Calculating the Inference FLOPs of an LLM Transformer Model

Understanding the number of Floating Point Operations (FLOPs) required for inference with a Large Language Model (LLM) Transformer is essential for evaluating computational efficiency, optimizing deployment strategies, and estimating energy consumption. This guide provides a comprehensive approach to calculating the inference FLOPs for the example model previously discussed.

Table of Contents

Introduction

Inference involves using a trained LLM Transformer to generate predictions or outputs based on new input data. Calculating the FLOPs required for inference helps in:

- **Resource Allocation**: Determining necessary computational resources.
- **Performance Benchmarking**: Comparing model efficiencies.
- **Optimization**: Identifying areas to reduce computational load.
- **Cost Estimation**: Estimating operational costs based on computational requirements.

Why FLOPs Matter

- **Computational Efficiency**: Higher FLOPs indicate more computations, affecting latency and throughput.
- **Energy Consumption**: More FLOPs generally lead to higher energy usage.
- **Hardware Selection**: Helps in choosing appropriate hardware accelerators (e.g., GPUs, TPUs).
- **Scalability**: Facilitates understanding how inference scales with model size and input complexity.

Components Contributing to FLOPs During Inference

The total FLOPs for inference are accumulated from:

1. **Embedding Layer**
2. **Multi-Head Attention (MHA)**
3. **Feed-Forward Network (FFN)**
4. **Layer Normalization**
5. **Activation Functions**
6. **Autoregressive Caching (for models like GPT)**

FLOPs Calculation per Component

Notation

- **Batch Size**: B
- **Sequence Length**: L
- **Model Dimension**: d_{model}
- **Feed-Forward Dimension**: d_{ff}
- **Number of Heads**: h
- **Number of Layers**: N
- **Bytes per Element**: Not directly relevant for FLOPs but impacts memory.

Embedding Layer

- **Operations**: Lookup and addition.
- **FLOPs per Token**:
  - **Token Embedding**: Lookup operations do not involve FLOPs.
  - **Positional Embedding**: Lookup operations do not involve FLOPs.
  - **Addition of Embeddings**: d_{model} FLOPs per token.
- **Total FLOPs**:

$$\text{Embedding FLOPs} = B \times L \times d_{\text{model}}$$

Multi-Head Attention

Steps:

1. **Linear Projections**: Query (Q), Key (K), Value (V)
2. **Scaled Dot-Product Attention**
   - Compute attention scores
   - Apply softmax
   - Compute attention output
3. **Output Projection**

Calculations:

1. **Linear Projections (Q, K, V)**

- **FLOPs per Projection**:

$$2 \times B \times L \times d_{\text{model}} \times d_{\text{model}}$$

  (Factor of 2 accounts for multiply and add operations in matrix multiplication.)

- **Total for Q, K, V**:

$$3 \times 2 \times B \times L \times d_{\text{model}}^2 = 6 \times B \times L \times d_{\text{model}}^2$$

2. **Attention Scores**
   - **Compute Scores**:

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

3. **Softmax**
   - **FLOPs**:
     - **Exponentials and Divisions**: Approximately 5 FLOPs per element.

$$5 \times B \times h \times L^2$$

4. **Weighted Sum**
   - **FLOPs**:

$$2 \times B \times L^2 \times d_{\text{model}}$$

5. **Output Projection**
   - **FLOPs**:

$$2 \times B \times L \times d_{\text{model}}^2$$

Total MHA FLOPs per Layer:

$$6 \times B \times L \times d_{\text{model}}^2 + 2 \times B \times L^2 \times d_{\text{model}} + 5 \times B \times h \times L^2 + 2 \times B \times L^2 \times d_{\text{model}} + 2 \times$$

Simplifying:

$$8 \times B \times L \times d_{\text{model}}^2 + 4 \times B \times L^2 \times d_{\text{model}} + 5 \times B \times h \times L^2$$

Feed-Forward Network

Consists of two linear transformations with an activation function in between.

