Generating Long Sequences with Sparse Transformers

Dominik Soós
Department of Computer Science
Old Dominion University
dsoos001@odu.edu

Sushmitha Halli Sudhakara
Department of Computer Science
Old Dominion University
shall051@odu.edu

May 30, 2024

Abstract

Transformers have revolutionized the modeling of sequential data due to their ability to capture long-range dependencies. However, their quadratic computational complexity with respect to sequence length limits their applicability for very long sequences. The "Generating Long Sequences with Sparse Transformers" paper introduces an innovative approach to mitigate this issue by employing sparse factorizations of the attention matrix, thus reducing the complexity to $O(n\sqrt{n})$. This study aims to reproduce the results presented in the original paper, focusing on the application of Sparse Transformers to text and image data using fixed and strided sparse factorization methods. Utilizing the EnWik8 and CIFAR-10 datasets, we replicated the experiments under varied computational constraints. Our results confirm that while Sparse Transformers maintain competitive performance, the replication of the original results depends heavily on specific model configurations and training durations. Our findings highlight the challenges in reproducing state-of-the-art methods and emphasize the necessity of detailed documentation for ensuring reproducibility in the machine learning community. Our code can be found on `https://github.com/sushmitha047/cs795-deeplearning/tree/main/Final-project`

1 Introduction

In the realm of machine learning, Transformers have set a new standard for handling sequential data by effectively capturing long-range dependencies. However, their widespread adoption is constrained by their quadratic computational need in relation to sequence length. Addressing this limitation, the paper "Generating Long Sequences with Sparse Transformers" introduces a method that leverages sparse factorizations of the attention matrix. This approach significantly curtails these computational complexities to $O(n\sqrt{n})$, making it feasible to handle much longer sequences. This innovation opens new possibilities for applying Transformers to a broader range of tasks involving both text and image data.

1.1 Project Type: Reproduction

Chosen paper: Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers.
`https://arxiv.org/pdf/1904.10509v1.pdf`

1.2 Motivation

The primary motivation for this project stems from the necessity to extend the applicability of Transformers to longer sequences without incurring prohibitive computational costs. By introducing sparse factorizations of the attention matrix, this research seeks to overcome the existing scalability challenges and democratize the use of advanced machine learning models across more extensive datasets and diverse applications.

1.3 Goal of the Project

The primary goal of this project is to evaluate the reproducibility of the results claimed in the "Generating Long Sequences with Sparse Transformers" paper, which introduced Sparse Transformers as a solution to the computational and memory inefficiencies encountered when using standard Transformer models for very long sequences. By implementing and testing the proposed Sparse Transformer models on the same datasets and tasks described in the original paper, this study aims to validate the efficacy and practicality of the Sparse Transformer in different settings and under different computational constraints.

1.4 Task to be Solved

The task involves replicating the Sparse Transformer architecture and its variants—specifically the fixed and strided sparse factorizations—to model long sequences of text and images. This includes:

1. Reconstructing the Sparse Transformer architecture as described in the original paper.
2. Training the models on the EnWik8 and CIFAR-10 datasets using the parameters and methods outlined in the source material.
3. Comparing the performance of these models against the reported results in terms of efficiency and accuracy, and under different hardware setups.

1.5 Major Contributions

This reproducibility study makes several important contributions to the field of machine learning, particularly in the domain of sequence modeling:

- **Verification of Sparse Transformer Performance**: By replicating the experiments from the original paper, this study provides an independent verification of the Sparse Transformer's ability to efficiently handle extremely long sequences using less computational resources compared to traditional full attention models.
- **Impact of Computational Resources**: This study explores how variations in available computational resources affect the performance of Sparse Transformers, providing insights into their scalability and practical applicability in resource-constrained environments.
- **Reproducibility and Transparency**: Our findings contribute to the ongoing discussion about reproducibility in machine learning research. We highlight potential discrepancies and elucidate the conditions necessary to replicate published results, emphasizing the need for detailed documentation and transparent sharing of model architectures and training procedures.

These contributions aim to solidify the understanding and trust in Sparse Transformer technology and encourage more robust and transparent research practices across the AI community.

