A Research Report on Attention Is All You Need

Author: Harshit Kumawat (Created using Agentic AI Pipeline)
Disclamer: This report is auto-generated by an AI-powered research assistant. Human verification is recommended for critical use.
Date of Creation: 2025-09-11

Abstract

This knowledge base synthesizes methods and findings from three recent works that extend, adapt, or economize Transformer-style attention across modalities and deployment constraints. The first paper investigates inference-time layer skipping in Llama-v2 models, formalizing removal of attention sublayers, MLP sublayers, or entire Transformer blocks at the tail of deep networks and measuring impacts on accuracy (OpenLLM benchmarks) and latency. It finds deeper attention sublayers are relatively redundant (e.g., removing ∼33% of attention layers in 13B Llama-v2 yields only ∼1.8% average drop) while MLP removal is more harmful; full-block dropping achieves largest speedups. The second paper introduces GLAM, a global-local attention module for image retrieval that jointly models local/global and spatial/channel attention (four attention maps: local-channel, local-spatial, global-channel, global-spatial), fusing them into a learned feature tensor that improves global descriptors and advances state-of-the-art retrieval. The third paper presents RITA, a scalable Transformer for long timeseries via group attention: dynamically cluster segments into a small number of groups, compute group-level attention with an embedding-aggregation and group-softmax scheme that preserves segment-level outputs, and use an adaptive scheduler (guided by approximation error bounds) plus batch-size modeling for efficient GPU usage; experiments show substantial speedups (up to 63×) with equal or better accuracy on long-timeseries tasks. Collectively, these works show attention can be restructured—via pruning, richer multi-dimensional attention, or coarse-grained grouping—to meet constraints of latency, memory and task demands while retaining performance.

Methodology

This research report was generated using an **Agentic AI pipeline** designed to simulate the process of academic research, writing, and review. The methodology combines automated information retrieval, structured extraction, natural language generation, and iterative critique to ensure reliability and coherence. The pipeline consists of the following components:

1. **Searcher Agent**
    - Retrieves relevant Wikipedia articles, arXiv research papers, and recent news using specialized tools.
    - Ensures coverage of both academic and practical sources within a defined time period.
2. **Extractor Agent**
    - Processes the raw sources and converts them into a structured **knowledge base (JSON format)**.
    - Summarizes each topic and subtopic into concise bullet points with references.
3. **Writer Agent**
    - Expands the structured knowledge into detailed, human-readable sections.
    - Produces coherent paragraphs while maintaining alignment with the knowledge base.
4. **Critic Agent**
    - Reviews the Writer's output against the knowledge base.
    - Detects hallucinations, unsupported claims, or factual drift.
    - Provides corrective feedback or validates correctness.
5. **Assembler Agent**
    - Integrates all validated sections into a unified document.
    - Produces the final **PDF report** with a Title page, abstract, table of contents, Main body, conclusion, references, appendix, and consistent styling.

This layered methodology ensures that the generated report is **factually grounded, logically structured, and stylistically coherent**, while also being transparent about its AI-assisted origin.

Inference-time layer skipping and sublayer removal in Transformer LLMs (Llama-v2 study)

Transformer-based large language models incur high inference cost because self-attention scales quadratically with sequence length and because modern models are deep, motivating the exploration of inference-time structural reductions to reduce compute and latency [1]. The study constructs and evaluates a set of targeted removal strategies that eliminate components at inference — either sublayers (attention or MLP) or entire Transformer blocks — in the deeper part of the network, with the explicit aim of trading off runtime against task performance on a multi-task benchmark suite [1]. Empirical analysis of internal representations shows increasing similarity between consecutive layer outputs with depth, indicating representational redundancy in deeper layers, while the final layer tends to diverge from its predecessor; this pattern motivates selective dropping of deeper components rather than uniform pruning throughout the network [1].

Quantitatively, the experiments on Llama-v2 indicate that removing a substantial fraction of deeper attention sublayers can yield large runtime benefits with only modest performance degradation: for the 13B model, removing roughly one-third of attention sublayers produced only about a 1.8% drop in average performance across the OpenLLM compilation (ARC, HellaSwag, MMLU, TruthfulQA) [1]. Ablation results further demonstrate that removing feedforward (MLP) sublayers causes larger performance declines than analogous removal of attention sublayers, implying that deeper MLP components contribute more to final task performance than attention in later layers [1]. Regarding runtime, full-block removal yields the largest wall-clock speedups, attention-sub-layer skipping provides intermediate gains, and MLP removal gives the smallest time improvements; measured token-prediction latency averaged over 1000 sequences confirms substantial improvements for skip rates in the 25–33% range [1].

