Calculating the Training Speed of an LLM Transformer Model in FLOPs

Estimating the number of floating-point operations (FLOPs) required to train a Transformer-based Large Language Model (LLM) is crucial for understanding the computational resources needed. This guide provides a detailed walkthrough on how to calculate the FLOPs for training the example model previously discussed.

Table of Contents

Introduction

When training a neural network, especially large models like Transformers, it's important to estimate the computational cost. The number of floating-point operations (FLOPs) gives a hardware-agnostic measure of computational complexity.

Why FLOPs Matter

- **Resource Planning**: Helps in selecting appropriate hardware.
- **Performance Benchmarking**: Allows comparison between different models.
- **Energy Consumption Estimation**: Higher FLOPs often mean more energy usage.
- **Optimization**: Identifying bottlenecks to improve efficiency.

Components Contributing to FLOPs

The FLOPs count comes from:

1. **Embedding Layer**
2. **Multi-Head Attention (MHA)**
3. **Feed-Forward Network (FFN)**
4. **Layer Normalization**
5. **Activation Functions**

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

Embedding Layer

- **Operation**: Lookup + Addition
- **FLOPs per Token**:
  - Token Embedding: None (lookup operation)
  - Positional Embedding: None (lookup operation)
  - Addition of embeddings: d_{model} FLOPs per token
- **Total FLOPs**:
  - $\text{Embedding Addition FLOPs} = B \times L \times d_{\text{model}}$

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**:
        - $3 \times 2 \times B \times L \times d_{\text{model}}^2$
2. **Attention Scores**
    - **Compute Scores**:
        - $B \times h \times L \times L \times \frac{d_{\text{k}}}{h}$
        - Since $d_{\text{k}} = \frac{d_{\text{model}}}{h}$, this simplifies.
        - **FLOPs**:
            - $2 \times B \times h \times L^2 \times \frac{d_{\text{model}}}{h} = 2 \times B \times L^2 \times d_{\text{model}}$
3. **Softmax**
    - **FLOPs**:
        - **Exponentials and Divisions**: Approximately 5 FLOPs per element.
        - Total FLOPs:
            - $5 \times B \times h \times L^2$
4. **Weighted Sum**
    - **FLOPs**:
        - $2 \times B \times h \times L^2 \times \frac{d_{\text{model}}}{h} = 2 \times B \times L^2 \times d_{\text{model}}$
5. **Output Projection**
    - **FLOPs**:
        - $2 \times B \times L \times d_{\text{model}}^2$

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**
   - **Assuming GELU Activation**:
     - Approximately 8 FLOPs per element.
     - **Total FLOPs**:
       - $8 \times B \times L \times d_{\text{ff}}$
3. **Second Linear Layer**
   - **FLOPs**:
     - $2 \times B \times L \times d_{\text{ff}} \times d_{\text{model}}$

Layer Normalization

- **FLOPs per Layer Norm**:
  - **Mean Calculation**: $B \times L \times d_{\text{model}}$ (sum and division)
  - **Variance Calculation**: $B \times L \times d_{\text{model}}$ (subtract mean, square, sum, divide)
  - **Normalization**: $B \times L \times d_{\text{model}}$ (subtract mean, divide by std)
  - **Total per Layer Norm**:
    - Approximately $5 \times B \times L \times d_{\text{model}}$

Activation Functions

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

Total FLOPs per Forward and Backward Pass

Forward Pass

- Sum the FLOPs from all components for the forward pass.

Backward Pass

- The backward pass requires computing gradients, which often involves similar computations to the forward pass.
- **Rule of Thumb**: The backward pass takes approximately **2 to 3 times** the FLOPs of the forward pass.
- For estimation, we'll assume the backward pass is **2 times** the forward pass FLOPs.

Total FLOPs per Training Step

- **Total FLOPs**:
  - $\text{FLOPs}_{\text{total}} = \text{FLOPs}_{\text{forward}} + \text{FLOPs}_{\text{backward}}$
  - $\text{FLOPs}_{\text{backward}} = 2 \times \text{FLOPs}_{\text{forward}}$
  - Therefore, $\text{FLOPs}_{\text{total}} = 3 \times \text{FLOPs}_{\text{forward}}$

Example Calculation

Given

- **Batch Size (B)**: Let's assume $B = 1$ for simplicity.
- **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 Layer

