Dynamic Masking Rate Schedules for MLM Pretraining

Zachary Ankner [1,2] **Naomi Saphra**[1,3] **Davis Blalock**[1] **Jonathan Frankle**[1] **Matthew Leavitt**[1]
[1]MosaicML [2]Massachusetts Institute of Technology [3] New York University

Abstract

Most works on transformers trained with the Masked Language Modeling (MLM) objective use the original BERT model's fixed masking rate of 15%. Our work instead dynamically schedules the masking ratio throughout training. We found that linearly decreasing the masking rate from 30% to 15% over the course of pretraining improves average GLUE accuracy by 0.46% in BERT-base, compared to a standard 15% fixed rate. Further analyses demonstrate that the gains from scheduling come from being exposed to both high and low masking rate regimes. Our results demonstrate that masking rate scheduling is a simple way to improve the quality of masked language models and achieve up to a 1.89x speedup in pretraining.

1 Introduction

BERT (Devlin et al., 2019) is a popular encoder-only Transformer (Vaswani et al., 2017) architecture that is pretrained using a masked language modeling (MLM) objective. Inspired by the Cloze task (Taylor, 1953), when performing MLM training, we mask out a subset of the input tokens and the model is trained to reconstruct the missing tokens. The proportion of tokens to be masked out is determined by the masking rate hyperparameter.

Most practitioners use a fixed masking rate of 0.15 (Devlin et al., 2019), but Wettig et al. (2022) found that the standard 15% masking rate is sub-optimal for a variety of model settings, and recommended a higher rate. We build on their work by studying the impact of dynamically scheduled masking rates.

Scheduling during training—i.e., changing the value of a hyperparameter, such as learning rate, dropout rate, batch size, sequence length, etc.—is a common practice in deep learning (Loshchilov and Hutter, 2017; Smith, 2017; Howard and Ruder, 2018; Morerio et al., 2017; Smith et al., 2018; Li et al., 2022). There are several reasons the masking rate is a good candidate for hyperparameter scheduling. A high masking rate directly reduces the amount of feature information available during a training step, which allows simulated annealing through scheduling. A higher masking rate also adds training signal, as loss is computed for a larger portion of tokens. Accordingly, the masking rate shares similarities with commonly-scheduled hyper-parameters. We therefore study the natural question of whether scheduling the masking rate during training could lead to model quality improvements.

This work presents a series of experiments to examine the effect of masking rate scheduling on the quality of BERT-base (Devlin et al., 2019). We evaluate our masking rate scheduled models on MLM loss and on downstream tasks.

Our contributions are:

- We introduce masking rate scheduling as a novel technique for improving MLM pretraining (Section 3.1), and find that performance improves only when starting at a higher ratio and decaying it (Appendix F).
- We show that the improvement from scheduling the masking rate is a Pareto improvement over fixed masking rates for all examined pretraining durations (Section 3.2).
- We find that dynamic scheduling attains both the improved linguistic performance of a lower masking rate (Section 3.3) and improved language modeling of a higher masking rate (Section 3.4).

2 Methods

We perform typical MLM training with the key difference that a scheduler sets the masking rate dynamically.

2.1 Masked language modeling

An MLM objective trains a language model to reconstruct tokens that have been masked out from an input sequence. Let $x \sim \mathcal{X}$ be the input sequence,

Correspondence to ankner@mit.edu.

and p_{mask} be the probability with which tokens are masked from the model, i.e., the masking rate. A mask $M = \{m_1, ..., m_k\}$ is defined as the indices of the tokens to be masked, where the probability of a given token index being included in the mask is a Bernoulli random variable with parameter p_{mask}. Following Devlin et al. (2019), we replace 80% of the masked tokens with a `[MASK]` token, substitute 10% with another random token, and leave 10% unchanged. The training objective is then defined as:

$$\mathcal{L}(x) = \frac{1}{|M|} \sum_{i=1}^{|M|} \log p(x_{m_i} | x_{-M}) \tag{1}$$

2.2 Schedulers

Let $\mathcal{T}_{\texttt{total}}$ be the total number of steps the model takes during training and t be the current step. Let p_i and p_f be the initial and final masking rate respectively. For each step, we set the masking rate $p_{\text{mask},t}$ according to the following schedules. We test several nonlinear schedules as well, but find no consistent advantage over the simpler linear schedule (Appendix F).

