LRA ListOps Benchmark

An Experimental Study

Abercha Zineb
Fri Yassir
Kemmoune Anass

Supervised by Pr. Hamza Alami Issam Ait Yahya

December 9, 2024

Table of Contents

Project Context

- Benchmarking multiple AI models on the ListOps dataset.
- Evaluates hierarchical reasoning in long-context scenarios.
- Essential for understanding model limitations and strengths.

Objectives

1. Assess performance of state-of-the-art models on ListOps.
2. Investigate impact of sequence length on hierarchical reasoning.
3. Establish baselines for long-context reasoning tasks.

Overview of the ListOps Dataset

- Proposed by Nangia and Bowman (2018).
- Involves hierarchical structures and logical operators: `MAX`, `MEAN`, `MEDIAN`, `SUM MOD`.
- Task: Parse structure to predict correct output.

Data Generation

Base Dataset:

- Generated using original ListOps script.
- Produced three TSV files: Training, Test, Validation sets.

Depth-20 Dataset:

- Additional dataset with tree depth of 20.

ListOps: Task Description

- Extended version with sequence lengths up to 2K tokens.
- Example Sequence:

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

- Ten-way classification task.
- Challenges for neural models in hierarchical reasoning.

Overview of Models

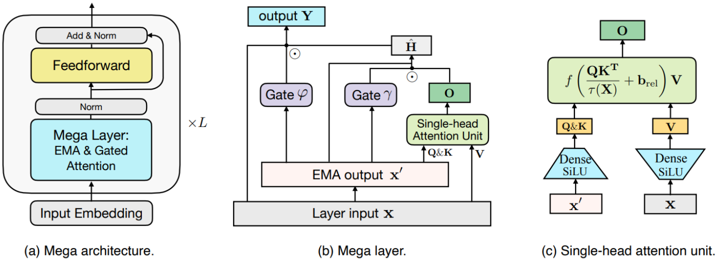
1. MEGA Model (Moving Average Equipped Gated Attention)
2. Longformer Model
3. BigBird Model
4. Comparative RNN
5. MobileBERT
6. Reformer
7. GPT-2

Challenges with Transformers

- **Low Inductive Bias:**
  - Transformers lack inherent assumptions about sequences, making them less efficient in learning hierarchical patterns.
- **$O(n^2)$ Complexity:**
  - Each token attends to every other token, leading to quadratic scaling with sequence length and high computational costs.

MEGA Model Overview

- Stands for Moving Average Equipped Gated Attention.
- Designed for long-context reasoning.
- Utilizes gated attention with moving averages.



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

Figure: MEGA Paper Reference

Training Approach

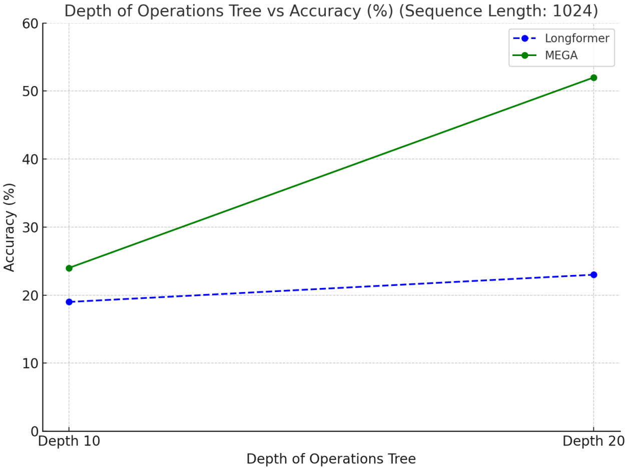
1. **Model Initialization:**
   - Loaded directly from Hugging Face's `MegaModel`.
2. **Loss Function:**
   - Standard cross-entropy loss with custom weighting.
3. **Training Duration and Early Stopping:**
   - Trained for 10 epochs with early stopping at epoch 9.

