Zefan Cai

zefncai@gmail.com · Google Scholar · GitHub · Main Page

Education

University of Wisconsin - Madison - Institute of Computer Science 2024 – Present
Ph.D. majoring in Computer Science
Supervisor: Prof. **Junjie Hu** from School of Computer, Data & Information Sciences

Peking University - School of Software & Microelectronics 2022 – 2024
M.Sc. majoring in Computer Science Technology; GPA: 3.71/4.0 (top 5%)
Supervisor: Prof. **Baobao Chang** from Institute of Computational Linguistics (ICL) from Peking University

Beijing Jiaotong University - College of Computer Science and Technology 2018 – 2022
B.Eng. majoring in Computer Science; GPA: 3.61/4.0 (Rank 2/31)

Selected Publications

1. R-KV: Redundancy-aware KV Cache Compression for Training-Free Reasoning Models Acceleration [PDF] [Code]
   Zefan Cai, Wen Xiao, Hanshi Sun, ..., Anima Anandkumar, Abedelkadir Asi, Junjie Hu **NeurIPS 2025, Poster**
2. PyramidKV: Dynamic KV Cache Compression based on Pyramidal Information Funneling [PDF] [Code]
   Zefan Cai, Yichi Zhang, ..., Junjie Hu, Wen Xiao **COLM 2025, Spotlight (24 / 418)**
3. Not All Heads Matter: A Head-Level KV Cache Compression Method with Integrated Retrieval and Reasoning [PDF] [Code]
   Yu Fu, **Zefan Cai**, Abedelkadir Asi, Wayne Xiong, Yue Dong, Wen Xiao **ICLR 2025, Poster**
4. Delta Attention Residuals [PDF] [Code]
   Cheng Luo*, **Zefan Cai***, Junjie Hu **arXiv 2026**
5. Multi-Head Attention Residuals [PDF] [Code]
   Cheng Luo*, **Zefan Cai***, Junjie Hu **arXiv 2026**

Open-Source Projects

1. KVCache-Factory: Unified KV Cache Compression Methods for Auto-Regressive Models [Code]
   Zefan Cai **∼1.4k GitHub stars**
2. R-KV: Decoding-Time KV Cache Implementation for Reasoning Models [Code]
   Zefan Cai **∼1.2k GitHub stars**

Work Experience

Adobe Research - San Jose, CA - Supervisor: Hao Tan May 2025 – July 2026

Research Intern: cutting the cost of carrying long context, in language models and in video generation. Built the distributed (FSDP) training and inference pipelines behind the work below, spanning language models from 124M to 8B and video diffusion at 1.3B and 14B. Also contributed to **Chimera** [PDF], a video diffusion backbone that replaces most full attention with linear-cost attention and routes tokens through a sparse mixture of experts, with the scaling recipe fitted rather than guessed: at 11B parameters with 2B active it is **7.3**× as compute-efficient in pretraining loss as a matched full-attention Wan-2.1 baseline, and it generates 30-second video after training only on 5-second clips, with no length-specific fine-tuning.

LastKV: Last Layer Key-Value Cache is All You Need [PDF]

- **Hypothesis:** Serving long context is expensive because every layer of a Transformer keeps its own copy of the cached keys and values, so the cache grows both with input length and with model depth. Prior work lets a few layers share one cache, always reading from a layer below. LastKV asks the opposite question: can every layer read the *last* layer's cache?
- **Approach:** Sharing token by token would destroy the parallelism that makes Transformer training fast, so LastKV shares once per chunk. Each chunk is processed as an ordinary Transformer and, when it finishes, keeps only its last layer's cache; every layer of every later chunk then reads that single cache through ordinary attention. Only the current chunk holds a full set, so history costs **one layer of cache instead of all of them**.
- **Validation:** Language modeling, novel view synthesis for 3D reconstruction, and autoregressive video generation.

- **Result:** Quality holds or improves in all three: language-model loss drops, long-context retrieval stays competitive on RULER, reconstructed views get sharper in PSNR, and VBench video scores hold. Continued pretraining converts existing Transformer weights into LastKV, released as one model per setting where this memory is the bottleneck: **LastKV-LLM-8B** and **LastKV-LLM-MoE-16B-A1B** for agentic workflows, keeping **one layer of cache out of 32** while retaining reasoning, agentic and long-context ability; **LastKV-VL-8B** for image and video understanding, where the visual cache dominates; **LastKV-NVS-Object-1B** and **LastKV-NVS-Scene-1B** for high-resolution 3D reconstruction; and **LastKV-Video-33B** for video generation.

Universal Test-Time Training [PDF] [Code]

- **Hypothesis:** Test-Time Training gives a model a small memory whose weights are rewritten as it reads, so earlier context can be recalled later. Each layer is normally handed its own private memory. That split is arbitrary: spending the same total budget on a single memory shared by every layer, or on one shared pool, should recall long-range context better.
- **Approach:** Spent a fixed budget either as one wide memory that every layer uses, or as a pool of memories each layer selects from, built as uTTT-Dense, TTT-MoE (pool kept private to a layer) and uTTT-MoE (pool shared across all layers). Sharing one memory across depth turns the layer index into a second time axis: the stack becomes a single recurrence over (chunk, layer) steps, and giving each layer its own memory is the special case where that sharing is switched off.
- **Validation:** Large-chunk language modeling at 124M and 760M parameters, and LVSM-style novel view synthesis.
- **Result:** Recall of long-range context improves steadily as sharing widens, with RULER retrieval accuracy rising from **8.5 to 15.5** at 124M and **23.2 to 27.8** at 760M, and rendering quality from **24.5 to 25.5 dB**.

Test-Time Generation [PDF]

- **Hypothesis:** A model that compresses long context into a fixed-size memory reads it back by regression: one key in, one value out. If the same key was stored with two different values, regression returns their average, often a value that was never stored. Answering instead with a *sample* from the distribution of values for that key should keep the alternatives apart.
- **Approach:** Replaced the read step of a LaCT-style fast-weight memory with a small conditional generative model: writes train it to denoise corrupted copies of what was stored, and a read answers a key by running a short denoising trajectory. A candidate is then scored on how plausible the value is, not only on how well the key matches.
- **Validation:** A stress test that stores two values under one key, long-context language modeling at 124M and 760M parameters, and novel view synthesis.
- **Result: Only the sampling read recovers both stored values**; no deterministic read does, including controls given the same memory capacity and the same injected noise. Freezing the sampler's randomness removes the ability, confirming that sampling is what causes it. TTG also improves on LaCT and on delta-rule baselines at both language-model scales, and raises median novel-view PSNR by **about 1 dB** at equal compute.