# Bangla Small Language Model From Scratch

Understanding Transformers from Scratch

Hosted by BARTA
(Bangla AI Research, Tools and Applications)

**Workshop Overview**

In this workshop, you will learn how large language models work by building a small Bangla language model from scratch. You'll start by manually calculating the core transformer operations to understand the mathematics, then trace how these operations flow through the entire model architecture.

## Part 1: Manual Calculation of Self-Attention

### The Heart of Transformers

### Context

Your Bangla GPT model uses the **Causal Self-Attention** mechanism. Let's understand it by calculating it by hand with a tiny example.

### The Scenario

Imagine we have tokenized a simple Bangla phrase: **"বাংলাদেশ ব্যাংক সুদ"** (Bangladesh Bank interest)

For this exercise, assume:

- We have 3 tokens (words): Token 1, Token 2, Token 3
- Embedding dimension ($n_{\text{embd}}$) = 4 (in your model it's 512, but we'll use 4 for hand calculation)
- Number of attention heads ($n_{\text{head}}$) = 2 (the SLM actually has 8)
- Each head dimension = $n_{\text{embd}}/n_{\text{head}} = 4/2 = 2$

## Step 1: Token Embeddings

After passing through the embedding layer, suppose we get these token embeddings:

$$\begin{aligned} \text{Token 1 (''বাংলাদেশ'')}: &\quad [1.0, 0.5, 0.2, 0.8] \\ \text{Token 2 (''ব্যাংক'')}: &\quad [0.3, 0.9, 0.4, 0.1] \\ \text{Token 3 (''সুদ'')}: &\quad [0.7, 0.2, 0.6, 0.5] \end{aligned}$$

This gives us input matrix $\boldsymbol{X}$ of shape $(3, 4)$:

$$\boldsymbol{X} = \begin{bmatrix} 1.0 & 0.5 & 0.2 & 0.8 \\ 0.3 & 0.9 & 0.4 & 0.1 \\ 0.7 & 0.2 & 0.6 & 0.5 \end{bmatrix} \tag{1}$$

## Step 2: Weight Matrices for One Attention Head

In the model code (url is attached with course material lecture 5), you see three linear transformations: `self.key`, `self.query`, and `self.value`. For Head 1, let's use simplified weight matrices:

$\boldsymbol{W}_Q$ **(Query weight matrix)** - shape $(4, 2)$:

$$\boldsymbol{W}_Q = \begin{bmatrix} 0.1 & 0.2 \\ 0.3 & 0.1 \\ 0.2 & 0.4 \\ 0.1 & 0.3 \end{bmatrix} \tag{2}$$

$\boldsymbol{W}_K$ **(Key weight matrix)** - shape $(4, 2)$:

$$\boldsymbol{W}_K = \begin{bmatrix} 0.2 & 0.1 \\ 0.1 & 0.3 \\ 0.4 & 0.2 \\ 0.3 & 0.1 \end{bmatrix} \tag{3}$$

$\boldsymbol{W}_V$ **(Value weight matrix)** - shape $(4, 2)$:

$$\boldsymbol{W}_V = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.4 \\ 0.1 & 0.3 \\ 0.4 & 0.1 \end{bmatrix} \tag{4}$$

# Questions for Students

## Question 1: Compute Query, Key, and Value Matrices

**[20 points]**

Calculate $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ by multiplying the input embeddings $\boldsymbol{X}$ with the respective weight matrices.

$$\boldsymbol{Q} = \boldsymbol{X} \times \boldsymbol{W}_Q \tag{5}$$
$$\boldsymbol{K} = \boldsymbol{X} \times \boldsymbol{W}_K \tag{6}$$
$$\boldsymbol{V} = \boldsymbol{X} \times \boldsymbol{W}_V \tag{7}$$

Show your complete matrix multiplication work for all three matrices. Remember: when multiplying a $(3, 4)$ matrix with a $(4, 2)$ matrix, you get a $(3, 2)$ result.

**Hint**

For the first element of $\boldsymbol{Q}$ (position $[0, 0]$), you would calculate:

$$Q_{0,0} = (1.0 \times 0.1) + (0.5 \times 0.3) + (0.2 \times 0.2) + (0.8 \times 0.1) = ?$$

*Show your work here:*

## Question 2: Calculate Attention Scores

**[25 points]**

Once you have $\boldsymbol{Q}$ and $\boldsymbol{K}$ matrices, calculate the attention scores matrix using:

$$\text{Attention Scores} = \frac{\boldsymbol{Q} \times \boldsymbol{K}^T}{\sqrt{d_k}} \tag{8}$$

where $d_k$ is the dimension of each key vector (in our case, $d_k = 2$, so $\sqrt{d_k} = 1.414$)

**Step 2a**: First, compute $\boldsymbol{K}^T$ (transpose of $\boldsymbol{K}$). If $\boldsymbol{K}$ is $(3, 2)$, what shape will $\boldsymbol{K}^T$ be?

