# Evolution of LLM Architectures

LLM Lab

November 26, 2025

## Contents

# 1 Evolution of LLM Architectures: From Transformers to Hybrid Models

This document traces the technical evolution of large language models, covering:

1. Original Transformer (2017)
2. Efficient / long-context Transformers
3. State Space Models (S4-style)
4. Selective SSMs (Mamba, Mamba-2)
5. Retention / hybrid sequence models (RetNet, Hyena/StripedHyena, RWKV)
6. Modern hybrids (Jamba, StripedHyena-2, etc.) as of late 2024-2025

---

## 1.1 1. The Original Transformer: Global Attention With $O(N^2)$ Cost

### 1.1.1 1.1 Scaled dot-product attention

Given a sequence of $N$ tokens, each is embedded into a $d_{\text{model}}$-dim vector. We form:

- Query matrix $Q \in \mathbb{R}^{N \times d_k}$
- Key matrix $K \in \mathbb{R}^{N \times d_k}$
- Value matrix $V \in \mathbb{R}^{N \times d_v}$

via learned linear projections:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V$$

The core operation is:

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

This gives a weighted sum of value vectors for each token, with weights determined by similarity between queries and keys. ([arXiv][1])

- **Complexity**: computing $QK^\top$ is $O(N^2 d_k)$.
- **Memory**: you must store an $N \times N$ attention matrix  the famous quadratic bottleneck.

### 1.1.2 1.2 Multi-head attention

Instead of one big attention, you use $h$ heads:

$$\text{MHA}(X) = \text{Concat}\big(\text{head}_1, \ldots, \text{head}_h\big)W_O$$

where each head has its own $(W_Q^{(i)}, W_K^{(i)}, W_V^{(i)})$ and computes attention in a lower-dimensional subspace. This lets the model attend to different "types" of relations in parallel (syntax vs coreference vs topic, etc.).

#### 1.1.3 1.3 Positional encodings

Because attention is permutation-invariant, Transformers add position information:

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin\Big(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\Big), \quad \mathrm{PE}_{(\mathrm{pos},2i+1)} = \cos\Big(\frac{\mathrm{pos}}{10000^{2i/d_{\mathrm{model}}}}\Big)$$

and use $X = \mathrm{TokenEmbedding} + \mathrm{PE}$. ([Wikipedia][2])

> **Key challenge**: Everything is great until you want millions of tokens. Then $O(N^2)$ attention is too slow and too memory-hungry.

---

### 1.2 2. Efficient / Long-Context Transformers (Still Attention-Centric)

A huge wave of work tried to keep attention semantics while reducing the cost.

#### 1.2.1 2.1 Sparse / local attention (Longformer, BigBird, etc.)

Idea: Instead of full $N \times N$ connectivity, define a sparse pattern $M$:

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\Big(\frac{QK^\top + M}{\sqrt{d_k}}\Big)V$$

where entries of $M$ are $-\infty$ for disallowed connections (local windows, global tokens, random blocks, etc.). This reduces the complexity to roughly $O(Nw)$ where $w$ is window size.

#### 1.2.2 2.2 Low-rank and kernelized approximations (Linformer, Performer)

$$\mathrm{softmax}(QK^\top)V \approx \mathrm{softmax}(Q(E_K K)^\top)(E_V V)$$

with learned projections $E_K, E_V \in \mathbb{R}^{N\times r}$, $r \ll N$.

$$\mathrm{softmax}(QK^\top)V \approx \frac{\phi(Q)\big(\phi(K)^\top V\big)}{\phi(Q)\big(\phi(K)^\top \mathbf{1}\big)}$$

making attention essentially **linear in** $N$.

These are important conceptual steps, but fundamentally still operate in an "attention mindset": every token can attend to every other, just more efficiently.

---

### 1.3 3. State Space Models (S4-style): From Attention to Dynamical Systems

The conceptual jump with S4 and friends is:

> What if long-range sequence modeling is better framed as a *dynamical system*?

#### 1.3.1 3.1 Classic (continuous) linear state space model

A linear time-invariant SSM defines hidden state $h(t)$, input $u(t)$, output $y(t)$:

$$\frac{dh(t)}{dt} = Ah(t) + Bu(t), \quad y(t) = Ch(t) + Du(t)$$

with learned matrices $A, B, C, D$. ([Hugging Face][3])

Discretize with step $\Delta t$:

$$h_{t+1} = \bar{A}h_t + \bar{B}x_t, \quad y_t = Ch_t + Dx_t$$

for sequences $x_t$.
If you unroll the recurrence, you see:

$$y_t = \sum_{\tau=0}^{t} K_\tau x_{t-\tau}$$

where $K_\tau$ is an **impulse response kernel** derived from $A, B, C, D$ (roughly $K_\tau = C\bar{A}^\tau \bar{B}$). So an SSM layer is mathematically just a **1D convolution over time** with a very structured kernel.

