Optimizing Packing and Shuffling Strategies for Enhanced Performance in Generative Language Models

Anonymous ACL submission

Abstract

Packing and shuffling tokens is a common practice in training auto-regressive language models to prevent overfitting and improve efficiency. Documents are typically concatenated to chunks of maximum sequence length (MSL) and shuffled in chunks of tokens (atom-size chunk), possibly breaking context within documents. An alternative approach is padding, which only includes one document per chunk. To optimize both packing strategies (concatenation vs padding), we explored the optimal atom size for shuffling and compared performance and efficiency. We found that in the most common setup (where average document length is greater than MSL), matching atom size to MSL yields the lowest perplexity, controlling for dataset. Also, padding yields lower final perplexity than concatenation at the cost of lower efficiency.[1] This trade-off informs the choice of shuffling and packing methods in training LMs.

1 Introduction

Shuffling removes underlying chronological or thematic order in the original dataset, which reduces the risk of overfitting and improves generalizability (Nicolae et al., 2016; Shen et al., 2020; Zhong et al., 2023). For example, training a classifier for cats versus dogs on an unshuffled dataset with 5,000 images of each can lead to bias. If the first 5,000 gradient updates are solely from cat images, the model develops a "cat bias," making inference on dogs problematic. This issue can be avoided by interleaving cat and dog images prior to training, and this shuffling process is now a standard approach.

Although shuffling helps unbiased learning with independent samples, the optimal packing method for shuffling remains unclear for language models (Press, 2019; Abdou et al., 2022). For GPT models (Radford et al., 2019), the commonly used PyTorch Dataloader class concatenates and packs documents into chunks of a fixed size (usually MSL) before shuffling (hereafter referred to as "concat"). Another method is padding, which shuffles documents after padding them to a fixed size. Both methods achieve the goal of generating fixed-length sequences, but it is still an open question which method is more effective for GPT models.

The optimal shuffling unit size for packing ("atom size" for conciseness) is also unclear for language models. While the "atom" for vision tasks is naturally images, finding an atom size for texts is hard since language datasets contain documents or sequences with varying lengths.

We hypothesize that shuffling data in an atom size of MSL is best since the contextual information within each chunk is maximized. Specifically, the previous contexts that transformers rely on for next-token prediction is disrupted when the atom size is smaller than the MSL, as unrelated contextual fragments are concatenated together. On the contrary, the context of consecutive sequences would be dependent when the atom size is larger than the MSL, introducing correlation and bias. Using the MSL as the shuffling unit thus maintains data integrity and randomness.

Our experiments confirmed that when average document length is greater than MSL, packing and shuffling data in atom sizes of MSL indeed optimizes performance for both concat and padding methods. We also showed that padding results in better model performance than concat, albeit at the cost of efficiency due to more training steps. This advantage however disappears when documents are significantly shorter than MSL, as padding becomes an inferior packing method with samples dominated by redundant padding tokens.

[1]The codebase available ongithub:https://anonymous.4open.science/r/data-shuffling-3A4D/README.md

2 Method

2.1 Model Pretraining Setting

We pretrained GPT-2 124M and GPT-2 XL 1.5B models (Radford et al., 2019) on WikiText (average document length of 143 tokens) with each packing method—concat or padding—across various atom sizes and MSLs. Table 1 shows different configurations tested, with justifications provided in Appendix A.2. We used Alibi (Press et al., 2021) as a positional encoding that introduces no additional learnable parameters in order to control for the total parameter size regardless of MSLs. Padding affects the total training steps for the same data, as discussed in A.10. GPT-2 models were trained on 1 NVIDIA A100 GPU, and GPT-2 XL models were trained on 8 NVIDIA H100 GPUs. Appendices A.1, A.3, A.4, and A.5 provide details on dataset and filtering, implementations for data packing, explanation for parameter sizes with Alibi, and justification for the step size, respectively.

2.2 Evaluation and Comparison Metric

We used final perplexity and perplexity ranking to determine the optimal atom size for both packing methods (concat or padding). Calculations for final perplexity and perplexity ranking are explained in Appendix A.6. Under each MSL, we compared concat and padding models by final perplexity, learning efficiency (perplexity at given steps) and step size efficiency (steps per epoch).

model	MSL	Atom Size Choice for Both Concat and Padding.
GPT-2	32	8, 16, 32, 64, 128
GPT-2	64	16, 32, 64, 128, 256
GPT-2	128	32, 64, 128, 256

Table 1: MSL and atom size choices in GPT-2 (124M)

3 Results

3.1 Concat Experiments

We found that when average document length is greater than MSL, atom sizes smaller or larger than MSL increased perplexity, indicating that MSL is indeed the optimal atom size for concat. Figure 1 shows the training perplexity of concat models with different atom sizes ($s \in \{$0.25MSL, 0.5MSL, 1MSL, 2MSL, 4MSL$\}$) when MSL is 64. Among all atom sizes, 0.25MSL (purple) and 0.5MSL (red) obviously lead to higher perplexity (worse performance). Although the differences in perplexity among 4MSL (blue), 2MSL (orange) and 1MSL (green) are less obvious, the zoom-in plot (right side of Figure 1) shows that 1MSL (green) consistently had lower perplexity (better performance) than 2MSL and 4MSL (blue and orange) in the second epoch. Table 2 shows final perplexity and perplexity ranking respectively. The model using 1MSL as the atom size has the lowest final perplexity (118.08) and highest average ranking (1.05), indicating optimal performance. MSL = 32 and 128 yielded similar results (see Appendix A.8).

Atom Size	Final Perplexity		Perplexity Ranking	
	Concat	Padding	Concat	Padding
0.25MSL	207.04	175.33	5.00	5.00
0.5MSL	157.39	134.43	4.00	4.00
1MSL	**118.08**	**102.82**	**1.05**	**1.18**
2MSL	119.66	104.46	1.96	2.03
4MSL	121.18	105.85	2.99	2.79

Table 2: Comparison of final perplexity values and average perplexity rankings across different atom sizes for concat and padding models when MSL is 64.

3.2 Padding Experiments

For padding experiments with GPT-2 124M, we found that **atom sizes smaller or larger than MSL increased perplexity, confirming MSL as the optimal atom size when average document length is greater than MSL**. Figure 2 shows the training perplexity of padding models with atom sizes $s \in \{$0.25MSL, 0.5MSL, 1MSL, 2MSL, 4MSL$\}$ when MSL is 64. Note that atom sizes affect the training steps per epoch for models with paddings, as discussed in Appendix A.10. Similar to the concat experiments, 0.25MSL (purple) and 0.5MSL (red) lead to higher perplexity, while differences between 4MSL (blue), 2MSL (orange) and 1MSL (green) are subtle yet present: The zoom-in figure 2 (right side) shows that 1MSL (green) had lower perplexity compared to 2MSL and 4MSL (blue and orange) at the end of the second epoch, improving fast despite starting with a higher perplexity. Table 2 presents the final perplexity and perplexity ranking. The model with atom size of 1MSL has the lowest final perplexity (102.82) and highest average ranking (1.18), indicating optimal performance. Experiments with MSL of 32 or 128 yielded similar results (See Appendix A.8 for details).

3.3 Comparison between Padding and Concat

Although padding resulted in lower final perplexities (better performance) than concat, it has lower learning efficiency (higher perplexity at given

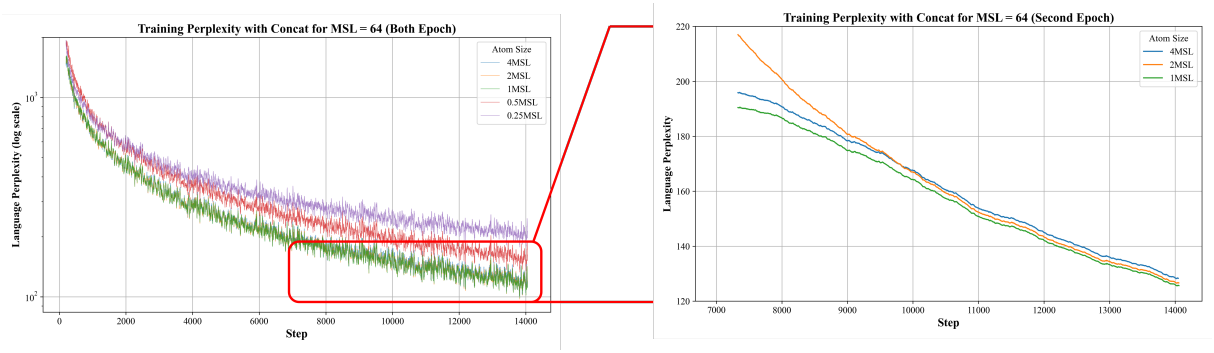


Figure 1: Comparison of concat models with different atom sizes when MSL is 64. The full training plot (left) shows that 1MSL (green) achieves the lowest perplexity, while smaller or larger atom sizes increase perplexity. The zoom-in plot (right) shows the second epoch, where 1MSL (green) consistently maintains the lowest perplexity.