1. **First Linear Layer**
   - **FLOPs**:

$$2 \times B \times L \times d_{\text{model}} \times d_{\text{ff}}$$

2. **Activation Function (e.g., GELU)**
   - **FLOPs**:
     - Approximately 8 FLOPs per element.

$$8 \times B \times L \times d_{\text{ff}}$$

3. **Second Linear Layer**
   - **FLOPs**:

$$2 \times B \times L \times d_{\text{ff}} \times d_{\text{model}}$$

Total FFN FLOPs per Layer:

$$2 \times B \times L \times d_{\text{model}} \times d_{\text{ff}} + 8 \times B \times L \times d_{\text{ff}} + 2 \times B \times L \times d_{\text{ff}} \times d_{\text{model}} = 4 \times B \times L \times d_{\text{mod}}$$

Layer Normalization

- **FLOPs per Layer Norm**:
  - **Mean Calculation**: Sum and division.
  - **Variance Calculation**: Subtract mean, square, sum, division.
  - **Normalization**: Subtract mean, divide by standard deviation.
  - **Total per Layer Norm**:

$$\approx 5 \times B \times L \times d_{\text{model}}$$

- **Assumption**: Two Layer Norms per layer (pre-attention and post-FFN).

Activation Functions

- **GELU Activation**: Approximately 8 FLOPs per element.
- **ReLU Activation**: Approximately 1 FLOP per element.

- **Assumption**: Using GELU for activations.

Total FLOPs per Inference Pass

To calculate the total FLOPs for a single inference pass:

1. **Embedding Layer FLOPs**:

$$B \times L \times d_{\text{model}}$$

2. **Per Layer FLOPs**:
   - **MHA**:

$$8 \times B \times L \times d_{\text{model}}^2 + 4 \times B \times L^2 \times d_{\text{model}} + 5 \times B \times h \times L^2$$

   - **FFN**:

$$4 \times B \times L \times d_{\text{model}} \times d_{\text{ff}} + 8 \times B \times L \times d_{\text{ff}}$$

   - **Layer Norms**:

$$10 \times B \times L \times d_{\text{model}}$$

   (Two Layer Norms per layer)

   Total per Layer:

$$8 \times B \times L \times d_{\text{model}}^2 + 4 \times B \times L^2 \times d_{\text{model}} + 5 \times B \times h \times L^2 + 4 \times B \times L \times d_{\text{model}} \times d_{\text{f}}$$

3. **Total for All Layers**:

$$N \times \Big(8 \times B \times L \times d_{\text{model}}^2 + 4 \times B \times L^2 \times d_{\text{model}} + 5 \times B \times h \times L^2 + 4 \times B \times L \times d_{\text{mod}}$$

4. **Total Inference FLOPs**:

$$\text{Total FLOPs} = \text{Embedding FLOPs} + \text{Total Layer FLOPs}$$

Example Calculation

Given

- **Batch Size (B)**: 1
- **Sequence Length (L)**: 1,024
- **Model Dimension (d_{model})**: 1,024
- **Feed-Forward Dimension (d_{ff})**: 4,096
- **Number of Heads (h)**: 16
- **Number of Layers (N)**: 24

Calculating FLOPs for One Inference Pass

1. **Embedding Layer FLOPs**:

$$B \times L \times d_{\text{model}} = 1 \times 1,024 \times 1,024 = 1,048,576 \text{ FLOPs}$$

2. **Per Layer FLOPs**:
   - **MHA**:

$$8 \times 1 \times 1,024 \times (1,024)^2 + 4 \times 1 \times (1,024)^2 \times 1,024 + 5 \times 1 \times 16 \times (1,024)^2$$

   Simplifying:

$$8 \times 1,024 \times 1,048,576 = 8,589,934,592 \text{ FLOPs}$$

$$4 \times 1,048,576 \times 1,024 = 4,294,967,296 \text{ FLOPs}$$

$$5 \times 16 \times 1,048,576 = 83,886,080 \text{ FLOPs}$$

   Total MHA FLOPs per Layer:

$$8,589,934,592 + 4,294,967,296 + 83,886,080 = 12,968,787,968 \text{ FLOPs}$$

- **FFN**:

  $$4 \times 1 \times 1,024 \times 4,096 + 8 \times 1 \times 1,024 \times 4,096 + 10 \times 1 \times 1,024 \times 1,024$$