The study also investigates the role of the final layer: preserving the last block while skipping earlier components sometimes mitigates performance loss, but extreme reductions that leave only the final layer can amplify upstream perturbations and produce brittle behaviour, particularly in larger models under heavy skipping [1]. As a practical recommendation, targeted skipping of deeper attention sublayers emerges as a cost-effective inference optimization that tends to preserve quality, whereas MLP removal is riskier; the authors propose light continued training or lightweight adaptation after skipping as a potential remedy for degraded robustness and identify dynamic/adaptive skipping as an open direction for further research and validation across architectures and contexts [1].

Layer skipping methodology

Layer skipping is formalized by replacing specified sublayers with the empty operation ∅ in the last k layers of a model M composed of L Transformer layers (each layer consisting of Attention_i and MLP_i), yielding distinct inference-time variants that retain the original trained weights elsewhere [1]. Three concrete variants are defined and evaluated: Mskip_MLP, in which feedforward (MLP) sublayers are removed in the final k layers; Mskip_Attention, where the attention sublayers are removed in the final k layers; and Mskip_Block, which removes entire final k Transformer blocks (both attention and MLP) [1]. Experiments are reported across configurations that correspond to keeping 66%, 75%, 90%, and 100% of the original network, which maps to the fraction of layers or sublayers retained under each skip strategy [1].

Performance under these skip strategies is assessed by comparing downstream benchmark metrics to the baseline intact model and by measuring per-token inference latency to quantify time savings. The benchmark suite comprises ARC, HellaSwag, TruthfulQA, and MMLU (the OpenLLM compilation), and latency is measured as per-token prediction time averaged over 1000 sequences for two sequence lengths (50 and 100 tokens) to capture realistic wall-clock behaviour under different input sizes [1]. These measurement protocols enable a direct comparison of accuracy versus compute trade-offs across the different skip variants and retention rates [1].

Empirical results and benchmarks

Across the OpenLLM benchmarks, skipping attention sublayers in deeper layers preserves much of the model's accuracy, with only modest average declines, whereas removing MLP sublayers produces substantially larger performance drops under comparable retention levels [1]. For example, in the Llama-v2-13B experiments, retaining 66% of the network (i.e., removing about one-third of deeper attention sublayers) corresponded to an average performance decrease of only approximately 1.8% when attention sublayers were removed, while analogous MLP removal produced larger degradations [1]. The representation analysis that motivates these interventions shows increasing cosine similarity between consecutive layers with depth, supporting the notion that deeper attention computations are relatively redundant and thus amenable to removal [1].

Timing experiments indicate that removing entire Transformer blocks yields the largest runtime improvements, followed by attention-sub-layer skipping, with feedforward-sublayer removal producing the smallest speedups; reported measurements quantify per-token latency (times reported per sequence length) and show notable percentage improvements for 50- and 100-token inputs when skipping 25–33% of components [1]. Experiments that selectively preserve the last block demonstrate that keeping the final layer while skipping earlier components often improves robustness of the reduced models, although some heavily reduced configurations (particularly for the 13B model) can still perform poorly despite last-layer preservation, suggesting limits to how much upstream computation can be removed without inducing brittle behaviour [1].

Practical implications, limitations, and future directions

The study indicates that inference-time sublayer removal, and in particular targeted skipping of deeper attention sublayers, offers a low-cost optimization for serving LLMs that can meaningfully reduce latency while incurring only modest accuracy loss, making it a practical option for latency-sensitive deployment scenarios [1]. However, several caveats apply: deeper MLP components are more sensitive than attention sublayers, so removing feedforward modules degrades downstream behaviour more severely, and extreme reductions that rely heavily on the final layer can amplify perturbations from upstream omission, producing brittle outputs [1]. As potential mitigations, the authors propose light continued training or other lightweight adaptation after skipping to recover robustness, and they identify dynamic or adaptive skipping (analogous to early-exit mechanisms) as an open research avenue that requires study of robustness and calibration [1].

Limitations of the present evaluation include its focus on Llama-v2 7B/13B models and the OpenLLM benchmark suite; the generality of the observed trends to other architectures, modalities, or very long-context regimes remains to be validated in future work [1].

