Estimating Total Memory Requirements for Inference

When deploying a Transformer-based Large Language Model (LLM) for inference, understanding the memory requirements is crucial for efficient resource allocation and ensuring low latency responses. This guide will help you estimate the total memory size needed to run inference with the example model previously discussed.

Table of Contents

Introduction

Inference is the process of using a trained model to make predictions on new data. For Transformer-based LLMs like GPT models, inference often involves generating text in an autoregressive manner. Estimating the memory requirements for inference helps in:

- **Resource Allocation**: Ensuring the deployment environment has sufficient memory.

- **Scalability**: Planning for multiple concurrent inference requests.
- **Optimization**: Identifying opportunities to reduce memory usage for faster inference.

Components of Memory Usage During Inference

1. **Model Parameters**: The weights of the neural network.
2. **Activation Maps**: Outputs of each layer during the forward pass.
3. **Cache Memory**: Stored past key and value tensors for efficient autoregressive generation.
4. **Input Data**: The input tokens and any associated embeddings.
5. **Temporary Buffers and Overheads**: Memory used during computations.

Parameter Memory Calculation

From the previous calculation:

- **Total Parameters**: 354,336,768
- **Memory per Parameter**:
  - **32-bit (FP32)**: 4 bytes
  - **16-bit (FP16/BF16)**: 2 bytes
  - **8-bit (Int8 Quantization)**: 1 byte

Parameters Memory

Using FP16/BF16 Precision

- **Parameters Memory**:

  $$ \text{Parameters Memory} = 354,336,768 \times 2 = 708,673,536 \text{ bytes} \approx 0.66 \text{ GB} $$

Using Int8 Quantization

- **Parameters Memory**:

  $$ \text{Parameters Memory} = 354,336,768 \times 1 = 354,336,768 \text{ bytes} \approx 0.33 \text{ GB} $$

Activation Memory Calculation

Activation memory depends on:

- **Batch Size ((B))**
- **Sequence Length ((L))**
- **Model Dimension ((d_{\text{model}}))**
- **Number of Layers ((N))**
- **Bytes per Element**

Calculation

Per Token Activation Memory

- **Per Layer Activation Memory**:

  For inference, we only need to store activations for the current token as we do not perform backpropagation. However, activations are needed for computations within the layer.

- **Per Layer Memory**:

  $$ \text{Per Layer Memory} = B \times d_{\text{model}} \times \text{Bytes per Element} $$

- **Total Activation Memory**:

  $$ \text{Total Activation Memory} = N \times \text{Per Layer Memory} $$

Example with FP16 Precision

- **Bytes per Element**: 2 bytes

Cache Memory for Autoregressive Models

In autoregressive generation, models like GPT cache past key and value tensors to avoid recomputing them at each time step.

Cache Memory Calculation

- **Per Layer Cache for One Position**:
  - **Key Tensor**: (B \times h \times d_k)
  - **Value Tensor**: (B \times h \times d_v)
  - **Since (d_k = d_v = \frac{d_{\text{model}}}{h})**, total per layer per position:

    $$ 2 \times B \times h \times \frac{d_{\text{model}}}{h} = 2 \times B \times d_{\text{model}} $$

- **Cache for Sequence Length (L)**:

  $$ \text{Cache Memory} = N \times L \times 2 \times B \times d_{\text{model}} \times \text{Bytes per Element} $$

Total Memory Estimation

Summing Up Components

1. **Parameters Memory**
2. **Activation Memory**
3. **Cache Memory**
4. **Input Data Memory** (usually negligible)
5. **Temporary Buffers and Overheads**

Total Memory Formula

$$ \text{Total Memory} = \text{Parameters} + \text{Activations} + \text{Cache Memory} + \text{Overheads} $$

Example Calculations

Given

- **Model Dimension ((d_{\text{model}}))**: 1,024
- **Number of Layers ((N))**: 24
- **Number of Heads ((h))**: 16
- **Bytes per Element**: 2 bytes (FP16)
- **Batch Size ((B))**: 1 (single inference)
- **Sequence Length ((L))**: Varies (e.g., 128 tokens)

Parameters Memory

Using FP16:

$$ \text{Parameters Memory} = 0.66 \text{ GB} $$

Activation Memory

- **Per Layer Memory**:

  $$ \text{Per Layer Memory} = B \times d_{\text{model}} \times 2 = 1 \times 1,024 \times 2 = 2,048 \text{ bytes} $$

- **Total Activation Memory**:

  $$ \text{Total Activation Memory} = N \times 2,048 = 24 \times 2,048 = 49,152 \text{ bytes} \approx 0.05 \text{ MB} $$

Cache Memory

- **Per Layer Cache per Position**:

  $$ 2 \times B \times d_{\text{model}} = 2 \times 1 \times 1,024 = 2,048 \text{ bytes} $$

- **Total Cache Memory for (L = 128)**:

  $$ \text{Cache Memory} = N \times L \times 2,048 = 24 \times 128 \times 2,048 = 6,291,456 \text{ bytes} \approx 6 \text{ MB} $$

Temporary Buffers and Overheads

- Estimated at **~100 MB**

Total Memory Estimate

- **Parameters**: 0.66 GB
- **Activations**: ~0.05 MB
- **Cache**: ~6 MB
- **Overheads**: ~100 MB
- **Total Memory**:

  $$ \text{Total Memory} \approx 0.66 \text{ GB} + 0.1 \text{ GB} = 0.76 \text{ GB} $$

Additional Considerations

1. **Batch Size**:
   - Increasing the batch size linearly increases activation and cache memory.
2. **Sequence Length**:
   - Longer sequences increase cache memory linearly.
3. **Quantization**:
   - Using 8-bit or lower precision for parameters can significantly reduce memory usage.
4. **Memory Optimization Techniques**:
   - **Layer Fusion**: Combine operations to reduce memory overhead.
   - **Efficient Implementations**: Use optimized libraries like NVIDIA TensorRT or ONNX Runtime.
5. **Hardware Limitations**:
   - Ensure that the deployment environment (e.g., GPU, CPU) has sufficient memory.
6. **Streaming Inference**:
   - For very long sequences, consider processing in chunks to manage memory usage.

Conclusion

For inference, the memory requirements are significantly lower than during training because:

- **No Gradients or Optimizer States**: These are not needed during inference.
- **Reduced Activation Memory**: Only the forward pass activations are needed.
- **Cache Memory**: Autoregressive models require cache memory for past keys and values.

Total Estimated Memory Requirements:

- **Approximately 0.76 GB** for a single instance with a sequence length of 128 tokens using FP16 precision.

This estimation helps in planning resource allocation for deploying the model in production environments.

References

- Vaswani, A., et al. (2017). "Attention is All You Need".
- Shoeybi, M., et al. (2019). "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism".

- NVIDIA. (2021). "NVIDIA TensorRT".
- OpenAI. (2020). "GPT-3 Technical Report".

Note: These calculations provide estimates. Actual memory usage may vary based on implementation details, hardware architecture, and framework optimizations.