Enhancing Large Language Model Efficiency through Sequence-Level Knowledge Distillation with Sparse Transformers

Ronald John C. Atanoso
Computer Science Department, College of Computing and Information Sciences, Caraga State University - Main Campus

Dan Victor B. Lofranco
Computer Science Department, College of Computing and Information Sciences, Caraga State University - Main Campus

Roland P. Abao
Computer Science Department, College of Computing and Information Sciences, Caraga State University - Main Campus

https://doi.org/10.5109/7395598

Enhancing Large Language Model Efficiency through Sequence-Level Knowledge Distillation with Sparse Transformers

Ronald John C. Atanoso[1], Dan Victor B. Lofranco[2], Roland P. Abao[3]
[1,2,3]Computer Science Department, College of Computing and Information Sciences, Caraga State University – Main Campus, Ampayon, Butuan City, 8600, Philippines
Corresponding author email: rpabao@carsu.edu.ph

Abstract: *This study explores the effectiveness of sparse attention mechanisms, specifically, local and sliding window attention in improving the efficiency of sequence-level knowledge distillation for decoder-only language models. While sliding window attention offers broader contextual coverage through overlapping receptive fields, its purely causal implementation introduced redundancy and led to diminished performance. In contrast, local attention, with its simpler non-overlapping design, achieved better training and validation outcomes, while also significantly reducing attention computation costs. Notably, the local attention model with large window size demonstrated the highest efficiency index, achieving a favorable trade-off between performance and computational overhead. These results highlight the importance of architectural alignment in sparse attention design, as well as the potential for substantial cost savings without major accuracy loss. To fully unlock the benefits of sparsity in causal models, future work should explore hybrid attention strategies, integrate global context or dynamic windowing, and assess generalizability across diverse sequence-level tasks.*

Keywords: Decoder-only transformer, Sparse transformer, Local window attention, Sliding window attention, Knowledge distillation

1. INTRODUCTION

One of the biggest breakthroughs in natural language processing in recent years was the creation of language models capable of speaking convincingly as if they were real people. However, that alone is not novel, as there had already been such models in the past—like Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs)—albeit with less efficiency. The real breakthrough was the introduction of Transformer-based models in the paper "Attention is All You Need" [1]. As the title suggests, the attention mechanism is what enables superior performance and scalability. It allowed dependencies inside a sentence to communicate without the short-memory limitations typical in RNNs and even LSTMs. Moreover, it enabled both training and inference to be parallelizable, making the model far more scalable.

The attention mechanism itself was not entirely new; it had already been used in fields like computer vision and speech recognition. The novelty introduced by Transformers was the concept of self-attention. While standard attention computes a weighted sum of value vectors using similarity scores between query and key vectors from different input and output sequences, self-attention uses queries, keys, and values from the same sequence—allowing the model to capture long-range dependencies and contextual information effectively.

Large language models on the commercial market tend to be more performant due to their higher count of learnable parameters. Parameter count alone does not fully determine model performance—factors such as data quality, optimization, and regularization also play important roles. However, more parameters often help in capturing complex patterns and enabling emerging behaviors within the model [2].

Today, the race for better language models is often framed around parameter count, as reflected in various dataset benchmarks. However, large models typically require vast clusters of Tensor Processing Units (TPUs) and are prohibitively expensive to train and deploy. They are also energy-intensive, both in training and inference.

This research aims to investigate whether sequence-level knowledge distillation can be effectively combined with sparse attention mechanisms in student models, and how such models compare to their full-attention counterparts under the same distillation conditions.

2. RELATED WORKS

Machine learning has demonstrated remarkable success in optimizing predictive tasks across diverse domains, from geotechnical property estimation [3] to dynamic system control [4]. While these applications leverage algorithmic efficiency to extract salient patterns from structured, low-dimensional data, natural language processing presents unique challenges due to its inherently high-dimensional and sequential nature. Building upon these cross-domain principles of efficient pattern extraction, we adapt and extend them to language modeling through structured sparsity and sequence-level distillation. Our approach systematically identifies and preserves only the most linguistically relevant attention patterns, transforming what is typically a dynamic, computation-heavy process into a static prior that maintains performance while eliminating redundant operations. This achieves comparable efficiency gains to those seen in other ML domains [3,4], but specifically addresses the quadratic complexity bottleneck inherent in Transformer-based language models.

Several attempts have been made to compress large language models while minimizing performance loss. One common approach is knowledge distillation, where

a pre-trained large model (teacher) transfers its knowledge to a smaller student model that mimics its outputs. This typically involves some trade-off: reduced accuracy and weaker ability to capture long-range dependencies due to limited memory, fewer computations, and smaller model size. For instance, DistilBERT is a 40% lighter version of the BERT model that retains approximately 97% of its performance on certain benchmarks [8].