2 Related work

The concept of optimizing the efficiency of Transformers has been an active area of research, leading to various adaptations of the original architecture. Child et al. [2019] introduced the concept of Sparse Transformers, utilizing a combination of strided and fixed sparse attention patterns to handle longer sequences with reduced computational overhead. In a related work, Kitaev et al. [2020] proposed the Reformer, which employs locality-sensitive hashing to reduce complexity by approximating the attention mechanism, enabling the processing of sequences up to lengths of one million tokens. Another significant advancement was presented by Beltagy et al. [2020], with the Longformer, which extends the Transformer model for long documents by using a combination of sliding window and global attention mechanisms. Wang et al. [2020] explored the use of Linformer, which linearizes the self-attention mechanism to scale linearly with sequence length, thus making it practical for very long sequences. Lastly, Zaheer et al. [2020] contributed with Big Bird, which uses a combination of random, window, and global attention mechanisms to manage sequence lengths previously unattainable with standard Transformers. These studies collectively represent a significant leap toward making Transformer models more scalable and applicable across various longer sequence tasks in natural language processing and beyond.

3 Methods

In our study, we aimed to replicate the findings of Child et al. [2019] by using a Transformer architecture that incorporates Sparse Attention. This method modifies the standard Transformer by implementing sparse factorizations of the attention matrix. This adjustment significantly cuts down on both computational and memory demands. The primary innovation of Sparse Transformers is the reduction of time and memory complexity from $O(n^2)$ to $O(n\sqrt{n})$ through various sparse factorization techniques of the attention mechanism.

3.1 Fixed Sparse Factorization

This method was applied to text data (Enwiki8) and image data (CIFAR10). It involves using fixed attention patterns that segment data into blocks, within which attention is computed. This method maintains connectivity between all positions over several steps of attention, ensuring each output position can still access global context in an efficient manner.

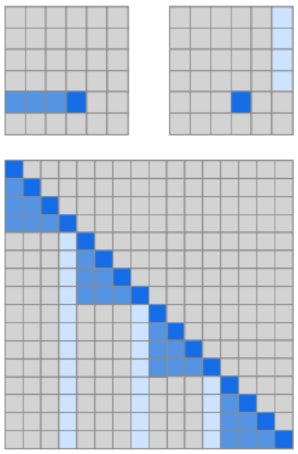


Figure 1: Fixed Sparse Factorization Techniques

3.2 Strided Sparse Factorization

Employed on text data (Enwiki8), this approach involves dividing the attention across every nth position (where n corresponds to the stride length, approximated by $\sqrt{n}$). This pattern was designed to optimize computational efficiency by reducing the number of positions each element needs to attend to, while still capturing the essential interactions across a sequence.

These methods underscore the flexibility and scalability of Sparse Transformers in modeling long sequences across different types of data by adapting the attention mechanism to reduce computational overhead while attempting to maintain or even enhance modeling capacity and efficiency.

4 Experiments

In our reproducibility study, we tested the Sparse Transformer model using fixed and strided sparse factorization techniques on different datasets to understand its performance across various configura-

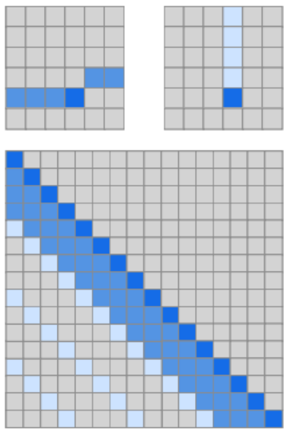


Figure 2: Strided Sparse Factorization Techniques

tions and compared these outcomes with those reported in the original paper. Below we detail the computational resources utilized

4.1 Settings

- **Original**
  The experiments were conducted over 7 days using 64 NVIDIA V100 GPUs, providing a total of 1,024 GB of memory. The configurations included models with 30 layers and 95M parameters trained for 80 epochs.
- **Our**
  Our experiments were facilitated using two main computational resources:
  1. **Google Colab Pro**
     - 1 NVIDIA V100 with 16 GB was used for most initial experiments.
     - 1 NVIDIA A100 with 40 GB for the Strided Sparse factorization on the EnWik8 dataset, managing 4 layers with 250K trainable parameters, with each epoch taking over 16 minutes for a batch size of 16.
  2. **Wahab Cluster**
     - 4 NVIDIA A100 GPUs totaling 320 GB were used for Fixed Sparse factorization on the EnWik8 and CIFAR-10 datasets.
     - An 8 layer model with 46M trainable parameters took more than 30 minutes per epoch for a batch size of 1024 maximizing the memory resources available.