Global-Local Attention Module (GLAM) for image

retrieval: spatial × channel × local × global

The Global-Local Attention Module (GLAM) is a learned attention module designed to enrich convolutional feature tensors for instance-level image retrieval by jointly applying four complementary attention types: local-channel, local-spatial, global-channel, and global-spatial. GLAM operates on a CNN feature tensor F of shape c × h × w and produces a fused feature tensor that encodes both locally pooled cues and pairwise long-range interactions across channels and spatial positions; this fused tensor is then used for global pooling to produce compact descriptors for retrieval tasks [2]. The central idea is that combining reweighting based on local pooled context with explicit pairwise modelling of interactions yields more discriminative global descriptors than using any single attention form alone [2].

Architecturally, the module splits processing into two parallel streams. The local-attention stream extracts a per-channel reweighting Al_c of shape $c \times 1 \times 1$ and a per-spatial-location map Al_s of shape $1 \times h \times w$, both designed to operate as efficient, learned reweighting mechanisms. The global-attention stream computes higher-order pairwise interactions, producing a channel–channel attention map Ag_c of shape $c \times c$ and a spatial affinity map Ag_s of shape $hw \times hw$; these capture long-range dependencies among channels and among spatial positions respectively. The outputs of the local and global streams are fused with the original feature tensor F through a learned fusion mechanism to form the final fused tensor Fgl, which is then pooled (for example using a GeM-like pooling) into a compact global descriptor for image retrieval [2].

GLAM is presented as an end-to-end-attached module for global descriptor learning that systematically covers the four quadrants of the local/global × channel/spatial taxonomy. Empirically, the paper reports that training image retrieval models with GLAM produces improved state-of-the-art global descriptors on standard retrieval benchmarks of the time, and provides experimental evidence that jointly modeling all four attention forms yields superior retrieval performance compared to methods that consider only a subset of these attention types [2]. The module is thus positioned as a general and learnable augmentation to CNN backbones for enhanced instance-level retrieval representations [2].

Local attention components and implementation

The local attention stream comprises two complementary components. The local channel attention produces a $c \times 1 \times 1$ vector by first applying global average pooling across spatial dimensions and then feeding the pooled vector into a one-dimensional convolution with kernel size k; the output is passed through a sigmoid to produce per-channel weights, thereby capturing local cross-channel context with controlled cross-channel interaction determined by k (this design is inspired by ECA-Net-style mechanisms) [2]. This lightweight channel reweighting allows the network to modulate each channel independently based on pooled contextual cues, which is particularly suitable for creating compact, discriminative embeddings for retrieval [2].

Local spatial attention is implemented by first reducing the channel dimensionality to c' via a 1×1 convolution and then applying a set of multi-scale dilated convolutions (3×3 kernels with dilations 1, 3, and 5) to capture spatial context at multiple receptive-field sizes. The multi-scale responses are combined through another 1×1 convolution and a sigmoid activation to yield a $1 \times h \times w$ spatial attention map, which reweights spatial locations according to local contextual cues. Together, the local channel and spatial components act as efficient, learned reweighting modules that emphasize informative channels and positions based on locally pooled context, making them well-suited for enhancing retrieval embeddings [2].

Global attention components and fusion

The global attention stream explicitly models pairwise interactions at both the channel and spatial levels. Global channel attention computes pairwise interactions between channels to produce a $c \times c$ attention map; this map is normalized with a softmax-like mechanism and subsequently used (often with a gating mechanism) to reweight channel activations, thereby capturing long-range dependencies and correlations across feature channels. This pairwise modelling enables the representation to account for relationships between feature detectors that are not captured by local pooling alone [2].

Global spatial attention computes an $hw \times hw$ affinity matrix that lets every spatial location attend to every other location in a non-local fashion, producing spatial reweighting that captures long-range spatial context and relationships that are critical for discriminative global descriptors. Such non-local interactions complement the local spatial module by introducing global positional dependencies into the pooled representation [2].

The outputs of the local and global streams (Fl and Fg) are merged with the original feature tensor F through a learned fusion mechanism to form the final fused tensor Fgl; this fusion integrates locally pooled cues and pairwise long-range interactions across both channel and spatial dimensions. The fused tensor Fgl is then aggregated by a global pooling operation (for example, GeM or a similar pooling strategy) into a compact global descriptor that is used for instance-level image retrieval [2].