Sparse Transformers are another line of work aimed at improving efficiency. They introduce sparsity in the attention computation by updating values only for a subset of elements instead of attending to the entire sequence. This reduces both computational complexity and memory usage, enabling the model to handle longer-range dependencies without sacrificing too much performance.

A notable example is the study by Child et al. [5] on Sparse Transformers, which introduced fixed and strided sparse attention patterns in autoregressive modeling. While their approach showed promising results on benchmark datasets, the sparsity was applied only to pre-trained Transformers using raw text corpora.
Another architecture using sparsity is the dilated sliding window attention model, where tokens attend only to local neighbors. The addition of dilation skips tokens at intervals, extending the context window without increasing the computational load.

This work builds on these foundations by applying sparse attention not just in pretraining but also during sequence-level knowledge distillation. It evaluates how sparse attention patterns affect student models' performance relative to full-attention models under the same training conditions. While models like Longformer by Beltargy et al. [6] utilize sparse attention in bidirectional contexts such as BERT, our study adapts the sparsity pattern concept for use in autoregressive decoding based on GPT-style architectures. This distinction is important as our aim is to evaluate the impact of sparsity in a generative, left-to-right setup rather than a masked language modeling context.

3. METHODS

3.1 Transformer Student Models

A total of 11 student models are produced in this study; 1 baseline model that uses **causal full attention**, 5 models that uses causal block-wise non-overlapping **local attention** with varying window sizes (2, 8, 16, 32, 64) as well as 5 models that uses **causal sliding window attention** with the same varying window sizes. Student models produced are based on GPT2 architecture and are all **causal autoregressive decoder-only architectures**. While the related studies did sparse attention mechanisms, they are all implemented under a bidirectional transformer architecture, this study will implement sparse attentions under a unidirectional transformer with respect to a decoder-only transformer. To only focus the findings on how sparse attention mechanisms work in student models, all student models will share the same hyperparameters and total model parameters count of 55 million, models will only differ in their forward pass function that calculates their attention scores.

Causal Full Attention: The baseline model employs standard full causal attention, where each token attends to all previous tokens on the sequence. This mechanism computes a dense attention matrix with quadratic complexity, enabling global contextual understanding but at high computational cost. Causality is enforced via a lower-triangular mask to prevent information leakage from future tokens. Fig. 1. shows a Causal Full Attention Connectivity Matrix, at current token (blue), future tokens (grey) are ignored while all previous tokens have computations representing attention scores.

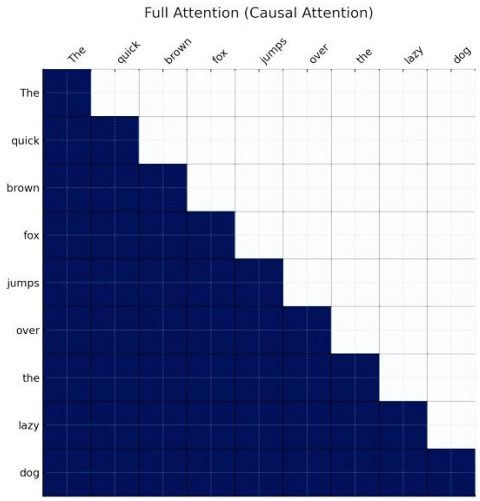

Fig 1. Causal Full Attention Connectivity Matrix with Lower-Triangular Masking for Token Dependency Control

Causal Local Attention (blockwise): Local attention restricts each token to a fixed, non-overlapping window of preceding tokens, reducing complexity to $O(n \cdot w)$ for window size w. While computationally efficient, this approach sacrifices long-range dependencies, as tokens cannot attend beyond their local block boundaries. Fig. 2. shows a Causal Local (blockwise) Connectivity Matrix, at current token (blue), future tokens (grey) are ignored while only previous tokens that belong to their own block or chunk have computations for attention scores.

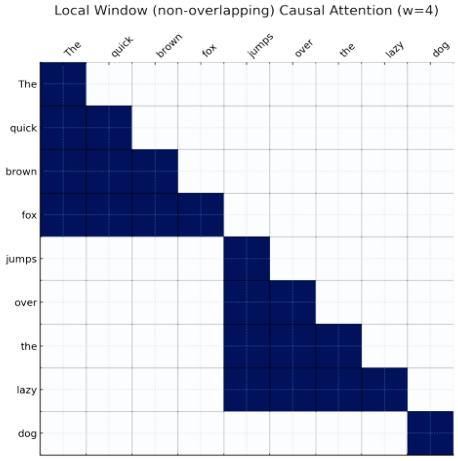

Fig. 2. Local Attention Connectivity Matrix with Constrained Window Context.

