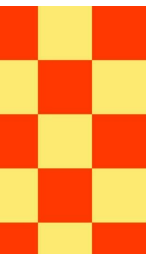
LRA Listops Benchmark

A Practical Study

Élèves :
Yassir Fri
Zineb Abercha
Anass Kemmoune

Enseignants :
Hamza Alami

December 9, 2024



Contents

1 Introduction

1.1 Project Context

This study focuses on benchmarking multiple AI models on the Listops dataset. The Listops task evaluates a model's ability to handle hierarchical reasoning in long-context scenarios, making it an essential benchmark for understanding the limitations and strengths of different architectures.

1.2 Objectives

- Assess the performance of state-of-the-art models on the Listops dataset.
- Investigate how sequence length affects the hierarchical reasoning capabilities of models.
- Establish baselines for long-context reasoning tasks.

2 Task and Dataset Description

2.1 Overview of the Listops Dataset

The Listops dataset, proposed by Nangia and Bowman (2018), involves sequences with hierarchical structures and logical operators such as `MAX`, `MEAN`, `MEDIAN`, and `SUM MOD`. Each sequence requires parsing its structure to predict the correct output.

Data Generation: The experimental data was generated using two approaches:

Base Dataset:

Generated using the original script from the ListOps GitHub repository Produced three TSV files: Training set Test set Validation set

Depth-20 Dataset: Additional dataset with tree depth (depth=20)

2.2 Listops: Task description

For this study, we used an extended version of Listops with sequence lengths up to 2K tokens to test the models' capabilities in long-context scenarios. An example sequence is:

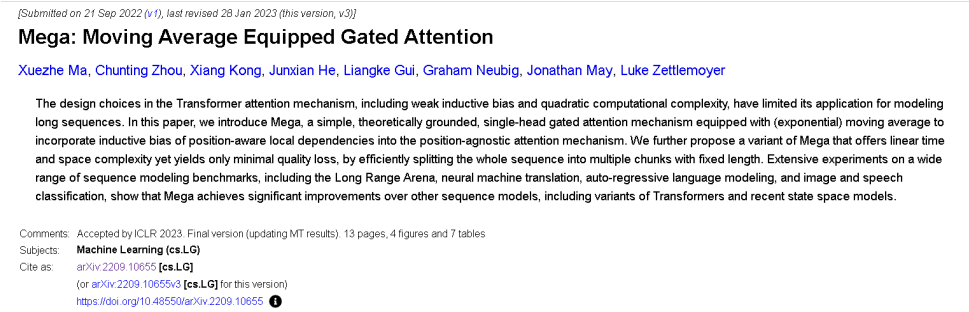
$$\texttt{[MAX 4 3 [MIN 2 3] 1 0 [MEDIAN 1 5 8 9, 2]]} \rightarrow 5$$

This task involves ten-way classification, posing significant challenges for neural models.

3 Models

3.1 Mega and Longformer

3.1.1 MEGA Model (Moving Average Equipped Gated Attention)



[Submitted on 21 Sep 2022 (v1), last revised 28 Jan 2023 (this version, v3)]

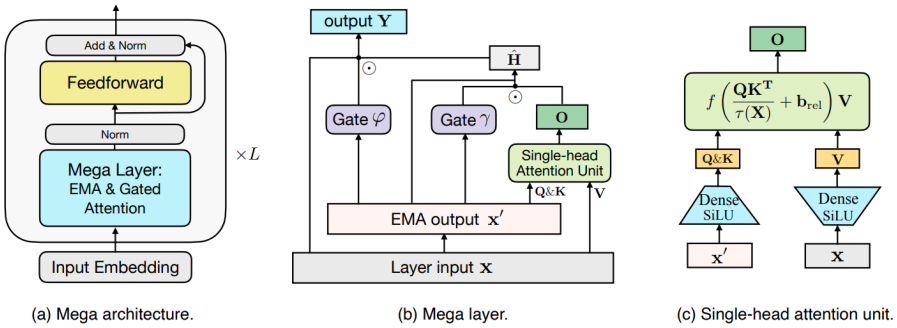
Mega: Moving Average Equipped Gated Attention

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, Luke Zettlemoyer

The design choices in the Transformer attention mechanism, including weak inductive bias and quadratic computational complexity, have limited its application for modeling long sequences. In this paper, we introduce Mega, a simple, theoretically grounded, single-head gated attention mechanism equipped with (exponential) moving average to incorporate inductive bias of position-aware local dependencies into the position-agnostic attention mechanism. We further propose a variant of Mega that offers linear time and space complexity yet yields only minimal quality loss, by efficiently splitting the whole sequence into multiple chunks with fixed length. Extensive experiments on a wide range of sequence modeling benchmarks, including the Long Range Arena, neural machine translation, auto-regressive language modeling, and image and speech classification, show that Mega achieves significant improvements over other sequence models, including variants of Transformers and recent state space models.