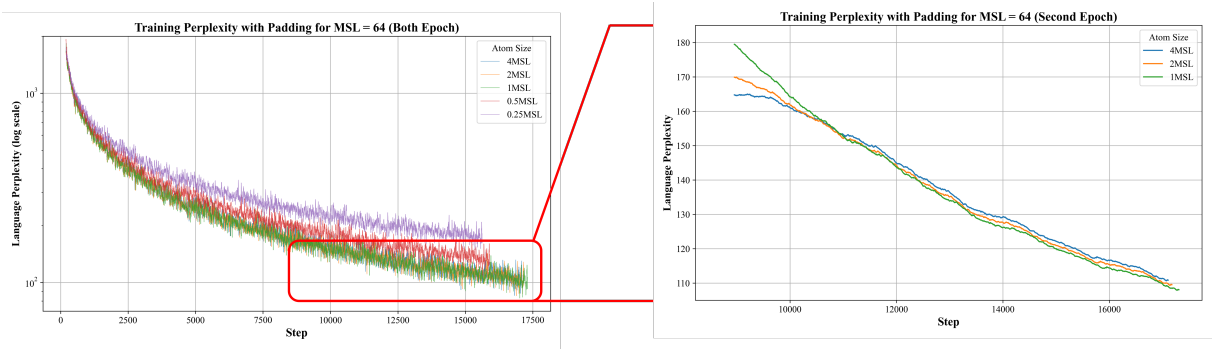


Figure 2: Comparison of padding models with different atom sizes when MSL is 64. The full training plot (left) shows that 1MSL (green) achieves the lowest perplexity, while smaller or larger atom sizes increase perplexity. The zoom-in plot (right) shows the second epoch, where 1MSL (green) achieves the best performance by the end.

steps) and step size efficiency (more steps per epoch). Table 3 compares the step size and final perplexity for concat and padding models when the atom size matches the MSL, showing that padding brings larger step (13% - 45% depending on MSL) and lower final perplexity (5% - 24% depending on MSL) than concat models across MSLs.

Figure 3 also shows the step-wise perplexity comparison for concat (blue) and padding (orange) models when the atom size matches the MSL (the first 2,000 high-perplexity steps are discarded for clarity). Again, we see that padding (orange) has lower final perplexities, while concat (blue) has fewer training steps.

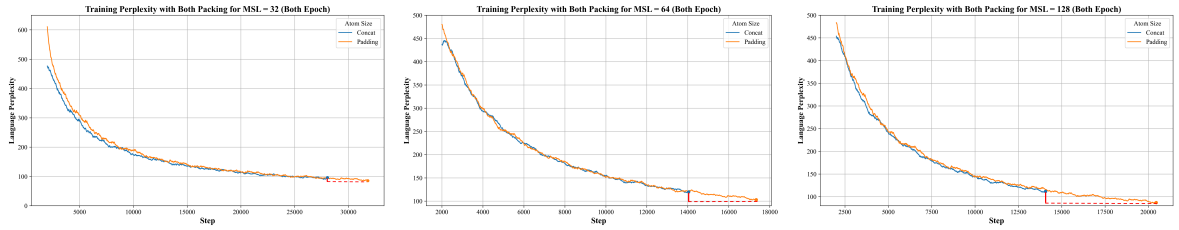
3.4 GPT-2 XL Experiments

We conducted ablation studies with GPT-2 XL 1.5B to test the scalability of our findings on a larger model and larger MSLs (256, 512, 1024). Unlike GPT-2 124M experiments, these MSLs exceed the average document length (143 tokens). This is uncommon in pretraining, as computational constraints from quadratic attention complexity typically force MSL to be shorter than documents.

In concat experiments, models with atom size of MSL showed the lowest perplexities (best performance), consistent with Section 3.1. However, in padding experiments, 1MSL, 2MSL, and 4MSL showed similar final perplexities and average perplexity rankings. In addition, the advantage of padding in reducing final perplexities, noted in Section 3.3, disappears at larger MSLs due to excessive padding tokens. See Appendix A.9 for details.

MSL	Batch Size	Step Size		Final Step Perplexity	
		Concat	Padding	Concat	Padding
32	256	**28120**	31816	91.01	**87.22**
64	256	**14058**	17308	110.45	**99.79**
128	128	**14056**	20496	102.42	**82.55**

Table 3: Concat models have fewer training steps while padding models have lower final perplexity.



(a) **MSL = 32.** (b) **MSL = 64.** (c) **MSL = 128.**

Figure 3: Step-wise perplexity comparison between padding and concat models under different MSLs. As marked by solid and dotted red lines, padding (orange) has lower final perplexities while concat (blue) has smaller steps.

4 Discussion

Effects of Matching MSL and Atom Size on Language Coherence and Bias When average document length is greater than MSL, matching MSL and atom size optimizes packing (padding and concat) by reducing language incoherence and bias as shown in Section 3.1 and 3.2. Our intuition is using an atom size smaller than MSL causes language incoherence, as it merges unrelated shuffling chunks into one sequence, damaging the contextual completeness. Conversely, using an atom size larger than MSL brings bias by splitting shuffling chunks into multiple consecutive sequences, creating unintended correlations.

Implication of the Trade-off Between Performance and Efficiency ML practitioners' choice of packing methods may be informed by the trade-off between performance and efficiency. With limited data, padding outperforms as it brings higher performance; with limited time, concat is preferred because it packs each epoch in fewer steps, leading to higher efficiency.

However, as shown in Section 3.4, padding should be avoided when MSL is significantly larger than average document lengths, as samples become largely dominated by padding tokens. This brings computational inefficiency and limits the amount of useful information in each training sample, preventing the model from learning adequate knowledge.

5 Related Work

Shuffle in PyTorch. While the DataLoader class in PyTorch shuffles data in concatenated chunks of a fixed atom size (usually MSL) to maximize training efficiency, our work explored multiple atom sizes (4MSL, 2MSL, 1MSL, 0.5MSL and 0.25MSL) as well as padding as an alternative packing method.

Data Shuffling Strategies for Context Preservation. Zhao et al. (Zhao et al., 2024) studied intra-document causal attention mask, which concatenates documents into fixed-length chunks and computes the likelihood of each token conditioned on previous tokens from the same document within that chunk. This is similar to padding because attention score is only calculated for intra-document tokens, but each token may not have full attention to other tokens in the same document since one document may be packed into different chunks. While this method saves padding tokens to improve efficiency, we implement the original padding method for better contextual completeness.

6 Conclusion

When average document length is greater than maximum sequence length (MSL) - a common setting as MSL is often constrained by quadratic complexity of attention computation while language documents can be much longer, we show that matching atom size with MSL optimizes packing performance (concat and padding). This finding shows the importance of aligning atom size with MSL during data shuffling to optimize language model training. We also found that padding yields lower final perplexity (higher performance) than concat at the cost of more training steps and lower efficiency. This trade-off guides packing choices in training models: padding is preferred when data is scarce, while concat is preferred when time is limited.

However, Section 3.4 shows that when MSL is greater than average document length, matching atom size with MSL only optimizes performance for concat. Padding is not ideal because wasteful padding tokens prevent efficient training.

Limitations and Future Work. We showed MSL as the optimal atom size for packing in GPT-2 124M and GPT-2 XL 1.5B models trained on WikiText. Future work can test datasets with longer documents and other model architectures to further confirm our findings.

References

Mostafa Abdou, Vinit Ravishankar, Artur Kulmizev, and Anders Søgaard. 2022. Word order does matter and shuffled language models know it. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6907–6919.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*.

Bogdan Nicolae, Carlos Costa, Claudia Misale, Kostas Katrinis, and Yoonho Park. 2016. Towards memory-optimized data shuffling patterns for big data analytics. In *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pages 409–412. IEEE.