Causal Sliding Window Attention: Sliding window attention also uses a fixed window size but slides incrementally (by one token per step), allowing overlapping receptive fields. Though theoretically richer

in context than local attention, it's purely causal implementation in this study introduced redundancy without performance gains, as tokens repeatedly attend to similar local contexts. Fig. 3. shows a Causal Sliding (1 step) Connectivity Matrix, at current token (blue), future tokens (grey) are ignored while only previous tokens within a set window are allowed to have attention scores.

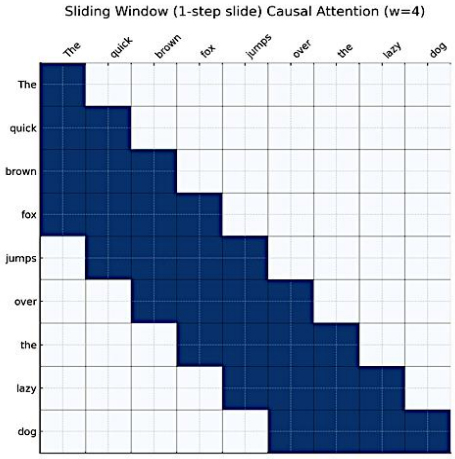

Figure 3. Sliding Window Attention Connectivity Matrix with Dynamic Shifted Focus.

Implementation Considerations

Notably, this study evaluates attention mechanisms at the algorithmic level rather than through optimized implementations. While prior work has demonstrated empirical speedups using custom CUDA kernels (e.g., Flash Attention) or specialized sparse attention libraries, our implementations use standard PyTorch operations to maintain functional equivalence across all attention variants. This design choice intentionally avoids hardware-specific optimizations that could introduce confounding variables when comparing architectural approaches. Instead of measuring wall-clock time or memory usage (which would favor implementations with dedicated kernel optimizations), we focus on theoretical complexity analysis through **attention cost** formulas quantifying FLOPs reductions in the attention computations part as well as an **efficiency Index** measuring performance-per-computation trade-offs.

3.2 Data Collection

This study utilized GPT-3.5 to generate a conversational dataset for sequence-level knowledge distillation, relying on the model's inference capabilities without accessing internal weights or parameters. This process is called the Sequence-Level Knowledge Distillation were instead of using the weights and biases of the Teacher Model for standard Knowledge Distillation, Teacher model provides the output or data used to train the preceding student models. The teacher model acts like a black-box as if the weights and biases are out of reach. Inference is used for this process.

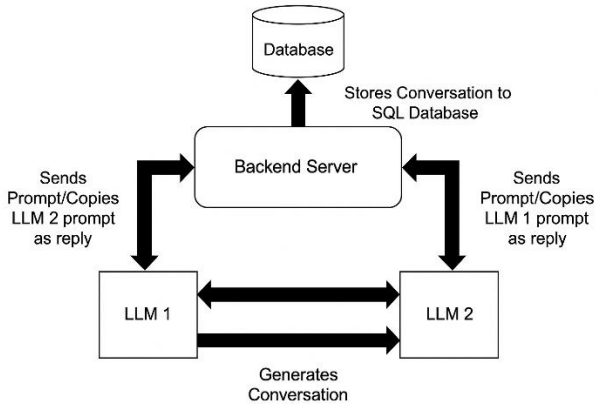

Fig. 4. Architecture of the Automated Conversation Generator.

For getting the Conversational Data from GPT-3.5, the use of OpenAPI usage is then used to extract its data. The process involves having 2 large language models of the same type to talk to each other. A set of random topics is then used for Prompt Engineering in order to make the conversation dataset diverse. The complete process involves:

- LLM 1 gets to start the conversation from the random set of topics. The response is later saved for the next stage.
- After LLM 1 provides its response, its value will be then used as input for the LLM 2, The response is then later used for the next stage.
- The process is now then reversed. The responses of the LLM 2 will be then used by LLM 1 as an input, and vice versa.

This process creates a response cycle, A set of conversation will be limited to a certain amount of response about 20 cycles. Since overtime as the LLMs conversation gets longer, the context of the discussion degrades as each cycle repeats. After the cycle is done the process is repeated with a new set of random topics to start the conversation.