4.2 Datasets

We employed two datasets: the EnWik8 dataset, which contains the first 100MB of cleanly formatted text from Wikipedia, and the CIFAR-10 dataset, consisting of 60,000 32x32 color images in 10 classes (50,000 training images and 10,000 test images). Each dataset was processed according to the model's input requirements, formatted as sequences of bytes for text and flat pixel arrays for images.

4.3 Baseline Methods

For both datasets, we used the results from the original paper by Child et al. [2019] as our baseline for comparison. These models utilized Adam optimizer, cross-entropy loss function, and were trained with specific dropout rates, number of layers, heads, and embedding sizes as reported.

4.4 Experiment Summaries

We conducted experiments using the fixed and strided sparse factorization techniques. The key hyperparameters and performance metrics for each setup are summarized in the Tables below. In

Tables 1 and 2, the bits per byte and bits per dimension serve as key indicators of model performance, quantifying how efficiently the model predicts the next byte or image pixel intensity. A lower score means the model needs fewer bits to make each prediction, revealing a tighter grasp on the dataset's patterns and a more effective compression capability. For instance, in the EnWik8 dataset from Table 1, the original model achieved an excellent score of 0.99 bits per byte, indicating its high efficiency in compressing textual data. The reproduced models, however, scored slightly higher, with the fixed version at 1.03 and the strided at 3.41, suggesting a decrease in predictive efficiency in these iterations.

Table 1: Experiment Summary for EnWik8 Dataset

Factorization Method	LR	Dropout	Layers	Heads	Embedding	Epochs	Params	Bits/Byte
Original (Fixed)	3.5e-4	0.4	30	8	512	120	95M	0.99
Reproduced (Fixed)	3.5e-4	0.4	12	8	512	10	46M	1.03
Original (Strided)	3.5e-4	0.4	30	8	512	80	-	1.13
Reproduced (Strided)	3.5e-4	0.25	2	2	64	10	253,440	3.41

When it comes to image data, the bits per dimension in Table 2 follows a similar principle, relating to how well the model predicts the color intensity of each pixel in the CIFAR-10 dataset. The original fixed model recorded 2.8 bits per dimension, a benchmark that the reproduced model was nearly able to meet with a score of 2.9. These metrics, derived from the Cross-Entropy Loss, which measures the model's accuracy in predicting the observed labels—directly reflect the model's understanding and representational capacity of the dataset. Lower scores in these metrics are desirable, as they signify a model's improved efficiency and potential for practical applications, although these results must be interpreted with an understanding that 'LR' denotes learning rate and 'Params' refers to the trainable parameters, which are the model's adjustable factors that influence its predictions.

Table 2: Experiment Summary for CIFAR-10 Dataset

Factorization Method	LR	Dropout	Layers	Heads	Embedding	Epochs	Params	Bits/Dim
Original (Fixed)	3.5e-4	0.25	128	2	256	120	59M	2.80
Reproduced (Fixed)	3.5e-4	0.25	128	2	256	15	189M	2.93

5 Discussion

In the reproduction of the models for both the EnWik8 and CIFAR-10 datasets, we observed a consistent pattern where the efficiency of data compression, as measured by bits per byte or bits per dimension, was not on par with the original models.

A pivotal factor in this divergence is the constrained computational resources at our disposal. Specifically, limitations in both memory and processing time posed significant restrictions on the extent of the models' training. For instance, the reduced number of epochs and the necessity to work with fewer layers and heads in the reproduced models led to a less nuanced understanding of the datasets, which is evident in the increased bits per byte/dimension values. It is clear that the resource-intensive nature of training deep learning models, such as Transformers, plays a critical role in achieving the benchmark results originally reported, and any shortfall in these resources directly translates to diminished model performance.

6 Conclusions

Our reproducibility efforts highlighted the challenges and impacts of altering model complexities and training durations. The outcomes affirm the importance of sufficient training epochs and appropriate layering in achieving close approximations to baseline performances. This study also underscored the substantial computational resources required for such deep learning experiments. The original paper's experiments were conducted using 64 NVIDIA V100 GPUs over a period of 7 days, signifying a high demand for both computational power and energy.