Ofir Press. 2019. Partially shuffling the training data to improve language models. *arXiv preprint arXiv:1903.04167*.

Ofir Press, Noah A Smith, and Mike Lewis. 2021. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Min Shen, Ye Zhou, and Chandni Singh. 2020. Magnet: push-based shuffle service for large-scale data processing. *Proceedings of the VLDB Endowment*, 13(12):3382–3395.

Yu Zhao, Yuanbin Qu, Konrad Staniszewski, Szymon Tworkowski, Wei Liu, Piotr Miłoś, Yuxiang Wu, and Pasquale Minervini. 2024. Analysing the impact of sequence composition on language model pretraining. *arXiv preprint arXiv:2402.13991*.

Tianle Zhong, Jiechen Zhao, Xindi Guo, Qiang Su, and Geoffrey Fox. 2023. Rinas: Training with dataset shuffling can be general and fast. *arXiv preprint arXiv:2312.02368*.

A Appendix

A.1 Dataset and Filtering

We conducted our studies on the generative language model training using the WikiText dataset (Merity et al., 2016), chosen for its generalizability. Specifically, we used the WikiText-103-raw subset, which comprises approximately 1.81M rows and over 100M words derived from filtered Wikipedia content. Notably, the dataset mostly consists of short paragraphs: Figure 4 shows the distribution of tokenized sequence lengths using 10,000 randomly sampled rows from the dataset.

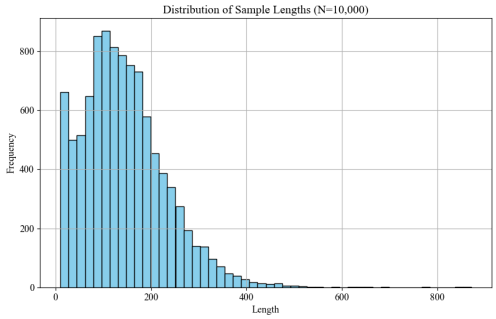


Figure 4: The distribution of tokenized sequence lengths in WikiText-103-raw with 10,000 random samples. The dataset mostly consists of short paragraphs with length 0 to 200.

Before tokenization or shuffling the WikiText dataset, we removed blank rows and short title rows that contained limited context information. We filtered out rows with fewer than 50 words. This filtered 55.62% rows (2.45% words) in the training set and 53.86% rows (2.33% words) in the validation set. The training and validation corpus sizes after filtering are 98,937,698 and 208,893 words respectively.

Before feeding dataset to models, we preprocessed the sequences by tokenization and packing. We first used GPT2TokenizerFast (Radford et al., 2019)to tokenize all sequences in parallel, then used one of the two packing methods (padding and concat) with shuffling to ensure that all sequences could be batched in MSL.

A.2 Choices of MSL and Atom Size Explained

We set MSL = 32, 64, 128 for GPT-2 124M to keep it smaller than average document lengths and save wasteful padding tokens. Atom sizes followed a geometric progression relative to MSL, set at 0.25MSL, 0.5MSL, 1MSL, 2MSL, 4MSL. However, for MSL = 128, we did not test on an atom size of 4 MSL = 512 because of wasteful padding tokens. The batch sizes were also adjusted according to MSL: we used batch sizes of 256 for MSLs of 32 and 64, while reducing to a batch size of 128 when MSL was 128. These adjustments were made to make optimal use of GPU memory.

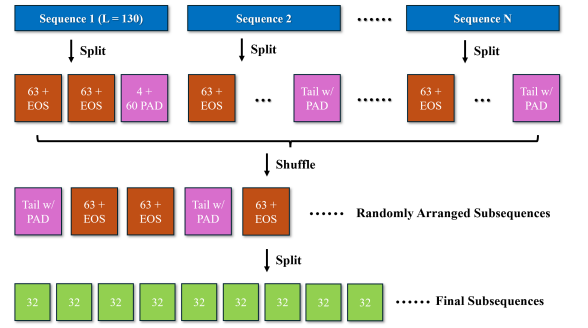


Figure 5: Illustration of packing steps of padding, when MSL is 32 and atom size is 64. The "tail" subsequence contains fewer tokens than the specified atom size and is padded to meet the MSL requirement, ensuring consistency in sequence length.

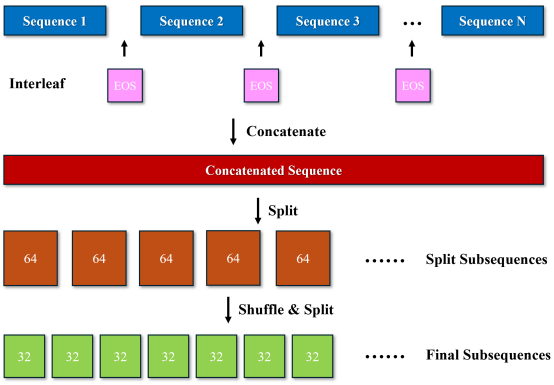


Figure 6: Illustration of packing steps of concat, with MSL of 32 and atom size of 64.

For GPT-2 XL 1.5B, we set MSL = 256, 512, and 1024, and tested all atom sizes regardless of padding token waste. The choices of MSL and atom size are detailed in table 4.

model	MSL	Atom Size Choice for Both Concat and Padding.
GPT-2	32	8, 16, 32, 64, 128
GPT-2	64	16, 32, 64, 128, 256
GPT-2	128	32, 64, 128, 256
GPT-2 XL	256	64,128,256,512,1024
GPT-2 XL	512	128, 256,512,1024,2048
GPT-2 XL	1024	256, 512,1024,2048,4096

Table 4: MSL and atom size choices in GPT-2 (124M) and GPT-2 XL (1.5B) models.

A.3 Concat and Padding Details

Padding. This method focuses on padding to generate sequences with lengths equal to MSL. The steps are shown in Figure 5. Each input document was segmented into smaller subsequences of length (atom size - 1) with an <EOS> token placed at the end. The role of the <EOS> token is to inform the model that the current sequence has ended. To maintain consistency in sequence length and ensure efficient batch processing, the tail end of any subsequence that does not meet the requirement of MSL would be padded. For example, in the case of MSL = 64, a sequence of length 130 ($L = 130$) would be segmented into 2 subsequences of length 64 (each with 63 word tokens and an <EOS> token at the end), then a tail subsequence composed of 4 word tokens and 60 padding tokens. We used <EOS> as the padding token for simplicity of the special token set. All resulted subsequences have a length of MSL regardless of the original sequence length.

Next, all subsequences were randomly shuffled with random seed set to 42. During this process, any underlying chronological or thematic order in the original dataset should be removed.

After shuffling, the subsequences were either merged or split to align with the predefined MSL. When atom size is less than MSL, we merged subsequences; when atom size is larger than MSL, we split subsequences. Finally, when atom size equals MSL, we kept the shuffled subsequences unchanged. For example, in the case where MSL is 32 and atom size is 64, we split every subsequences to get 2 final subsequences of length 32 to feed to the model.

Notably, when atom size is larger than MSL, we do not pad every tail end to atom size, but to MSL instead as shown in figure 5. This is because all subsequences will be split into size of MSL after shuffling. If we pad tail end to atom size instead of MSL, we will produce some training sequences that are completely composed of padding tokens. For example, when MSL is 32 and atom size is 128, if we pad a document of 35 tokens to an atom size of 128, we will yield a subsequence with 35 word tokens and 93 padding tokens, which will lead to completely meaningless training samples after split.

Concat. While the padding method handles different sequence lengths with padding tokens, the concat method employs a concatenating and splitting process. The steps are shown in Figure 6. In this approach, we firstly concatenate all sequences together to obtain an extremely long sequence interleaved with <EOS> tokens. Then we split the long sequence into subsequences according to atom size, shuffled them, and adjusted them to fit the max-

imum context length by merging or splitting as needed.

The two methods reflect different strategies to achieve the goal of generating fixed-length sequences for model training. The default method for GPT models is concat, but we hope to test whether padding outperforms concat in terms of efficiency and model performance. Our experiments in Section 3 would show that padding had better model performance than concat with the cost of lower efficiency. Since our experiments on shuffling atom size would focus on optimizing model performance, we decided to employ padding during those experiments.