3.3 Evaluation Method

Evaluating a Large Language Model output is not yet standardized, however there is some research particularly on the paper by Eldan. et., al (2023)[7] in the paper "Tiny Stories: How Small Can Language Models Be and Still Speak Coherent English?"[4]. The study shows the use of a state-of-the-art Large Language Models such as GPT-4 to evaluate the inferences of the student models. The research shows the use of evaluating various large language models with increasing size of parameters. The result shows that the GPT-4 Evaluation is indeed an effective method for qualitatively evaluating the quality of inferences of student models.

In order for this study to apply similar methods, the criteria design for this study selects the following metrics: **Grammatical Correctness**, **Readability**, **Descriptiveness**, **Coherence** and **Conciseness**. These criteria were carefully selected in order to further understand the result of the inferences of the student models based on the basic conversational dataset from GPT-3.5.

The process for automating the GPT-4 Evaluation, uses a prompt engineering framework called *Kiln AI*. This framework allows creating a "task" to do the process, this task is assigned with a "prompt message" which is an input for prompt engineering giving a task to the large language model on what to do the next proceeding prompts from the user. For this particular study, the task is designed to make the large language model GPT-4 to evaluate the string of conversational text and evaluate the results based on the given 5 criteria. The text below shows how the conversation is evaluated based on **JSON** format.

```
[{
      "sentence": "I love the feel of mornings, but I also get how a quiet moment can be just as inspiring. I'd say I lean a bit toward quieter moments, but I also enjoy the calm of a busy day sometimes--just a chance to pause and reflect. There's something special about that balance of peace and quiet. Do you ever find it hard to choose between the two, or do you feel like you've got a balance with your thoughts?",
      "grammatical_correctness": 9,
      "readability": 9,
      "descriptiveness": 9,
      "coherence": 9,
      "conciseness": 9,
      "explanation": "The sentence flows well, engaging the reader effectively without unnecessary complexity. All aspects are strong in this communication."
}]
```

The key-value pair called ***"explanation"*** is used to check on how the GPT-4 gives the result of the conversation sentence, this is to ensure that the evaluation makes sense from a researcher's standpoint. Once the results are provided, it is then stored to a database for further analysis.

4. RESULTS AND DISCUSSION

The data collection was processed in the span of 8 months, With the amount of training and time the amount of dataset that is collected is around **500k conversations** or around 1 million replies and more than 50 million tokens. A diverse collection from various topics such as Books, Movies, Cartoons, Small talks, etc. The collected data is available on the Kaggle website.
https://www.kaggle.com/datasets/danvictorlofranco/llms-basic-conversation-dataset/data

Model Evaluation: Cross-entropy Analysis
All models were trained for 100k iterations (98,304 tokens/iteration) under identical conditions:

- Batch: 6 sequences × 1,024 tokens
- Validation: 200 held-out sequences evaluated every 200 steps
- 50M tokens of GPT3.5 generated conversations (90%/10% train/val split).

Key Metrics

- **Training Loss**: Measures fit to teacher-generated data.
- **Validation Loss:** Tests generalization on unseen conversations

Table 1. Performance across attention types

Model	Type	W	TL	VL
full-wfull-54M-r5	Full	N/A	1.35	1.84
local-w128-54M-r8	Local	128	1.55	1.93
local-w64-54M-r9	Local	64	1.7	2
local-w32-54M-r10	Local	32	1.9	2.13
local-w16-54M-r11	Local	16	2.19	2.2
local-w2-54M-r15	Local	2	3.53	3.35
slide-w128-54M-r9	Sliding Window	128	4.02	3.76
slide-w64-54M-r7	Sliding Window	64	4.09	4.02
slide-w32-54M-r6	Sliding Window	32	4.12	4.05
slide-w16-54M-r5	Sliding Window	16	4.23	4.18
slide-w2-54M-r10	Sliding Window	2	4.3	4.22

Legend:

Type - Attention Type
TL - Train Loss
VL - Descriptiveness
W - Window Size

Full Attention baseline achieved lowest validation loss (1.84), confirming its representational capacity. **Local blockwise attention** models demonstrated strong window-size dependency; with larger window sizes achieving higher performance gain and local-w128 as its best performer (val loss 1.93, +4.9%).. Sliding window attention consistently underperforms (val loss 3.76-4.22) and shows minimal improvement with larger windows sizes (+0.46 val loss reduction from w=2 -> 128).

GPT-Evaluation
The GPT-Eval framework was able to get around 15k+ evaluations based on the criteria from Grammatical Correctness, Readability, Descriptiveness, Coherence and Conciseness. Each of which are from the **11 Models** (1 full attention, 5 local, and 5 sliding of corresponding window sizes), All are sampled **1370** for each model.
The sampled evaluations are then analysed from various perspectives from its performance to its architectural significance as a transformer based on its type full-attention, local and sliding.