Such extensive use of computational resources not only incurs significant financial costs but also has a considerable environmental impact due to the energy consumption involved in powering and cooling the GPUs. The carbon footprint associated with training state-of-the-art machine learning models is a growing concern, as it contributes to the broader environmental impact of the tech industry.

Given these factors, it is essential to consider both the economic and environmental costs when designing and executing machine learning experiments. Further investigations with adjustments to training durations and configurations are recommended to validate the Sparse Transformer's efficacy comprehensively. Additionally, efforts to optimize computational efficiency and reduce energy consumption should be integral to future research in machine learning, aiming to balance performance achievements with sustainable practices.

References

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

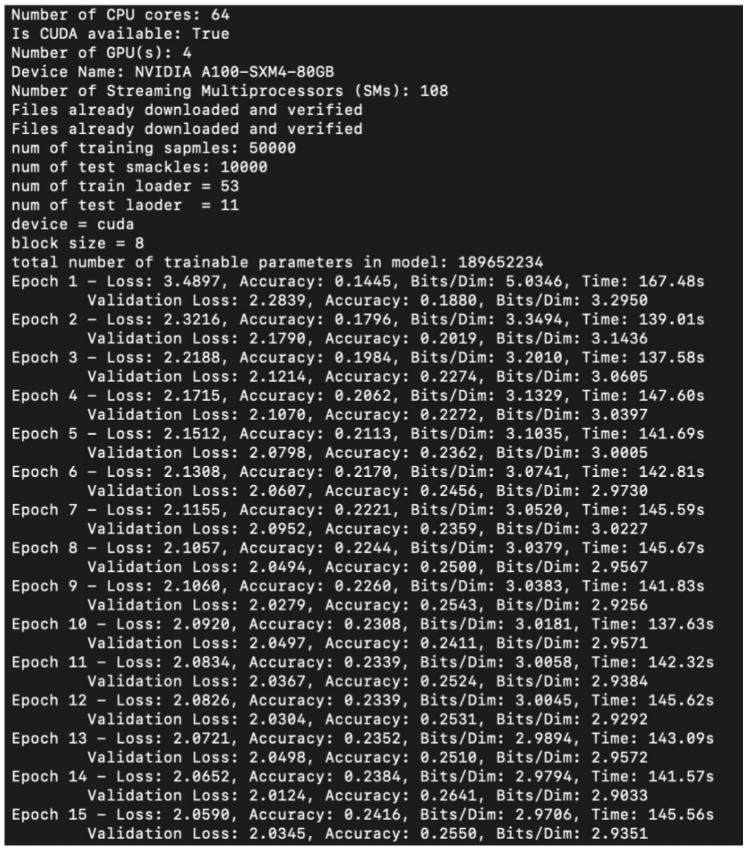
Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

