Week 1 Assignment: Modern Attention Mechanisms

*From Theory to Implementation*

# **📋 Assignment Overview**

Duration: 3-4 days (6-8 hours total)

Difficulty: PhD Research Level

Focus: GQA, MQA, MLA + Implementation

# **🎯 Learning Objectives**

By completing this assignment, you will:

* ☐ Deeply understand GQA and MQA from original papers
* ☐ Master Multi-Head Latent Attention (MLA) - DeepSeek innovation
* ☐ Implement all three variants from scratch
* ☐ Benchmark real performance on your hardware
* ☐ Make informed architecture decisions for production systems

# **📚 Part 1: Paper Study (2-3 hours)**

## **Paper 1: GQA - Grouped-Query Attention**

📄 Paper: "GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints"

🔗 Link: https://arxiv.org/abs/2305.13245

⏱️ Time: 45 minutes

**Focus on these sections:**

* ☐ Section 2: Method - How GQA groups heads
* ☐ Section 3.1: Conversion from MHA - Uptraining technique
* ☐ Figure 2: Architecture diagram comparing MHA vs GQA vs MQA
* ☐ Table 1: Performance comparison (quality vs speed)
* ☐ Section 4.2: Inference throughput analysis

**Answer these questions:**

**1.** Why does GQA achieve better quality than MQA while still getting most of the speed benefits?

**2.** What is "uptraining" and why does it help convert MHA → GQA without full retraining?

**3.** Looking at Table 1, at what group size (g) do we get the best quality/speed tradeoff?

**4.** Llama 2 70B uses g=8 (8 KV heads for 64 query heads). Calculate the memory reduction compared to full MHA.

## **Paper 2: MQA - Multi-Query Attention**

📄 Paper: "Fast Transformer Decoding: One Write-Head is All You Need"

🔗 Link: https://arxiv.org/abs/1911.02150

⏱️ Time: 30 minutes

**Focus on these sections:**

* ☐ Section 2: Multi-Query Attention mechanism
* ☐ Section 3: Why single KV head works (theoretical justification)
* ☐ Figure 1: Architecture comparison
* ☐ Section 4.1: Translation experiments and quality analysis

**Answer these questions:**

**5.** The paper shows minimal quality degradation. What is the perplexity difference between MHA and MQA on their test set?

**6.** Why is MQA particularly beneficial for autoregressive generation (inference) vs training?

**7.** The paper mentions "incremental decoding" - explain how MQA specifically helps with KV cache management.

## **Paper 3: MLA - Multi-Head Latent Attention (DeepSeek)**

📄 Paper: "DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model"

🔗 Link: https://arxiv.org/abs/2405.04434

⏱️ Time: 1 hour (this is the new, advanced one!)

**Focus on these sections:**

* ☐ Section 2.1: Multi-Head Latent Attention architecture
* ☐ Figure 2: MLA mechanism diagram
* ☐ Section 2.1.2: Low-rank KV compression technique
* ☐ Table 2: KV cache comparison (MHA vs GQA vs MLA)
* ☐ Section 3.2: Training details and stability

**Answer these questions:**

**8.** MLA uses "latent dimensions" for compression. What is d\_latent in DeepSeek-V2 and how does it compress KV cache?

**9.** Explain the two projection steps: c^KV = W^DKV h and then K/V = W^UK/W^UV c^KV. Why this two-step process?

**10.** Compare KV cache size: For d=5120, h=128 heads, DeepSeek uses d\_latent=512. Calculate memory reduction vs standard MHA.

**11.** Why does MLA achieve BETTER quality than GQA despite more compression? (Hint: read about decoupled RoPE)

**📝 Deliverable: Create "paper\_analysis.pdf" with your answers to all 11 questions**

Include diagrams, equations, and your own explanations. Aim for 3-4 pages.

