Escaping The Plateau:
Dynamic Context Length Adaptation for Efficient BERT Pretraining

Anonymous ACL submission

Abstract

We present a technique for dynamically shortening the pretraining time of BERT-based models. BERT-based models are a popular choice for pretraining research on a low budget. However improvements can still be made to further lower monetary and time investments. We propose an approach that dynamically shortens the context length when a plateau, a region of slow loss reduction rate, is detected, then returns to the original value after the plateau is escaped. We show that this change forces an abrupt exit from the plateau, which reduces the time it takes to reach 90% of the final baseline performance by a factor of 2.

1 Introduction

Language models use gradient descent for optimization. These methods sometimes exhibit a rapid loss function decrease at the beginning of pretraining but may slow down significantly after a relatively small number of steps (see Figure 1). The loss appears to *plateau* over a large number of steps, and then start decreasing again, sometimes significantly so (Ainsworth and Shin, 2020).

This work provides three main contributions to identify, mitigate, and visualize plateaus during pretraining. First, in section 3.1 we propose a method of dynamically detecting plateaus during the training process. We achieve this by using a sliding window based technique that dynamically tracks the average change in loss during training.

Second, in Section 3.2 we propose an intervention strategy that aims to escape training plateaus by employing a dynamic context length based method. This is achieved by briefly changing the context length of the inputs until the plateau is escaped, then going back to the original context length. Our method manages to reduce the plateau length significantly, leading to faster convergence.

Third, in Section 3.3 we formulate a loss visualization technique which focuses around important stages during the training process, rather than global visualizations employed in previous research (Li et al., 2017; Hao et al., 2019). This is achieved by visualizing around a specific checkpoint rather than using the initial weights as an origin, and by using PCA on vectors relevant to the training process, i.e., checkpoints close to the origin point. Using the visualization we further attest that our method manages to escape the plateau via manipulation of the loss landscape topology.

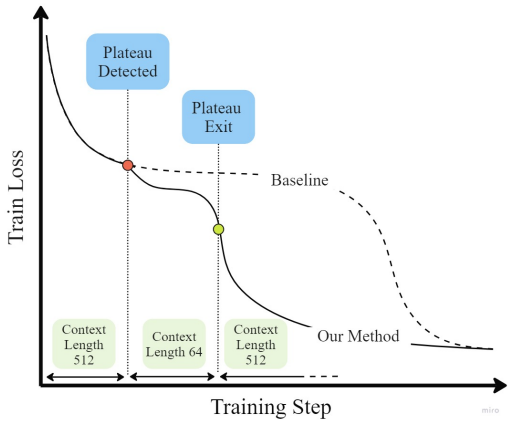

Figure 1: Illustration of the dynamic context length pretraining method compared to pretraining with a constant context length. The method starts with a context length of 512, then when a plateau is detected the context length is switched to length 64. Once we exit the plateau we switch back to the original context length.

In section 4 we use all three proposed techniques on BERT. We find that dynamically shortening the context length leads to an early plateau escape. We find that loss function plateaus correspond to low curvature areas in the loss landscape.

Beyond efficient pretraining for BERT, our methodology suggests a dynamic hyperparameter intervention based on the training behaviour. This stands in contrast with common practice of setting static hyperparameters which only change between training runs.

2 Background

The learning curve plateau phenomenon is a common occurrence during training in which the loss function exhibits rapid descent during initial training steps, then the descent rate of the loss function substantially slows down, only to finally start another rapid descent (Park et al., 2000) (as can be seen in Fig. 1). This behavior has been observed and researched in previous studies aiming to identify the causes of the phenomenon and find ways to early escape or avoid entering it (Ainsworth and Shin, 2020).

Dauphin et al. (2014) have shown that saddle points are common in high dimensional non-convex optimization problems, rather than local minima. Saddle points and local minima are both surrounded by flat regions with minimal changes in the error curvature, which can slow down the error rate reduction of the gradient descent. For saddle points however the plateau can be escaped, which points to the need of devising methods to deal with that type of plateaus in a time and cost effective manner.