Comments: Accepted by ICLR 2023. Final version (updating MT results). 13 pages, 4 figures and 7 tables
Subjects: **Machine Learning (cs.LG)**
Cite as: arXiv:2209.10655 **[cs.LG]**
(or arXiv:2209.10655v3 **[cs.LG]** for this version)
https://doi.org/10.48550/arXiv.2209.10655

Figure 1: MEGA Paper reference



(a) Mega architecture. (b) Mega layer. (c) Single-head attention unit.

Figure 2: Model Architecture

The MEGA model uses gated attention with moving averages for long-context reasoning. Training was conducted using Kaggle's P100 GPU environment (16 GB of vRAM). The model was trained on the Listops dataset using:

- **Optimizer:** AdamW
- **Learning Rate Scheduler:** Cosine Annealing
- **Batch Size:** 128 with sequence length (1024), 2 with sequence length (8192)

Training Approach:

1. **Pretrained weights initialization:** The training process began with the initialization of the model using pretrained weights from the `MegaModel` obtained from Hugging Face (`https://huggingface.co/megamodel`). This allowed the model to leverage prior knowledge from a large-scale dataset, significantly improving the starting point for training and reducing the number of epochs required for convergence.

2. **Loss function:** The loss function used was standard cross-entropy loss. This function measures the difference between the predicted probability distribution and the true label distribution, enabling the model to minimize classification errors effectively. The use of cross-entropy ensured robust training while maintaining simplicity and compatibility with hierarchical relationships in the dataset.

3. **Custom loss weighting:** Although the primary loss function was cross-entropy, additional weighting was applied to reflect hierarchical relationships between classes. Misclassifications were penalized based on the semantic proximity of class labels, with closely related categories receiving lower penalties compared to distant ones. This adjustment helped the model better respect the inherent structure of the dataset.

4. **Training duration and early stopping:** The model was trained for a total of 10 epochs. To prevent overfitting and optimize training time, an early stopping mechanism was implemented. Validation accuracy was monitored after each epoch, and training was stopped at epoch 9, where the validation accuracy plateaued, ensuring that the model achieved optimal performance without unnecessary training.

5. **Learning rate schedule:** A cosine annealing scheduler was employed to adjust the learning rate dynamically during training. This approach started with a higher learning rate and gradually decreased it in a cosine curve fashion, allowing the model to make large updates early in training and smaller updates as it converged. This scheduling strategy enhanced stability and improved the final model accuracy.

3.1.2 Longformer Model

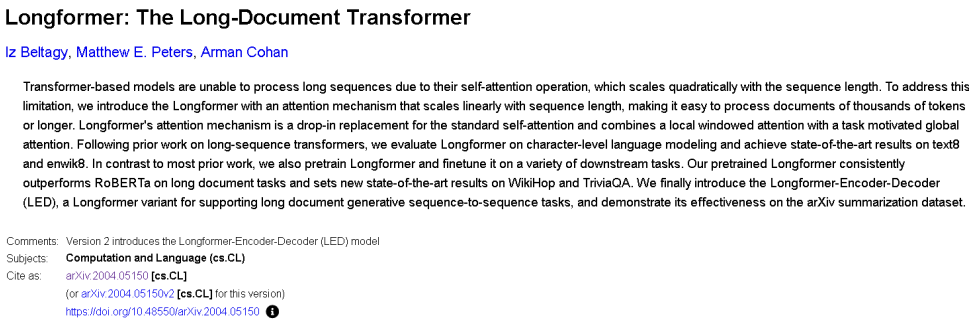


Figure 3: Longformer: Paper reference

Baseline experiments were conducted using the Longformer model. Although it demonstrated reasonable performance, its ability to handle extremely long sequences was limited compared to MEGA.