# **💻 Part 2: Implementation (2-3 hours)**

## **Task 2.1: Implement Three Attention Variants**

Create attention.py with these three classes:

class MultiHeadAttention(nn.Module):  
 """Standard MHA from 'Attention is All You Need'"""  
 # Each head has its own Q, K, V projections  
   
class GroupedQueryAttention(nn.Module):  
 """GQA from the paper you just read"""  
 # num\_heads > num\_kv\_heads  
 # Heads share K,V within groups  
   
class MultiHeadLatentAttention(nn.Module):  
 """MLA from DeepSeek-V2 paper"""  
 # Low-rank compression: d\_model -> d\_latent -> d\_model  
 # This is the advanced one!

**Requirements:**

* ☐ All three must work with same input: (batch, seq\_len, d\_model)
* ☐ All three must produce same output shape
* ☐ Use proper scaled dot-product: softmax(QK^T / sqrt(d\_k)) V
* ☐ MLA must include the two-step projection (down to latent, then up)
* ☐ Add comments explaining each step

## **Task 2.2: Memory Benchmark**

Create benchmark.py that measures:

**Metrics to collect:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variant** | **Seq Len** | **KV Cache (MB)** | **Tokens/sec** | **Quality (ppl)** |
| MHA | 1024 | ? | ? | baseline |
| MHA | 4096 | ? | ? | baseline |
| GQA (g=8) | 1024 | ? | ? | ? |
| GQA (g=8) | 4096 | ? | ? | ? |
| MLA | 1024 | ? | ? | ? |
| MLA | 4096 | ? | ? | ? |

**How to measure:**

# KV Cache Memory  
torch.cuda.reset\_peak\_memory\_stats()  
output = model(x)  
peak\_memory = torch.cuda.max\_memory\_allocated() / 1e6  
  
# Speed (tokens/second)  
start = time.time()  
for \_ in range(100):  
 \_ = model(x)  
tokens\_per\_sec = (seq\_len \* 100) / (time.time() - start)  
  
# Quality: Train small LM, measure perplexity on validation set

## **Task 2.3: Llama-Style Configuration**

Implement a function that creates Llama 2 70B attention configuration:

def create\_llama\_attention(model\_size='70B'):  
 """  
 Llama 2 70B config:  
 - d\_model = 8192  
 - num\_heads = 64  
 - num\_kv\_heads = 8 (GQA with g=8)  
 - head\_dim = 128  
   
 Return GQA module with these settings  
 """  
 pass  
  
# Test it  
llama\_attn = create\_llama\_attention('70B')  
x = torch.randn(1, 4096, 8192) # batch=1, seq=4k, dim=8192  
output = llama\_attn(x)  
print(f"Output shape: {output.shape}") # Should be (1, 4096, 8192)  
  
# Calculate KV cache for full 80 layers  
kv\_cache\_per\_layer = calculate\_kv\_cache(llama\_attn, seq\_len=4096)  
total\_cache = kv\_cache\_per\_layer \* 80 # 80 layers  
print(f"Total KV cache for 4k context: {total\_cache / 1e9:.2f} GB")

**📝 Deliverables:**

* ☐ attention.py - Three working implementations
* ☐ benchmark.py - Complete benchmarking script
* ☐ results.csv - Filled benchmark table
* ☐ llama\_config.py - Llama configuration function

# **🔬 Part 3: Research Analysis (1-2 hours)**

## **Experiment 1: Quality vs Efficiency Trade-off**

Using your implementations, answer:

**Research Question:** "At what point does compression hurt quality significantly?"

**What to do:**

* ☐ Train 5 small models (6 layers, 512 dim) on WikiText-2:
* ☐ • Baseline: MHA with 8 heads
* ☐ • GQA with g=4 (8→4 KV heads)
* ☐ • GQA with g=2 (8→2 KV heads)
* ☐ • MQA (8→1 KV head)
* ☐ • MLA with d\_latent=128
* ☐ Measure final validation perplexity for each
* ☐ Plot: Perplexity vs KV Cache Size

**Answer:**

• Which variant gives best quality/efficiency tradeoff?  
• Is there a "cliff" where quality suddenly drops?  
• Does MLA achieve better quality than GQA at same cache size?

