NVIDIA V100 GPU using CUDA version 12.4.1. Because GPU performance is highly dependent on hardware specifications, the GFLOP/s subsequently reported are specific to this GPU environment.

Next, a brief implementation overview of both MLP inference schedules. The baseline schedule executes *cublasSgemm*, a custom *bias_add_kernel*, and a custom *activation_kernel*. The fused schedule, on the other hand, executes *cublasSgemm* followed by a single *fused_bias_activation_kernel*, eliminating one round-trip to global memory. In terms of GEMM configuration, weights are stored in row-major format [Out, In]. To accommodate cuBLAS’s column-major requirement, the GEMM is executed as Output^T = W* X^T, where W is transposed during the call via CUBLAS_OP_T (param of *cublasSgemm*). Furthermore, the command-line arguments permitted specifying the activation function the caller could use, thereby allowing each MLP inference schedule to run on either of the two available per-layer activation kernels: ReLU or GELU.

Finally, the performance metric. To quantify and compare throughput, this report utilizes GFLOPS as its sole performance metric. For a standard GEMM operation, the operation count is 2*M*N*K. Performance is thus calculated as: GFLOP/s = 2*M*N*K / Time (s) * 10^9.

Performance Analysis: Throughput across Layer Widths

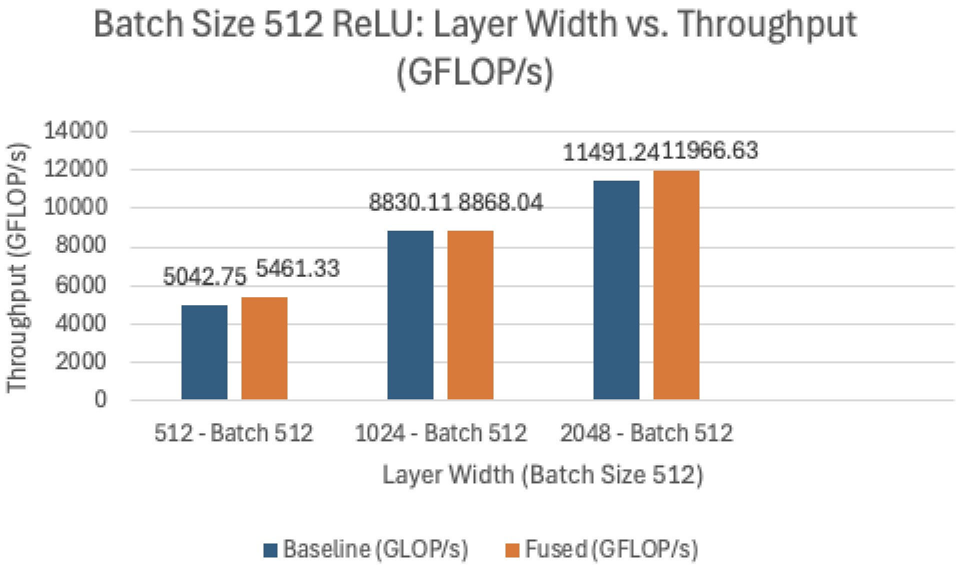


Figure 1

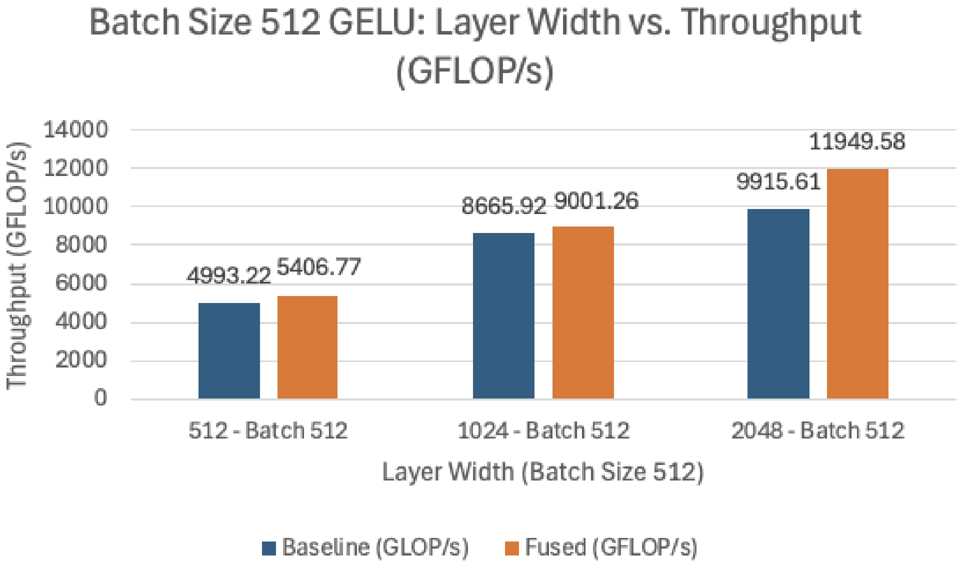


Figure 2

Figures 1 and 2 compare the throughput of “baseline” and fused schedules at a fixed batch size of 512 across three layer configurations: 512 x 512 x 512, 1024 x 2048 x 1024, and 2048 x 2048 x 2048. Figure 1 corresponds to the ReLU activation function, whereas Figure 2 corresponds to the GELU activation function.

As illustrated in both figures, throughput increases significantly with layer width. The activation-fused schedule with a 2048 x 2048 x 2048 layer width achieved the peak performance of 11966.63 GFLOP/s for ReLU and 11949.58 GFLOP/s for GELU. Not surprisingly, the ReLU-activation-fused version achieved the highest throughput, given its lower implementation complexity.

Similarly, the speedup of the fused kernel over the baseline scales with layer width, with narrower layers generally achieving the largest gains. At a layer width of 512 x 512 x 512, the fused ReLU kernel demonstrated an 8.3% speedup over the baseline. At this scale, the GEMM duration is relatively short, making the overhead of launching separate kernels for bias and activation a significant portion of the total execution time. As the layer configuration increases to 1024 x 2048 x 1024, GEMM dominates the workload, resulting in a relative speedup of only 0.43%. The GELU implementation showed even more dramatic speedup gains (up to 20.5% at 2048 width), likely due to the higher arithmetic intensity of the GELU calculation being better hidden within the fused memory access.