Training Configuration

- **Optimizer:** AdamW
- **Learning Rate Scheduler**
- **Batch Size:** 128 (seq length 1024), 2 (seq length 8192)
- **Runtime Environment:**
  - Kaggle P100 GPU (16 GB vRAM)
  - Python 3.8, PyTorch, Transformers

Performance Metrics

- Evaluated using classification accuracy on the ListOps dataset.
- MEGA achieved higher accuracy compared to Longformer.



PyTorch Lightning Wrapper

- **Simplified Training Pipeline:** Reduces boilerplate code, focusing on model architecture.
- **Enhanced Scalability:** Easily scales from single GPU to multiple GPUs or TPUs.
- **Improved Reproducibility:** Consistent training with automated logging and checkpointing.
- **Modular Code Structure:** Clear separation of data, model, and training logic.
- **Integrated Tools and Optimizations:** Supports mixed-precision training and performance optimizations.

PyTorch Lightning Implementation

```python
class LRADataModule(pl.LightningDataModule):
    def __init__(self, train_df, test_df, tokenizer, max_len, batch_size):
        super().__init__()
        self.train_df = train_df
        self.test_df = test_df
        self.tokenizer = tokenizer
        self.max_len = max_len
        self.batch_size = batch_size

    def setup(self, stage=None):
        self.train_dataset = LRADataset(
            texts=self.train_df["Source"].to_numpy(),
            labels=self.train_df["Target"].to_numpy(),
            tokenizer=self.tokenizer,
            max_len=self.max_len,
        )
        self.test_dataset = LRADataset(
            texts=self.test_df["Source"].to_numpy(),
            labels=self.test_df["Target"].to_numpy(),
            tokenizer=self.tokenizer,
            max_len=self.max_len,
        )

    def train_dataloader(self):
        return DataLoader(self.train_dataset, batch_size=self.batch_size, shuffle=True)

    def val_dataloader(self):
        return DataLoader(self.test_dataset, batch_size=self.batch_size)
```

Figure: PyTorch Lightning Training Loop

Longformer Model Overview

- Implements sliding window attention mechanism.
- Designed for processing longer sequences efficiently.
- Achieves linear complexity with respect to sequence length.

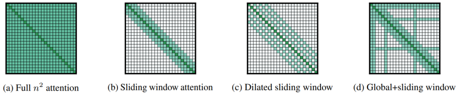


Figure: Longformer attention window

Training Configuration

- **Epochs:** 3
- **Batch Size:** 2
- **Max Sequence Length:** 8192
- **Runtime Environment:**
    - Kaggle P100 GPU
    - Python 3.8, PyTorch, Transformers

Performance Metrics

- Classification accuracy lower than MEGA.
- Peak accuracy: 23% on 8K tokens.
- Advantages:
  - Faster inference times.
  - Efficient memory usage.

BigBird Model Overview

- Implements sparse attention combining global, window, and random patterns.
- Efficient for processing longer sequences.
- Achieves linear complexity while maintaining performance.

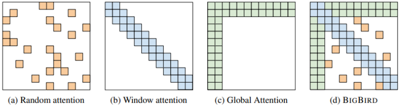


Figure: BigBird: Transformers for Longer Sequences (Zaheer et al., 2020)

Training Configurations - Common Setup

- **Common Training Setup:**
  - Optimizer: AdamW
  - Loss Function: Standard cross-entropy loss
  - Runtime Environment: Kaggle T4 GPU
  - Framework: Python 3.8, PyTorch, Transformers

1. **Initial Configuration:**
   - Hidden size: 512
   - Attention heads: 8
   - Intermediate size: 2048
   - Hidden layers: 6
   - Block size: 64
   - Max position embeddings: 4096

Training Configurations - Alternative Setups

- sun **Configuration with Depth-20:**
  - Hidden size: 8
  - Attention heads: 2
  - Intermediate size: 512
  - Hidden layers: 2
  - Block size: 64
  - Max position embeddings: 1024

Performance Comparison

Configuration	Accuracy	Training Time/Epoch	Batch Size
Initial	19.22%	15 min	8
Depth-20	22.19%	15 min	10

Table: BigBird Performance Across Configurations

RNN-Based Model Architecture

- Bidirectional LSTM with custom attention mechanism.
- **Architecture Details:**
  - Hidden size: 256
  - Number of layers: 2
  - Dropout: 0.3
- **Attention Mechanism:** Custom attention with learned weights.
- **Input Processing:**
  - Custom vocabulary mapping.
  - Padding and attention masking.
  - Sequence truncation for long inputs.

