Coursework 1

Machine learning at scale [SEM2]

**Date: 08-03-2024**

**Exam number: B222009**

Content

[Question 1 3](#_Toc189418506)

[Question 2 3](#_Toc189418507)

[Question 3 5](#_Toc189418508)

# Question 1

Here's a summary of the performance optimizations implemented:

1. Data Loading Optimizations:

* Added memory mapping for large files
* Implemented data caching with configurable cache size
* Enabled persistent workers and prefetching
* Optimized number of workers based on CPU cores
* Added pin memory for faster GPU transfers
* Vectorized normalization operations

1. Model Architecture Optimizations:

* Implemented Flash Attention when available
* Added gradient checkpointing for memory efficiency
* Optimized attention computation
* Improved memory efficiency in forward passes
* Added support for PyTorch 2.0 compile mode

1. Training Loop Optimizations:

* Enabled Automatic Mixed Precision (AMP) training
* Added gradient accumulation for larger effective batch sizes
* Implemented gradient clipping
* Optimized device transfers with non-blocking operations
* Added cuDNN benchmarking
* Improved validation loop efficiency

1. Memory Optimizations:

* Added memory-efficient attention variants
* Implemented gradient checkpointing in transformer blocks
* Optimized tensor operations and reshaping
* Improved memory management in data loading

To use these optimizations, update your configuration file with the following parameters:

base:

# Training optimizations

use\_amp: true

compile\_mode: true

use\_checkpoint: true

gradient\_accumulation\_steps: 4

max\_grad\_norm: 1.0

# Data loading

num\_workers: 8

prefetch\_factor: 2

cache\_size: 1000

use\_mmap: true

pin\_memory: true

# Evaluation

eval\_every: 1

Expected Performance Improvements:

1. Training Speed:

* AMP should provide 2-3x speedup
* Flash Attention can provide up to 2x speedup for attention computation
* Improved data loading can reduce I/O bottlenecks

1. Memory Usage:

* Gradient checkpointing can reduce memory usage by up to 60%
* Memory-efficient attention reduces peak memory usage
* Better memory management in data loading

1. Training Stability:

* Gradient clipping helps prevent exploding gradients
* Gradient accumulation allows larger effective batch sizes
* Improved normalization stability

To monitor these improvements:

1. Watch the training speed (samples/sec) in the logs
2. Monitor GPU memory usage
3. Track validation metrics for stability
4. Check CPU utilization for data loading efficiency