Constant scheduling. Constant scheduling is the standard approach to setting the masking rate for MLM pretraining (typically $p_{\text{mask}} = 0.15$) where the same masking rate is used throughout all of training. The masking rate is set as:

$$p_{\text{mask},t} = p_i = p_f$$

We refer to constant schedules as `constant-`$\{p_{\texttt{mask}}\}$.

Linear scheduling. Linear scheduling is a common scheduler type for setting model learning rate. In linear scheduling, the schedule is the linear interpolation between the initial and final masking rate:

$$p_{\text{mask},t} = p_i + \frac{t}{\mathcal{T}_{\texttt{total}}} * (p_f - p_i)$$

We refer to linear schedules as `linear-`$\{p_i\}$`-`$\{p_f\}$.

3 Experiments and Results

In this section we evaluate the performance of masking rate scheduling on a collection of downstream tasks and probe why our schedule is successful.

We pretrain all models on the Colossal Cleaned Common Crawl (C4) dataset (Raffel et al., 2019), and then fine-tune and evaluate on the GLUE benchmark (Wang et al., 2018). For all experiments we use a `BERT-base` model implemented in HuggingFace (Wolf et al., 2020). To manage the training of models we use the Composer library (Tang et al., 2022). We list further details of our experimental setup in Appendix A.

3.1 Improvement in downstream tasks

We first examine the effects of the best linear schedule on downstream performance on GLUE (Table 1). We find that `linear-0.3-0.15` outperforms both constant masking rate baselines. We focus our comparison to be between `linear-0.3-0.15` and the better constant baseline, `constant-0.3-0.3`. We find that `linear-0.3-0.15` improves performance over the baseline on 3 of the 8 GLUE tasks and achieves parity on all other tasks. Overall, `linear-0.3-0.15` achieves an average GLUE accuracy of 84.29%, a statistically significant improvement. These results show that scheduling the masking rate during pretraining produces higher quality models for downstream tasks.

We further analyze different schedule parameters in Appendix C and other scheduler functions in Appendix F.

3.2 Improvement in training efficiency

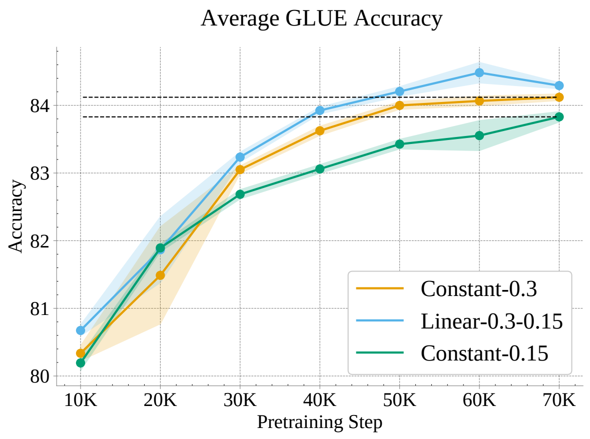

Figure 1: GLUE accuracy evaluated over the course of pretraining. The horizontal lines correspond to the best observed GLUE accuracies for `constant-0.15` and `constant-0.3`

Masking rate scheduling not only improves final model quality, but also improves efficiency across the entire duration of pretraining.

Linear scheduling matches the mean GLUE score of the best `constant-0.15` checkpoint in 37K steps and matches the best `constant-0.3` checkpoint in 42K steps, which correspond to speedups of 1.89x and 1.65x, respectively. Furthermore, `linear-0.3-0.15` is a Pareto improvement over both constant baselines as for each pretraining step evaluated, `linear-0.3-0.15` achieves better or equal performance for no increase in training time (Figure1). Appendix E contains further details on evaluating model speedups.