Performance Analysis: Throughput across Batch Sizes

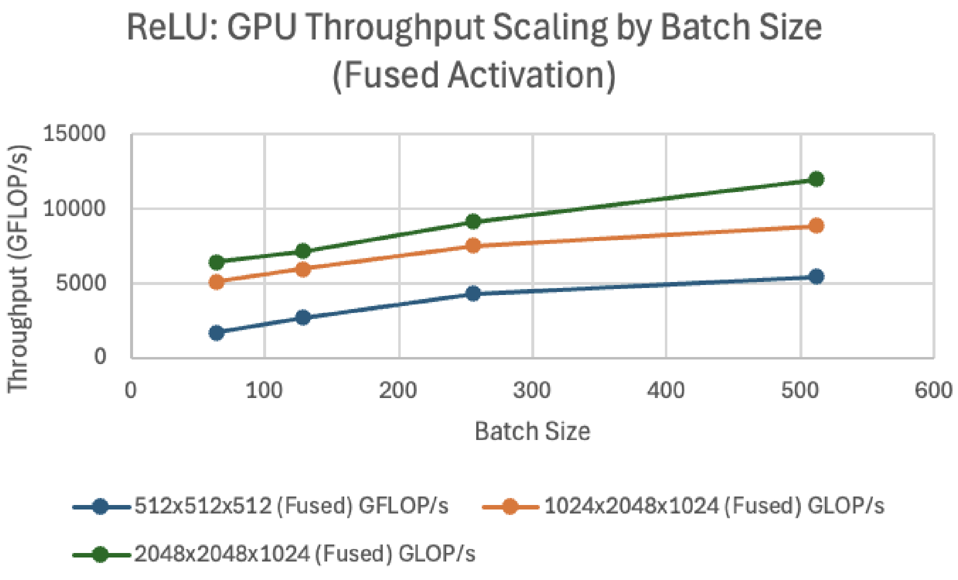


Figure 3

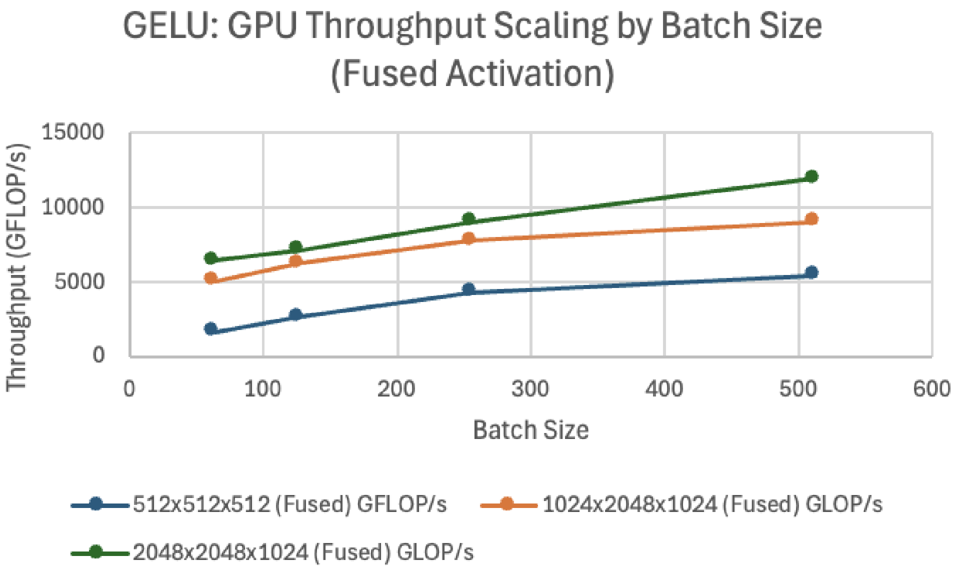


Figure 4

Figures 3 and 4 illustrate how throughput scales with batch sizes of 64, 128, 256, and 512 for the activation-fused schedule. Figure 3 corresponds to the ReLU activation function, whereas Figure 4 corresponds to the GELU activation function.

As illustrated by both figures, absolute GPU efficiency is lowest at small batch sizes. For the 512 x 512 x 512 configuration, throughput at a batch size of 64 was only 1724.63 GFLOP/s, compared to 5461.33 GFLOP/s at a batch size of 512— a 31% performance improvement. This disparity highlights the under-utilization of the GPU, where the overhead of data movement and kernel launches outweighs the time spent on actual computation.

In Figure 3, as the batch size increases to 512, the GPU achieves much higher efficiency, reaching up to 11,966.63 GFLOP/s for the largest layer configuration (2048x2048x1024). When compared to Figure 4, the same trend can be observed, though for the same largest layer configuration and batch size of 512, throughput remains a bit lower at (11949.58 GFLOP/s) due to the aforementioned implementation complexity of the GELU activation function.

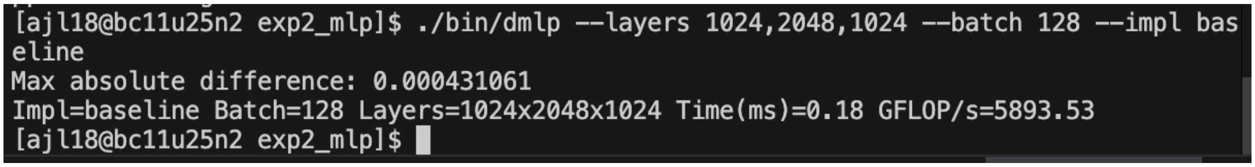
In addition, Figures 3 and 4 further show that the performance gain from fusion is most pronounced at batch sizes 64 and 128. In these small-batch regimes, kernel launch latency and the memory bandwidth cost of writing intermediate results to Global Memory are major bottlenecks. Fusion mitigates these costs by keeping data in registers between the bias and activation steps.

Key Takeaways and Conclusion

In this experiment, the data support the claim that speedup is more noticeable for smaller batch sizes or narrower layers. In these cases, the GEMM operation takes less time, and fixed costs, such as kernel launch overhead, become a larger fraction of the total execution time. Furthermore, for larger batches, the relative speedup is smaller because the O(N^3) computational cost of the GEMM dwarfs the O(N^2) memory-bound cost of bias and activation. However, even at peak throughput, fusion remains visible and thus advantageous for reducing overall memory traffic. Congruently, implementations with higher arithmetic intensity (GELU) benefit more from fusion than simpler ones (ReLU) because fusion allows the extra computation to overlap with the existing memory access pattern. Finally, all implementations met the accuracy requirement, maintaining an absolute difference of ≤ 1e-3 relative to the CPU double-precision reference.