Empirical findings and comparative positioning

Empirical evaluations reported for GLAM indicate that jointly employing local/global and spatial/channel attentions produces improved retrieval performance compared to methods that use only a subset of these attention forms, yielding state-of-the-art global descriptors on standard benchmarks at the time of the study. The experiments demonstrate that the richer attention modelling afforded by GLAM—combining both pairwise interactions and locally pooled cues across channel and spatial dimensions—strengthens compact global descriptors for instance-level retrieval [2].

The work also offers a taxonomy of prior attention methods along the local/global $\times$ spatial/channel axes and situates GLAM as the first learned, end-to-end module to systematically and jointly employ all four attention types for global descriptor learning. This positioning emphasizes GLAM's contribution as a unifying, systematic approach that extends prior channel-only or spatial-only attention mechanisms by integrating them into a single module tailored for retrieval tasks [2].

Group attention for timeseries analytics (RITA): scalable approximate self-attention

Transformer self-attention incurs quadratic time and memory cost in sequence length n, which limits its applicability to long timeseries composed of tens of thousands of segments and similar long-range problems [3]. RITA addresses this scalability bottleneck by replacing elementwise attention with a grouped computation: input timeseries are partitioned into segments and similar segments are dynamically clustered into N groups, so that attention is computed at the group granularity rather than across all n segments. This grouping yields a compressed $N \times N$ attention matrix and substantially reduces both compute and memory requirements compared to the canonical $n \times n$ attention form [3].

To preserve per-segment expressivity while operating on compressed group-level attention, RITA combines an embedding aggregation strategy with a customized group-aware softmax. The aggregation strategy computes Queries, Keys, and Values at the segment level, then aggregates these into per-group representations and performs attention on the compact group matrix; a redistribution mechanism reconstructs distinct segment-level outputs from the group results without materializing the full $n \times n$ attention matrix. The theoretical formulation supports correctness guarantees showing that, under the proposed scheme, the original attention matrix can be effectively restored within provable bounds, so no uncontrolled embedding error is introduced beyond these bounds [3].

RITA further introduces an adaptive scheduler and batching strategy to maximize runtime efficiency while enforcing a user-specified approximation tolerance. The scheduler adaptively selects the number of groups N per layer and iteration, iteratively merging groups until a provable approximation error bound is met; concurrently, RITA models and adapts the attainable batch size B as a function of N to improve GPU utilization by learning a mapping from sampled $<N,B>$ pairs. Implementation-level choices—including a GPU-friendly parallel grouping algorithm, group-level attention kernels, and dynamic group-merging logic—enable practical deployment. Empirically, across multiple public timeseries benchmarks and a clinical EEG dataset, RITA achieves substantial speedups (reported between $4\times$ and as high as $63\times$) over baseline attention mechanisms while matching or improving predictive accuracy on forecasting, classification, and anomaly-detection tasks, demonstrating that clustered attention with a tunable approximation bound makes Transformer-style models practical for long-timeseries analytics [3].

Group attention algorithmic components

RITA first chunks the input timeseries into segments and clusters segments with similar embeddings into N groups; this clustering is dynamic and may vary by layer and iteration, allowing the grouping to adapt to evolving representations within the model. The method computes Queries, Keys, and Values at the original segment granularity but aggregates these into per-group representations so that subsequent attention operates on a compressed group-level attention matrix A_group of size $N \times N$ rather than the full $n \times n$ matrix, thereby reducing computational and memory overhead [3].

A core component is the embedding aggregation and reconstruction pipeline: after computing group-level attention and group outputs, RITA applies an aggregation strategy to redistribute group outputs back to individual segments so that each original segment receives a distinct embedding without requiring expansion to the full attention matrix. To enable correct redistribution, RITA replaces the classical softmax with a grouping-aware normalization ("group softmax") that preserves the necessary normalization properties across grouped keys and queries; this grouping-aware softmax ensures attention weights are correctly redistributed to segment-level outputs while operating only on the compressed representation [3].

Adaptive scheduler, error bounds, and batching

RITA uses an adaptive scheduler that begins with a comparatively large number of groups N and iteratively merges similar groups while monitoring a provable approximation error bound; merging decisions are made when two groups are determined mergeable under the user-specified tolerance, so the scheduler balances approximation quality against computational savings. These error bounds are formalized to guide merging decisions, enabling the system to maintain a user-specified approximation quality while progressively reducing computational load [3].