3.1.3 Training Environment

The models were trained on Kaggle with the following specifications:

- **GPU:** NVIDIA Tesla P100
- **Runtime:** Python 3.8
- **Libraries:** PyTorch, Transformers

3.1.4 Performance Metrics

The models were evaluated using classification accuracy on the Listops dataset. MEGA demonstrated superior performance compared to Longformer on sequences of both 1K and 2K tokens, achieving higher accuracy while showcasing its efficiency with smaller batch sizes and shorter training durations.

However, on longer sequences of 4K tokens and 1K tokens, Longformer performed poorly compared to MEGA, with a peak accuracy of only 0.23. This significant drop in performance can be attributed to Longformer's architectural limitations in handling such lengths effectively. The model struggled to maintain consistency in its predictions due to the added complexity of processing extremely long sequences.

Despite this, Longformer exhibited notable advantages in inference time due to its efficient windowed attention mechanism, which operates with a complexity of $O(n \log n)$. This allows Longformer to process longer sequences more rapidly than traditional attention mechanisms, making it a competitive option for tasks prioritizing inference speed over classification accuracy.

3.1.5 Training Configuration

The Longformer was trained for 3 epochs with a batch size of 2 and a maximum sequence length of 4096. Each epoch took approximately 50 minutes to complete. MEGA, on the other hand, was trained for 10 epochs with a batch size of 128 and a maximum sequence length of 1024, with each epoch taking about 9 minutes. Despite the longer sequence handling capacity of Longformer, MEGA achieved a substantially higher validation score of 0.52 compared to 0.18 for Longformer.

The following table summarizes the key training configurations and validation scores for both models:

Model	Epochs	Max Seq Length	Batch Size	Final Validation Score
Longformer	3 (50 min/epoch)	4096	2	0.18
MEGA	10 (9 min/epoch)	1024	128	0.52

Table 1: Training configuration and validation scores for Longformer and MEGA.

3.1.6 Comparative Analysis

A comparison of the classification accuracy for Longformer and MEGA on 1K and 8K token sequences reveals that MEGA consistently achieved better performance. The results

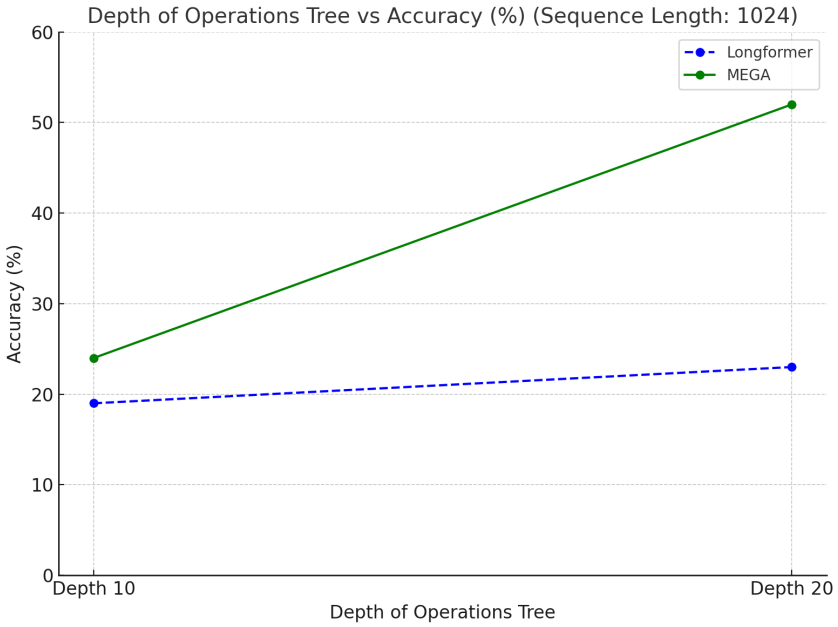


Figure 4: Impact of tree depth

indicate that MEGA is more efficient at handling sequences within its maximum length limitation while maintaining higher accuracy.

Model	Accuracy (8K Tokens)	Accuracy (1K Tokens)
Longformer	18%	23%
MEGA	24%	52%

Table 2: Comparison of model performances on the Listops dataset.