#### 1.3.2 3.2 S4: Structured State Space for Long Sequences

S4 (Efficiently Modeling Long Sequences with Structured State Spaces) has two key tricks: ([iclr-blog-track.github.io][4])

1. **HiPPO-based initialization** of $A$ to preserve information over long horizons.
2. **Structured** $A$ (diagonal + low-rank) so that the convolution kernel $K$ can be computed efficiently via FFT (or other structured transforms).

The layer looks like:

$$h_{t+1} = \bar{A}h_t + \bar{B}x_t, \quad y_t = Ch_t$$

implemented as:

$$y = K * x$$

with $K$ computed in $O(N \log N)$ from the SSM parameters.

**Compared to attention:**

* Attention: explicit pairwise token interactions, $O(N^2)$. * S4: implicit long-range interactions via convolution kernel, $O(N)$ or $O(N \log N)$. * You trade explicit content-based interactions for **learned, stable long-range filters**.

---

### 1.4 4. Selective SSMs (Mamba, Mamba-2): Making SSMs Input-Dependent

Plain SSMs use fixed $A, B, C$ for the whole sequence. Mamba's key insight:

Make the SSM *selective*: let the transition matrices depend on the input token.

#### 1.4.1 4.1 Mamba: "Linear-time sequence modeling with selective SSMs"

Mamba defines a **selective SSM**: ([arXiv][5])

$$h_t = A_t h_{t-1} + B_t x_t, \quad y_t = C_t^\top h_t$$

where now $A_t, B_t, C_t$ are **functions of the current input** $x_t$:

$$A_t = A(x_t),\ B_t = B(x_t),\ C_t = C(x_t)$$

implemented with small learned networks (e.g., linear layers + gates).
This gives you:

- **Content-dependent** recurrence (like attention),
- **Linear-time** scan over sequence (like RNN/conv),
- **Streaming** ability (no need to store full context).

On hardware, Mamba is implemented via a "selective scan" kernel that fuses the recurrence and gating efficiently, giving big throughput gains and enabling million-token sequences. ([arXiv][5])

#### 1.4.2 4.2 Mamba-2: state space duality

Mamba-2 refines this and clarifies the math: starting from the same SSM form

$$h_t = A_t h_{t-1} + B_t x_t, \quad y_t = C_t^\top h_t$$

but highlighting an exact duality between:

- a **recurrent view** (good for streaming), and
- a **convolutional / parallel view** (good for training). ([tridao.me][6])

This dual view is similar in spirit to how RWKV can be run both as a recurrent model and a parallelizable "GPT-style" model.

---

### 1.5 5. Retention, Hyena, RWKV: Re-inventing Long-Range Memory

#### 1.5.1 5.1 RetNet: Retentive Networks

RetNet replaces attention with a **retention** operator that looks like exponentially decayed attention over time.

At a high level (simplified):

$$s_n = \gamma \odot s_{n-1} + K_n^\top V_n$$

$$\text{Retention}(x)_n = Q_n s_n$$

where $\gamma \in (0, 1)$ controls exponential decay, and $Q_n, K_n, V_n$ come from projections of $x_n$. ([arXiv][7])

- This can be computed **recurrently** (constant-time per token for inference),
- or **in parallel** (like attention) during training.
- It mathematically connects recurrence and attention through a unified operator.

#### 1.5.2 5.2 Hyena & HyenaDNA: implicit long convolutions

Hyena operators model long-range dependencies via **implicit long convolutions** plus gated mixing: ([NeurIPS Proceedings][8])

Basic form:

$$y = \sum_{\tau=0}^{L-1} K_\tau \odot x_{t-\tau}$$

where the **kernel** $K$ is not directly learned as a long vector, but generated by a small MLP as a function of $\tau$, then applied via FFT-based convolution in $O(L \log L)$.

HyenaDNA uses this in genomics, reaching **1M-token context** on nucleotide-level sequences while training much faster than Transformers. ([arXiv][9])

StripedHyena and StripedHyena-2 then **hybridize**:

- Hyena-style long convolutions (for bulk long-range modeling),
- plus attention layers (for precise pattern recall and local interactions). ([Together AI][10])

#### 1.5.3 5.3 RWKV: RNN with attention-like update

RWKV is an RNN-like architecture that can be trained like a Transformer (parallel) but run as an RNN (streaming). ([arXiv][11])

At a simplified level, each layer maintains a **running weighted average** of keys and values with time-dependent decay and a softmax-like normalization, giving an **attention-like** effect with **linear complexity**. From a math perspective, it's another instance of:

$$\text{state}_t = f(\text{state}_{t-1}, x_t)$$

with cleverly chosen $f$ so that:

- you can derive a parallelizable formulation for training,
- but also maintain a compact hidden state for streaming inference.