Several works attack the plateau phenomena using learning rate schedulers. Shi et al. (2020) have shown that training a model using a decreasing learning rate, a common scheduler practice, leads to faster decrease in error rate. Another learning rate based method was suggested by Smith (2017), in which they present a cyclical scheduler working within a band of values. Wang et al. (2022) explained that a cyclical learning rate helps escaping plateaus as the increase in learning rate allows for rapid traversal of saddle point areas. Nagatsuka et al. (2021) have shown that using a curriculum learning method which gradually increases the input context length of a BERT model (Devlin et al., 2019) leads to faster convergence speed.

To better understand the behavior around plateaus, we visualize the loss surface in these areas. Visualization of loss surfaces is widely used as a tool to better understand the learning process of models. In Li et al. (2017) a number of visualization strategies are reviewed, with a method based on Principal Component Analysis (PCA) being the preferred method for optimization trajectory visualization. The method utilizes a 2D surface technique as seen in Goodfellow and Vinyals (2014); Im et al. (2016), with the basis for the space constructed from the outputs of running PCA on the weight vectors of the model at different stages of training.

3 Methodology

In this section we describe the plateau detection method (Section 3.1), describe our method for escaping a plateau using context length variation (Section 3.2), and present the visualization routine we used for analysing loss landscapes around areas of interest, which was used to analyse the impact of context length on the loss behavior (Section 3.3).

3.1 Plateau Detection

In this phase, we aim to identify when the convergence of the training process slows down. This is done dynamically in the process of the pretraining run, and is agnostic to the model and data used. The exact method is described in Algorithm 1. The algorithm saves the values of the step's loss while training in order to calculate the average over a moving window. It then calculates the ratio between the averages over a window starting 2 window sizes ago and a window starting right after that (lines 6-9). Finally, we check whether the number of steps classified as plateau in near history surpasses a predefined threshold to identify the present state as a plateau (line 12). This approach allows to filter out noise in the detection process.

3.2 Dynamic Intervention

In order to reduce the time spent in a plateau we briefly reduce the context length to a low value after detecting the entrance to the plateau. This is supported by the results of the CR method (Wang et al., 2022), which show quicker convergence in smaller context lengths. We achieve this by dynamically changing the context length training parameter, as well as tokenizing the dataset using the matching context length on the fly for the required number of steps. After the plateau detection method shows we have exited the plateau, we switch back to the original context length, since we still want to preserve the model's ability to perform on tasks requiring the original large context size. Performance wise, the additional tokenization time is offset by the decrease in step duration due to smaller context length, since less data is processed each step.

3.3 Visualization Methodology

We employ a PCA based visualization method (Li et al., 2017), focusing specifically on the loss surface around the plateau, in order to understand how context length affect it. This is achieved by offsetting the trajectory and surface vectors so that the

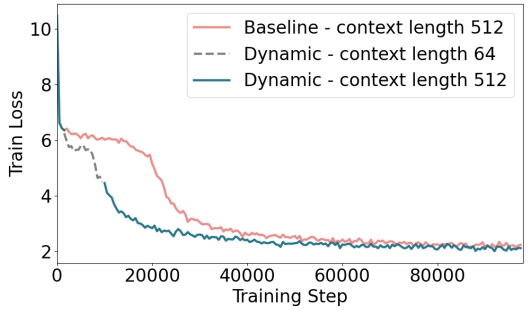

Figure 2: We compare between the baseline pretraining loss and our method, which uses a dynamic context length switch when a plateau is detected. The dashed section is the part where the context length is shortened to 64 tokens. Once we exit the plateau, the method switches back to 512 tokens. Training with a dynamic context length significantly reduces the plateau duration.

origin is a checkpoint on the center of the plateau. This puts the focus on the interesting parts as opposed to visualization techniques utilizing the entire training history that give a more general insight into the training process. We then flatten the weights at checkpoints around the plateau area as an input to the PCA algorithm and retrieve the eigenvectors of the 2 principal components. Let v_1, v_2 be the eigenvectors, and θ_c be the position vector corresponding to the weights at the plateau center checkpoint after being projected to the span of v_1, v_2. The loss surface function will then be:

$$f(x, y) = L(xv_1 + yv_2 + \theta_c)$$

where L is the model loss function.

