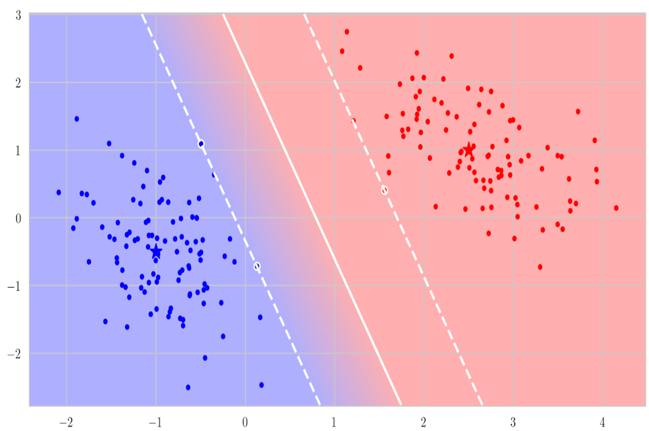
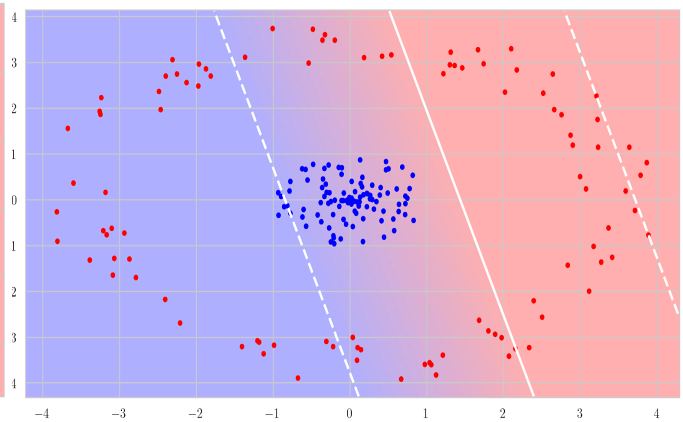
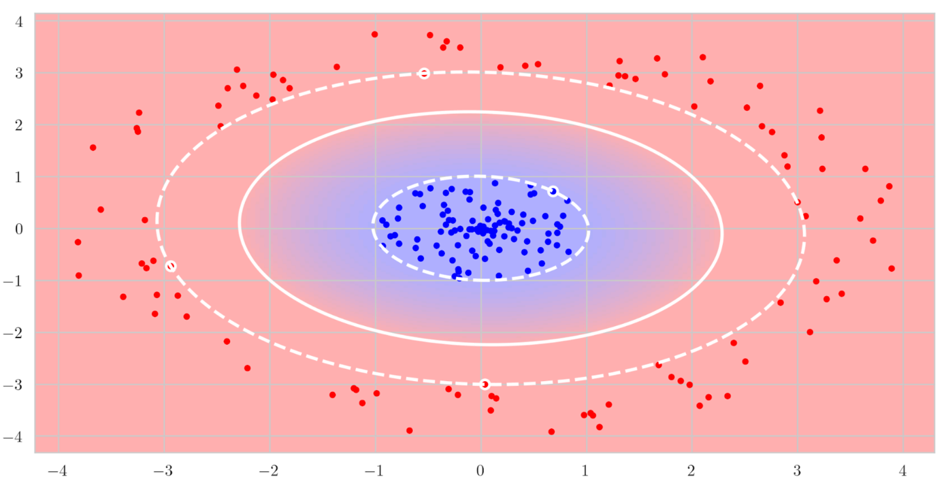
"Beyond Transformers: Exploring Modern AI Architectures"

🔍 ***From Transformers to State Space Models (SSMs), Mamba, Jamba, Zamba, Bamba, Mamba-2, Zyphra, FlashAttention, and Beyond***