---

### 1.6 6. Modern Hybrids (Jamba, StripedHyena-2, etc.) SoTA Pattern circa 2024-2025

By 20242025, the emerging pattern in SoTA long-context LLMs is: **hybrid architectures** that combine:

- **Attention** for flexible, content-based, "who should I look at?" reasoning
- **SSMs / convolutions / retention** for cheap long-range memory and streaming
- **Mixture-of-Experts (MoE)** for scaling parameter count without linearly scaling compute

#### 1.6.1 6.1 Jamba: Hybrid TransformerMamba MoE

Jamba interleaves **Transformer blocks** and **Mamba blocks**, with MoE in some layers: ([arXiv][12])

Roughly:

$$\text{Block}_\ell(x) = \begin{cases} \text{TransformerBlock}_\ell(x) & \ell \in \mathcal{T} \\ \text{MambaBlock}_\ell(x) & \ell \in \mathcal{M} \end{cases}$$

plus MoE FFNs:

$$\text{FFN}_{\text{MoE}}(x) = \sum_k g_k(x)\, \text{Expert}_k(x)$$

with sparse routing $g_k(x)$.

This gives:

- long context (reported up to 256k tokens),
- strong benchmark performance,
- good throughput and memory usage.

#### 1.6.2 6.2 StripedHyena-2 / Hyena-based LLMs

StripedHyena-2 and related systems use **multi-hybrid convolutional architectures**: Hyena convolutions + attention + sometimes SSM-like components. ([arXiv][13])

Key points:

- Hyena blocks handle long-range modeling efficiently.
- Attention blocks provide sharp, content-based lookup.
- The combination outperforms pure Transformers in throughput at large scale and on long-context benchmarks.

#### 1.6.3 6.3 RWKV, ARWKV & others

RWKV and successors (e.g. ARWKV) embody an **RNNTransformer duality**: train with parallelizable transformer-like math, serve as a streaming, compact-state RNN. ([arXiv][11])

---

### 1.7 7. Conceptual Summary: What Changed, in Equations

You can think of the evolution like this:

1. **Transformer (2017): global pairwise interactions**

$$y = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

1. **Efficient Transformers: same formula, but approximate or sparsify $QK^\top$**

- Sparse masks
- Low-rank projections
- Kernel tricks

1. **SSMs (S4): convert sequence modeling into learned convolution**

$$h_{t+1} = \bar{A}h_t + \bar{B}x_t, \quad y_t = Ch_t \quad \Rightarrow \quad y = K * x$$

1. **Selective SSMs (Mamba): make $A_t, B_t, C_t$ depend on $x_t$**

$$h_t = A(x_t)h_{t-1} + B(x_t)x_t, \quad y_t = C(x_t)^\top h_t$$

1. **Retention / RWKV / Hyena: alternative ways to maintain long-range memory**

- RetNet: exponentially decayed, attention-like recurrence

- RWKV: RNN with attention-shaped gating
- Hyena: implicit long convolutions via FFT

1. **Hybrids (Jamba, StripedHyena-2, etc.): combine all of the above**

- Attention + SSM + long conv + MoE
- Choose the right tool for the right scale (local vs global vs ultra-long).

---

## 1.8 8. Applications and Future Directions

From a bio + agentic workflows perspective:

- **For very long genomic sequences**: HyenaDNA / Mamba-like / SSM-based models are strongly aligned with these needs. ([arXiv][9])
- **For multi-modal, agentic, tool-using LLMs**: hybrid designs (Jamba-style, StripedHyena-2-like, or proprietary long-context GPT/Claude/Gemini) are increasingly the norm.

### 1.8.1 Potential Next Steps

- Derive the SSM  convolution kernel more explicitly
- Compare attention vs SSM in the frequency domain
- Sketch a **BioGraphLab note** section (LaTeX-ready) summarizing "Transformers  SSM Hybrids" for GNN/GRL notes
- Design a **toy Mamba-like layer in PyTorch** to make concepts concrete

---

## 1.9 References

[1] Attention Is All You Need [2] Attention Is All You Need [3] Introduction to State Space Models (SSM) [4] The Annotated S4 - The ICLR Blog Track [5] Linear-Time Sequence Modeling with Selective State Spaces [6] State Space Duality (Mamba-2) Part I - The Model [7] Retentive Network [8] HyenaDNA: Long-Range Genomic Sequence Modeling at ... [9] HyenaDNA: Long-Range Genomic Sequence Modeling at ... [10] Paving the way to efficient architectures: StripedHyena-7B ... [11] RWKV: Reinventing RNNs for the Transformer Era [12] Jamba: A Hybrid Transformer-Mamba Language Model [13] Systems and Algorithms for Convolutional Multi-Hybrid ...