In order to visualize the optimization trajectory over the loss landscape we project each of the given model checkpoints during training. To project model weights w onto the span of v_1, v_2, we first define the matrix $\mathbf{A} = \begin{bmatrix} v_1 & v_2 \end{bmatrix}$, and use the following formula:

$$\begin{bmatrix} x \\ y \end{bmatrix} = (\mathbf{A}^\intercal \mathbf{A})^{-1} \mathbf{A}^\intercal w$$

As v_1, v_2 are eigenvectors of eigenvalues calculated by the PCA, they are orthogonal and thus $\mathbf{A}$ is invertible, justifying the above calculation. The visualization relies on having reasonably frequent checkpoints of the pretraining process. These are used both as inputs for the PCA process, and as points on the optimization trajectory.

Algorithm 1: Plateau Detection

Input: $w \in \mathbb{N}$ - window size
$loss \in \mathbb{R}$ - current training step loss
$pThresh \in \mathbb{N}$ - plateau threshold
$dThresh \in \mathbb{N}$ - window ratio threshold
P - plateau detection history
L - loss history
Output: True if at plateau step, False otherwise

$L.append(loss)$
if $|L| < 2 * w$ **then**
  return False
else
  `// Average on previous window`
  $d1 = avg(L[|L| - 2 * w : |L| - w])$
  `// Average on last window`
  $d2 = avg(L[|L| - w : |L|])$
  $d = d1/d2$
  $isPlateau = d < dThresh$
  $P.append(isPlateau)$
  return $sum(P[|P| - 2 * w : |P|]) > pThresh$
end if
§

4 Results

In this section we present the results of evaluating our method. We find that changing the context length dynamically being beneficial towards pretraining optimization. Additionally shorter context lengths correlated with narrower plateaus in the landscape.

Experimental setup. We pretrain a medium sized model, BERT-base-uncased, which consists of 110 million parameters. We used the BookCorpus and the English Wikipedia dump (March 2022) datasets to pretrain all model variations, in line with the original BERT training routine. The datasets were mixed together and then split using a standard 90/10%, with the latter portion to be used as a test set in the evaluation phase. We trained all models for 100K steps. A base context length of 512 tokens was used, and we change the context length to 64 when reaching a plateau. To evaluate the effect on performance of the above technique we use the mean reciprocal rank (MRR) metric, which correlates with perplexity, over masked tokens. We compare MRR scores and convergence times between a baseline with a constant context length of

512, a model trained with dynamic context length as described above and the curriculum method (CR) (Nagatsuka et al., 2021). Additional details regarding training, plateau detection and visualization parameters can be found in Appendix A.

Shorter context length leads to faster convergence time. While shorter context length is faster per step by virtue of processing less data each iteration, we additionally show that utilizing shorter context length leads to faster convergence time. This was done by pretraining 4 models with 4 different context lengths: 64, 128, 256, 512. As can be seen in Fig. 4 in the appendix, the shorter the context length the faster we reach convergence. Additionally we can observe that the number of steps spent in a plateau is getting smaller together with smaller context length. For the smallest context length in the experiment - 64, we do not get stuck in a plateau at all.

Dynamic context length shortens plateau duration. By comparing loss graphs in Fig. 2, we can see that our method reduces the number of steps spent in plateau by half compared to the constant model, eventually leading to a $\sim 2.5x$ improvement in convergence time.

In Fig. 3 we demonstrate the time optimizations provided by our method. At 15% of baseline training time we can see that our method dramatically outperforms the other methods. In particular, utilizing our method and stopping at 20K steps yield an MRR score that is $\sim 85\%$ of the maximum achieved, while taking $\sim 2.5x$ less time to train than the baseline equivalent, and $\sim 2x$ less time to train than the CR method. Moreover, all 3 models converge to a similar score, pointing to no degradation of performance when using our method. This enables to pretrain for a much shorter duration while getting MRR scores very similar to the ones from a long training routine.

The above observations give the user the freedom to apply the method automatically and decide when to stop training on the go. Additionally, based on the results the gradual approach taken in the CR paper might be overly cautious for the task at hand, and a single jump between context sizes might be faster while maintaining quality.

Different plateau lengths can also be observed in static context lengths. In order to better understand the effect of context changes on the training process, we used a PCA visualization method with several context lengths and investigated the differences in the landscape and trajectory between them, as shown in Fig. 5 in the Appendix. It can be seen that the optimization trajectory remains unaffected by context size changes. This points mainly to the loss landscape remaining constant in its general topology, with the only change being to the area of the plateau around the starting point. This effectively mirrors the reduction in plateau length, expressed in training steps, with longer context length correlated with more expansive plateaus in the landscape. These changes are then directly correlated with the convergence speed due sharper gradients in earlier stages of training.