It's important to note here that we also applied padding and concat respectively to the testing set during padding vs concat experiments. This choice is natural and reasonable because the training and testing sets should be preprocessed in the same way to accurately reflect performance. To make a fair comparison, we calculated perplexity with cross entropy loss normalized by max length, preventing bias due to differences in max length.

A.4 Total Parameter Size

When we experiment on models with different MSL, it is important to control the parameter size across all models. In the vanilla GPT-2 architecture, different MSLs lead to different parameter sizes. This is because the positional encoding layer's parameter size has a positive linear relationship with maximum context length. To eliminate this difference, we decide to replace the positional encoding with Attention with Linear Biases (Alibi) (Press et al., 2021), which is a non-parametric positional encoding algorithm that biases attention scores in accordance with the distance between tokens. This algorithm was originally designed to improve the processing of long sequences in language models and to reduce the computational load associated with longer inputs. However, our smaller-scale model with shorter maximum context lengths did not benefit from these advantages. Instead, we focused on one particular characteristic of Alibi: it does not introduce additional trainable parameters, unlike the default positional encoding in the GPT-2 architecture. As a result, the total parameter size of all models trained in our experiments were fixed to 124M.

A.5 Total Step Size

Our objective is to schedule and use computational resources in ways that minimize training time and adhere to optimal training practices. We estimate the optimal number of tokens for training based on the estimation table from Chinchilla (Hoffmann et al., 2022). The table indicates that a model with 400M parameters requires 8B tokens, so our 124M-parameter model will need $(124/400) \times 8 = 2.48$B tokens. Then we divided this number by batch size and maximum context length to calculate the optimal step size for training. However, due to limitations in computational resources, it would take us one day to train one model on the optimal number of tokens. In this case, we decided to run two full epochs (114,400,095 word tokens per epoch) for all experiments instead, bringing the perplexity to nearly convergence in 2 to 3 hours on one GPU (NVIDIA Tesla V100-PCIE-32GB).

A.6 Detailed Calculation of Final Perplexity and Perplexity Ranking

After evaluating with perplexity, we compared all models based on their average perplexity ranking and final perplexity value. Here, we need to be careful of how to calculate these two comparison metrics in detail.

We chose to calculate the comparison metrics by epochs rather than training steps due to variations in the number of word tokens learned at each step in padding models. For ranking, we divided the last epoch into 100 segments, each covering 0.01 of an epoch, calculated the average perplexity for all models within each segment, and ranked them. We then averaged the rankings from all 100 ranges as our final ranking. For the final perplexity value, we selected the last range (0.99 epoch - 1.00 epoch) and calculated its average perplexity.

A.7 Exponential Moving Average (EMA)

We use Exponential Moving Average (EMA) to visualize smooth perplexity curves for our models. EMA computes a weighted average of past data points, with exponentially decreasing weights that effectively smooth out fluctuations in original perplexity values. Specifically, EMA is computed iteratively using the following formula:

$$S_t = \alpha \cdot y_t + (1 - \alpha) \cdot S_{t-1}$$

where S_t represents the smoothed perplexity at step t, y_t is the observed perplexity at step t, and

α is the smoothing parameter. α is designed to dynamically adjust based on changes in training steps Δt:

$$\alpha_t = \min(\sqrt{\alpha}, 0.999)^{\Delta t}$$

where Δt is always 1 since the training step is a discrete variable. Therefore, the smoothing parameter remains constant during the process.

A.8 Concat and Padding Results under MSL = 32 and 128 with GPT2 124M

Concat. See Figure 7, Figure 8 for details.

Padding. See Figure 9, Figure 10 for details.

A.9 Concat and Padding Results under MSL = 256, 512 and 1024 with GPT-2-XL 1.5B

Concat. See Figure 11, Figure 12 and Figure 13 for details.

Padding. See Figure 14, Figure 15 and Figure 16 for details.

When training GPT-2 XL models with MSLs (256, 512 and 1024) that are significantly larger than average document lengths, aligning atom size with MSL no longer always leads to the lowest perplexity. Instead, models with atom size of 1MSL, 2MSL and 4MSL take turn to achieve the lowest perplexity in different MSLs. This is because when document length < MSL < atom size, we only pad to MSL instead of padding to atom size. This implementation prevents us from getting training samples that purely consist of padding tokens. Since WikiText has mostly short documents ($\tilde{1}$50-200 tokens), there is actually no difference between atom size = 1MSL, 2MSL and 4MSL, as we are just padding to 1MSL.

A.10 Step Size Differences in Padding

As mentioned in Section 3.2, padding causes model's training steps per epoch to depend on the atom sizes, due to the different amounts of padding tokens in the training sequences. We added padding tokens to the dataset in two ways.

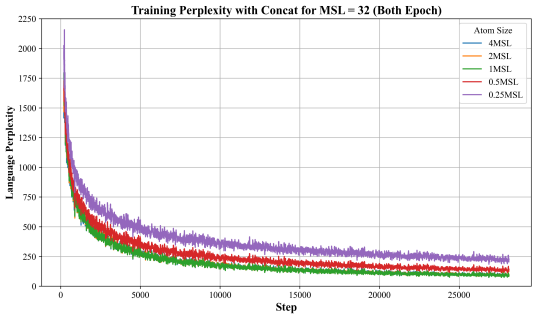
(a) Between each Shuffling Chunk. As mentioned in Appendix A.3, documents are split to shuffling chunks with an <EOS> token (our padding token) added to their ends. A smaller atom size will get more padding tokens in this way, since the dataset is split into more chunks.

(b) At the End of an Unfilled Chunk. If a shuffling chunk is not completely filled by word tokens, we added padding tokens to fill the chunk. A larger atom size will get more padding tokens in this way, since it is more likely that the word tokens cannot fill in each chunk.

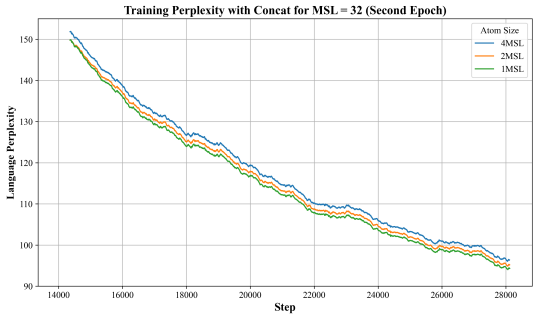
In the case of atom size = 64 (as in Figure 2), we see that atom size = 1/2/4 MSL is having more steps than atom size = 0.25MSL / 0.5MSL, which is a consequence of (b). However, within atom size = 1/2/4 MSL, we see that 1MSL is having the most steps, which is a consequence of (a).

A.11 License for GPT-2

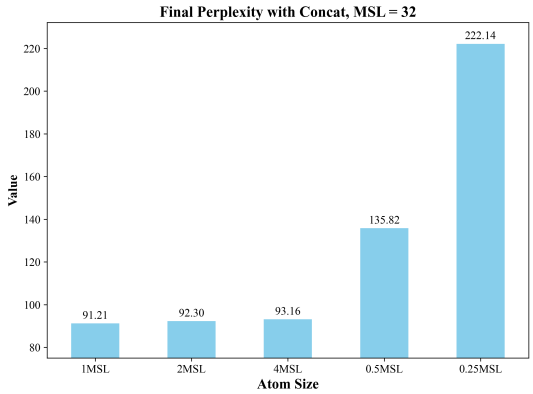
GPT-2 used a Modified MIT License, which can be seen on this: `https://github.com/openai/gpt-2/blob/master/LICENSE`. We only use GPT-2 for research, which is consistent to its intended use.



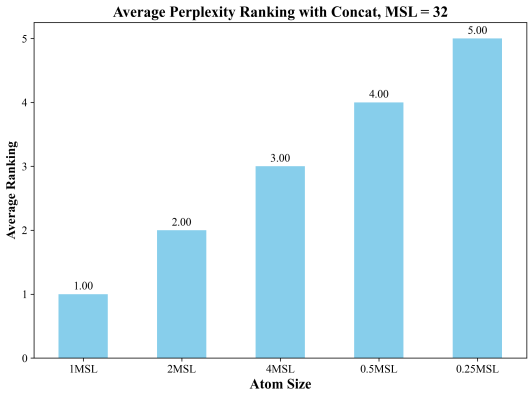
(a) **Full Training Perplexity.** The models with atom sizes of 0.5MSL (red) and 0.25MSL (purple) have higher perplexity than the others. 1MSL (green) stabilizes at a low perplexity after an initial drop.