SCHEDULE	MNLI-M/MM	QNLI	QQP	RTE	SST-2	MRPC	COLA	STS-B	AVG
CONSTANT-0.15	84.3/84.71	90.38	**88.31**	**76.65**	**92.91**	**91.94**	55.89	89.38	83.83
CONSTANT-0.3	84.5/84.83	**90.82**	**88.31**	**76.56**	**92.79**	**92.18**	57.24	**89.85**	84.12
LINEAR-0.3-0.15 (OURS)	**84.61/85.13**	**90.89**	**88.34**	**76.25**	**92.71**	**91.87**	**58.96**	**89.87**	**84.29**

Table 1: Downstream performance for different masking rate schedules. For each model we report the average accuracy for each task in GLUE. Bold indicates no significant difference from highest-performing schedule, *P > 0.05*, t-test.

SCHEDULE	AVG BLIMP ACCURACY
LINEAR-0.3-0.15	**82.70**
CONSTANT-0.15	**82.44**
CONSTANT-0.3	82.13

Table 2: Average accuracy across all BLiMP tasks. Bold indicates mean + standard error matches best average.

3.3 Improvement in grammatical understanding

In order to better understand how masking rate scheduling affects the linguistic capabilities of masked language models, we evaluated our models on the BLiMP benchmark (Warstadt et al., 2020), a collection of tasks designed to test understanding of syntax, morphology, and semantics.

We find that the average accuracy of `linear-0.3-0.15` on BLiMP tasks significantly improves over `constant-0.3` and is comparable to `constant-0.15` (Table 2). These results suggest that a dynamic schedule provides the linguistic understanding offered by a lower masking rate.

3.4 Improvement in the pretraining objective

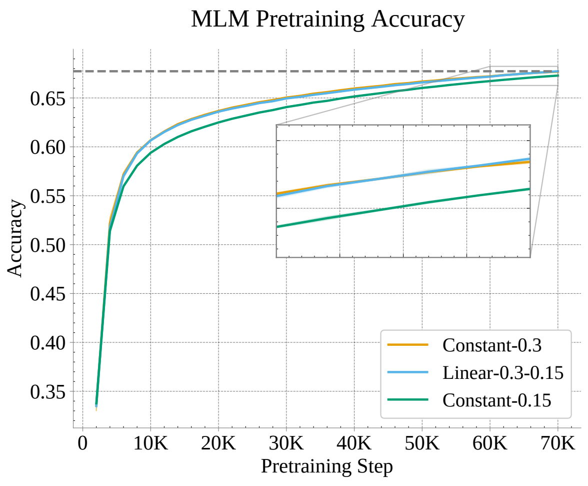

Figure 2: Average MLM accuracy of different masking rate schedules over the entire pretraining duration. Linear scheduling has a high overlap in performance with the 30% constant masking model throughout the entirety of pretraining.

Similarly to the case of grammatical understanding, we find that performance on the MLM pretraining objective also matches the better of the constant baselines. While MLM performance is not necessarily indicative of downstream model capabilities, it *is* informative about a model's ability to model language and gives insight into how scheduling affects pretraining dynamics.

In Figure 2, we graph each schedule's MLM accuracy on the test set throughout pretraining, where the evaluation masking rate is fixed at 15%. There is significant overlap between the performance of linear scheduling and the higher 30% constant scheduling. Specifically, we find that `linear-0.3-0.15` and `constant-0.3` have have MLM accuracies of 67.73% and 67.69% respectively. However, `constant-0.15` performs significantly worse throughout the entirety of pretraining, with a best MLM accuracy of 67.29%.

Although scheduling only temporarily sets a masking ratio in a regime close to 30%, scheduled models match the language modelling capabilities of 30% masking throughout the entire pretraining duration.

4 Discussion

A decreasing masking rate schedule significantly improves the average downstream performance over the typical constant masking rate baselines. We hypothesize that this is because the dynamic schedule receives both the benefits of higher and lower masking rates. At higher masking rates the model learns more robust representations to handle the increased loss of information, and also receives a greater training signal. In contrast, at lower masking rates, the model has more usable context, which more closely mimics the setting in which the model will be used for inference.