## **Experiment 2: Real-World System Design**

You are building a production chatbot. Choose the right attention variant:

**Scenario A: Customer Support Bot**• Hardware: 1× A100 (40GB)  
• Context: 8k tokens average  
• Users: 50 concurrent  
• Latency requirement: < 2 seconds  
Your choice: \_\_\_\_\_ because \_\_\_\_\_

**Scenario B: Long Document Analysis**• Hardware: 4× A100 (40GB each)  
• Context: 32k-128k tokens  
• Users: 10 concurrent  
• Quality critical (legal documents)  
Your choice: \_\_\_\_\_ because \_\_\_\_\_

**Scenario C: Code Assistant**• Hardware: 8× H100 (80GB each)  
• Context: 16k tokens (full file context)  
• Users: 500 concurrent  
• Need maximum throughput  
Your choice: \_\_\_\_\_ because \_\_\_\_\_

**For each scenario, provide:**

* ☐ Your chosen attention variant
* ☐ Calculated max batch size with your choice
* ☐ Estimated KV cache memory usage
* ☐ Justification based on benchmarks from Part 2

## **Experiment 3: Future Research Proposal**

Based on papers and your experiments, propose ONE research direction:

**Format (1 page):**

**Title:** [Your research question]  
**Motivation:** Why this matters (2-3 sentences)  
**Hypothesis:** What you think will happen  
**Method:** How you would test it (3-4 bullet points)  
**Expected Impact:** If successful, what improves?

Example ideas:

• Can we learn optimal group size (g) dynamically during training?  
• Does MLA compression work better for Indic languages (less training data)?  
• Hybrid attention: MHA for early layers, MLA for later layers?  
• Adaptive attention: switch between GQA/MQA based on input complexity?

**📝 Deliverables:**

* ☐ experiment\_results.pdf - Plots and analysis for Experiment 1
* ☐ system\_design.pdf - Your choices and calculations for Experiment 2
* ☐ research\_proposal.pdf - Your 1-page proposal for Experiment 3

# **📦 Final Submission Checklist**

**Part 1: Paper Study**

* ☐ paper\_analysis.pdf (answers to 11 questions)

**Part 2: Implementation**

* ☐ attention.py (MHA, GQA, MLA implementations)
* ☐ benchmark.py (benchmarking script)
* ☐ results.csv (filled benchmark table)
* ☐ llama\_config.py (Llama configuration)

**Part 3: Research Analysis**

* ☐ experiment\_results.pdf (quality vs efficiency analysis)
* ☐ system\_design.pdf (3 scenarios with justifications)
* ☐ research\_proposal.pdf (1-page research idea)

# **📊 Grading Rubric (100 points)**

|  |  |  |
| --- | --- | --- |
| **Section** | **Points** | **Criteria** |
| Part 1: Paper Study | 30 | Deep understanding, correct answers, clear explanations |
| Part 2: Implementation | 40 | Working code, correct implementations, benchmarks run successfully |
| Part 3: Research Analysis | 30 | Thoughtful analysis, justified decisions, creative proposals |
| Total | 100 | 85+ = Excellent, 70-84 = Good, <70 = Needs more work |

# **💡 Pro Tips**

✓ Start with paper reading - understanding WHY is more important than coding

✓ For MLA, the two-step projection is key. Read DeepSeek paper Section 2.1.2 carefully

✓ Use small models for experiments (6 layers is enough to see patterns)

✓ Your benchmark script will be useful for the entire roadmap - make it good!

✓ For research proposal, pick something YOU are genuinely curious about

✓ Ask questions if stuck - better to clarify than waste time going wrong direction

# **⏰ Suggested Timeline**

|  |  |
| --- | --- |
| **Day** | **Tasks** |
| Day 1 (2-3 hrs) | Read all 3 papers, answer questions, create paper\_analysis.pdf |
| Day 2 (2-3 hrs) | Implement MHA and GQA, test they work, start MLA |
| Day 3 (2 hrs) | Finish MLA, run benchmarks, collect results |
| Day 4 (1-2 hrs) | System design scenarios, research proposal, final submission |

**Good luck! This assignment will set you up for success in the entire roadmap. 🚀**