Concurrently, RITA models how available memory and compute change as groups are merged and leverages a learned, simple predictive mapping from N to an effective batch size B. By sampling a small set of $<N,B>$ pairs and learning this mapping offline or during warm-up, the system increases batch sizes opportunistically as N is reduced, improving throughput and GPU utilization without violating the approximation constraints enforced by the scheduler [3].

Empirical scalability and accuracy outcomes

Empirical evaluation on multiple public timeseries datasets and a clinical EEG dataset shows that RITA outperforms existing timeseries attention approximations in predictive accuracy while substantially lowering runtime and memory footprints, demonstrating both efficiency and effectiveness on forecasting, classification, and anomaly-detection tasks [3]. The method scales to very long sequences (empirically in the regime of ≥ 2000 segments), a region where standard self-attention becomes infeasible, and reports speedups up to $63\times$ over canonical attention baselines in favorable settings [3].

In practice, choosing the number of groups N controls the trade-off between computational speed and approximation error; RITA's scheduler automates this trade-off on a per-layer and per-iteration basis, meeting user-specified error bounds while exploiting increased parallelism through larger batch sizes when memory is freed. These empirical and practical characteristics suggest that group attention extends Transformer-style architectures to long-timeseries analytics by offering a tunable approximation mechanism that yields large efficiency gains with formally guided error control [3].

Conclusion

Across language, vision, and timeseries domains, attention-centric modifications offer complementary routes to practical Transformers: (1) inference-time skipping (preferentially of deeper attention sublayers or whole blocks) provides a low-cost latency reduction with modest accuracy loss and is attractive for serving large LLMs, (2) richer attention factorization (GLAM) that jointly models local/global and spatial/channel interactions yields stronger compact descriptors for image retrieval, and (3) coarse-grained group attention (RITA) enables provably approximate, highly efficient self-attention for very long sequences by clustering and adaptive scheduling. Each approach trades approximation or model capacity for compute/memory savings; combining these ideas (e.g., adaptive skipping informed by representation similarity, richer local/global reweighting inside grouped attention) and validating cross-architecture/generalization remain promising directions. Careful attention to last-layer robustness, approximation error bounds, and small adaptation training can help recover or preserve performance when applying these optimizations in production.

References

[1] G. Tyukin, G. J. Dovonon, J. Kaddour, and P. Minervini, "Attention Is All You Need But You Don't Need All Of It For Inference of Large Language Models", arXiv, [Online]. Available: https://arxiv.org/abs/2407.15516

[2] C. H. Song, H. J. Han, and Y. Avrithis, "All the attention you need: Global-local, spatial-channel attention for image retrieval", arXiv, [Online]. Available: https://arxiv.org/abs/2107.08000

[3] J. Liang, L. Cao, S. Madden, Z. Ives, and G. Li, "RITA: Group Attention is All You Need for Timeseries Analytics", arXiv, [Online]. Available: https://arxiv.org/abs/2306.01926

Appendix A: Key points of Report

1. **Inference-time layer skipping and sublayer removal in Transformer LLMs (Llama-v2 study):**

    - Motivation: inference cost of Transformer-based LLMs scales poorly due to quadratic self-attention cost and deep architectures; layer skipping at inference is explored to reduce compute and latency.

    - Model variants: construct three skip strategies for a model M with L layers (each layer = Attention_i + MLP_i): (a) Mskip_MLP — remove MLP sublayers in the last k layers, (b) Mskip_Attention — remove attention sublayers in the last k layers, (c) Mskip_Block — remove entire final k Transformer blocks.

    - Empirical observation on representation redundancy: cosine similarity between consecutive Llama-v2 layers increases with depth (features become more similar), except the final layer which diverges most from the previous one; this motivates dropping deeper components.

    - Key quantitative result: for Llama-v2 13B, removing ∼33% of attention sublayers yields only a ∼1.8% drop in average OpenLLM benchmark performance (ARC, HellaSwag, MMLU, TruthfulQA), showing attention sublayers in deeper layers are relatively redundant.

    - Ablations: dropping MLP (ffwd) sublayers causes larger performance degradation than dropping attention sublayers, indicating deeper MLPs have a larger contribution to final performance than attention in later layers.

