Attention Mechanism Enhancements in Modern Machine Learning

Bijo Benjamin Thomas
University of Cincinnati
USA

Abstract

Attention mechanisms have become a cornerstone of modern machine learning, enabling models (especially Transformers) to weigh the importance of different input elements dynamically. Since the introduction of the Transformer and its Multi-Head Self-Attention (MHA) in 2017, attention-based models have achieved state-of-the-art results in NLP, computer vision, and other domains. However, the flexibility of attention comes at a high computational and memory cost: naïve self-attention requires $O(n^2)$ operations and memory with respect to sequence length n. This poses challenges for long sequences and efficient inference, motivating a surge of research into faster and more memory-efficient attention mechanisms. In this article, we review the fundamentals of attention and highlight recent advances aimed at improving attention speed and efficiency, including Multi-Query and Grouped-Query Attention, Rotary Positional Embeddings (RoPE), Query-Key Normalization (QK Norm), hybrid Transformer models, and high-performance implementations like FlashAttention.

Keywords: Attention mechanisms, Transformers, Multi-Query Attention, FlashAttention, Rotary Positional Embeddings

1. Introduction

In a self-attention layer, each token in a sequence attends to all other tokens to compute contextualized representations. For a sequence of length n, the attention mechanism computes attention weights using dot-products between a query vector and a set of key vectors, then produces an output by weighted summation of corresponding value vectors [3], [4]. The ubiquitous Scaled Dot-Product Attention uses a softmax over these dot-products (scaled by $1/\sqrt{d}$) to generate weights [5]. In Multi-Head Attention (MHA), multiple attention "heads" (parallel attention computations with different learned projections) allow the model to attend to information from different representation subspaces[3], [7]. Formally, for each head h:

$$\text{Attention\_h}(Q_h,K_h,V_h) = \text{softmax}(Q_h.K_h^T/\sqrt{d_h}).V_h$$

Where Q, K, and V are derived from the learned projections of input X.

$$Q_h = X.W_h^Q$$
$$K_h = X.W_h^K$$
$$V_h = X.W_h^V$$

The head outputs are concatenated and transformed to produce the final attention output. This mechanism allows each token to adaptively focus on relevant context, which has proved exceptionally powerful in language modeling and other tasks.

Computational Cost

The main drawback of vanilla self-attention is its quadratic complexity in n. Computing the n×n attention matrix and multiplying by V is $O(n^2 \cdot d)$ for each head (or $O(n^2)$ if d scales with n). Memory usage is also quadratic in n because the attention matrix (and often the key-value representations) must be stored [2]. This becomes prohibitive for long sequences (e.g., long documents, high-resolution images, or DNA sequences), where standard Transformers struggle beyond a few hundred or thousands of tokens. For example, the original Transformer was limited to sequences of ~512 tokens. Moreover, during autoregressive decoding (generating one token at a time), each new token's self-attention must process all previous tokens, leading to latency that grows with sequence length.

Efficient Attention Variants and Optimizations

Researchers have developed a wide range of strategies to make attention more efficient, either by approximating the attention computation or by optimizing the exact computation. Early efforts introduced sparse or localized attention patterns, limiting each token to attend to a subset of the sequence. Models like Longformer and BigBird use fixed windows or block patterns (with occasional global tokens) to bring complexity down to O(n) [2]. These approaches enable processing sequences 8× longer than standard Transformers on the same hardware, achieving new state-of-the-art results on tasks like long-document question answering and summarization. Another line of work uses low-rank or

 | DOI: 10.20533/ijicr.2042.4655.2025.0165

kernel-based approximations: for example, Linformer projects keys and values to a lower-dimensional subspace to reduce memory, and Performer replaces the softmax with a feature map, enabling linear complexity attention [5]. While such approximations can drastically cut complexity, they may introduce some degradation in model accuracy if not carefully applied, especially on tasks requiring global context.

Instead of approximating the attention computation, recent research (especially in the last 2–3 years) has focused on exact attention optimizations that maintain full accuracy while improving speed or memory use. Below, we discuss several leading developments in this area (see Figure 1).