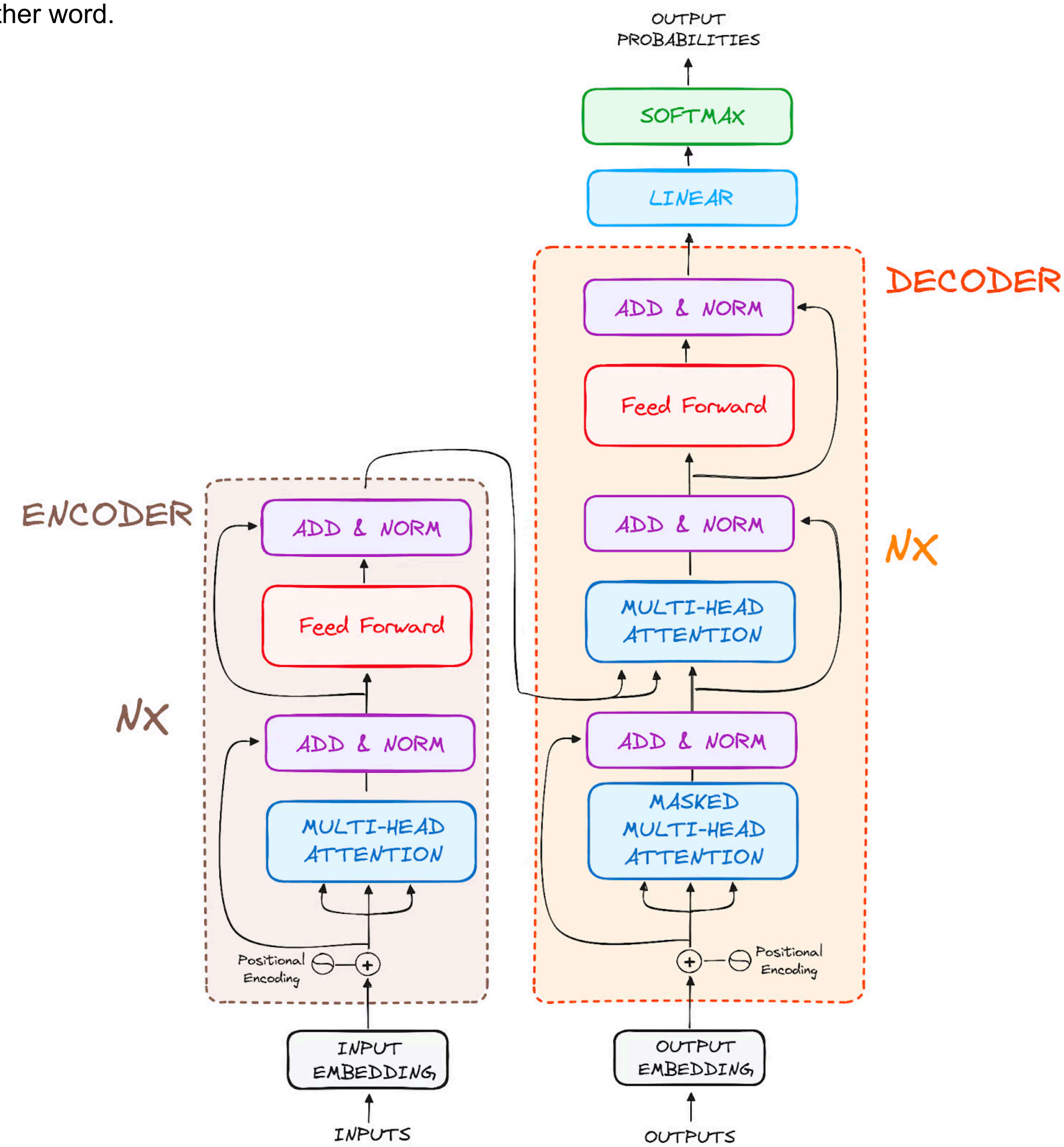


Transformers – The Standard Approach

🧠 **How do Transformers work?**

- **Self-Attention:** Every word compares itself with every other word.
- **Advantages:** Excellent at capturing relationships in text.
- **Limitations:**
  - **Slow for long sequences.**
  - **High memory usage.**