Table 2. Average Metric Table Model Performance

model	G	R	D	C	Co
out-full-wfull-54M-r5	8.79	8.64	8.47	8.67	8.14
out-local-w128-54M-r8	8.77	8.50	8.47	8.58	8.07
out-local-w64-54M-r9	8.56	8.37	8.35	8.34	7.89
out-local-w32-54M-r10	8.32	8.11	8.12	7.92	7.67
out-local-w16-54M-r11	7.52	7.43	7.53	7.23	6.94
out-slide-w128-54M-r9	5.86	5.75	6.30	5.40	5.44
out-local-w2-54M-r15	5.90	5.79	6.26	5.28	5.44
out-slide-w64-54M-r7	5.75	5.75	6.22	5.34	5.36
out-slide-w32-54M-r6	5.82	5.67	6.24	5.23	5.26
out-slide-w16-54M-r5	5.61	5.55	6.09	5.09	5.15
out-slide-w2-54M-r10	5.55	5.48	5.98	5.09	5.11

Legend:

G - Grammatical Correctness
R - Readability
D - Descriptiveness
C - Coherence
Co - Conciseness

The table II shows the performance average for each metrics and ranked by its performance. By its average performance to each metric, it shows that the full attention model (out-full-wfull-54M-r5) perform the best performance out of all types of models, the local models follows after, and the sliding models perform worst with some model (out-slide-w128-54M-r9) beating local model. There is a positive correlation with the performance by its window size, such that **as the window size increases the quality of the output increases**. The trend is more prominent to local models, having each window size increase significantly, while the sliding models barely show differences but slight improvement still showing the pattern as the ranking shows ascending order for each window size increase. This trend and the training loss provides a predictable outcome of its quality of output.

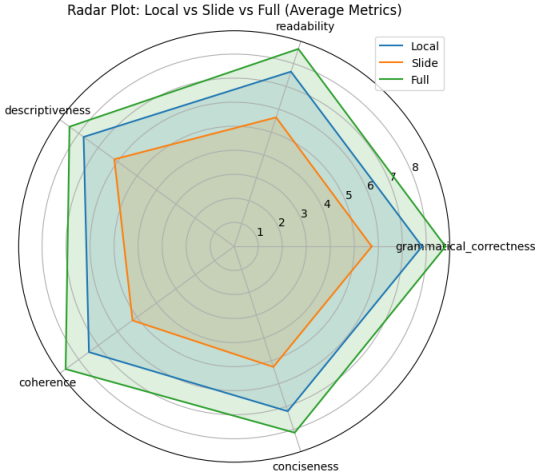

Fig. 5. Radar Plot Comparing Full, Local, and Sliding Attention Models

Figure 5 presents a radar plot comparing the performance of models using full, local, and sliding attention mechanisms. The full attention model, represented in green, consistently achieved the highest scores across all evaluation metrics. Local attention models, shown in blue, demonstrated near-parity with full attention, indicating minimal performance degradation despite reduced computation. Sliding window models, depicted in yellow, underperformed across all criteria.

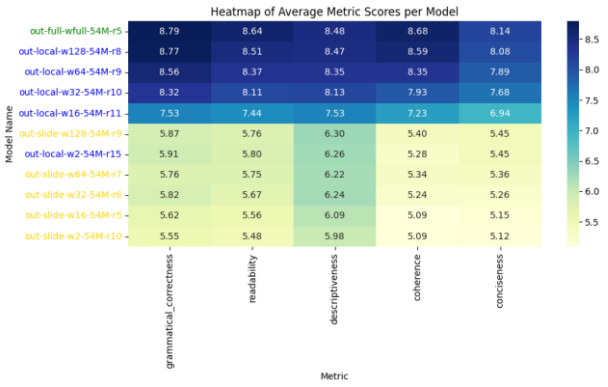

Fig. 6. Heatmap of Average Metric Scores Across All Models

The Heatmap shows the trend of its performance from various criteria, showing the trend from light yellow (low) to dark blue (high) scores. The Full Attention shows a dark blue hue as benchmark providing the highest score and quality of inferences. The trend of local models from **w2** to **w128** significantly changed from light yellow to dark blue solidifying the relationship of output quality and window size via direct proportionality.