We would like to highlight two of our results: 1) masking rate scheduling is a Pareto improvement over fixed masking rates on GLUE for *all examined pretraining durations*, and 2) masking rate scheduling yielded models that were of equivalent or better quality than all constant masking rate models for *every* GLUE task. These results together show that we did not observe any scenario in which scheduled masking rates are detrimental to model quality—and that scheduled masking rates are usually an improvement. This demonstrates the broad utility of scheduled masking rates.

Our method of beginning with a larger masking ratio and decaying parallels the motivation behind *simulated annealing* (Kirkpatrick et al., 1983). Simulated annealing is a general method for avoiding local minima by smoothing the loss surface early in training. It is implemented by adding noise early in training and gradually reducing the amount of noise as training continues. By adding noise, we remove more information and detail from the loss surface; similarly, by increasing the masking rate, we remove more information about input tokens.

5 Related work

5.1 Masked Language Modeling

With the advent of Transformer networks (Vaswani et al., 2017), self-supervised pretraining has become the dominant paradigm in NLP. In the seminal BERT work by Devlin et al. (2019), they propose to use encoder-only transformers pretrained using the MLM objective. Many works since BERT have made architectural changes to the model while still using the original MLM objective, including the 15% constant masking rate (Liu et al., 2019; Lan et al., 2020; Zaheer et al., 2020; He et al., 2021).

Other works on encoder-only models have modified the MLM objective itself to mask out spans of tokens instead of individual tokens (Joshi et al., 2020; Zhang et al., 2019; Levine et al., 2021). We note that the strategy of masking whole clusters of tokens is compatible with our proposed masking rate scheduling.

ELECTRA (Clark et al., 2020) proposes an alternate denoising objective to masking, where a subset of tokens in the input sequence are replaced using a separate "generator" encoder language model. While ELECTRA may implicitly schedule the masking rate as the generator makes fewer mistakes in reconstructing the input, there is a benefit to explicitly scheduling the rate since accuracy can be sensitive to this parameter (Appendix F). Additionally, the generative model is trained using MLM and as such could benefit from masking rate scheduling.

Most related to our work is that of Wettig et al. 2022 which investigates whether the standard 15% masking rate is optimal. However, their work still uses constant masking rates.

5.2 Hyperparameter scheduling

Learning rate is the most commonly-scheduled hyperparameter (Loshchilov and Hutter, 2017; Smith, 2017; Howard and Ruder, 2018), but other parameters are often scheduled in machine learning.

Our approach is not the first to schedule a hyperparameter that removes information content from the model such as dropout (Morerio et al., 2017; Zhou et al., 2020) or input resolution (Howard and Gugger, 2020). However, unlike these works, our masking rate scheduling also impacts the training signal provided to the model.

Scheduling has also been applied to hyperparameters which control the training signal to the model such as batch size (Smith et al., 2018) and sequence length (Li et al., 2022). However, these parameters do not affect the noise in the model's representations.

6 Conclusion

We introduce the technique of scheduling the masking rate during MLM pretraining. We demonstrate that masking rate scheduling can lead to model improvements on downstream tasks. We experiment with a variety of schedulers and scheduling rates, finding that decreasing masking schedules outperform their increasing counterparts. Specifically, we find that linearly scheduling the masking rate from 30% to 15% signifigantly improves average downstream accuracy on GLUE as compared to constant masking rates of 15% and 30%. The quality improvement is also fungible with time—when training to the same final GLUE accuracy as the 30% or 15% constant masking rate models, using a decreasing scheduled masking rate speeds up training by 1.65x or 1.89x, respectively.

Additionally, we investigate the source of masking rate scheduling's success, demonstrating that dynamic scheduling receives both the benefit of improved syntactical understanding from a lower masking rate regime, and improved language modeling capabilities from a higher masking rate regime.