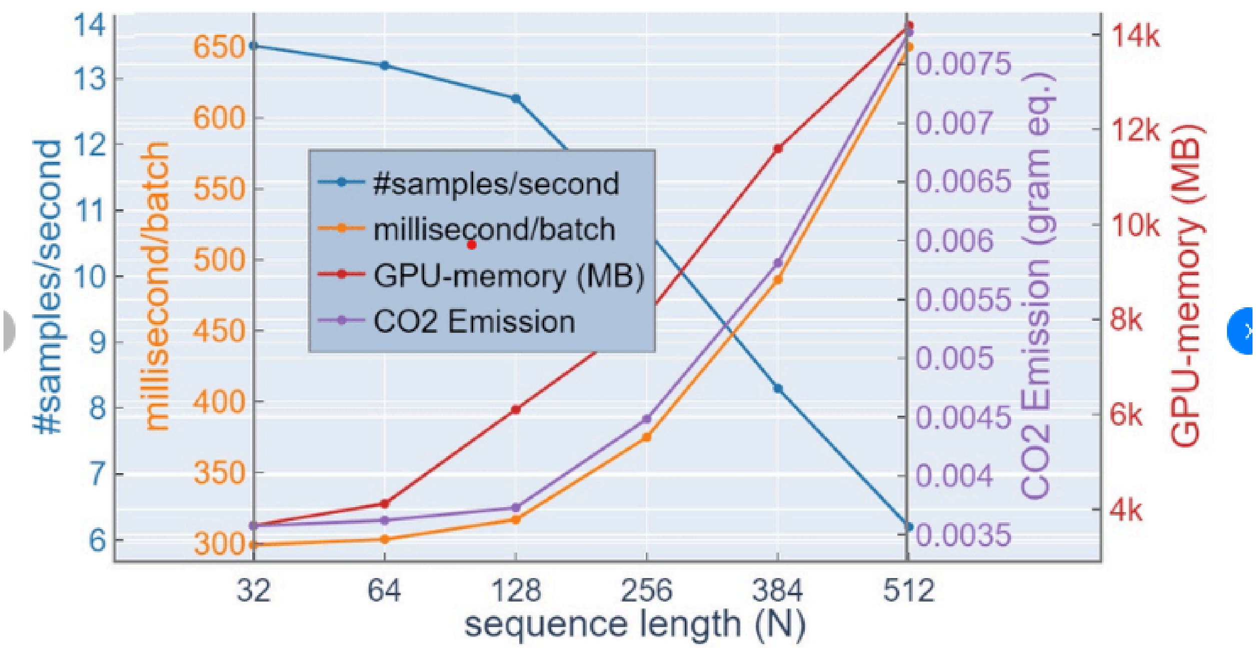
📌 *Example:* Processing an entire book is inefficient because every word interacts with every other word.



The Challenge – Processing Long Sequences

💡 **Why is this important?**

- AI models need to handle **long texts, time-series data, audio, and genomics.**
- **Transformers (like GPT-4) struggle with long sequences** due to **quadratic complexity (O(N2)).**
- **Inefficiency in handling extremely long contexts**

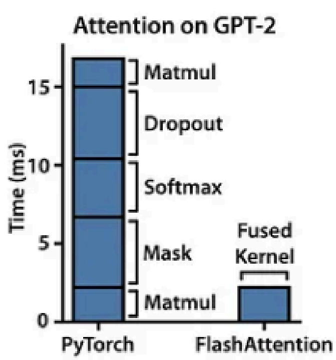


FlashAttention – Optimizing Transformers

⚡ **What is FlashAttention?**

- **An optimization technique for Transformers** to reduce memory usage.
- **Speeds up computation** but still **follows self-attention(O(N2)).**
- Used in **models like GPT-4 and Claude.**

FlashAttention vs. standard attention memory usage.