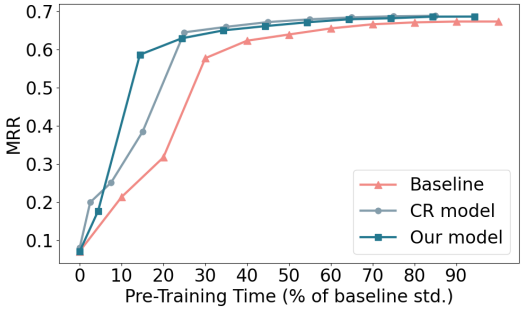

Figure 3: MRR score and processing time of our approach (blue squares, dynamic context length of 512-64-512), baseline (pink triangles, pretraining with constant context length of 512) and the CR approach (gray dots, context length gradually increased). Each data point is 10K steps apart from the previous one in its series. Left and higher is better. Our method performs as good as the CR method in later stages of training, and significantly outperforms the other methods when stopping early.

5 Conclusions

In this work we present a dynamic context length pretraining method as a way to escape a detected plateau early.This method leads to a convergence speed faster by a factor of 2 compared to the baseline. We also use a PCA based visualization method to explore the effect of context length on the model's loss landscape and optimization trajectory. The visualization shows that the trajectory remains similar for all context lengths, however the landscape plateau is shorter for shorter context lengths, in a manner that correlates to a faster convergence speed due to sharper gradients in earlier stages of training.

6 Limitations

This work is subject to a number of limitations. First, the methods presented were only applied to the BERT model. While plateaus are reported in architectures other than BERT, we can provide no guarantee that this method will be effective when applied to them. Secondly, as with any empirical machine learning research, the hyper parameters chosen for the plateau detection algorithm may not be universal and require adjustment when ran with a different experimental setup.

References

Mark Ainsworth and Yeonjong Shin. 2020. Plateau phenomenon in gradient descent training of relu networks: Explanation, quantification and avoidance. *SIAM J. Sci. Comput.*, 43:A3438–A3468.

Yann Dauphin, Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *ArXiv*, abs/1406.2572.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*.

Ian J. Goodfellow and Oriol Vinyals. 2014. Qualitatively characterizing neural network optimization problems. *CoRR*, abs/1412.6544.

Yaru Hao, Li Dong, Furu Wei, and Ke Xu. 2019. Visualizing and understanding the effectiveness of bert. *ArXiv*, abs/1908.05620.

Daniel Jiwoong Im, Michael Tao, and Kristin Branson. 2016. An empirical analysis of the optimization of deep network loss surfaces. *arXiv: Learning*.

Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. 2017. Visualizing the loss landscape of neural nets. In *Neural Information Processing Systems*.

Koichi Nagatsuka, Clifford Broni-Bediako, and Masayasu Atsumi. 2021. Pre-training a bert with curriculum learning by increasing block-size of input text. In *Recent Advances in Natural Language Processing*.

Hyeyoung Park, Shun-ichi Amari, and Kenji Fukumizu. 2000. Adaptive natural gradient learning algorithms for various stochastic models. *Neural networks : the official journal of the International Neural Network Society*, 13 7:755–64.

Bin Shi, Weijie J Su, and Michael I Jordan. 2020. On learning rates and schr\" odinger operators. *arXiv preprint arXiv:2004.06977*.

Leslie N. Smith. 2017. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472.

Weixuan Wang, Choon Meng Lee, Jianfeng Liu, Talha Çolakoğlu, and Wei Peng. 2022. An empirical study of cyclical learning rate on neural machine translation. *Nat. Lang. Eng.*, 29:316–336.

A Experimental Setup Details

Training Details. For dataset tokenization we use a the bert-base-uncased tokenizer, pretrained on the same datasets used for the training data. For the masked language model, we use the default setting - generating 15% mask tokens and 10% random tokens for each input sequence (Devlin et al., 2019).

We trained all models for 100K steps, with a batch size of 16 examples and gradient accumulation of 2. These were chosen to emulate a batch size of 32 within our resource limitations. The learning rate was set to 5e-5 (Devlin et al., 2019).