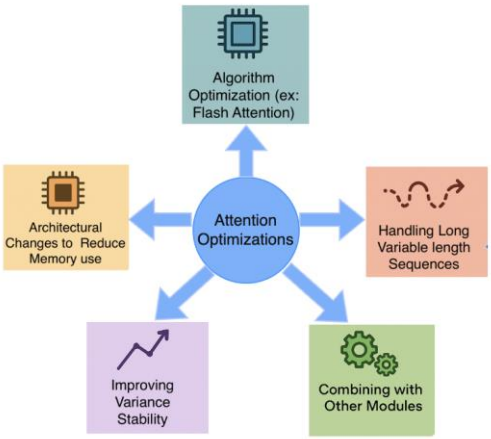

Figure 1. Attention variants and optimizations

- Multi-Query Attention (MQA) and Grouped-Query Attention (GQA) – architectural changes to reduce memory bandwidth and caching costs during decoding.
- High-performance attention implementations (FlashAttention) – algorithmic optimizations to use hardware more efficiently.
- Positional encoding improvements (RoPE) – better handling of long sequences without extra cost.
- Normalization and stability tweaks (QK norm) – making training of attention more stable for faster convergence.
- Hybrid architectures – combining attention with other sequence modules (like state-space models) to get the best of both.

Each of these provides significant speed-ups or memory savings, often with minimal impact on task performance. We now examine them in detail, correcting some common misconceptions and highlighting results.

2. Multi-Query Attention (MQA): Reducing Memory Redundancy

One major source of inefficiency in MHA is that each of the H heads maintains its own set of key and value vectors. This means at inference time, a decoder must store H key-value pairs per past token (the KV cache), and load all H of them from memory at each step. Multi-Query Attention (MQA) addresses this by sharing the key and value across all heads [6]. In an MQA layer, there are still H independent query projections (so each head can attend differently), but only one key and one value projection for the entire layer. In effect, all heads attend to the same keys and values, rather than each head attending to their own. This drastically reduces the size of the KV cache and memory bandwidth requirements for attention.

MQA was shown to enable 10–100× smaller key-value caches and about 12× faster decoder inference, with only minor drops in model quality. Noam Shazeer's 2019 paper introducing MQA demonstrated that Transformers with MQA decode much faster while incurring only a small accuracy degradation compared to MHA [3]. For instance, when converting a large Transformer to use one shared KV head, inference throughput improved an order of magnitude [6]. Google's PaLM language model adopted MQA in its decoder layers to cut memory usage, finding no significant performance loss at scale [7]. By reducing memory load, MQA can also speed up training by enabling larger batch sizes (fitting more sequences in memory) [6].

The main trade-off is a reduction in modeling flexibility. With shared keys/values, the heads have less capacity to represent diverse information, which can hurt accuracy. Empirically, MQA can cause a measurable drop in performance on certain tasks compared to full MHA[6,7]. For example, Ainslie et al. [6] reported that a T5-XXL model using MQA underperformed an MHA baseline on summarization benchmarks, even when model size was kept constant. Similarly, Llama-2 experiments showed that an MQA variant (with feed-forward layers scaled up to compensate for fewer parameters) still had slightly lower average accuracy than the MHA version [7]. Another practical limitation is that one cannot simply retrofit a pre-trained MHA model to MQA without retraining; the change in architecture requires fine-tuning or training from scratch to adapt [6]. Finally, in multi-GPU training with tensor-parallelism (where different heads reside on different GPUs), MQA forces those GPUs to duplicate the same key/value data, leading to some wasted computation [6], [7] (though it's still more memory-efficient overall than MHA).

Despite these caveats, MQA can be effective where speeding up autoregressive decoding is of higher priority compared to generation quality. When quality requirements are high, developers often prefer

 | DOI: 10.20533/ijicr.2042.4655.2025.0165

a middle ground, which brings us to grouped-query attention.

3. Grouped-Query Attention (GQA): A Balance Between Speed and Accuracy

Grouped-Query Attention (GQA) is a generalization of MQA introduced by Ainslie et al. [6] to strike a balance between the efficiency of MQA and the expressiveness of MHA. The idea is simple: instead of one global key/value or H independent ones, use G groups of query heads, each group sharing its key and value. In other words, partition the H heads into G groups; heads in the same group use a shared key and value projection. GQA with G=1 is equivalent to MQA, and G=H recovers standard MHA [6]. Intermediate values of G allow a trade-off: as G increases, you have more distinct key/value sets (improving capacity), but also more memory use.