(b) **Second Epoch Perplexity.** Models with atom size of 4MSL (blue) has higher perplexity than the other two. 1MSL (green) consistently maintains the lowest perplexity in the second epoch.

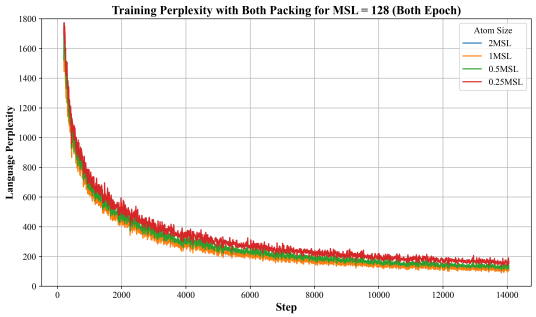


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value(222.14), while model with atom size of 1MSL has the lowest final perplexity value(91.21) for 2 epochs.

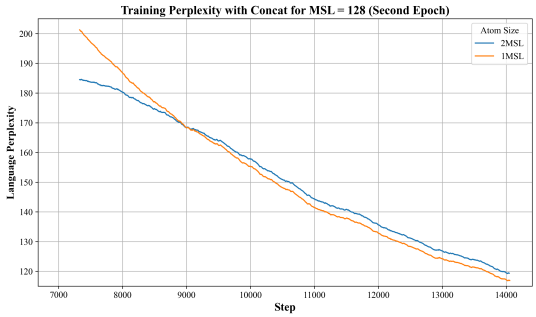


(d) **Perplexity Ranking.**The model with atom size of 0.25MSL has the highest perplexity ranking(5), while model with atom size of 1MSL has the lowest perplexity ranking(1) for 2 epochs.

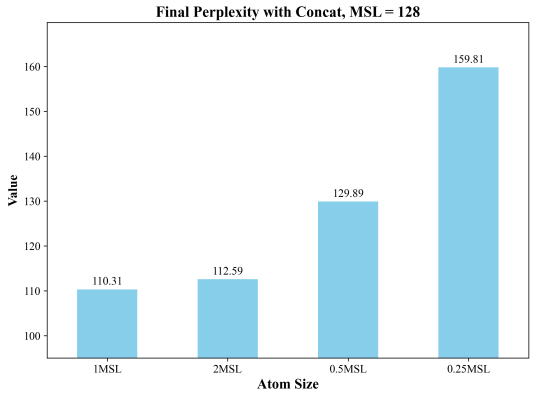
Figure 7: Comparisons across concat models with different atom sizes when MSL is 32. Smaller or larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the lowest final perplexity value and the smallest average perplexity ranking at the end of 2 epochs, indicating better performance.



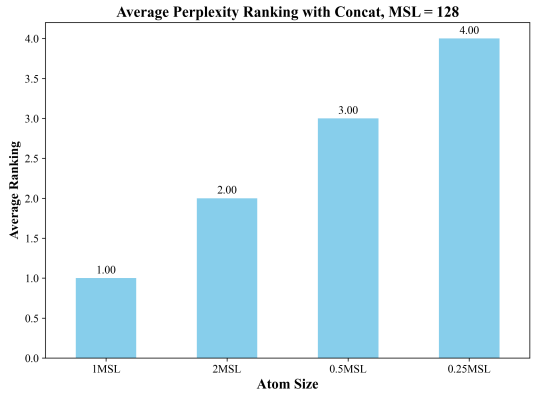
(a) **Full Training Perplexity.** The models with atom sizes of 0.25MSL (red) and 0.5MSL (green) have higher perplexity than the others. 1MSL (orange) stabilizes at a low perplexity after an initial drop.



(b) **Second Epoch Perplexity.** Initially, the model with atom size of 1MSL (orange) shows higher perplexity than 2MSL (blue). 1MSL continuously decreases and achieves the lowest perplexity by the end of the second epoch.

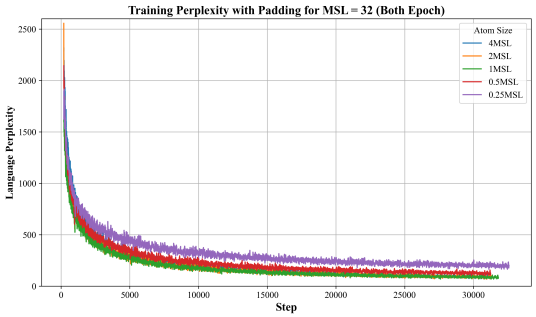


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value(159.81), while model with atom size of 1MSL has the lowest final perplexity value(110.31) for 2 epochs.

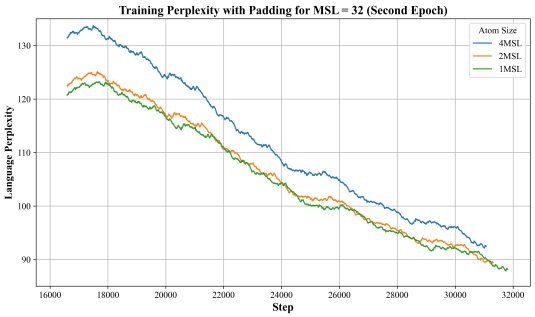


(d) **Perplexity Ranking.**The model with atom size of 0.25MSL has the highest perplexity ranking (4), while model with atom size of 1MSL has the lowest perplexity ranking (1) for 2 epochs.

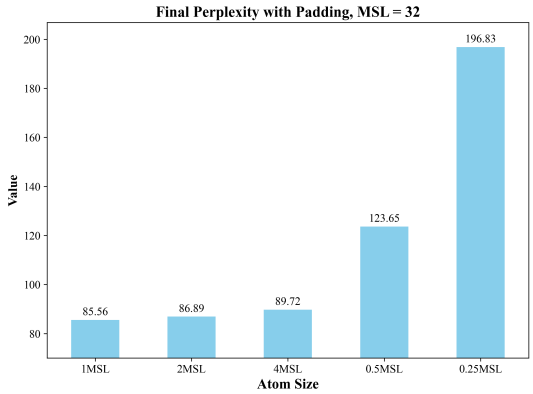
Figure 8: Comparisons across concat models with different atom sizes when MSL is 128. Smaller or larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the lowest final perplexity value and the smallest average perplexity ranking at the end of 2 epochs, indicating better performance.



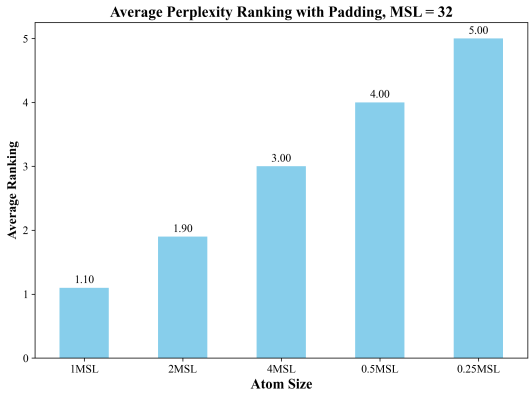
(a) **Full Training Perplexity.** The models with atom sizes of 2MSL (orange) and 0.5MSL(red) have higher perplexity than the others. 1MSL (green) stabilizes at a low perplexity after an initial drop.



(b) **Second Epoch Perplexity.** The models with atom size of 4MSL (blue) has higher perplexity than the other two. 1MSL (green) has the lowest perplexity at the end of second epoch.

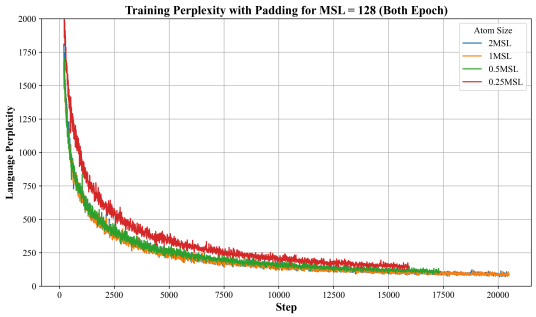


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value (196.83), while model with atom size of 1MSL has the lowest final perplexity value (85.56) for 2 epochs.