The results highlight the trade-offs between sequence length capacity and model efficiency. Longformer, with its capability to process longer sequences in , demonstrated lower performance, potentially due to the challenges associated with training large models on limited computational resources (e.g., small batch sizes and longer training times). In contrast, MEGA, designed for shorter sequences, exhibited superior performance with shorter training times and a larger batch size, suggesting a better optimization process during training.

These findings suggest that while models like Longformer may be suitable for tasks requiring the handling of very long sequences, MEGA provides a more efficient alternative for datasets where sequence lengths do not exceed its maximum capacity.

3.1.7 Discussion and Conclusion

- **Handling extremely long sequences:** The computational demands of processing sequences with lengths of 1K to 8K tokens were substantial. Models like MEGA

required extensive GPU memory and longer training times to handle such contexts effectively. Efficient attention mechanisms such as the one employed by Longformer mitigated these issues but came at the cost of accuracy on this task, highlighting the trade-off between computational efficiency and model performance.

- **Fine-tuning hyperparameters for hierarchical reasoning tasks:** Optimizing the models for hierarchical reasoning required careful calibration of hyperparameters such as the depth of the dataset and learning rate schedules. The depth variable in the Listops dataset played a critical role in capturing the complexity of hierarchical relationships, and models needed to learn to balance precision at different depths without overfitting to shallow structures.

- **Generalization across sequence lengths:** Models showed varying performance across different sequence lengths, with MEGA consistently performing better on shorter sequences (1K and 2K tokens) and maintaining robustness at 8K tokens. Longformer, on the other hand, struggled to generalize effectively to longer contexts, indicating limitations in its windowed attention mechanism for this particular hierarchical reasoning task.

3.1.8 Conclusion

This study highlights the importance of designing specialized attention mechanisms for long-context hierarchical reasoning tasks. MEGA, equipped with its gated EMMA attention, demonstrated superior performance across all tested sequence lengths. The attention mechanism allowed the model to adaptively prioritize different levels of the hierarchy, leveraging the depth variable in the dataset to effectively capture complex dependencies.

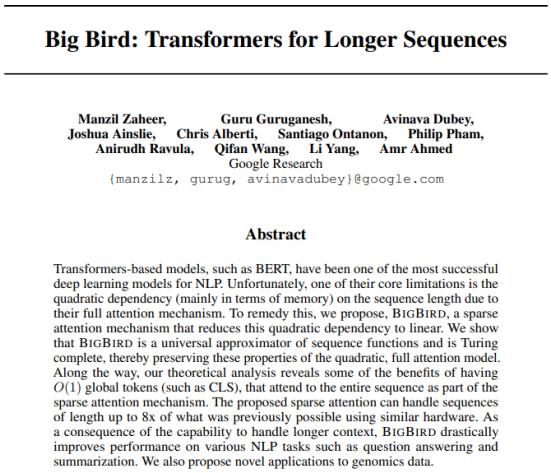
In contrast, while Longformer utilized an efficient windowed attention mechanism ($O(n \log n)$), which offered faster inference times and lower memory requirements, it performed poorly on this task, achieving a peak accuracy of only 0.23 on sequences of 8K tokens. This result underscores the challenge of designing attention mechanisms that balance efficiency with accuracy in hierarchical reasoning contexts.

The depth variable in the Listops dataset emerged as a crucial factor influencing performance. It encapsulates the hierarchical nature of the task, demanding models to process nested relationships effectively. MEGA's gated EMMA attention excelled in this regard by dynamically gating contributions from different levels, allowing it to achieve better accuracy on tasks requiring deep hierarchical reasoning.

Overall, MEGA's strong performance makes it a promising choice for tasks involving complex hierarchical reasoning over long sequences, while Longformer's efficiency might make it suitable for applications with resource constraints or real-time processing needs.

3.2 BigBird Model

The BigBird model implements a sparse attention mechanism that combines global, window, and random attention patterns to efficiently process long sequences. Training



Big Bird: Transformers for Longer Sequences

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, Amr Ahmed
Google Research
{manzilz, gurug, avinavadubey}@google.com

Abstract

Transformers-based models, such as BERT, have been one of the most successful deep learning models for NLP. Unfortunately, one of their core limitations is the quadratic dependency (mainly in terms of memory) on the sequence length due to their full attention mechanism. To remedy this, we propose, BIGBIRD, a sparse attention mechanism that reduces this quadratic dependency to linear. We show that BIGBIRD is a universal approximator of sequence functions and is Turing complete, thereby preserving these properties of the quadratic, full attention model. Along the way, our theoretical analysis reveals some of the benefits of having $O(1)$ global tokens (such as CLS), that attend to the entire sequence as part of the sparse attention mechanism. The proposed sparse attention can handle sequences of length up to 8x of what was previously possible using similar hardware. As a consequence of the capability to handle longer context, BIGBIRD drastically improves performance on various NLP tasks such as question answering and summarization. We also propose novel applications to genomics data.