**Step 2b**: Multiply $\boldsymbol{Q} \times \boldsymbol{K}^T$. What shape will this result be, and why?

**Step 2c**: Divide each element by $\sqrt{2} \approx 1.414$ to get scaled attention scores.

## Question 3: Apply the Causal Mask

**[15 points]**

In your model code, you see this critical line:

```
att = att.masked_fill(self.mask[:, :, :T, :T] == 0, float('-inf'))
```

This is what makes the attention "causal" – each token can only attend to previous tokens and itself, not future tokens.

**Step 3a**: Create a causal mask matrix for our 3 tokens. It should be a lower triangular matrix:

$$\text{Mask} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \tag{9}$$

**Step 3b**: Apply the mask to your attention scores. Replace positions where the mask is 0 with $-\infty$ (very large negative number like $-999$). Why do we use negative infinity?

## Question 4: Apply Softmax

**[20 points]**

Apply the softmax function to each row of your masked attention scores:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{10}$$

**Step 4a**: For the first row (how token 1 attends to itself), calculate:

$$\frac{e^{\text{score}_{11}}}{e^{\text{score}_{11}}}$$

**Step 4b**: For the second row (how token 2 attends to tokens 1 and 2), calculate:

$$\frac{e^{\text{score}_{21}}}{e^{\text{score}_{21}} + e^{\text{score}_{22}}}$$
$$\frac{e^{\text{score}_{22}}}{e^{\text{score}_{21}} + e^{\text{score}_{22}}}$$

**Step 4c**: Complete the calculation for the third row.

This gives you the **Attention Weights** matrix – each row sums to 1.0.

## Question 5: Compute Final Attention Output

**[15 points]** Now multiply your attention weights with the Value matrix:

$$\text{Output} = \text{Attention\_Weights} \times \boldsymbol{V} \tag{11}$$

Show your calculation. This output is what each token "learns" from attending to previous tokens!

# Part 2: Understanding the Full Model Flow

## Question 6: Trace Through the Architecture

**[25 points]**

Based on the model code and your manual calculations, answer these questions:

**6a**: In the Bangla SL< model, the block size is 128 tokens. If we process a batch of 64 sequences, what is the shape of the input tensor `idx` that enters the forward pass?

**6b**: After token embedding and positional embedding are added together, what is the shape of the resulting tensor `x`?

**6c**: The model has 8 layers (`N_LAYER = 8`). Each layer contains:

- Layer normalization
- Self-attention (with 8 heads)
- Another layer normalization
- MLP (feed-forward network)

If a single token representation starts as a vector of 512 dimensions, trace its shape as it flows through ONE transformer block. At which points does the shape change, if at all?

**6d**: The final output `logits` has shape (Batch, Sequence_Length, Vocab_Size). In your model:

- Batch = 64
- Sequence_Length = 128
- Vocab_Size = 32,100

What does each element of this tensor represent? Why do we need 32,100 values for each token position?

## Question 7: Loss Calculation and Learning

**[30 points]**

**7a**: The model uses Cross-Entropy Loss. If the model predicts probabilities for the next token and the actual next token has index 5432 in the vocabulary, write out the mathematical formula for the loss for this single prediction.

**Cross-Entropy Loss Formula**

$$\mathcal{L}_{\text{CE}} = -\log(p_{\text{true}}) \tag{12}$$

where $p_{\text{true}}$ is the predicted probability for the true class.

**7b**: During training, the model achieved:

- Training loss: 2.6058 at iteration 8500
- Validation loss: 4.0778 at iteration 8500

What does this tell you about the model's performance? Is there evidence of overfitting or underfitting? Explain your reasoning.

**7c**: The learning rate scheduler starts at $3 \times 10^{-4}$ and includes:

- Warmup phase: 1000 iterations
- Cosine decay: remaining iterations

Calculate the learning rate at:

- Iteration 500 (during warmup)
- Iteration 10,000 (during decay)

Use the formula:

$$\text{Warmup:} \quad lr = LR \times \frac{\text{iteration}}{\text{WARMUP\_ITERS}} \tag{13}$$

$$\text{Decay:} \quad lr = 0.1 \times LR + 0.9 \times LR \times 0.5 \times (1 + \cos(\pi \times \text{pct})) \tag{14}$$

where $\text{pct} = \frac{\text{iter}-\text{warmup}}{\text{max\_iters}-\text{warmup}}$

## Question 8: Generation Process

**[25 points]**

When generating text with the prompt **"বাংলাদেশ ব্যাংক"**, the model produced a news article about Bangladesh Bank.