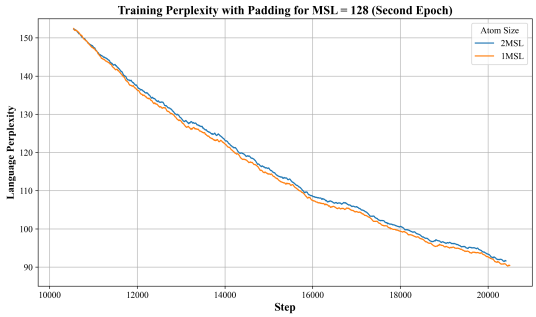


(d) **Perplexity Ranking.** The model with atom size of 0.25MSL has the highest perplexity ranking (5), while model with atom size of 1MSL has the lowest perplexity ranking (1.1) for 2 epochs.

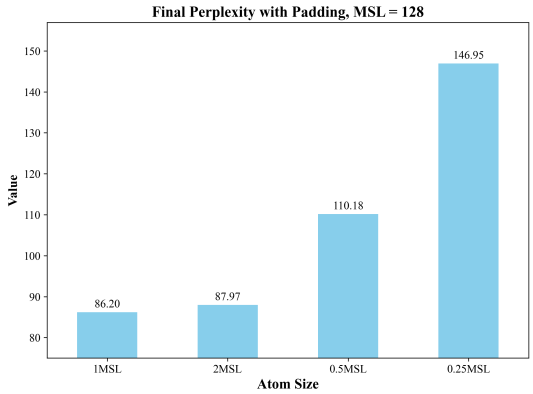
Figure 9: Comparisons across padding models with different atom sizes when MSL is 32. Smaller or larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the lowest final perplexity value and the smallest average perplexity ranking at the end of 2 epochs, indicating better performance.



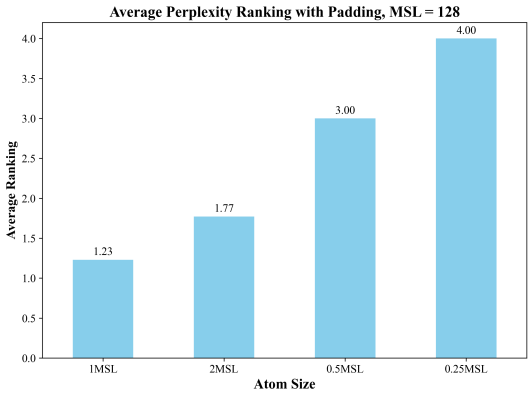
(a) **Full Training Perplexity.** The model with atom sizes of 0.25MSL (red) has higher perplexity than the others. 1MSL (orange) stabilizes at a low perplexity after an initial drop.



(b) **Second Epoch Perplexity.** Initially, the model with atom size of 2MSL (blue) and 1MSL(orange) have similar perplexity. 1MSL (orange) has the lowest perplexity at the end of second epoch.

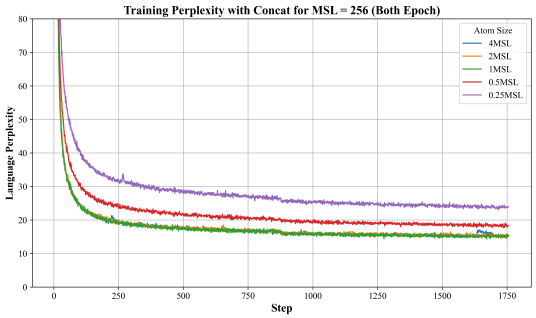


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value (146.95), while model with atom size of 1MSL has the lowest final perplexity value (86.20) for 2 epochs.

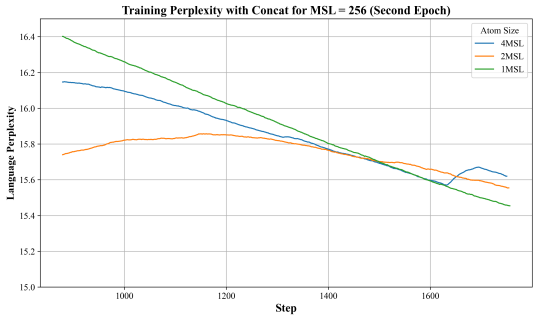


(d) **Perplexity Ranking.**The model with atom size of 0.25MSL has the highest perplexity ranking (4), while model with atom size of 1MSL has the lowest perplexity ranking (1.23) for 2 epochs.

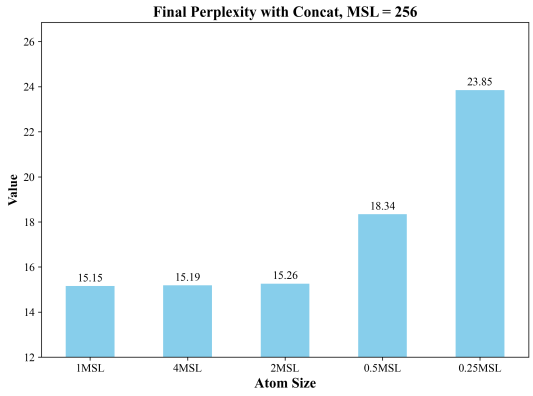
Figure 10: Comparisons across padding models with different atom sizes when MSL is 128. Smaller or larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the lowest final perplexity value and the smallest average perplexity ranking at the end of 2 epochs, indicating better performance.



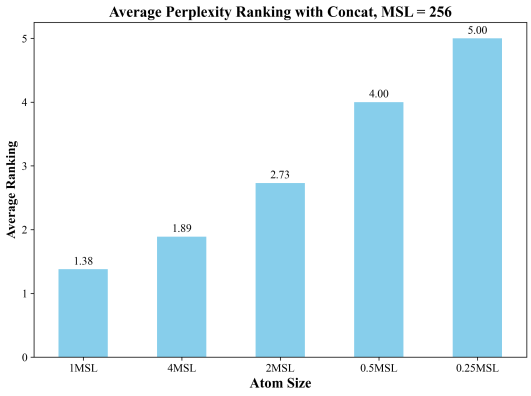
(a) **Full Training Perplexity.** The models with atom sizes of 0.5MSL (red) and 0.25MSL (purple) have higher perplexity than the others. 1MSL (green) stabilizes at a low perplexity after an initial drop.



(b) **Second Epoch Perplexity.** Models with atom size of 4MSL (blue) has higher final perplexity than the other two. 1MSL (green) starts with relatively high perplexity, but eventually reaches to the lowest perplexity in the second epoch.

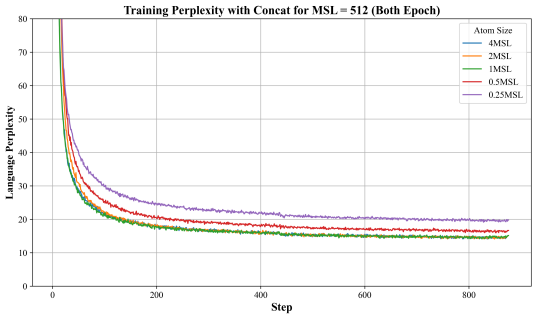


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value(23.85), while model with atom size of 1MSL has the lowest final perplexity value(15.15) for 2 epochs.

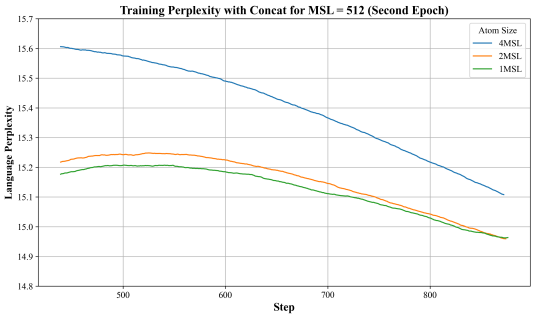


(d) **Perplexity Ranking.** The model with atom size of 0.25MSL has the highest perplexity ranking(5), while model with atom size of 1MSL has the lowest perplexity ranking(1.38) for 2 epochs.