Training Configuration

- **Batch Size:** 32
- **Optimizer:** AdamW
- **Learning Rate:** 1e-3
- **Mixed Precision Training:** Enabled via GradScaler
- **Maximum Sequence Length:** 4096

Performance Metrics

- Validation Accuracy: 19.22%
- Comparable to BigBird's initial configuration.

Reformer Model Overview

- Introduced by Nikita Kitaev et al. in 2020.
- Reduces complexity using locality-sensitive hashing (LSH) attention.
- Handles long sequences with efficient memory usage.

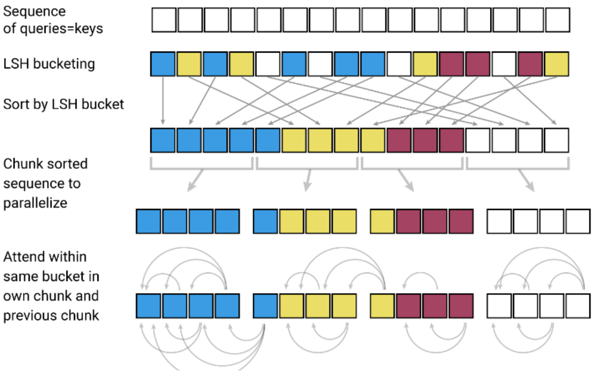


Figure: Reformer: LSH Attention

Training Configuration

- **Epochs:** 5
- **Batch Size:** 64
- **Max Sequence Length:** 1024
- **Optimizer:** AdamW
- **Learning Rate:** 1e-3
- **Runtime Environment:** Kaggle P100 GPU

Reformer Model Configuration

The following are the key configuration parameters used for the Reformer model:

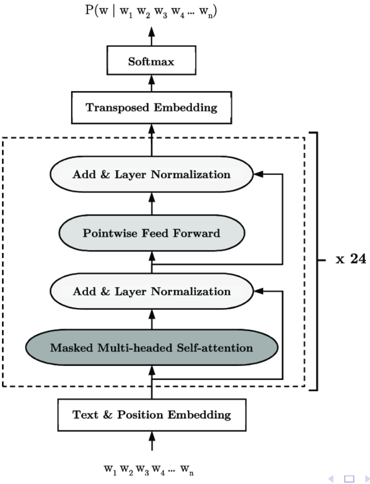
- **Hidden Size:** 64 (dimensionality of hidden states)
- **Number of Layers:** 6 (stacked hidden layers)
- **Attention Heads:** 8 (multi-head attention)
- **Attention Head Size:** 32
- **Axial Positional Embedding Dimensions:** [32, 32]
- **Axial Position Shape:** [32, 32]
- **Attention Layers:** ["local", "lsh", "local", "lsh", "local", "lsh"]
- **Feed Forward Size:** 512 (intermediate layer size)
- **Dropout: 10%**
- **Local Chunk Length:** 64

Performance Metrics

- Achieved validation accuracy of 25.60% on the ListOps dataset (depth 20).
- Advantages:
  - Scales to very long sequences.
  - Low memory footprint compared to other models.
- Disadvantes:
  - Low performance compared to other models.

GPT-2 Model Overview

- Transformer-based model with autoregressive architecture.
- Pretrained on diverse datasets for language generation tasks.
- Fine-tuned for classification tasks on ListOps.