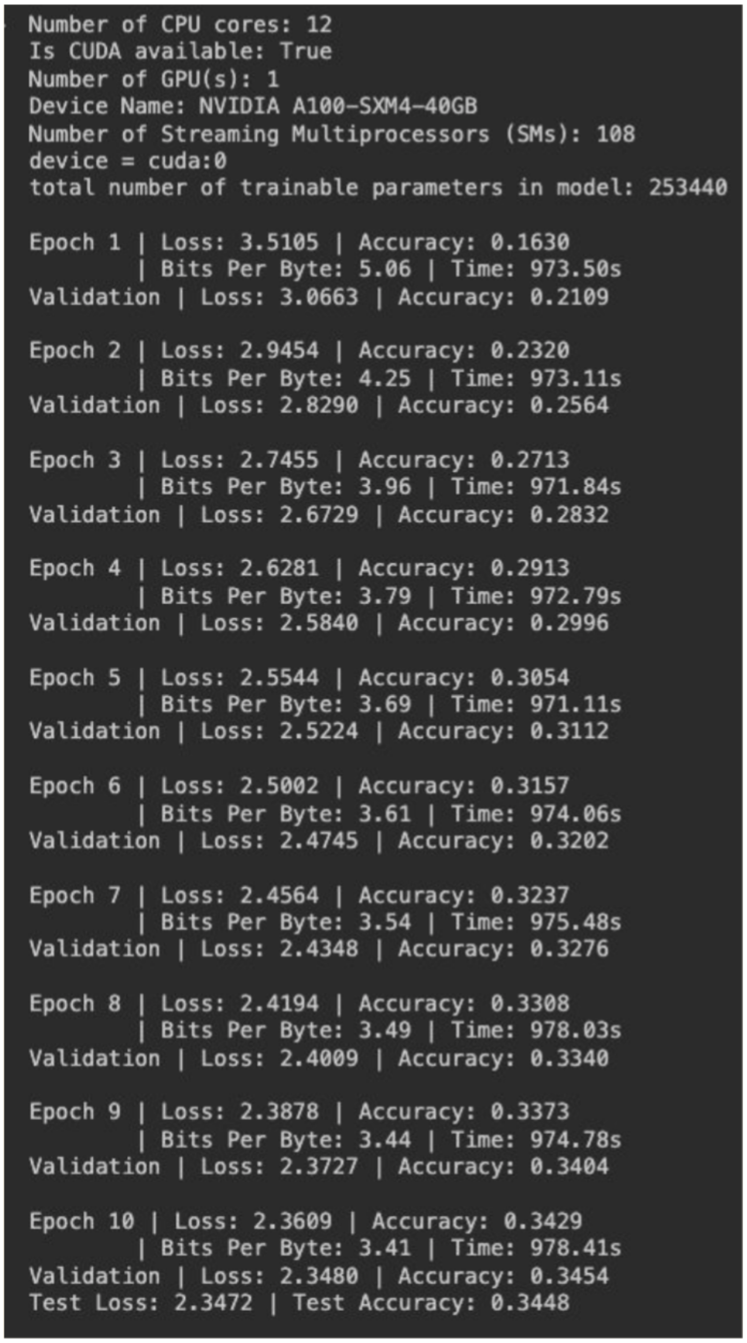
Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

A Appendix



```
Number of CPU cores: 64
Is CUDA available: True
Number of GPU(s): 4
Device Name: NVIDIA A100-SXM4-80GB
Number of Streaming Multiprocessors (SMs): 108
Files already downloaded and verified
Files already downloaded and verified
num of training sapmles: 50000
num of test smackles: 10000
num of train loader = 53
num of test laoder  = 11
device = cuda
block size = 8
total number of trainable parameters in model: 189652234
Epoch 1 - Loss: 3.4897, Accuracy: 0.1445, Bits/Dim: 5.0346, Time: 167.48s
        Validation Loss: 2.2839, Accuracy: 0.1880, Bits/Dim: 3.2950
Epoch 2 - Loss: 2.3216, Accuracy: 0.1796, Bits/Dim: 3.3494, Time: 139.01s
        Validation Loss: 2.1790, Accuracy: 0.2019, Bits/Dim: 3.1436
Epoch 3 - Loss: 2.2188, Accuracy: 0.1984, Bits/Dim: 3.2010, Time: 137.58s
        Validation Loss: 2.1214, Accuracy: 0.2274, Bits/Dim: 3.0605
Epoch 4 - Loss: 2.1715, Accuracy: 0.2062, Bits/Dim: 3.1329, Time: 147.60s
        Validation Loss: 2.1070, Accuracy: 0.2272, Bits/Dim: 3.0397
Epoch 5 - Loss: 2.1512, Accuracy: 0.2113, Bits/Dim: 3.1035, Time: 141.69s
        Validation Loss: 2.0798, Accuracy: 0.2362, Bits/Dim: 3.0005
Epoch 6 - Loss: 2.1308, Accuracy: 0.2170, Bits/Dim: 3.0741, Time: 142.81s
        Validation Loss: 2.0607, Accuracy: 0.2456, Bits/Dim: 2.9730
Epoch 7 - Loss: 2.1155, Accuracy: 0.2221, Bits/Dim: 3.0520, Time: 145.59s
        Validation Loss: 2.0952, Accuracy: 0.2359, Bits/Dim: 3.0227
Epoch 8 - Loss: 2.1057, Accuracy: 0.2244, Bits/Dim: 3.0379, Time: 145.67s
        Validation Loss: 2.0494, Accuracy: 0.2500, Bits/Dim: 2.9567
Epoch 9 - Loss: 2.1060, Accuracy: 0.2260, Bits/Dim: 3.0383, Time: 141.83s
        Validation Loss: 2.0279, Accuracy: 0.2543, Bits/Dim: 2.9256
Epoch 10 - Loss: 2.0920, Accuracy: 0.2308, Bits/Dim: 3.0181, Time: 137.63s
        Validation Loss: 2.0497, Accuracy: 0.2411, Bits/Dim: 2.9571
Epoch 11 - Loss: 2.0834, Accuracy: 0.2339, Bits/Dim: 3.0058, Time: 142.32s
        Validation Loss: 2.0367, Accuracy: 0.2524, Bits/Dim: 2.9384
Epoch 12 - Loss: 2.0826, Accuracy: 0.2339, Bits/Dim: 3.0045, Time: 145.62s
        Validation Loss: 2.0304, Accuracy: 0.2531, Bits/Dim: 2.9292
Epoch 13 - Loss: 2.0721, Accuracy: 0.2352, Bits/Dim: 2.9894, Time: 143.09s
        Validation Loss: 2.0498, Accuracy: 0.2510, Bits/Dim: 2.9572
Epoch 14 - Loss: 2.0652, Accuracy: 0.2384, Bits/Dim: 2.9794, Time: 141.57s
        Validation Loss: 2.0124, Accuracy: 0.2641, Bits/Dim: 2.9033
Epoch 15 - Loss: 2.0590, Accuracy: 0.2416, Bits/Dim: 2.9706, Time: 145.56s
        Validation Loss: 2.0345, Accuracy: 0.2550, Bits/Dim: 2.9351
```