GQA reduces the KV cache size by a factor of G compared to MHA (and requires loading G sets of keys/values instead of H each time). In practice, using even a moderate number of groups can yield almost the same speedup as MQA while recovering most of the lost accuracy [8]. Ainslie et al. showed that a Transformer could be up-trained (post-trained) to GQA and achieve quality very close to MHA with inference speed nearly as fast as MQA. Larger models benefit particularly from GQA: as model size grows, the wasted memory in MHA (duplicating large KV tensors per head) becomes more severe, so grouping heads provides a favorable speed-quality trade-off [8]. Moreover, unlike MQA, it's possible to convert a pre-trained multi-head model into a GQA model (by merging head weights) and then fine-tune, rather than training from scratch [8]. This makes GQA a practical retrofit to existing models (see Figure 2).

Figure 2. MHA vs. MQA vs. GQA [7]

Owing to these advantages, GQA has been rapidly adopted in state-of-the-art LLMs. Meta first used GQA in LLaMA-2 (2023) – for the 70B model, they set 8 query groups (instead of 32 heads, i.e., groups of 4 heads), and for smaller 7B/13B, they effectively used 1 group (MQA) [7]. The open-source Mistral 7B (2023) model similarly employs GQA for efficient generation[6]. IBM's recent 2024 flagship models (Granite series) also use GQA to enable faster inference [6]. Empirical results indicate that with a reasonable number of groups (e.g., 8), the accuracy loss is minimal – GQA-LLaMA70B performs on par with a full-attention model on many benchmarks, while significantly cutting memory use [8]. Thus, GQA is emerging as a best-of-both-worlds solution, bringing near-MHA quality with MQA-level speedups.

4. Fast Attention via Software/Hardware Optimization: FlashAttention

A breakthrough in exact attention efficiency came with FlashAttention, an algorithm that computes the same output as standard attention but uses memory much more efficiently [5]. The key insight by Dao et al. [5], [9] was that reading/writing the large n×n attention matrix to global memory is a bottleneck. FlashAttention instead tiles the attention computation to keep intermediate results in high-speed on-chip memory (SRAM), avoiding materializing the full matrix in GPU memory. It computes attention in small blocks that fit in the cache, performing the softmax reduction on the fly. This yields linear memory usage in sequence length (only a small buffer is needed) instead of quadratic. By trading some extra FLOPs for far fewer memory access, FlashAttention achieves significantly higher speed on long sequences.

In benchmarks, FlashAttention attained up to 2-4x faster runtime than the best CUDA implementations of standard attention for sequence lengths 128 to 2048 [5], [9], and over 10× memory footprint reduction. Notably, it maintains mathematical exactness, there is no approximation error compared to normal attention, so model quality is unaffected. This has made it a drop-in replacement in many libraries. FlashAttention enabled training Transformers on sequences as long as 16k–64k tokens without running out of memory, something previously impractical [5]. For example, using FlashAttention, one can train GPT-2 with a 4× longer context length in the same memory budget, achieving better perplexity on long-context tasks [13].

Building on this, FlashAttention-2 [9] further optimizes thread parallelism and latency, roughly doubling the speed of FlashAttention v1. Combined, these innovations have made exact self-attention competitive with, or even faster than, many approximate methods for typical sequence lengths. As a result, frameworks like PyTorch and Jax have integrated FlashAttention or similar kernels, and many recent LLMs leverage these optimizations under the hood. In summary, FlashAttention demonstrates that we can often have our cake and eat it too: preserving the full accuracy and flexibility of softmax attention while dramatically improving efficiency. It exemplifies how algorithmic refinements (tiling, kernel fusion, better memory layouts) can complement architectural changes like GQA.

 | DOI: 10.20533/ijicr.2042.4655.2025.0165

5. Rotary Positional Embeddings (RoPE): Extending Attention's Context

An "efficiency" improvement in terms of speed, Rotary Position Embeddings (RoPE) have become an important technique to enhance Transformer attention for longer sequences without increasing cost. Proposed in 2021 [10], RoPE is a form of relative positional encoding that multiplies queries and keys by a rotation matrix that depends on their positions [10]. This imparts a phase to the vector components such that the dot product QK^T effectively includes relative position information (difference in angles). Crucially, RoPE encodes absolute positions implicitly yet allows the model to generalize to positions beyond those seen in training by extrapolating the rotations [10] (see Figure 3).