Training Configuration

- **Epochs:** 5
- **Batch Size:** 8
- **Max Sequence Length:** 512
- **Optimizer:** AdamW
- **Learning Rate:** 1e-3
- **Runtime Environment:** Kaggle P100 GPU

GPT-2 Model Configuration

The following are the key configuration parameters used for the GPT-2 model:

- **Max Positions:** 512 (optimized for ListOps sequence length)
- **Hidden Size:** 32 (dimensionality of hidden states)
- **Number of Layers:** 4 (smaller model size)
- **Number of Attention Heads:** 4 (efficient memory usage)
- **Feed-Forward Size:** 512 (inner layer dimensionality)
- **Activation Function:** GELU (Gaussian Error Linear Unit)
- **Dropout: 10%**

GPT-2 Performance Metrics

- Achieved validation accuracy of 50.7%.
- Advantages:
  - High performance in comparison with other models.
  - Faster training (3min / epoch).
- Disadvantages:
  - Performance decreases with longer sequences.
  - High memory cost.

Training Configuration and Validation Scores

Model	Epochs	Max Seq Length	Batch Size	Final Validation Score
Longformer	3 (50 min/epoch)	4096	2	18%
MEGA	10 (9 min/epoch)	1024	128	52%
BigBird (Initial)	10 (15 min/epoch)	4096	8	19.22%
BigBird (Depth-20)	10 (15 min/epoch)	1024	10	22.19%
RNN-Based	10 (8 min/epoch)	4096	32	19.22%
MobileBERT	5 (15 min/epoch)	512	8	11.10%
Reformer (Depth-20)	5 (16 min/epoch)	1024	64	25.60%
GPT-2 (Depth-20)	5 (3 min/epoch)	512	8	50.70%

Table: Training Configurations and Validation Scores

Comparative Analysis

- **MEGA:** Consistently outperforms Longformer and BigBird.
- **Efficiency:** MEGA is more efficient with higher accuracy on shorter sequences.
- **Longformer:** Offers faster inference times but lower accuracy.
- **BigBird:** Shows variable performance based on configuration.
- **RNN-Based Model:** Provides a solid baseline.
- **Reformer:** Efficient handling of longer sequences with axial embeddings but scores poorly.
- **GPT-2:** Achieves similar performances to MEGA using a reduced configuration.

Generalization Across Sequence Lengths

- MEGA and GPT-2 excels on shorter sequences (1K and 2K tokens).
- MEGA Maintains robustness at 8K tokens.
- Longformer and Reformer struggle with longer contexts despite efficient attention mechanism.

Conclusion

- **MEGA:**
  - Superior performance across tested sequence lengths.
- **Longformer and Reformer:**
  - Efficient windowed attention.
  - Lower accuracy on hierarchical reasoning tasks.
- **BigBird:**
  - Sparse attention mechanism.
  - Performance heavily dependent on configuration.
- **RNN-Based Model:**
  - Provides a strong baseline.
  - Comparable performance to BigBird's initial configuration.
- **GPT-2:**
  - Good performance on short-length sequences.

Final Thoughts

- Specialized attention mechanisms are crucial for long-context hierarchical reasoning.
- MEGA shows promise for complex tasks involving deep hierarchical structures.
- Future work could explore hybrid models or further optimization of existing architectures.

References I

- Zaheer, M., Guruganesh, G., Dubey, K., Ainslie, J., Alberti, C., Ontañón, S., ... & Stoyanov, V. (2020). *BigBird: Transformers for Longer Sequences*. Retrieved from https://arxiv.org/pdf/2007.14062
- Google Research. (2023). *Long Range Arena*. Retrieved from https://github.com/google-research/long-range-arena
- Mega: Moving Average Equipped Gated Attention. Retrieved from https://doi.org/10.48550/arXiv.2209.10655
- Hugging Face. (2024). *MegaModel and GPT-2*. Retrieved from https://huggingface.co/models
- Kitaev, N., Kaiser, Ł., Levskaya, A. (2020). *Reformer: The Efficient Transformer*. Retrieved from https://arxiv.org/abs/2001.04451