    - Last-layer importance: preserving the last layer when skipping earlier layers affects results—keeping the final block sometimes mitigates degradation, but skipping everything except the last layer can amplify errors because the final feedforward must process perturbed upstream representations.

    - Compute vs. accuracy trade-off: full-block removal gives the largest runtime improvement, followed by attention sublayer skipping; feedforward sublayer skipping yields smaller runtime gains. Measured token-prediction latency (averaged over 1000 sequences) shows substantial wall-clock improvements for 25–33% skipping.

    - Practical implications: targeted skipping of deeper attention sublayers is a low-cost inference optimization for LLMs that preserves quality; MLP removal is riskier. Small additional fine-tuning after skipping (continued training) is proposed as a potential remedy for degraded robustness.

    - Formalization: define skipping variants by replacing the relevant sublayer(s) with ∅ in the last k layers; evaluate performance vs. baseline on downstream benchmarks.

    - Variants: Mskip_MLP (remove MLP in last k), Mskip_Attention (remove Attention in last k), Mskip_Block (remove whole blocks), with experiments across keeping 66%, 75%, 90%, 100% of original network.

    - Measurement protocols: report benchmark metrics (ARC, HellaSwag, TruthfulQA,

MMLU) and measure per-token inference latency averaged over 1000 sequences for sequence lengths 50 and 100 to quantify time improvements.

- Benchmarks used: OpenLLM compilation including ARC, HellaSwag, MMLU, TruthfulQA to measure multi-task LLM performance.
- Performance trends: skipping attention sublayers preserves much of accuracy (average falls modestly), while skipping MLP sublayers leads to larger drops; examples: 13B-66% (keeping 66% of network) shows only ~1.8% average drop when removing attention vs larger drops when removing MLP.
- Timing results: for Llama-v2-7B, skipping full layers produced the most significant time savings, attention sublayer skipping gave intermediate speedups, feedforward skipping gave the least; tables show time(s) $\times$ 10^2 per token and percentage improvements for 50/100-length inputs.
- Last-layer inclusion experiments: keeping the final block while skipping earlier components often improves robustness of reduced models, but sometimes extremely reduced configurations still perform poorly (especially for 13B when heavy skipping but last-layer preserved).
- Implication: inference-time sublayer removal (especially attention sublayers in deeper layers) is a cost-effective optimization for serving LLMs with lower latency and modest accuracy loss.
- Caveats: MLP components are more sensitive in deeper layers and removing them degrades behaviour more than attention removal; last-layer processing can amplify upstream perturbations making some skip patterns brittle.
- Suggested mitigations: small continued training or lightweight adaptation after skipping could recover robustness; dynamic or adaptive skipping (early-exit style) remains an open avenue tied to robustness and calibration studies.
- Limitations: experiments focused on Llama-v2 7B/13B and the OpenLLM benchmark; behaviour on other architectures, modalities, or very long-context regimes remains to be validated.

2. **Global-Local Attention Module (GLAM) for image retrieval: spatial $\times$ channel $\times$ local $\times$ global:**
   - Core idea: GLAM combines four attention types—local-channel, local-spatial, global-channel, global-spatial—applied jointly to a CNN feature tensor to produce a fused feature tensor used for global pooling and image retrieval.
   - Architecture overview: input feature tensor F ($c \times h \times w$) is processed by two parallel streams: local attention (extracts Al_c: $c \times 1 \times 1$, and Al_s: $1 \times h \times w$) and global attention (extracts Ag_c: $c \times c$ and Ag_s: $hw \times hw$), their outputs are fused with F to produce Fgl before pooling.
   - Local attention design: local channel attention uses GAP + 1D conv (kernel k controls cross-channel interaction) followed by sigmoid (inspired by ECA-Net); local spatial attention uses multi-scale/dilated convolutions to capture contextual spatial cues.
   - Global attention design: global channel attention models interactions between

channels (produces $c \times c$ map) using pairwise computations and a softmax-like normalization; global spatial attention models interactions across spatial positions producing $hw \times hw$ maps (non-local style).