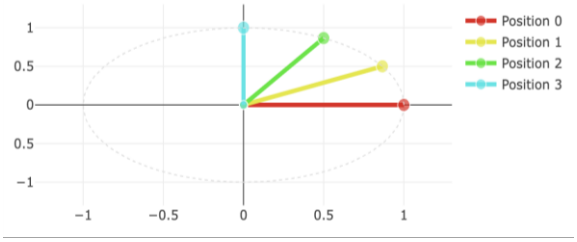

Figure 3. Vector Rotation by Position

Traditional absolute position embeddings (learned or sinusoidal) fix a maximum sequence length and don't generalize to longer sequences than those trained on. RoPE, in contrast, enables flexibility of sequence length – models trained on (say) 2048 tokens can seamlessly handle 4096 or more at inference with graceful degradation [10]. This is because the rotation for position n+1 is a predictable extension of that for n. Empirically, RoPE-enhanced Transformers (called "RoFormers") achieved better performance on long text classification tasks than models with other positional encodings. The rotary embedding causes attention scores to naturally decay for distant positions (the dot products decrease as the relative angle grows), which provides an inductive bias for long sequences [10]. RoPE also works nicely with efficient attention mechanisms – for example, it can be integrated into linear attention variants to provide relative position information without sacrificing linear complexity [10] (see Figure 4).

RoPE has been adopted in many recent large language models (GPT-NeoX, EleutherAI's models, and Meta's LLaMA family all use RoPE instead of fixed sinusoidal embeddings). With RoPE, transformers can extrapolate to very long sequences even if trained on shorter ones, all without additional computation [10]. In summary, RoPE doesn't make attention faster per se, but it improves the performance of attention on longer inputs and removes the need for costly retraining to extend context length. It is an

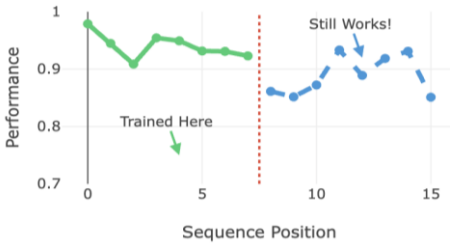

Figure 4. RoPE is not limited by the token limit during training

elegant example of a positional encoding that aligns well with the structure of dot-product attention.

6. Query-Key Normalization (QK Norm): Stabilizing Attention for Better Training

Another recent insight is that the scale of the query and key vectors in attention can impact training stability and performance. QK Normalization (QK Norm) refers to normalizing the query and key vectors (usually to unit length) before computing attention scores [11]. Instead of the raw $q \cdot k^T/\sqrt{d}$, the dot product is effectively turned into a cosine similarity (often with a learnable scale parameter) [11]. This technique, introduced in 2020 and refined in later work, aims to prevent the softmax attention from becoming too peaked or "saturated" early in training [11], [4].

In standard attention, nothing explicitly constrains the magnitude of queries and keys. During training, the model can increase the L^2 norm of q and k vectors, which in turn increases the dot-product logits uniformly [4]. Since softmax is invariant to adding the same constant to all logits, the model doesn't "see" an issue with making all logits large, except that eventually numerical precision and gradient magnitudes become problematic. This can lead to an effect called "attention entropy collapse", where the softmax outputs become nearly one-hot (one very large logit dominates) [4]. When that happens, training can become unstable or diverge (loss spikes) because gradients through the nearly discrete attention are ill-behaved [4]. This issue is exacerbated in very deep or multi-modal transformers. For example, experiments showed an 8B-parameter vision Transformer (ViT) exhibited diverging loss after a few thousand steps due to unnormalized Q, K causing logits to blow up to 50,000+ in magnitude [4].