BERT Implementation	Training time (minutes)
Nvidia MLPerf 1.1 [58]	20.0 ± 1.5
FLASHATTENTION (ours)	**17.4** ± 1.4

Figure 4: A comparison between a standard-attention and flash attention. (Left) Flash attention delivers a 7.6x Speedup over the PyTorch implementation. (Right) Flash attention is 15% faster over an Nvidia implementation that set the training speed record for MLPerf 1.1.

Vanilla Attention	Flash Attention
1. Matmul_op (Q,K) a. Read Q,K to SRAM b. Compute matmul A=QxK c. Write A to HBM 2. Mask_op a. Read A to SRAM b. Mask A into A' c. Write A' to HBM 3. Softmax_op a. Read A' to SRAM b. Softmax A' into A" c. Write A" to HBM	1. Read Q,K to SRAM 2. Compute A = QxK 3. Mask A into A' 4. Softmax A' into A" 5. Write A" to HBM

Figure 3: A comparison between standard attention (left) and flash attention (right). This comparison leverages three operations (matmul, mask, softmax) only. Other operations (e.g., dropout) are omitted for presentation purposes.

State Space Models (SSMs) – A Different Approach

🧩 **What are SSMs?**

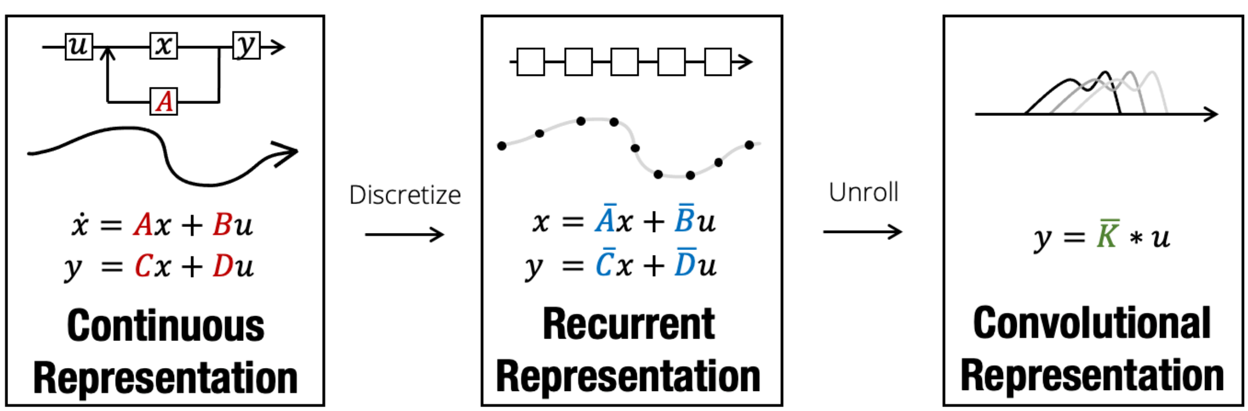
- Instead of self-attention, **SSMs track hidden states over time.**
- Used in **control systems, time-series forecasting, and deep learning.**
- **More efficient for long sequences** because they **don't compare every token.**

📌 *Think of it like summarizing key points instead of remembering every word.*

✅ **Linear Scaling:**.✅ **Memory Efficiency:**.✅ **Parallelized Training:**✅ **Handles Long-Range Dependencies:** Unlike RNNs, SSMs can capture **long-term information** without vanishing gradients

Discretization is one of, if not the most important point in SSM. All the efficiency of this architecture lies in this step, since it enables us to pass from the continuous view of the SSM to its two other views: the recursive view and the convolutive view.

If there's one thing to remember from this, it's this.