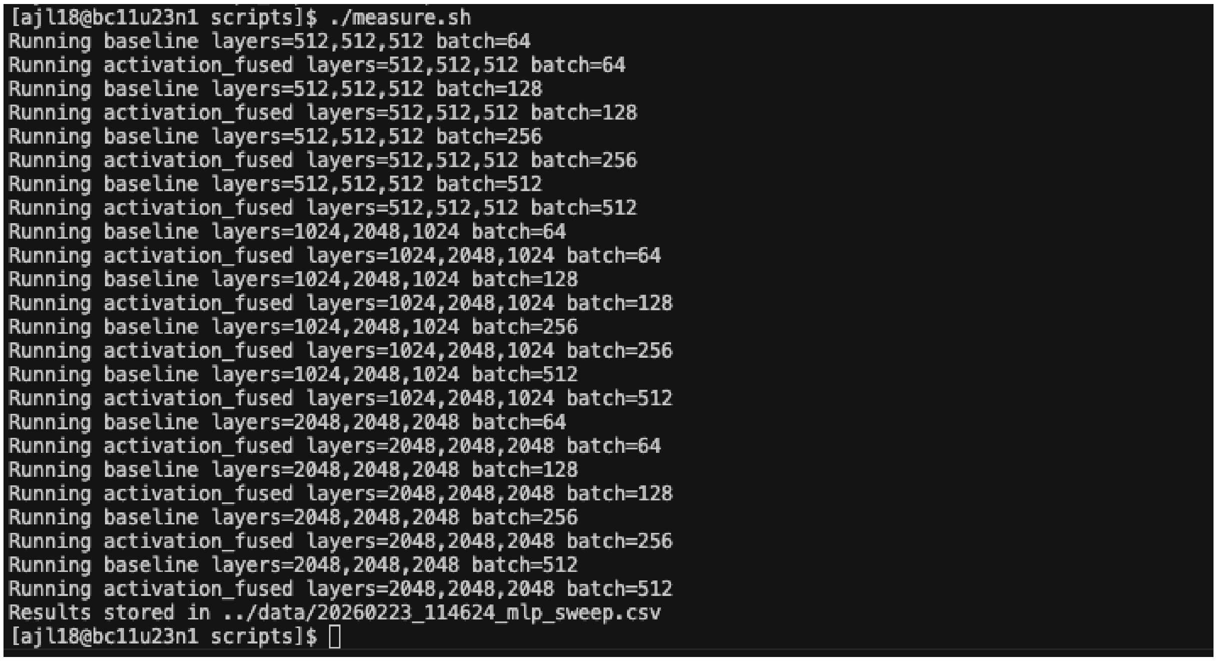
Logs

Max absolute difference ≤ 1e-3:



```
[ajl18@bc11u25n2 exp2_mlp]$ ./bin/dmlp --layers 1024,2048,1024 --batch 128 --impl bas
eline
Max absolute difference: 0.000431061
Impl=baseline Batch=128 Layers=1024x2048x1024 Time(ms)=0.18 GFLOP/s=5893.53
[ajl18@bc11u25n2 exp2_mlp]$ 
```

Script console output



```
[ajl18@bc11u23n1 scripts]$ ./measure.sh
Running baseline layers=512,512,512 batch=64
Running activation_fused layers=512,512,512 batch=64
Running baseline layers=512,512,512 batch=128
Running activation_fused layers=512,512,512 batch=128
Running baseline layers=512,512,512 batch=256
Running activation_fused layers=512,512,512 batch=256
Running baseline layers=512,512,512 batch=512
Running activation_fused layers=512,512,512 batch=512
Running baseline layers=1024,2048,1024 batch=64
Running activation_fused layers=1024,2048,1024 batch=64
Running baseline layers=1024,2048,1024 batch=128
Running activation_fused layers=1024,2048,1024 batch=128
Running baseline layers=1024,2048,1024 batch=256
Running activation_fused layers=1024,2048,1024 batch=256
Running baseline layers=1024,2048,1024 batch=512
Running activation_fused layers=1024,2048,1024 batch=512
Running baseline layers=2048,2048,2048 batch=64
Running activation_fused layers=2048,2048,2048 batch=64
Running baseline layers=2048,2048,2048 batch=128
Running activation_fused layers=2048,2048,2048 batch=128
Running baseline layers=2048,2048,2048 batch=256
Running activation_fused layers=2048,2048,2048 batch=256
Running baseline layers=2048,2048,2048 batch=512
Running activation_fused layers=2048,2048,2048 batch=512
Results stored in ../data/20260223_114624_mlp_sweep.csv
[ajl18@bc11u23n1 scripts]$ 
```