By ℓ_2 normalizing queries and keys, the dot-product is bounded between -1 and 1 (before scaling), eliminating the incentive to simply escalate norms. Any growth in logits must come from changing the angles (i.e., better feature representations) rather than just norms. This has been found to greatly stabilize training, especially for models with heterogeneous inputs (text+image tokens together) where the

 | DOI: 10.20533/ijicr.2042.4655.2025.0165

distribution of token embeddings can be very different [4]. In unified multi-modal transformers, QK norm is considered essential to prevent one modality from causing attention logits to dominate and to avoid training crashes [4]. Even in purely textual models, applying QK norm can allow using higher learning rates or fewer warmup steps without divergence. The original QKNorm paper showed an average +0.928 BLEU improvement in low-resource machine translation tasks by making attention less prone to spurious peaks [11].

Importantly, QK norm does not change the model's expressiveness (it's essentially inserting a normalization layer and adjusting the scaling). At inference time, it adds only a tiny overhead (normalizing vectors) but otherwise doesn't slow down attention computation. Some large-scale models (e.g., Google's ViT-22B, as reported by Zhai et al. in [4] have incorporated QK normalization and found it alleviated issues of logit drift (gradually shifting output distributions) during training. In summary, QK Norm is a stability optimization: by keeping attention logits well-behaved, it indirectly improves training speed (by enabling more aggressive training schedules) and ensures the benefits of deep or multi-modal attention can be realized without numerical problems.

7. Hybrid Transformer Models: Combining Attention with Alternative Layers

Beyond optimizing the attention mechanism itself, another promising direction is to combine attention with other efficient sequence processing modules within a model. The motivation is that while self-attention excels at modeling long-range dependencies, it might not be necessary (or optimal) to use it exclusively in every layer of a deep model. Recent research has explored hybrid architectures where some Transformer layers (or heads within a layer) are replaced by or run in parallel with lower-cost alternatives, such as convolution or state-space models (SSMs).

One striking example is Hymba (Hybrid Memory Bandwidth Attention) by NVIDIA [12], which introduces a hybrid-head Transformer that integrates standard attention heads alongside state-space model (SSM) components in each layer. SSMs (like those from models such as S4 or Mega) can handle long sequences with O(n) time via recurrence and have a small memory footprint, but they lack the dynamic global querying ability of attention [12]. By pairing them, Hymba aims to get the best of both: attention heads provide high-fidelity recall for important tokens, while SSM heads efficiently capture longer-term or background context [12]. These two representations are merged in parallel at each layer.

Impressively, a small 1.5 B-parameter model with this hybrid design was shown to outperform a purely-attention LLaMA-3B model in accuracy, while using 11.7× less KV cache memory and achieving 3.49× higher throughput during inference[12], [13]. In other words, by offloading some of the sequence processing to the lightweight SSM, the burden on attention (and its memory cache) is greatly reduced with no loss in quality – a slight gain was observed [12].

Another hybrid approach is to restrict full self-attention to certain layers or tokens and use cheaper attention patterns elsewhere. For instance, some models use sliding window attention in lower layers and full attention in a few top layers, or share the same KV cache across multiple layers to avoid redundant storage [12]. Hymba does this as well: it shares key/value caches between consecutive layers (since they tend to attend to similar things) and applies a local sliding window in most layers, reserving full attention for a few global "meta" tokens [12]. These strategies can cut memory use dramatically without much impact on the model's overall ability to capture dependencies.

Lastly, we note there are also efforts to replace attention entirely with other primitives (such as Fourier transforms, multi-layer perceptrons, or long convolution kernels as in models like FNet, gMLP, Hyena). Some of these alternatives can be more efficient for extremely long sequences. However, pure replacements often struggle to match attention's performance on complex tasks. The trend in 2023 - 2024 has favored hybrid designs instead of outright replacements: use attention where it's most needed, and augment or support it with computationally cheaper components elsewhere. This hybrid philosophy is likely to play a big role in next-generation Transformer architectures.

8. Conclusion

The landscape of attention mechanisms in machine learning is rapidly evolving, driven by the need to scale to longer sequences and larger models efficiently. We have seen many of the recent advancements, from multi-query and grouped-query attention reducing memory bandwidth, to FlashAttention rethinking the algorithmic execution, to RoPE and QK norm improving effectiveness and stability. These techniques do not fundamentally alter the $O(n^2)$ nature of attention, but rather mitigate its practical bottlenecks. These improvements often complement each other: for example, one can use GQA and FlashAttention together, achieving both lower memory usage and faster computation.