Overall, our work shows that masking rate scheduling is a simple and reliable way to both improve the quality and efficiency of models trained with an MLM objective.

References

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Yosef Hochberg. 1988. A sharper Bonferroni procedure for multiple tests of significance. *Biometrika*, 75(4):800–802.

J. Howard and S. Gugger. 2020. *Deep Learning for Coders with Fastai and Pytorch: AI Applications Without a PhD*. O'Reilly Media, Incorporated.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2020. SpanBERT: Improving Pre-training by Representing and Predicting Spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science*, 220(4598):671–680.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Karim Lasri, Alessandro Lenci, and Thierry Poibeau. 2022. Word order matters when you increase masking.

Yoav Levine, Barak Lenz, Opher Lieber, Omri Abend, Kevin Leyton-Brown, Moshe Tennenholtz, and Yoav Shoham. 2021. Pmi-masking: Principled masking of correlated spans. In *International Conference on Learning Representations*.

Conglong Li, Minjia Zhang, and Yuxiong He. 2022. The stability-efficiency dilemma: Investigating sequence length warmup for training GPT models. In *Advances in Neural Information Processing Systems*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Ilya Loshchilov and Frank Hutter. 2017. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Pietro Morerio, Jacopo Cavazza, Riccardo Volpi, Rene Vidal, and Vittorio Murino. 2017. Curriculum dropout. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

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.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. 2018. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*.

Hanlin Tang, Ravi Rahman, Mihir Patel, Moin Nadeem, Abhinav Venigalla, Landan Seguin, Daya S. Khudia, Davis Blalock, Matthew L Leavitt, Bandish Shah, Jamie Bloxham, Evan Racah, Austin Jacobson, Cory Stephenson, Ajay Saini, Daniel King, James Knighton, Anis Ehsani, Karan Jariwala, Nielsen Niklas, Avery Lamp, Ishana Shastri, Alex Trott, Milo Cress, Tyler Lee, Brandon Cui, Jacob Portes, Laura Florescu, Linden Li, Jessica Zosa-Forde, Vlad Ivanchuk, Nikhil Sardana, Cody Blakeney, Michael Carbin, Hagay Lupesko, Jonathan Frankle, and Naveen Rao. 2022. Composer: A PyTorch Library for Efficient Neural Network Training.

Wilson L. Taylor. 1953. "Cloze procedure": a new tool for measuring readability. *Journalism Quarterly*, 30:415–433. Place: US Publisher: Association for Education in Journalism & Mass Communication.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Alexander Wettig, Tianyu Gao, Zexuan Zhong, and Danqi Chen. 2022. Should you mask 15% in masked language modeling?

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics.

Wangchunshu Zhou, Tao Ge, Furu Wei, Ming Zhou, and Ke Xu. 2020. Scheduled DropHead: A regularization method for transformer models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1971–1980, Online. Association for Computational Linguistics.

A Training Details

Modeling details. We use a `BERT-base` model as implemented in HuggingFace (Wolf et al., 2020). To manage the training of models we use the Composer library (Tang et al., 2022). All training is conducted on 8 NVIDIA A100 GPUs.

Pretraining. For our experiments we perform 3 trials of MLM pretraining on a 275 million document subset of the Colossal Cleaned Common Crawl (C4) dataset (Raffel et al., 2019). Following a learning rate warm-up period of 6% of the total training duration, we linearly schedule the learning rate from 5e-4 to 1e-5. We use the AdamW optimizer (Loshchilov and Hutter, 2019) with parameters $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = $ `1e-6`, and a weight decay of `1e-5`. All models are trained using a sequence length of `128` and a batch size of `4096`.

Downstream evaluation. We fine-tune and evaluate all models on the GLUE benchmark (Wang et al., 2018) which is composed of a variety of tasks evaluating different natural language tasks. All fine-tuning results are repeated for 5 trials for each pretraining trial.