  Simplifying:

  $$4 \times 1,024 \times 4,096 = 16,777,216 \text{ FLOPs}$$

  $$8 \times 1,024 \times 4,096 = 33,554,432 \text{ FLOPs}$$

  $$10 \times 1,024 \times 1,024 = 10,485,760 \text{ FLOPs}$$

  Total FFN FLOPs per Layer:

  $$16,777,216 + 33,554,432 + 10,485,760 = 60,817,408 \text{ FLOPs}$$

- **Total FLOPs per Layer**:

  $$12,968,787,968 + 60,817,408 = 13,029,605,376 \text{ FLOPs}$$

3. **Total for All Layers**:

   $$24 \times 13,029,605,376 = 312,710,529,024 \text{ FLOPs}$$

4. **Total Inference FLOPs**:

   $$1,048,576 + 312,710,529,024 = 312,711,577,600 \text{ FLOPs}$$

   Approximately **312.71 billion FLOPs** per inference pass.

Additional Considerations

1. **Batch Size Impact**:

- **Linear Scaling**: FLOPs scale linearly with batch size.
- **Example**: For $B = 8$, total FLOPs would be $8 \times 312.71 \text{ billion} = 2.5017 \text{ trillion}$ FLOPs.

2. **Sequence Length Impact**:
   - **Quadratic Scaling**: Particularly in the attention mechanism, FLOPs scale quadratically with sequence length.
   - **Example**: Doubling L quadruples the attention-related FLOPs.
3. **Optimizations**:
   - **Sparse Attention**: Reduces FLOPs by limiting attention to certain token pairs.
   - **Efficient Transformer Variants**: Models like Performer, Longformer reduce computational complexity.
   - **Quantization**: Lower precision can marginally affect FLOPs but significantly reduce memory usage.
4. **Hardware Efficiency**:
   - **Throughput and Parallelism**: Actual inference speed depends on hardware capabilities, such as the number of cores and memory bandwidth.
   - **Batch Processing**: Efficiently utilizing batch processing can lead to better hardware utilization.
5. **Autoregressive Caching**:
   - **Reusing Computations**: Caching key and value tensors reduces redundant computations in subsequent token generations.
   - **Impact on FLOPs**: While caching saves computational steps, the initial token generation still incurs full FLOPs costs.
6. **Model Pruning and Distillation**:
   - **Pruning**: Removing redundant weights can decrease FLOPs.
   - **Distillation**: Transferring knowledge to a smaller model reduces FLOPs while maintaining performance.

Conclusion

Calculating the FLOPs required for inference with an LLM Transformer model involves aggregating the computational costs of each component within the model architecture. For the example model with:

- **Batch Size (B)**: 1
- **Sequence Length (L)**: 1,024
- **Model Dimension (d_{model})**: 1,024
- **Feed-Forward Dimension (d_{ff})**: 4,096
- **Number of Heads (h)**: 16
- **Number of Layers (N)**: 24

Total Inference FLOPs: Approximately **312.71 billion FLOPs** per inference pass.

Understanding these FLOPs helps in:

- **Selecting Appropriate Hardware**: Ensuring that computational resources meet the model's demands.
- **Optimizing Deployment**: Balancing performance with computational efficiency.
- **Scaling Applications**: Planning for higher throughput based on computational capabilities.

References

- Vaswani, A., et al. (2017). "Attention is All You Need".
- Kaplan, J., et al. (2020). "Scaling Laws for Neural Language Models".
- Shoeybi, M., et al. (2019). "Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism".
- OpenAI. (2020). "GPT-3 Technical Report".
- Patil, V., et al. (2021). "Efficient Transformers: A Survey".

Note: These calculations provide estimates. Actual FLOPs may vary based on implementation details, optimizations, and hardware-specific operations.