- Empirical contribution: GLAM is trained and evaluated for instance-level image retrieval, yielding improved state-of-the-art global descriptors on standard benchmarks and providing experimental evidence that combining all four attention forms is beneficial.
- Positioning vs prior work: prior methods typically address only one or two attention forms (e.g., channel-only, spatial-only, local or global); GLAM is a systematic, learned, end-to-end-attached module for global descriptor learning.
- Local channel attention: obtain $c \times 1 \times 1$ via global average pooling then 1D conv (kernel size k) → sigmoid to produce per-channel weights; captures local cross-channel context.
- Local spatial attention: reduce channels to c' via 1×1 conv, then apply multi-scale (dilated) convolutions (kernels 3×3 with dilations 1,3,5) and combine via 1×1 conv + sigmoid to obtain $1 \times h \times w$ spatial map; captures multi-scale local spatial context.
- Design role: local modules weigh elements independently based on pooled/ contextual cues; they are learned and operate as efficient reweighting mechanisms suited for retrieval embeddings.
- Global channel attention: compute interactions between channels leading to a $c \times c$ attention map (pairwise channel interactions) and apply normalization (softmax-like) and gating to reweight channel activations.
- Global spatial attention: compute $hw \times hw$ affinity to allow every spatial location to attend to every other (non-local mechanism) producing spatial reweighting that captures long-range context beneficial for discriminative global descriptors.
- Fusion: outputs of local and global streams (Fl and Fg) are combined with original features F through a learned fusion mechanism to produce a final feature tensor Fgl which is pooled (e.g., GeM or similar) into a compact global descriptor for retrieval.
- GLAM's empirical results show that combining local/global and spatial/channel attentions improves retrieval performance vs. methods that use only a subset; experiments demonstrated state-of-the-art performance on standard retrieval benchmarks at the time of publication.
- The paper provides a taxonomy of prior attention methods (local/global × spatial/channel) and situates GLAM as the first to jointly employ all four forms in a learned module applied to global descriptor learning.
- Implication: richer attention modelling (capturing both pairwise interactions and locally pooled cues across both spatial and channel dimensions) strengthens compact global descriptors for instance-level retrieval.

3. **Group attention for timeseries analytics (RITA): scalable approximate self-attention:**

- Problem: canonical Transformer self-attention has O(n^2) time and memory in sequence length n, limiting applicability to long timeseries (e.g., tens of thousands of segments).
- Key idea: group attention clusters segments (sequence chunks) dynamically into N groups (segments in same group are similar) and computes attention at the group granularity, yielding a compressed group-level attention matrix and drastically reducing compute/memory.
- Embedding quality and correctness: introduce an embedding aggregation strategy and a customized group softmax that allow producing distinct embeddings per original segment while operating on the compressed attention matrix; theoretical results show equivalence to restoring the original attention matrix under the scheme (i.e., no additional embedding error beyond provable bounds).
- Adaptive scheduler and batching: RITA adapts the number of groups N per layer and iteration via a scheduler guided by a user-specified approximation error bound; it also models and adapts batch size B as a function of N to maximize GPU utilization (learned mapping from sampled $<$N,B$>$ pairs).
- Implementation details: GPU-friendly parallel grouping algorithm, group-level attention computations, and a dynamic strategy to merge groups to meet approximation bounds while minimizing computation.
- Empirical results: across public timeseries benchmarks and a real EEG dataset, RITA achieves speedups between $4\times$ and up to $63\times$ over baseline attention mechanisms while matching or improving predictive accuracy on forecasting, classification and anomaly tasks.
- Implication: group attention makes Transformer-style models practical for long-timeseries analytics by trading a tunable approximation (controlled error bound) for large efficiency gains; it generalizes the attention mechanism to clustered computations.
- Clustering: chunk input timeseries into segments and cluster segments with similar embeddings into N groups (dynamic per layer/iteration).
- Compressed attention: compute Queries/Keys/Values per segment, aggregate per-group, compute group-level attention matrix A_group ($N\times N$) and use an embedding aggregation strategy to reconstruct segment-level outputs without materializing full $n\times n$ attention.
- Custom group softmax: replace classical softmax with a grouping-aware normalization enabling correct redistribution of attention weights back to segment-level outputs without expanding to the full matrix.
- Adaptive scheduler: starts with a large N and iteratively merges similar groups guided by a provable approximation error bound, deciding when two groups are mergeable to satisfy user tolerance.
- Batch size adaptation: learns a simple predictive model mapping N→B (batch size) using a small set of measured $<$N,B$>$ pairs to scale batch size as memory is freed when N reduces, improving throughput.