B Significance testing

For a given task, to determine whether a masking rate schedule has performance comparable to the masking rate schedule with the best mean performance across seeds, we compute a one-sided t-test of the hypothesis "Schedule X performance is less than schedule Y performance", where X is the schedule being compared and Y is the schedule with the best mean performance. Since we are computing multiple pair-wise t-tests, we correct the pairwise t-tests using the Hochberg step-up procedure (Hochberg, 1988). If the corrected P-value is less than 0.05 we reject the null hypothesis and conclude that the schedule with the greater mean performance significantly outperforms the alternative schedule.

C Sweeping Schedule Hyperparameters

In scheduling the masking rate, we introduce two new parameters: the initial masking rate and the final masking rate. To determine the optimal configuration of these parameters for the `BERT-base` experiments, we performed the following search over parameter configurations. For all experiments, we used the same training setup as presented in Appendix A and selected the best hyperparameters based on the model's performance on the GLUE benchmark. We first determined the optimal constant rate, by pretraining with constant masking rates in $\{15\%,20\%,25\%,30\%,35\%\}$. After determining that 30% was the optimal masking rate for constant masking schedules, we fixed 30% to be the starting masking rate for our linear schedules, and swept over final masking rates of $\{15\%,20\%,25\%,35\%,40\%,45\%\}$. From this sweep, we determined that `linear-0.3-0.15` was the optimal linear schedule. Furthermore, decreasing masking rate schedules consistently outperform constant masking rate schedules (Table 3).

Schedule	MNLI-m/mm	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Avg
Constant									
constant-0.15	84.3/84.71	90.38	**88.31**	**76.65**	**92.91**	**91.94**	55.89	89.38	83.83
constant-0.3	84.5/84.83	**90.82**	**88.31**	**76.56**	**92.79**	**92.18**	57.24	**89.85**	84.12
Decreasing									
linear-0.3-0.15	**84.61/85.13**	**90.89**	**88.34**	76.25	**92.71**	**91.87**	**58.96**	**89.87**	**84.29**
linear-0.3-0.2	**84.57**/84.89	**90.87**	**88.33**	**77.04**	**92.84**	91.38	57.29	**89.78**	84.11
linear-0.3-0.25	**84.63**/84.93	**90.84**	**88.33**	76.1	**92.84**	**92.02**	57.33	**89.19**	84.02
Increasing									
linear-0.3-0.35	84.31/84.85	**90.73**	**88.28**	**76.9**	**92.91**	**91.68**	55.85	89.7	83.91
linear-0.3-0.4	84.19/84.71	**90.74**	**88.31**	**76.82**	92.49	**91.79**	55.67	87.83	83.62
linear-0.3-0.45	84.07/84.68	**90.85**	**88.29**	**77.02**	92.43	**91.98**	55.84	**89.92**	83.9

Table 3: Downstream performance for different schedules and configurations. For each model we report the average accuracy for each task in GLUE. Bold indicates no significant difference from highest-performing schedule, $P > 0.05$, t-test.

	Schedule		
Task	linear-0.3-0.15	constant-0.15	constant-0.3
Anaphor Agreement	**98.72**	**98.77**	**98.63**
Argument Structure	**76.13**	**76.59**	75.36
Binding	**76.13**	**75.76**	**74.91**
Control Raising	**78.31**	**79.17**	77.13
Determiner	**95.51**	**95.72**	**95.43**
Ellipsis	**85.38**	**84.63**	**85.88**
Filler Gap	**79.71**	**78.37**	77.38
Irregular Forms	**91.02**	**90.0**	**90.87**
Island Effects	**78.11**	**76.17**	**78.34**
Npi Licensing	**80.62**	**80.26**	**81.63**
Quantifiers	**81.08**	**81.79**	79.93
Subject Verb Agreement	**90.17**	**90.37**	89.47
Overall	**82.7**	**82.44**	82.13

Table 4: Average accuracy for each super-task in BLiMP. Bold indicates mean + standard error matches best average.

D Grammatical Understanding

In this section we further detail the BLiMP (Warstadt et al., 2020) benchmark.

