From LSTMs to Mamba Passing Through Attention

Nicolò Monti

1 Introduction

Sequence modeling is a cornerstone task in machine learning, with applications going from natural language processing to time series forecasting. Various architecures have been proposed over the year, each with its own pros and cons. The scope of this talk is to give you an insight on how architectures for such problem have evolved.

2 Long Short-Term Memory Networks (LSTMs)

Long Short-Term Memory networks (LSTMs) are a type of recurrent neural network (RNN) designed to address the vanishing gradient problem in training deep RNNs. LSTMs introduce a memory cell and gating mechanisms that regulate the flow of information.

2.1 LSTM Architecture

An LSTM cell consists of several components: the cell state C_t, the hidden state h_t, the input gate i_t, the forget gate f_t, and the output gate o_t. The mathematical operations within an LSTM cell are as follows:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{1}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{4}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$
$$h_t = o_t \odot \tanh(C_t) \tag{6}$$

Where σ is the sigmoid function, tanh is the hyperbolic tangent function, $\odot$ denotes element-wise multiplication, and W_f, W_i, W_C, W_o and b_f, b_i, b_C, b_o are the weights and biases of the gates.

2.2 Advantages and Limitations of LSTMs

LSTMs effectively capture long-term dependencies due to their gating mechanisms, but they suffer from limitations such as high computational cost and difficulty in parallelization due to their sequential nature.

3 Attention Mechanisms

Attention mechanisms address some of the limitations of LSTMs by allowing the model to focus on different parts of the input sequence when making predictions. This is particularly useful in tasks like machine translation, where different words in a sentence can have varying levels of importance.

3.1 Self-Attention

Self-attention, or scaled dot-product attention, is a key component of the Transformer architecture. The self-attention mechanism computes a weighted sum of the input values, where the weights are derived from the similarity between queries and keys. The equations are as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{7}$$

Where Q (queries), K (keys), and V (values) are matrices derived from the input embeddings, and d_k is the dimensionality of the queries and keys.

3.2 Multi-Head Attention

Multi-head attention extends the self-attention mechanism by allowing the model to jointly attend to information from different representation subspaces at different positions:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{8}$$

Where each head is computed as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{9}$$

Here, W_i^Q, W_i^K, W_i^V are learned projection matrices, and W^O is the output projection matrix.

3.3 Advantages and Limitations of Attention

Attention mechanisms allow for parallelization and capture long-range dependencies effectively. However, they can be computationally expensive, especially for long sequences, due to the quadratic complexity of the attention operation.

4 Mamba Architecture

The Mamba architecture represents a modern approach to sequence modeling, leveraging structured state space models (SSMs) and efficient computation techniques such as parallel scan and kernel fusion.

4.1 Structured State Space Models (SSMs)

SSMs combine principles from recurrent neural networks (RNNs), convolutional neural networks (CNNs), and classical state space models. They are characterized by the ability to handle longrange dependencies and efficient computation as recurrences or convolutions.

4.1.1 SSM Equations

An SSM is parameterized by matrices A, B, C, and a discretization parameter Δ, defining the continuous-time system:

$$h'(t) = Ah(t) + Bx(t) \tag{10}$$

$$y(t) = Ch(t) \tag{11}$$

To apply these dynamics in discrete-time settings, the parameters are discretized:

$$A_d = e^{A\Delta} \tag{12}$$

$$B_d = A^{-1}(e^{A\Delta} - I)B \tag{13}$$

Where $e^{A\Delta}$ is the matrix exponential of $A\Delta$, and I is the identity matrix.

4.1.2 Discretization and Implementation

Discretization ensures that the continuous system's behavior is faithfully represented in a discrete-time framework. Techniques like zero-order hold (ZOH) transform the continuous parameters to discrete ones, maintaining resolution invariance and normalization.

4.2 Efficient Computation with Parallel Scan

Parallel scan, or prefix sum, is used in Mamba to implement recurrent computation efficiently. The parallel scan algorithm involves an up-sweep (reduce) phase and a down-sweep phase:

4.2.1 Up-Sweep (Reduce) Phase

$$x[k] = x[k] + x[k - 2^d] \quad \text{for} \quad k = 2^{d+1} - 1, 2^{d+2} - 1, ..., n - 1 \tag{14}$$

4.2.2 Down-Sweep Phase

$$x[n-1] = 0 \tag{15}$$
$$\text{temp} = x[k] \tag{16}$$
$$x[k] = x[k] + x[k-2^d] \tag{17}$$
$$x[k-2^d] = \text{temp} \tag{18}$$

This two-phase approach ensures parallel computation, reducing the time complexity to $O(\log n)$.

4.3 Kernel Fusion

Kernel fuseon combines multiple kernel functions into a single kernel, reducing overhead and memory accesses. For instance, the SSM update equations:

$$h_t = A_d h_{t-1} + B_d x_t \tag{19}$$
$$y_t = C h_t \tag{20}$$

Can be fused into:

$$y_t = C(A_d h_{t-1} + B_d x_t) \tag{21}$$

Fusing these operations minimizes global memory accesses and kernel launches, improving performance.

4.4 GPU Hierarchy

Understanding the GPU hierarchy is crucial for optimizing algorithms. The hierarchy consists of multiple levels of memory and computational resources:

4.4.1 Stream Multiprocessors (SMs)

SMs contain CUDA cores and manage parallel execution:

- Warp Scheduling/: Threads are grouped into warps, executing the same instruction simultaneously. - /SIMT Architecture/: Single Instruction, Multiple Threads (SIMT) architecture optimizes parallel execution.

4.4.2 CUDA Cores

CUDA cores are the basic processing elements, performing arithmetic operations:

- /Execution Units/: Perform integer and floating-point operations, supporting mixed-precision arithmetic.

4.4.3 Memory Hierarchy

The memory hierarchy includes registers, shared memory, L1 cache, L2 cache, and global memory:

Registers Fastest and smallest memory, used for temporary variables:
- /Scope/: Private to each thread. - /Latency/: Extremely low. - /Capacity/: Limited.

Shared Memory Shared among threads in a block, used for data sharing and communication:
- /Scope/: Shared within a block. - /Latency/: Low. - /Capacity/: Typically 48KB per SM.

L1 Cache Stores frequently accessed data:
- /Scope/: Shared among threads in an SM. - /Latency/: Higher than shared memory. - /Capacity/: Around 128KB per SM.

L2 Cache Larger, slower cache shared among all SMs:
- /Scope/: Shared across the GPU. - /Latency/: Higher than L1 cache. - /Capacity/: Several megabytes.

Global Memory Main memory with the highest latency:
- /Scope/: Accessible by all threads. - /Latency/: High. - /Capacity/: Several gigabytes.

5 Challenge

Try coding one of those architecture *from scratch*! You're allowed to use whatever language or framework you wish. Whoever gets more point will win Wolfram Swag
- LSTM: 1 point - Transformer: 2 points - Mamba: 6 points