Embedding Layer FLOPs

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

Multi-Head Attention FLOPs

1. **Linear Projections (Q, K, V)**
   - $3 \times 2 \times B \times L \times d_{\text{model}}^2$
   - $6 \times 1 \times 1,024 \times (1,024)^2 = 6 \times 1,024 \times 1,048,576$
   - **Result**: $6,442,450,944$ FLOPs
2. **Attention Scores**
   - $2 \times B \times L^2 \times d_{\text{model}}$
   - $2 \times 1 \times (1,024)^2 \times 1,024 = 2 \times 1,048,576 \times 1,024$
   - **Result**: $2,147,483,648$ FLOPs
3. **Softmax**
   - $5 \times B \times h \times L^2$
   - $5 \times 1 \times 16 \times (1,024)^2 = 5 \times 16 \times 1,048,576$
   - **Result**: $83,886,080$ FLOPs
4. **Weighted Sum**
   - Same as Attention Scores: $2,147,483,648$ FLOPs
5. **Output Projection**
   - $2 \times B \times L \times d_{\text{model}}^2$
   - $2 \times 1 \times 1,024 \times (1,024)^2 = 2,147,483,648$ FLOPs

Total MHA FLOPs per Layer:

- Sum of the above: $6,442,450,944 + 2,147,483,648 + 83,886,080 + 2,147,483,648 + 2,147,483,648$
- **Result**: $12,968,788,968$ FLOPs

Feed-Forward Network FLOPs

1. **First Linear Layer**
   - $2 \times B \times L \times d_{\text{model}} \times d_{\text{ff}}$
   - $2 \times 1 \times 1,024 \times 1,024 \times 4,096 = 8,589,934,592$ FLOPs
2. **GELU Activation**
   - $8 \times B \times L \times d_{\text{ff}}$
   - $8 \times 1 \times 1,024 \times 4,096 = 33,554,432$ FLOPs
3. **Second Linear Layer**
   - $2 \times B \times L \times d_{\text{ff}} \times d_{\text{model}}$
   - Same as the first linear layer: $8,589,934,592$ FLOPs

Total FFN FLOPs per Layer:

- Sum of the above: $8,589,934,592 + 33,554,432 + 8,589,934,592$
- **Result**: $17,213,423,616$ FLOPs

Layer Normalization FLOPs

- Assuming two Layer Norms per layer:
  - $2 \times 5 \times B \times L \times d_{\text{model}}$
  - $10 \times 1 \times 1,024 \times 1,024 = 10,485,760$ FLOPs

Total FLOPs per Layer

- **Sum**:
  - Embedding Layer: $1,048,576$ (only once)
  - MHA: $12,968,788,968$
  - FFN: $17,213,423,616$
  - Layer Norms: $10,485,760$
  - **Total per Layer**: $30,192,698,344$ FLOPs

Total FLOPs for All Layers

- **Total for All Layers**:
  - $N \times \text{Total per Layer}$
  - $24 \times 30,192,698,344 = 724,624,760,256$ FLOPs
- **Add Embedding Layer FLOPs** (only once):
  - Total Forward FLOPs: $724,624,760,256 + 1,048,576 \approx 724,625,808,832$ FLOPs

Total FLOPs for Forward and Backward Pass

- **Forward Pass FLOPs**: $\approx 724,625,808,832$
- **Backward Pass FLOPs**: $2 \times 724,625,808,832 = 1,449,251,617,664$
- **Total FLOPs per Training Step**:
  - $724,625,808,832 + 1,449,251,617,664 = 2,173,877,426,496$ FLOPs

---

Additional Considerations

1. **Batch Size Impact**:
   - FLOPs scale linearly with batch size.
   - For $B = 16$, total FLOPs would be $16\times$ the calculated value.
2. **Sequence Length Impact**:
   - FLOPs scale quadratically with sequence length in the attention mechanism.
3. **Optimizations**:
   - **Sparse Attention**: Reduces FLOPs by focusing on local interactions.
   - **Efficient Transformers**: Models designed to reduce computational complexity.
4. **Hardware Efficiency**:
   - **Throughput**: Actual training speed depends on hardware (GPUs, TPUs) and their efficiency.
   - **Parallelism**: Utilizing data and model parallelism can affect training speed.

Conclusion

Calculating the FLOPs required to train a Transformer-based LLM involves summing the operations from embeddings, multi-head attention, feed-forward networks, and other components. For the example model with $B = 1$ and $L = 1,024$:

- **Total FLOPs per Training Step**: Approximately **2.17 trillion FLOPs**.

Understanding the computational requirements helps in selecting appropriate hardware and optimizing the training process.

References

- Vaswani, A., et al. (2017). "Attention is All You Need".

- Kaplan, J., et al. (2020). "Scaling Laws for Neural Language Models".
- Patterson, D., et al. (2021). "Carbon Emissions and Large Neural Network Training".
- Narang, S., et al. (2021). "Do Transformer Modifications Transfer Across Implementations and Applications?".

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