BLiMP sub-tasks are organized into a collections of super-tasks which categorize a given linguistic phenomenon. Each sub-task is composed of minimal pairs of correct (positive) sentences and incorrect (negative) examples. The model correctly evaluates an example pair if it assigns a higher probability to the positive sentence in the pair than the negative sentence. However, we note that BERT is not a true language model as it does not produce a probability score over a sequence of tokens. Accordingly, following Salazar et al. 2020, we use the *pseudo-log-likelihood (PLL)* to score each sentence. The PLL is computed by iteratively masking each position in the input sequence and then summing the log likelihood of each masked token.

We present and discuss the average model performance across all tasks in Section 3.3, finding that `linear-0.3-0.15` outperforms `constant-0.3` and has similar performance to `constant-0.15`. In table 4, we present the performance on each individual super-task. We find that `linear-0.3-0.15` and `constant-0.15` have accuracies within one standard error of each other across all super-tasks in BLiMP. Additionally, `linear-0.3-0.15` outperforms `constant-0.3` on 5 out of the 12 BLiMP super-tasks and achieves parity on all other taks.

Lasri et al. (2022) found that in a synthetic setting, higher masking rates increase model dependence on positional information and thus improve syntactic understanding. Interestingly, we find the opposite effect: `constant-0.15` significantly outperforms `constant-0.3` on BLiMP. This observation, combined with the improved overall performance of scheduling, suggests that the improvement in grammar from scheduling is not simply due to being exposed to a higher masking rate.

E Computing Scheduling Speedup

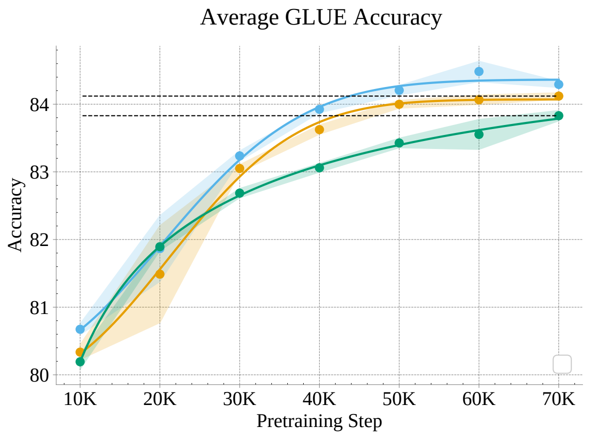

Figure 3: Pretraining step vs interpolated average GLUE accuracy.

To compute the efficiency gain of linear scheduling, we evaluate all models on GLUE after every 10K model steps. We then perform a regression on the number of model steps and the corresponding average GLUE performance using a model of the form:

$$c_1 - c_2 \exp\{(-(c_3 t)^{c_4}\}$$

where c_i are the regression variables and t is the pretraining step. After fitting a model to each schedule's step vs. GLUE performance, we compute the expected speedup by solving for the step in which one schedule achieves the best GLUE performance of the schedule being compared to. We demonstrate the regressed pretraining step vs GLUE performance curves in Figure 3

We evaluate speedup as a function of pretraining step instead of wall-clock time as dynamic schedules and constant schedules have identical throughput.

F Nonlinear Schedules

Let $\mathcal{T}_{\texttt{total}}$ be the total number of steps the model takes during training and t_i be the current model step. Let p_i and p_f be the initial and final masking rate respectively. For each step, we set the masking rate p_{mask} according to the following schedules. In Figure 4 we provide a graphical representation of the different schedules experimented with which we detail below.