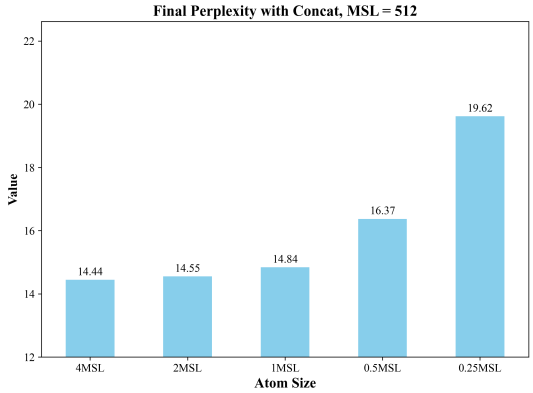
Figure 11: Comparisons across concat models with different atom sizes when MSL is 256. Smaller or larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the lowest final perplexity value and the smallest average perplexity ranking at the end of 2 epochs, indicating better performance.



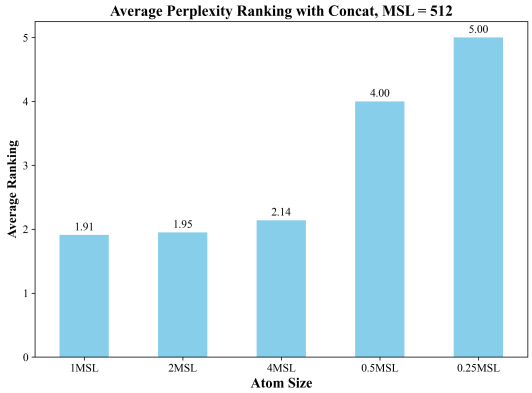
(a) **Full Training Perplexity.** The models with atom sizes of 0.25MSL (purple) and 0.5MSL (red) have higher perplexity than the others. 1MSL (green) stabilizes at a low perplexity after an initial drop.



(b) **Second Epoch Perplexity.** 1MSL maintains as the lowest perplexity during most of the second epoch, but it slows down at the very end of training and gets surpassed by 2MSL.

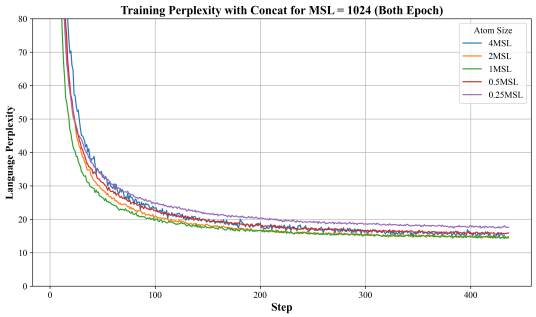


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value(19.62), while model with atom size of 4MSL has the lowest final perplexity value(14.44) for 2 epochs.

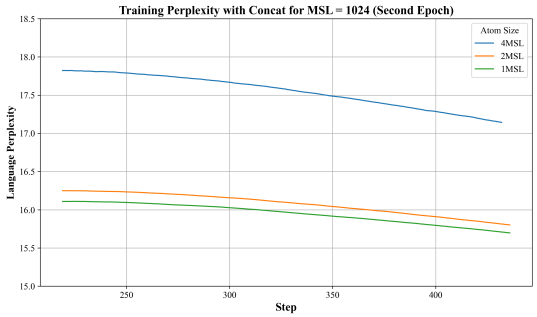


(d) **Perplexity Ranking.**The model with atom size of 0.25MSL has the highest perplexity ranking (5), while model with atom size of 1MSL has the lowest perplexity ranking (1.91) for 2 epochs.

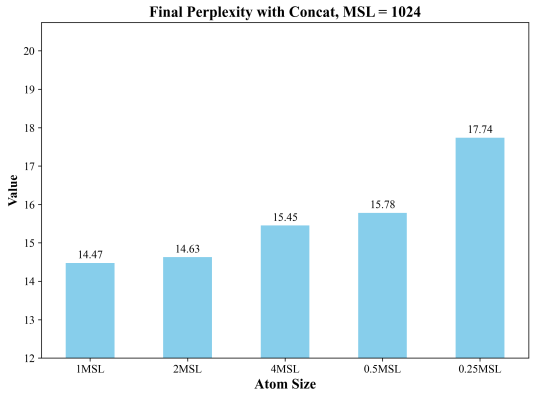
Figure 12: Comparisons across concat models with different atom sizes when MSL is 512. Smaller or larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the smallest average perplexity ranking at the end of 2 epochs, indicating better performance. However, it does not have the best final perplexity as the perplexity decrease slows down



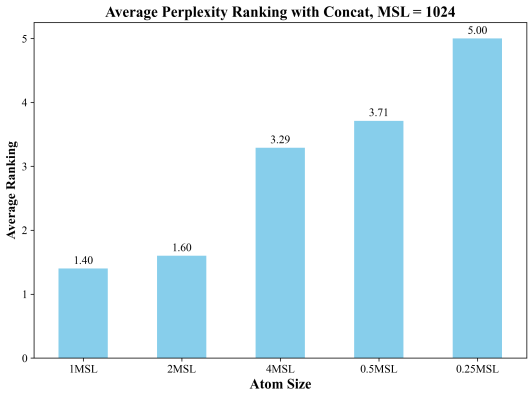
(a) **Full Training Perplexity.** The models with atom sizes of 0.5MSL (red) and 0.25MSL (purple) have higher perplexity than the others. 1MSL (green) stabilizes at a low perplexity after an initial drop.



(b) **Second Epoch Perplexity.** Models with atom size of 4MSL (blue) has higher final perplexity than the other two. 1MSL (green) maintains to have the lowest perplexity in the second epoch.

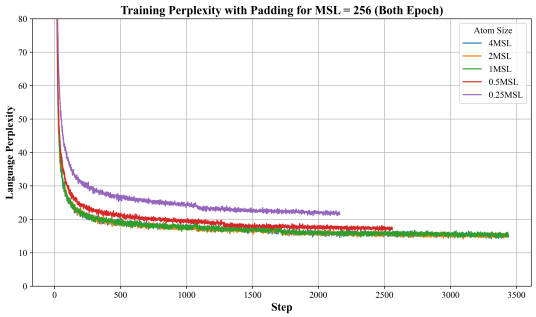


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value(17.74), while model with atom size of 1MSL has the lowest final perplexity value(14.47) for 2 epochs.

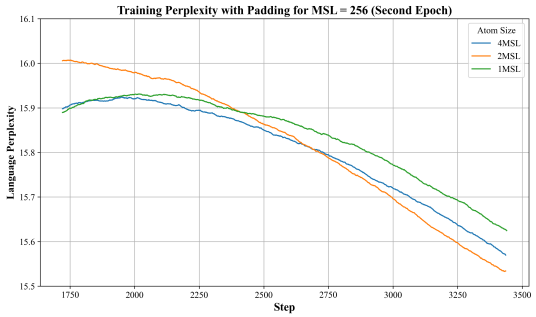


(d) **Perplexity Ranking.** The model with atom size of 0.25MSL has the highest perplexity ranking(5), while model with atom size of 1MSL has the lowest perplexity ranking(1.40) for 2 epochs.

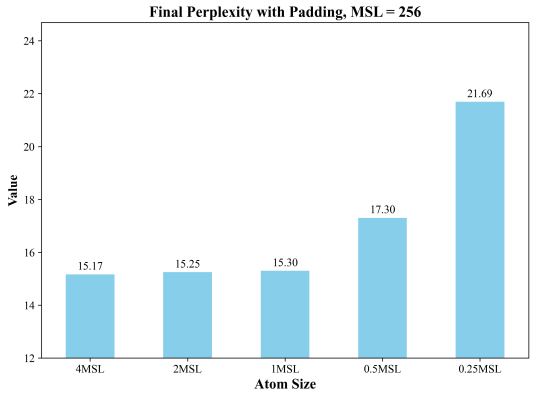
Figure 13: Comparisons across concat models with different atom sizes when MSL is 1024. Smaller or larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the lowest final perplexity value and the smallest average perplexity ranking at the end of 2 epochs, indicating better performance.



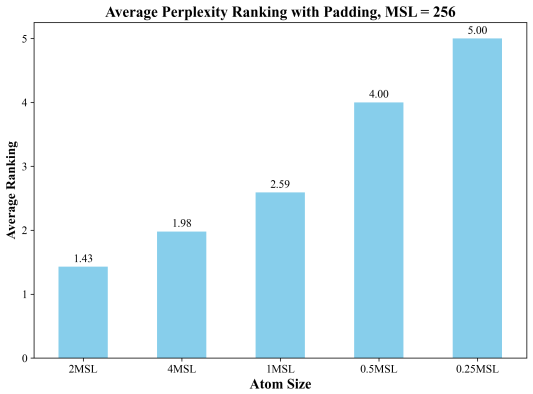
(a) **Full Training Perplexity.** The models with atom sizes of 0.25MSL (purple) and 0.5MSL(red) have higher perplexity than the others. 1MSL (green), 2MSL(orange) and 4MSL(blue) stabilizes at a low perplexity during training.