We used 4 GPUs (either NVIDIA A6000s, RTX6000s or A5000s). All hyperparameters are identical in all iterations of the experiment, except for the varying context length.

Plateau Detection Details. Using the plateau detection algorithm described at section 3.1, we used $w = 500, dThresh = 1.008$ and $pThresh = 700$. These constants were the most stable out of a number of different configuration we tried, with other configurations resulting in either late detections or jittery output.

Visualization Details For the landscape visualization we chose checkpoints from steps at and around the plateau. For the trajectory, we chose checkpoints from steps 2.5K, 5K, 7.5K, 10K, 12.5K, 15K and 25K. In our experiments we set the range of x and y dynamically to contain all projected trajectory checkpoints, which resulted in range [-60,60]. We then plotted the above function with a resolution of 40 samples per axis, 1,600 sample points in total.

B Additional Graphs

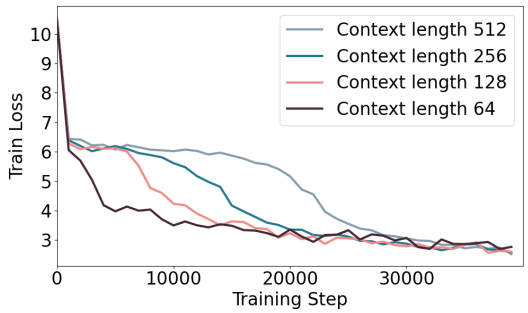

Figure 4: Comparing usage of different context lengths in pretraining shows a consistent relation between context length and plateau duration. Final loss values for all series are very close, only the plateau exit point is affected.

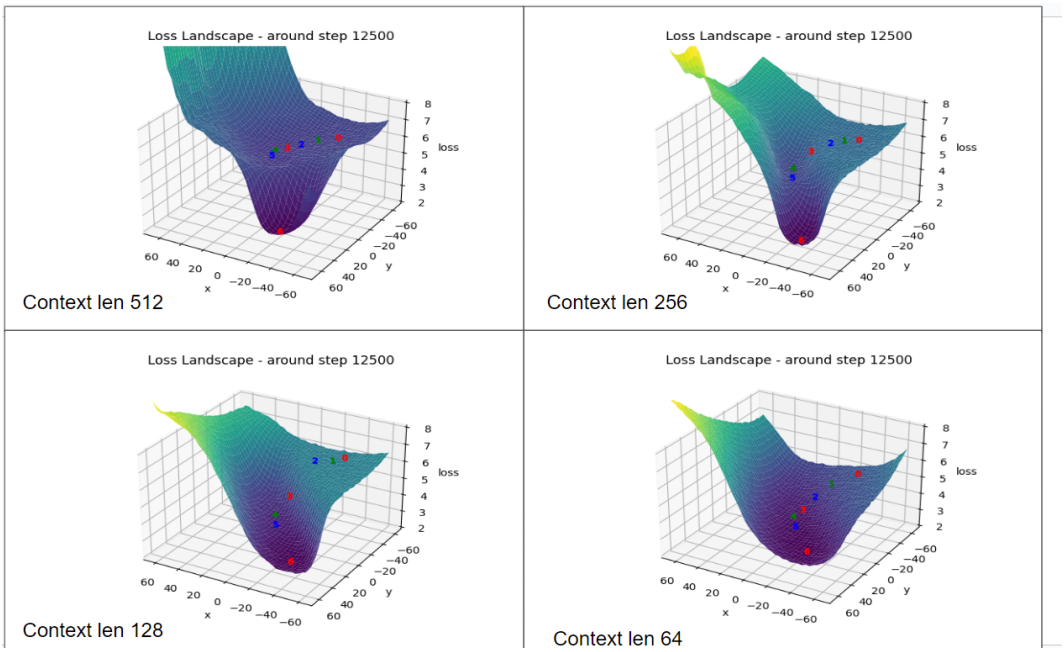

Figure 5: Comparison of the loss landscape for different context lengths. PCA was used to compute the principal vectors used as axes here. All 4 graphs were made with the same checkpoint as an origin point. The optimization trajectory is described by the numbers, in ascending order. All 4 graphs exhibit a plateau in the upper right part of the landscape, with its size decreasing as we decrease the context length. The trajectory remains very similar between the graphs, pointing to overall topological similarity.