- Theoretical guarantees: provide error bounds on attention approximation that drive merging decisions so RITA maintains a user-specified approximation quality while reducing computational load.
- Benchmarks: experiments on multiple public timeseries datasets and a clinical EEG dataset demonstrate that RITA outperforms state-of-the-art timeseries attention approximations in accuracy while substantially lowering runtime and memory.
- Scalability: RITA is shown to scale to very long sequences (≥ 2000) where standard self-attention becomes infeasible; reported speedups reach up to $63\times$ compared to canonical attention baselines in favorable settings.
- Practical note: choosing N trades speed vs approximation error; the scheduler automates this trade-off per layer/iteration to meet constraints while leveraging increased parallelism via larger batch sizes when possible.

Appendix B: Recent News

- **Paper Walkthrough: Attention Is All You Need - Towards Data Science**
  - Towards Data Science - Published on Sun, 03 Nov 2024 07:00:00 GMT
  - For more details click here.
- **It turns out that attention is not all you need in LLMs - Substack**
  - Substack - Published on Tue, 10 Dec 2024 08:00:00 GMT
  - For more details click here.
- **What is a Transformer Model? - IBM**
  - IBM - Published on Fri, 28 Mar 2025 07:00:00 GMT
  - For more details click here.
- **Attention is all you need — until you need people - Medium**
  - Medium - Published on Thu, 29 May 2025 07:00:00 GMT
  - For more details click here.
- **What Researchers Behind OpenAI's Foundational Framework Are Doing Today - CCN.com**
  - CCN.com - Published on Sun, 13 Oct 2024 07:00:00 GMT
  - For more details click here.
- **ChatGPT's success could have come sooner, says former Google AI researcher - Ars Technica**
  - Ars Technica - Published on Thu, 14 Nov 2024 08:00:00 GMT
  - For more details click here.
- **AI Explained: How Attention Mechanisms Took Over AI - PYMNTS.com**
  - PYMNTS.com - Published on Mon, 09 Dec 2024 08:00:00 GMT
  - For more details click here.
- **A look under the hood of transfomers, the engine driving AI model evolution - VentureBeat**
  - VentureBeat - Published on Sat, 15 Feb 2025 08:00:00 GMT
  - For more details click here.
- **What If We're Doing AI All Wrong? - Bloomberg**
  - Bloomberg - Published on Wed, 03 Sep 2025 09:00:21 GMT
  - For more details click here.
- **'Attention is All You Need' Author Suggests LLMs 'Reflect' in Pre-Training - Analytics India Magazine**
  - Analytics India Magazine - Published on Mon, 14 Apr 2025 07:00:00 GMT
  - For more details click here.

- **You’re Being Alienated From Your Own Attention - The Atlantic**
  - The Atlantic - Published on Wed, 22 Jan 2025 08:00:00 GMT
  - For more details click here.
- **What is an attention mechanism? - IBM**
  - IBM - Published on Wed, 11 Dec 2024 16:27:47 GMT
  - For more details click here.
- **Tracing the Transformer in Diagrams - Towards Data Science**
  - Towards Data Science - Published on Thu, 07 Nov 2024 08:00:00 GMT
  - For more details click here.
- **What is self-attention? - IBM**
  - IBM - Published on Tue, 18 Feb 2025 08:00:00 GMT
  - For more details click here.
- **Hands-On Attention Mechanism for Time Series Classification, with Python - Towards Data Science**
  - Towards Data Science - Published on Fri, 30 May 2025 07:00:00 GMT
  - For more details click here.
- **What is grouped query attention (GQA)? - IBM**
  - IBM - Published on Fri, 06 Dec 2024 08:00:00 GMT
  - For more details click here.
- **Increasing Transformer Model Efficiency Through Attention Layer Optimization - Towards Data Science**
  - Towards Data Science - Published on Mon, 18 Nov 2024 08:00:00 GMT
  - For more details click here.
- **DeepSeek-V3 Explained 1: Multi-head Latent Attention - Towards Data Science**
  - Towards Data Science - Published on Fri, 31 Jan 2025 08:00:00 GMT
  - For more details click here.
- **How I Studied LLMs in Two Weeks: A Comprehensive Roadmap - Towards Data Science**
  - Towards Data Science - Published on Fri, 18 Oct 2024 07:00:00 GMT
  - For more details click here.
- **“Attention is all you need” - The Ken**
  - The Ken - Published on Sun, 24 Aug 2025 07:00:00 GMT
  - For more details click here.