Figure 3: Training Output of Fixed Sparse Attention mechanism on the CIFAR-10 dataset



```
Number of CPU cores: 12
Is CUDA available: True
Number of GPU(s): 1
Device Name: NVIDIA A100-SXM4-40GB
Number of Streaming Multiprocessors (SMs): 108
device = cuda:0
total number of trainable parameters in model: 253440

Epoch 1 | Loss: 3.5105 | Accuracy: 0.1630
        | Bits Per Byte: 5.06 | Time: 973.50s
Validation | Loss: 3.0663 | Accuracy: 0.2109

Epoch 2 | Loss: 2.9454 | Accuracy: 0.2320
        | Bits Per Byte: 4.25 | Time: 973.11s
Validation | Loss: 2.8290 | Accuracy: 0.2564

Epoch 3 | Loss: 2.7455 | Accuracy: 0.2713
        | Bits Per Byte: 3.96 | Time: 971.84s
Validation | Loss: 2.6729 | Accuracy: 0.2832

Epoch 4 | Loss: 2.6281 | Accuracy: 0.2913
        | Bits Per Byte: 3.79 | Time: 972.79s
Validation | Loss: 2.5840 | Accuracy: 0.2996

Epoch 5 | Loss: 2.5544 | Accuracy: 0.3054
        | Bits Per Byte: 3.69 | Time: 971.11s
Validation | Loss: 2.5224 | Accuracy: 0.3112

Epoch 6 | Loss: 2.5002 | Accuracy: 0.3157
        | Bits Per Byte: 3.61 | Time: 974.06s
Validation | Loss: 2.4745 | Accuracy: 0.3202

Epoch 7 | Loss: 2.4564 | Accuracy: 0.3237
        | Bits Per Byte: 3.54 | Time: 975.48s
Validation | Loss: 2.4348 | Accuracy: 0.3276

Epoch 8 | Loss: 2.4194 | Accuracy: 0.3308
        | Bits Per Byte: 3.49 | Time: 978.03s
Validation | Loss: 2.4009 | Accuracy: 0.3340

Epoch 9 | Loss: 2.3878 | Accuracy: 0.3373
        | Bits Per Byte: 3.44 | Time: 974.78s
Validation | Loss: 2.3727 | Accuracy: 0.3404

Epoch 10 | Loss: 2.3609 | Accuracy: 0.3429
        | Bits Per Byte: 3.41 | Time: 978.41s
Validation | Loss: 2.3480 | Accuracy: 0.3454
Test Loss: 2.3472 | Test Accuracy: 0.3448
```

Figure 4: Training Output of Strided Sparse Attention mechanism on the EnWik8 dataset