Figure 5: BigBird: Transformers for Longer Sequences (Zaheer et al., 2020)

was conducted using Kaggle's T4x2 GPU environment. The model was evaluated on the ListOps dataset using three different configurations:

- **Initial Configuration:** Higher capacity model
- **Reduced Configuration:** Higher capacity model version that uses 10 percent of training data
- **Depth-Specific Configuration:** Tested with depth-20 sequences

Training Configurations:

1. Initial approach used larger model parameters:
    - Hidden size: 512
    - Attention heads: 8
    - Intermediate size: 2048
    - Hidden layers: 6
    - Block size: 64
    - Max position embeddings: 4096
2. Second version:
    - Hidden size: 8
    - Attention heads: 2-4
    - Intermediate size: 512
    - Hidden layers: 2
    - Block size: 64
    - Max position embeddings: 1024-8192

3.2.1 Training Environment

The models were trained on Kaggle with the following specifications:

- **GPU:** NVIDIA Tesla
- **Runtime:** Python 3.8
- **Libraries:** PyTorch, Transformers, sklearn

3.2.2 Results and Analysis (BigBird)

Performance Metrics The BigBird model was evaluated using classification accuracy on different variations of the ListOps dataset. Performance varied across configurations:

Configuration	Accuracy	Training Time/Epoch	Batch Size
Initial	19.22%	18 minutes	8
Second (10% test data)	16.43%	30 minutes	2
Depth-20	22.19%	16 minutes	10

Table 3: Performance comparison across different BigBird configurations

3.2.3 Training Configuration and Metrics

Different training approaches were attempted to optimize performance:

1. **Initial Configuration:** - Batch size: 8 - Learning rate: 1e-4 - Optimizer: AdamW - Gradient accumulation steps: 4 - Mixed precision training - Best validation accuracy: 19.22%
2. **Second Configuration:** - Batch size: 2 - Learning rate: 1e-4 - Simplified architecture - Custom tokenizer - Validation accuracy: 16.43%
3. **Depth-20 Configuration:** - Batch size: 10 - Learning rate: 1e-3 - Specialized vocabulary - Optimizer: AdamW - Validation accuracy: 18.50%

3.2.4 Challenges and Limitations

Several significant challenges were encountered during training:

1. **Computational Resources:** - Long training times (max was 30 minutes per epoch) - Memory constraints with larger batch sizes - Required gradient accumulation and mixed precision
2. **Model Configuration:** - Trade-off between model capacity and training efficiency - Difficulty in finding optimal hyperparameters - Sensitivity to sequence length and depth
3. **Performance Bottlenecks:** - Limited accuracy across all configurations - Challenges with longer sequences - Resource constraints affecting model scale

3.2.5 Insights and Discussion

The experimental results highlight several key findings:

1. **Model Scaling:** - Larger model configurations didn't necessarily yield better results - Memory-optimized versions showed comparable performance - Training efficiency was crucial for iteration

2. **Resource Utilization:** - GPU memory constraints significantly influenced design choices - Training time impacted experimentation capacity - Batch size limitations affected optimization

These findings suggest that while BigBird's sparse attention mechanism theoretically allows for processing longer sequences, practical implementation faces significant challenges in the ListOps task context.

3.2.6 Comparative RNN Implementation

To establish a baseline and compare with BigBird's performance, an LSTM-based model with attention mechanism was implemented: **Architecture Details:**

- Base: Bidirectional LSTM
- Hidden size: 256
- Number of layers: 2
- Attention: Custom attention mechanism over LSTM outputs
- Dropout: 0.3 for regularization

Training Configuration:

- Batch size: 32
- Optimizer: AdamW
- Learning rate: 1e-3
- Mixed precision training: Enabled

The RNN model achieved comparable performance to BigBird, reaching a validation accuracy of 19.22

4 References