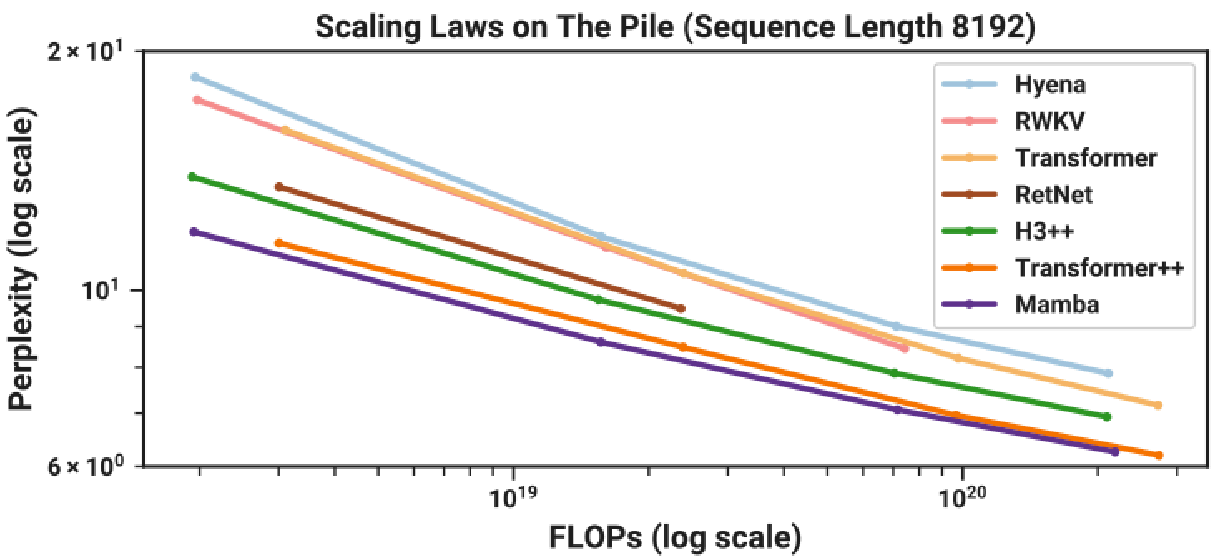
Mamba – An Alternative to Transformers

🐍 **What is Mamba?**

- **Inspired by SSMs but designed for deep learning.**
- **Processes sequences in O(N)O(N) (linear) instead of O(N2)O(N^2) (quadratic).**
- **More efficient for long-sequence tasks like genomics, audio, and language.**
- **Does not use self-attention but structured state updates.**

📌 *Think of it like tracking important points in a discussion instead of listening to everything at once.*

Mamba vs. Transformer efficiency comparison chart.



Expanding on Mamba & Emerging Innovations

🛠️ Variants of Mamba

✅ **Jamba** – Hybrid model blending Transformer-like properties for performance boost.
✅ **Zamba** – Optimized for audio and genomics, excelling in domain-specific tasks.
✅ **Zyphra** – Enhances sequence processing, ideal for long-context applications.

✨ **Bamba** – A further optimized variant of Mamba with efficiency refinements.
✨ **Mamba 2** – An improved, next-gen version with enhanced stability & scalability.

📢 Others

🔹 **EfficientVMamba** – Introduces Atrous Selective Scan, enabling lightweight deployment & better global-local feature extraction.

🔹 **Cobra** – Extends Mamba into a multi-modal AI for vision-language reasoning, achieving faster inference.

🔹 **SiMBA** – A simplified Mamba-based model with EinFFT for stable scaling, excelling in vision & time-series tasks.

🌐 ***Explore more: GitHub - state-spaces/mamba***

🌐 ***The Potential Transformer Replacement: Mamba***:

https://medium.com/@zilliz_learn/the-potential-transformer-replacement-mamba-f982a9d2aa12

🌐 ***Mamba Will Never Beat the Transformer:***

https://nathanpaull.substack.com/p/mamba-will-never-beat-the-transformer-24-03-08