(b) **Second Epoch Perplexity.** The models with atom size of 1MSL (green) has higher final perplexity than the other two. 2MSL (orange) has the lowest perplexity at the end of second epoch.

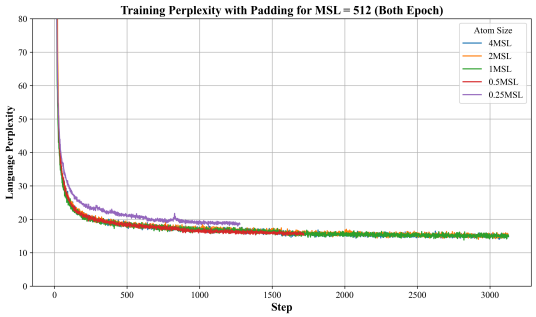


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value (21.69), while model with atom size of 4MSL has the lowest final perplexity value (15.17) for 2 epochs.

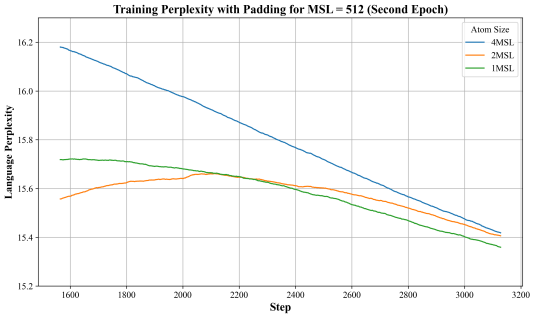


(d) **Perplexity Ranking.** The model with atom size of 0.25MSL has the highest perplexity ranking (5), while model with atom size of 2MSL has the lowest perplexity ranking (1.43) for 2 epochs.

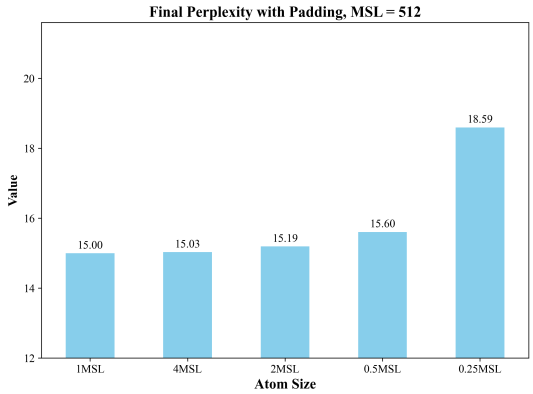
Figure 14: Comparisons across padding models with different atom sizes when MSL is 256. Larger atom sizes than 1MSL increase perplexity. The model with 4MSL as the atom size has the lowest final perplexity value, and the model with 2MSL has the smallest average perplexity ranking at the end of 2 epochs.



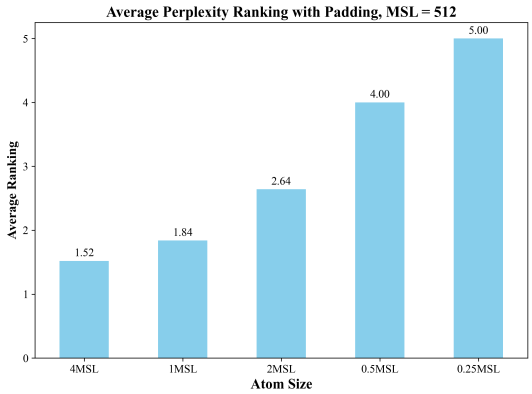
(a) **Full Training Perplexity.** The model with atom sizes of 0.25MSL (purple) has higher perplexity than the others. The other models have comparable perplexity levels during training.



(b) **Second Epoch Perplexity.** The model with atom size of 4MSL (blue) has the highest perplexity. 1MSL (green) has the lowest perplexity at the end of second epoch.

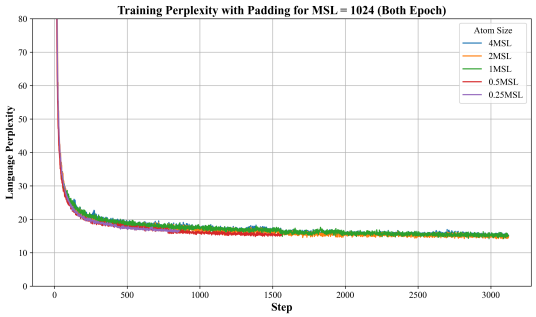


(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value (18.59), while model with atom size of 1MSL has the lowest final perplexity value (15.00) for 2 epochs.

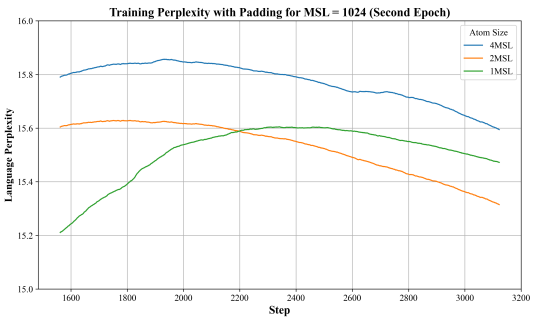


(d) **Perplexity Ranking.**The model with atom size of 0.25MSL has the highest perplexity ranking (5), while model with atom size of 4MSL has the lowest perplexity ranking (1.52) for 2 epochs.

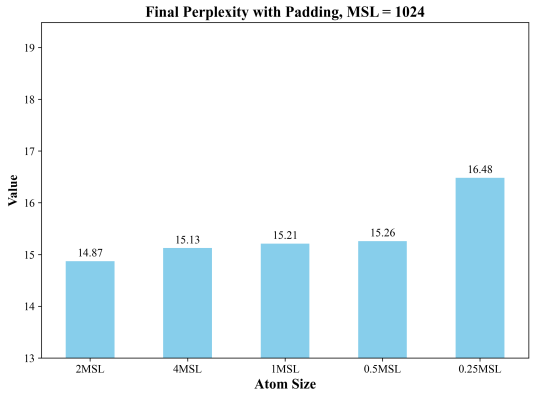
Figure 15: Comparisons across padding models with different atom sizes when MSL is 512. Larger atom sizes than 1MSL increase perplexity. The model with 1MSL as the atom size has the lowest final perplexity value, while the model with 4MSL has smallest average perplexity ranking at the end of 2 epochs.



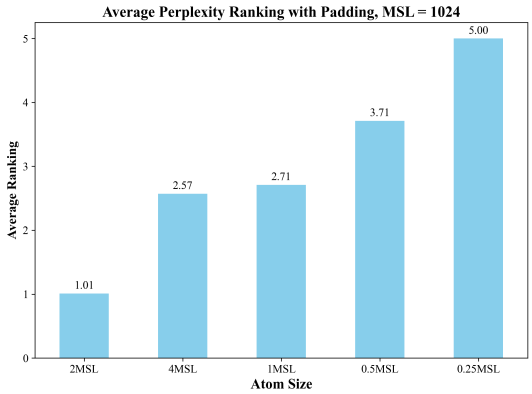
(a) **Full Training Perplexity.** The model with atom size of 0.5MSL (red) has slightly lower perplexity than the others. The models with 1MSL (green), 2MSL (orange) and 4MSL (blue) have comparable perplexity levels.



(b) **Second Epoch Perplexity.** The model with atom size of 4MSL (blue) has the highest perplexity. 2MSL (orange) has the lowest perplexity at the end of second epoch.



(c) **Final Perplexity.** The model with atom size of 0.25MSL has the highest final perplexity value (16.48), while model with atom size of 2MSL has the lowest final perplexity value (14.87) for 2 epochs.



(d) **Perplexity Ranking.** The model with atom size of 0.25MSL has the highest perplexity ranking (5), while model with atom size of 2MSL has the lowest perplexity ranking (1.01) for 2 epochs.

Figure 16: Comparisons across padding models with different atom sizes when MSL is 1024. The model with 2MSL as the atom size has the lowest final perplexity value and smallest average perplexity ranking at the end of 2 epochs.