Cosine scheduling. We directly adopt cosine scheduling as proposed in (Loshchilov and Hutter, 2017). We perform cosine scheduling by annealing the masking rate following half a cycle of a cosine

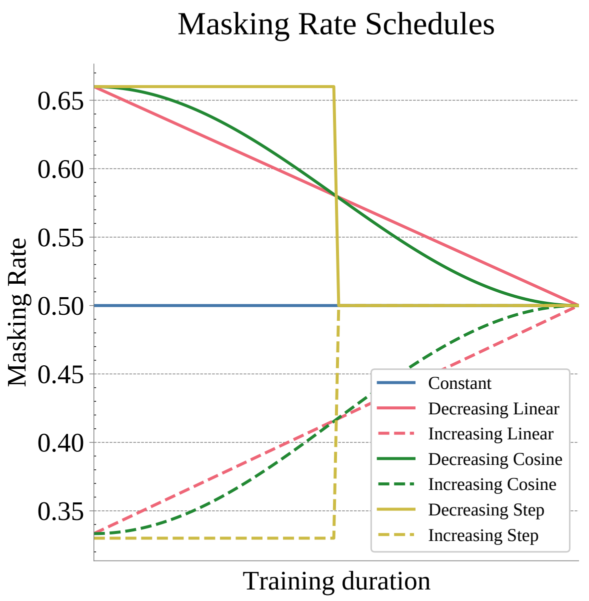

Figure 4: Various masking rate schedules we considered. Schedules can be constant, increasing or decreasing, and change following a linear, cosine, or step function. In the figure, increasing schedules have parameters $p_i = 0.33$, $p_f = 0.5$ and decreasing schedules have parameters $p_i = 0.66$, $p_f = 0.5$

curve. The masking rate is then defined as:

$$p_{\text{mask},t} = p_i + \frac{1}{2} * (p_f - p_i) * (1 + \cos((1 - \frac{t}{\mathcal{T}_{\texttt{total}}})\pi))$$

We refer to cosine schedules as `cosine-`$\{p_i\}$`-`$\{p_f\}$.

Step-wise scheduling. Step wise scheduling is defined by a decay rate, γ, and a set of timesteps, $\Gamma = \{t_1,...,t_m\}$, for when the masking rate is decayed. The schedule is then defined as:

$$p_{\text{mask},t} = \begin{cases} \gamma * p_{\text{mask},t-1}, & t \in \Gamma \\ p_{\text{mask},t-1} \end{cases}$$

Our experiments are restricted to step-wise schedules that apply the decay to the masking rate only once, halfway through the training duration. As such, for ease of notation, we ignore the decay rate when talking about step-wise schedules and instead describe our step-wise schedules in terms of their initial and final masking rates. We refer to step-wise schedules as `step-`$\{p_i\}$`-`$\{p_f\}$.

F.1 Results

Following the same pretraining and evaluation setup (Section A), we evaluate the performance of `cosine-0.3-0.15` and `step-0.3-0.15`. We find that for

SCHEDULE	MNLI-M/MM	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	AVG
Constant									
CONSTANT-0.15	84.3/84.71	90.38	88.31	76.65	**92.91**	**91.94**	55.89	89.38	83.83
CONSTANT-0.3	84.5/84.83	**90.82**	88.31	76.56	**92.79**	**92.18**	57.24	**89.85**	84.12
Dynamic									
LINEAR-0.3-0.15	**84.61/85.13**	**90.89**	**88.34**	76.25	**92.71**	**91.87**	**58.96**	**89.87**	**84.29**
COSINE-0.3-0.15	**84.55**/84.97	**90.94**	**88.39**	**77.67**	**92.91**	**91.94**	57.45	89.64	**84.27**
STEP-0.3-0.15	**84.65/85.09**	**90.85**	**88.37**	**77.71**	**92.76**	**91.56**	57.47	89.59	**84.23**

Table 5: Downstream performance for different scheduler functions. For each model we report the average accuracy for each task in GLUE.

linear, cosine, and step-wise scheduling there is no statistically significant difference in average GLUE performance (Table 5). `linear-0.3-0.15` outperforms `cosine-0.3-0.15` on 3 tasks, under performs on 1 task, and achieves parity on the rest of the tasks in GLUE. Similarly, `linear-0.3-0.15` outperforms `step-0.3-0.15` on 2 tasks, under performs on 1 task, and achieves parity on the rest of the tasks in GLUE. In the context of these results, we conclude that the scheduler type is less significant than the schedule parameters, and as such conduct the primary experiments in our paper with respect to the simple linear scheduler.