Crucially, these optimizations allow us to push the limits of models (longer context, faster responses) without sacrificing the powerful modeling capabilities that full attention provides. As summarized, Multi-Query/Grouped attention slashes inference costs for large LMs, high-performance kernels like

 | DOI: 10.20533/ijicr.2042.4655.2025.0165

FlashAttention make training and generation with long sequences feasible, and techniques like RoPE and QK Norm ensure the models remain accurate and stable when pushed to these new limits. Hybrid models offer a glimpse at a future where "attention is all you need" becomes "attention is mostly what we need," combined with other tools for efficiency.

Moving forward, we expect continued innovation in this space. Possible directions include learnable sparsity patterns (dynamic pruning of the attention graph), improved up-training methods to convert existing models to efficient variants, and further integration of hardware-aware optimization in model design. The goal is clear: maintain the richness of attention-based models while bringing down their time and memory requirements closer to linear scaling. The developments of the past two years have made significant strides toward that goal, making Transformers more practical and versatile than ever.

9. References

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems. https://arxiv.org/pdf/1706.03762 (Access Date: 30 March 2025).

[2] Beltagy, I., Peters, M. E., and Cohan, A. (2020). Longformer: The long-document transformer. Zaheer, M., et al. (2020). Big Bird: Transformers for longer sequences. Google's BigBird Model Improves Natural Language and Genomics Processing. https://www.infoq.com/news/2020/09/google-bigbird-nlp/ (Access Date: 11 June 2025).

[3] Shazeer, N. (2019). Fast transformer decoding: One write-head is all you need. arXiv preprint arXiv:1911.02150. https://ar5iv.org/abs/1911.02150 (Access Date: 19 May 2025).

[4] Henry, A., Dachapally, P. R., Pawar, S. S., and Chen, Y. (2020). Query-key normalization for transformers. arXiv preprint arXiv:2010.04245. https://rossjtaylor.com/blog/qk-norm-and-the-curious-case-of-logit-drift/ (Access Date: 26 April 2025).

[5] Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. (2022). FlashAttention: Fast and memory-efficient exact attention with IO-awareness. https://deepsense.ai/wp-content/uploads/2023/04/2205.14135.pdf (Access Date: 17 August 2025).

[6] Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. (2023). GQA: Training generalized multi-query transformer models from multi-head checkpoints. arXiv preprint arXiv:2305.13245. https://www.ibm.com/think/topics/grouped-query-attention (Access Date: 21 June 2025).

[7] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., et al. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288. https://tinkerd.net/blog/machine-learning/multi-query-attention/ (Access Date: 22 August 2025).

[8] Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebrón, F., and Sanghai, S. (2023). GQA: Training generalized multi-query transformer models from multi-head checkpoints. https://arxiv.org/pdf/2305.13245 (Access Date: 29 April 2025).

[9] Dao, T. (2023). FlashAttention-2: Faster attention with better parallelism and work partitioning. arXiv preprint arXiv:2307.08691. https://tridao.me/publications/flash2/flash2.pdf (Access Date: 13 August 2025).

[10] Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. (2021). Former: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864. https://arxiv.org/abs/2104.09864 (Access Date: 25 June 2025).

[11] Henry, A., Dachapally, P. R., Pawar, S. S., and Chen, Y. (2020). Query-key normalization for transformers. arXiv preprint arXiv:2010.04245. https://arxiv.org/abs/2010.04245 (Access Date: 12 July 2025).

[12] NVIDIA (2024). Hybrid attention and state space models for breakthrough performance. NVIDIA's Hybrid: Combining Attention and State Space Models. https://syncedreview.com/2024/12/14/self-evolving-prompts-redefining-ai-alignment-with-deepmind-chicago-us-eva-framework-14/ (Access Date: 17 June 2025).

[13] OpenReview (2024). Hybrid model performance analysis. https://openreview.net/pdf?id=H4DqfPSibmx (Access Date: 14 July 2025).

 | DOI: 10.20533/ijicr.2042.4655.2025.0165