Table 3. Model Performance Ranking with Percent Benchmark

model name	**average score**	**percent of benchmark**
out-full-wfull-54M-r5	8.54	100
out-local-w128-54M-r8	8.48	99.26
out-local-w64-54M-r9	8.30	97.18
out-local-w32-54M-r10	8.03	93.99
out-local-w16-54M-r11	7.33	85.82
out-slide-w128-54M-r9	5.75	67.34
out-local-w2-54M-r15	5.73	67.14
out-slide-w64-54M-r7	5.68	66.55
out-slide-w32-54M-r6	5.64	66.09
out-slide-w16-54M-r5	5.50	64.37
out-slide-w2-54M-r10	5.44	63.71

The Sliding models on the other hand, shows small changes from ***w2*** to ***w128***. The light shades show that the sliding models and local modals gaps in quality of performance are huge across all metrics, concluding

better outlook for local models as a better suitable for sequence-level knowledge distillation technique.

The Percent benchmark shows the performance relative to the full attention model, The Local Models performance is very close relative to the *wfull* model, this tells that the local model tradeoffs on its implementation to its attention can perform similar performance of that of the full attention model. In contrast, The Sliding window model shows a more pronounced drop in relative performance. It operates at approximately 67% of the speed of the full attention model, indicating a substantial trade-off in efficiency.

Table 4. Standard Deviation Metric Score Average

model name	std average
out-full-wfull-54M-r5	1.028794
out-local-w128-54M-r8	1.071948
out-local-w64-54M-r9	1.162879
out-local-w32-54M-r10	1.236150
out-local-w16-54M-r11	1.648251
out-local-w2-54M-r15	1.679374
out-slide-w32-54M-r6	1.790947
out-slide-w64-54M-r7	1.797253
out-slide-w128-54M-r9	1.810470
out-slide-w16-54M-r5	1.866170
out-slide-w2-54M-r10	1.869903

In order to derive the consistency of each model's performance. The use of standard deviation is key to understanding how the result differs from each criteria.

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

The Standard Deviation average shows the consistency of each model output. The lower the std, the better, It tells that the model output score is of similar quality across all metrics. The Full Attention benchmark model shows the consistent scores across all frequencies. Followed by the local model *out-local-w128-54M-r8* that has close performance of 1.07, and its least consistent around 1.67 achieved by the *out-local-w2-54M-r15*. A similar trend of window sizes shows that **as the window size increases the consistency of the scores increases**.

In contrast to that, the sliding model doesn't have a similar trend like the local model. The best sliding model in terms of consistency is *out-slide-w32-54M-r6* instead of *out-slide-w128-54M-r9*. The result tells that the sliding model is very noisy in terms of its scores across all metrics. The greater variability indicates that the sliding attention mechanism induces greater model performance fluctuations, leading to less consistency than full and local attention models. This variability might be undesirable in situations requiring consistent and predictable output.