**8a**: Explain step-by-step what happens during generation:

1. How is the prompt converted to numbers?
2. Why does the code use `out[:, -BLOCK_SIZE:]` to condition on recent tokens?
3. What role does temperature play? What happens if temperature = 0.1 vs temperature = 2.0?
4. What is the purpose of `top_k=40` filtering?

**8b**: The model generates one token at a time. If we want to generate 150 new tokens, approximately how many forward passes through the model are required? Why?

# Part 3: Dataset and Training Analysis

## Question 9: Data Processing

**[25 points]**

**9a**: The dataset contains 7,653 samples split into:

- Training: 7,270 samples (95%)
- Validation: 383 samples (5%)

After tokenization:

- Training tokens: 2,312,741
- Validation tokens: 122,216

Calculate the average number of tokens per sample in both splits. What does this tell you about Bangla text tokenization?

**9b**: The tokenizer (BanglaT5) has a vocabulary of 32,100 tokens. The tokens are stored as `uint16` (16-bit integers). Why is `uint16` sufficient, and what would be the maximum vocabulary size that `uint16` can represent?

**9c**: Each training iteration processes a batch of 64 sequences, each 128 tokens long. Calculate:

1. Total tokens processed per iteration
2. Total tokens seen by the model after 8,500 iterations
3. How many times does the model see the complete training set by iteration 8,500?

## Question 10: Model Capacity and Computation

**[30 points]**

**10a**: The model has 58.14 million parameters. Given:

- 8 layers
- 8 attention heads per layer
- Embedding dimension of 512
- Vocabulary size of 32,100

Estimate how many parameters are in:

1. The token embedding layer
2. One attention head (hint: Q, K, V projections plus output projection)
3. One MLP block (hint: it expands to $4 \times n_{\text{embd}}$ then back)

Do your estimates add up approximately to 58.14M?

**Parameter Calculation Hints**

- Embedding layer: vocab_size $\times$ n_embd
- Linear layer: input_dim $\times$ output_dim (+ bias if applicable)
- For attention: Consider Q, K, V each project from n_embd to n_embd

**10b**: The model uses approximately 4.33 GB of GPU memory during training. Explain what components occupy this memory:

- Model parameters (in float32)
- Gradients
- Optimizer state (Adam stores first and second moments)
- Activations for one batch

Calculate the approximate memory for model parameters:

$$58.14M \text{ parameters} \times 4 \text{ bytes (float32)} = ? \text{ GB} \tag{15}$$

# Bonus Challenge: Multi-Head Attention

## Question 11: Why Multiple Heads?

**[20 bonus points]**

You calculated attention for one head with dimension 2. The actual model uses 8 heads, each with dimension $512/8 = 64$.

**11a**: What is the computational advantage of using 8 heads of dimension 64 instead of 1 head of dimension 512? (Hint: think about what happens in parallel)

**11b**: Different attention heads can learn to focus on different types of relationships. Speculate on what different heads might learn when processing Bangla text:

- What might one head focus on?
- What might another head focus on?
- Why is this diversity useful?

**11c**: After computing 8 separate attention outputs, each of shape (batch, sequence, 64), how are they combined back into shape (batch, sequence, 512)? Show this operation conceptually.

# Reflection Questions

## Question 12: Bringing It All Together

**[30 points]**

**12a**: Draw a complete flowchart showing how the input Bangla text □□বাংলাদেশ ব্যাংক সুদ বৃদ্ধি" flows through the entire model from tokenization to final probability distribution. Label each transformation with tensor shapes.

**12b**: The validation loss plateaued around 4.08–4.13 while training loss continued decreasing. Propose three specific techniques you could implement to improve the model's generalization. For each technique, explain WHY it should help.

**12c**: Compare this 58M parameter Bangla model to large language models like GPT-4 (rumored $\sim$1.7 trillion parameters). Despite the massive size difference, the fundamental architecture is similar. What are the key advantages and limitations of this smaller model for Bangla language tasks?

## Learning Objectives

By completing this workshop, you will:

- Understand the mathematical foundations of self-attention mechanisms
- Trace data flow through a complete transformer architecture
- Connect theoretical concepts to practical implementation
- Analyze training dynamics and model performance
- Appreciate the challenges and considerations in building SLMs for Bangla language

**Good luck, and enjoy building your understanding of language models from the ground up!**