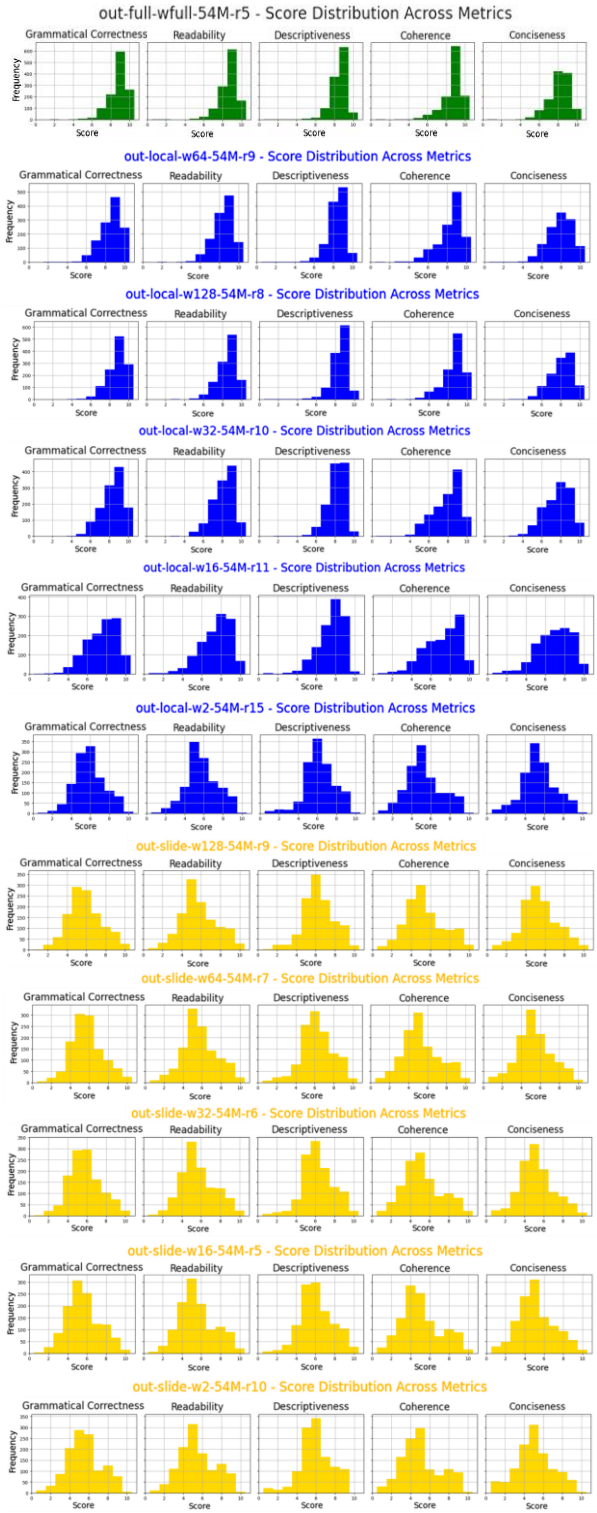

Fig. 7. Score distribution of Models from Highest to lowest by Attention Type

Legend:

Full Attention - Green
Local Attention - Blue
Sliding Attention - Yellow

Based on the Score Distribution across metrics, the local model's performance improves significantly as the number of window size increases, there is a positive correlation based on the result. The local model in ascending order (see fig. 7, from bottom to top) based on

window size shows that the frequency metric goes from left to right or increasingly gets better scores. And despite the trade-offs in its performance and shorter training time (approximately twice faster compared to full attention with 128 window size, and quadruple for smaller window sizes) it is promising that it is very similar to the graphs of the full attention model *out-full-wfull-54M-r5*. The local models are very compatible with sequence-level knowledge distillation, with almost similar performance to the full attention models across all metrics.
In contrast, to the sliding models. There is small correlation with its window size and a small pattern emerges as the window size increases. Although it is clear that a slight improvement is observed in the output with increasing frequency of score on 8 to 10 scores.

Evaluation Discussion
The cross-entropy results reveal fundamental insights about sparse attention in causal language models. While full attention unsurprisingly achieved the lowest validation loss (1.84), local attention with large windows (w≥64) demonstrated remarkable efficiency, maintaining within 9% of the baseline's performance despite drastic theoretical cost reductions. This suggests that for basic conversational tasks – where local context often suffices – carefully sized non-overlapping windows can effectively replace global attention without significant quality degradation. However, the sharp performance decline in smaller local windows (w≤16) indicates a critical threshold below which context becomes insufficient for coherent generation.

The consistent underperformance of sliding window models is particularly notable. Unlike bidirectional implementations (e.g., Longformer), our purely causal sliding attention failed to leverage its overlapping receptive fields effectively. We hypothesize two failure modes: (1) redundant attention to similar local contexts across sliding positions, and (2) lack of global "anchor" tokens to propagate long-range information. This contrasts with local attention's explicit segmentation, which appears to force more efficient intra-window feature learning. The minimal improvement from increasing sliding window sizes (+0.46 loss reduction from w=2→128) further suggests this pattern's fundamental incompatibility with unidirectional decoding.

Theoretical Attention Cost Analysis
Computation Complexity Formulas derived from the number of attention operations that happens in a forward pass:

- Causal Full Attention:

$$Cost_{\text{full}} = 2n(n+1)$$

- Causal Local Attention (blockwise):

$$Cost_{\text{local}} = \sum_{c=1}^{|n/w|} 2lc(l_c + 1),$$

$$\text{where } l_c = \min(w, n - w \cdot (c-1)$$

- Causal Sliding Window Attention:

$$Cost_{\text{sliding}} = \frac{w(w+1)}{2} + (n-w) \cdot w$$

Table 5. Model Attention Costs

model name	W	BS	AC
out-full-wfull	N/A	1024	524800
out-local-w2	2	1024	1536
out-local-w16	16	1024	8704
out-local-w32	32	1024	16896
out-local-w64	64	1024	33280
out-local-w128	128	1024	66048
out-slide-w2	2	1024	2047
out-slide-w16	16	1024	16264
out-slide-w32	32	1024	32272
out-slide-w64	64	1024	63520
out-slide-w128	128	1024	122944

Legend:

W - Window Size
BS - Block Size
AC - Attention Cost

To identify the cost-efficiency trade-off of the models, we compare their attention cost and validation loss against the full attention baseline. This can be quantified using the following formula:

$$Efficiency\ Index = \frac{Cost\ Reduction\%}{Loss\ Increase\%}$$

Table 6. Model Efficiency Index

Model	AC	VL	C-R%	L-I %	EF
local-w2	1,536	3.35	99.71	82.04	1.22
local-w16	8,704	2.2	98.34	19.55	5.03
local-w32	16,896	2.13	96.78	15.74	6.15
local-w64	33,280	2	93.66	8.68	10.79
local-w128	66,048	1.93	87.41	4.87	17.93
slide-w2	2,047	4.22	99.61	129.31	0.77
slide-w16	16,264	4.18	96.9	127.14	0.76
slide-w32	32,272	4.05	93.85	120.07	0.78
slide-w64	63,520	4.02	87.9	118.44	0.74
slide-w128	122,944	3.76	76.57	104.31	0.73

Legend:

AC - Attention Cost
VL - Validation Loss
C-R% - Cost-Reduction Percentage
L-I% - Loss-Increase Percentage
EF - Efficiency Index

The Efficiency Index tells you how many percent of attention-cost you "save" for each 1 % of extra validation loss relative to the baseline. Higher is better. Table VI shows that model local-w128 had the highest efficiency index, as well as showing that local window type models had better efficiency compared to the sliding window model counterparts. Local Attention window-size-128

(out of 1024 block size) increases validation loss of baseline by 5% but only uses 13% of the baseline's total attention.

5. CONCLUSION

This study conducted a systematic evaluation of sparse attention mechanisms—specifically local and sliding window attention—in the context of sequence-level knowledge distillation for decoder-only transformer models. The findings highlight that local attention, particularly with a window size of 128, delivers a strong balance between computational efficiency and output quality. The full attention model still achieved the best performance across all evaluation metrics, but large-window local models demonstrated only a marginal drop in performance while offering substantial reductions in attention computation costs.

Conversely, sliding window attention consistently underperformed. Despite its overlapping receptive fields, it introduced redundant attention computations that did not translate to better language generation. This suggests a fundamental incompatibility of the sliding window mechanism, at least in its causal form, for tasks requiring coherent long-range generation. Notably, evaluation metrics from GPT-4, including grammar, coherence, and descriptiveness, reinforced the superiority of full and local attention models over sliding variants. Moreover, the Efficiency Index validated the theoretical gains, with the local-w128 model achieving a 17.93 score, the highest among all tested architectures.

Importantly, the experimental design emphasized functional equivalence by using consistent model sizes and non-optimized PyTorch implementations, allowing for a fair, architecture-level comparison. These findings provide empirical and theoretical validation that structured sparsity—when carefully designed—can significantly accelerate transformer models with minimal quality degradation.

6. FUTURE WORK

While this study focused on static sparse patterns, future research could explore dynamic or hybrid attention architectures that adaptively combine local, global, and dilated patterns based on input context. Integrating global tokens or landmark-based routing mechanisms may also improve the ability of sparse models to capture long-range dependencies, particularly for more complex tasks beyond basic conversation.

Furthermore, implementing hardware-optimized kernels (e.g., FlashAttention, Triton-based attention layers) will provide a more realistic benchmark for inference and training speedups. Comparative studies using real-world deployment metrics such as latency, throughput, and energy consumption would give further validation beyond theoretical FLOPs.

Finally, incorporating human evaluations alongside GPT-Eval, especially in domains with subtle linguistic nuance or creative generation, would allow a more holistic understanding of model quality. Extending this framework to cross-lingual tasks, multimodal inputs, or domain-specific conversations (e.g., medical, legal) could also broaden its practical applicability.

7. REFERENCES

[1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. arXiv. https://arxiv.org/abs/1706.03762

[2] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, fS., Usman, M., Akhtar, N., Barnes, N., & Mian, A. (2023). A comprehensive overview of large language models. arXiv. https://arxiv.org/abs/2307.06435

[3] Galupino, Joenel & Dungca, Jonathan. (2023). Estimation of Permeability of Soil-Fly Ash Mix using Machine Learning Algorithms. Proceedings of International Exchange and Innovation Conference on Engineering & Sciences (IEICES). 9. 28-33. 10.5109/7157938.

[4] Shaqour, A., & Hagishima, A. (2022). Recent Advances in Reinforcement Learning Applications for Building Energy Management: A Mini Review. International Exchange and Innovation Conference on Engineering and Sciences, 239-245. https://doi.org/10.5109/5909098

[5] Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. arXiv. https://arxiv.org/abs/1904.10509

[6] Beltagy, I., Peters, M. E., & Cohan, A. (2020). Longformer: The long-document transformer. arXiv. https://arxiv.org/abs/2004.05150

[7] Eldan, R., Li, X. L., & Zhang, Y. (2023). TinyStories: How small can language models be and still speak coherent English? arXiv. https://arxiv.org/abs/2305.07759

[8] Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108. https://arxiv.org/abs/1910.01108