Dynamic Masking Rate Schedules for MLM Pretraining

Zachary Ankner [1,2] **Naomi Saphra**[3] **Davis Blalock**[1]
Jonathan Frankle[1] **Matthew Leavitt**[4]
[1]MosaicML [2]Massachusetts Institute of Technology
[3]Harvard University [4]DatologyAI

Abstract

Most works on transformers trained with the Masked Language Modeling (MLM) objective use the original BERT model's fixed masking rate of 15%. We propose to instead dynamically schedule the masking rate throughout training. We find that linearly decreasing the masking rate over the course of pretraining improves average GLUE accuracy by up to 0.46% and 0.25% in `BERT-base` and `BERT-large`, respectively, compared to fixed rate baselines. These gains come from exposure to both high and low masking rate regimes, providing benefits from both settings. Our results demonstrate that masking rate scheduling is a simple way to improve the quality of masked language models, achieving up to a 1.89x speedup in pretraining for `BERT-base` as well as a Pareto improvement for `BERT-large`.

1 Introduction

BERT (Devlin et al., 2019) is a popular encoder-only Transformer (Vaswani et al., 2017) architecture that is pretrained using a Cloze-inspired (Taylor, 1953) masked language modeling (MLM) objective. During MLM training, we mask out a subset of the input tokens and train the model 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 suboptimal 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.

Correspondence to ankner@mit.edu.

Hyperparameter scheduling—i.e., changing the learning rate, dropout rate, batch size, sequence length, etc., during training—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). Masking rate is a good candidate for hyperparameter scheduling for a number of reasons. First, a high masking rate, like a high dropout rate, directly reduces the amount of feature information available during a training step. This information removal may smooth the loss landscape, which permits simulated annealing if performed earlier in training. Furthermore, a higher masking rate adds training signal, as loss is computed for a larger portion of tokens, similar to a larger sequence length or batch size. We therefore study whether scheduling the masking rate during training could lead to model quality improvements, as scheduling these other hyperparameters does.

We present a series of experiments to assess the effects 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 downstream tasks. Our contributions are:

- We introduce a method of masking rate scheduling[1] for improving MLM pretraining (Section 3.1), and find that performance improves only when starting at a higher ratio and decaying it (Section 3.3).
- We show that the improvement from scheduling the masking rate is a Pareto improvement over fixed masking rates (Section 3.2, Appendix E), and that our method transfers to other pretraining objectives (Appendix H).

[1]After submitting this work, we were made aware of recent work (Yang et al., 2023) that also applies dynamic masking rates to MLM pretraining. Our method for scheduling masking rates differs slightly but our analysis of the technique substantially differs by focusing on understanding how scheduling improves MLM performance. We discuss these differences in Section 4.

- We find that dynamic scheduling attains both the improved linguistic performance of a lower masking rate (Section 3.5) and improved language modeling of a higher masking rate (Section 3.6).

2 Methods

We perform typical MLM pretraining, 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, and p_{mask} be the probability with which tokens are masked from the model, i.e., the masking rate. A mask $\mathcal{M} = \{m_1, ..., m_k\}$ 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 defined as:

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

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 G).

Constant scheduling. Constant scheduling, which we call constant-{p_{mask}}, 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:

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

Linear scheduling. In the linear schedule linear-{p_i}-{p_f}, the masking rate is set to a linear interpolation between the initial and final masking rate:

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

3 Experiments and Results

In this section, we evaluate the performance of masking rate scheduling on a collection of downstream tasks and determine 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). We use `BERT-base` and `BERT-large` models as implemented in HuggingFace (Wolf et al., 2020), and train models with 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 focus on comparing between `linear-0.3-0.15` and `constant-0.3-0.3` for `BERT-base`, and between `linear-0.4-0.25` and `constant-0.4-0.4` for `BERT-large`. These settings provide the best-performing linear and constant schedules, respectively. (Results for other schedule hyperparameters are in Appendix C.) For `BERT-base`, 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, leading to an average GLUE accuracy of 84.29%, a statistically significant improvement over the `constant-0.3-0.3` baseline of 84.12%. For `BERT-large` we find that `linear-0.4-0.25` improves performance over the baseline on 4 of the 8 GLUE tasks and achieves parity on all other tasks, leading to an average GLUE accuracy of 87.22%, a statistically significant improvement over the `constant-0.4-0.4` baseline of 86.97%. These results show that scheduling the masking rate during pretraining produces higher-quality models for downstream tasks.

3.2 Improvement in training efficiency

In addition to improving final model quality, pretraining with masking rate scheduling is more efficient in wall clock time. For `BERT-base`, 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,

SCHEDULE	MNLI-M/MM	QNLI	QQP	RTE	SST-2	MRPC	COLA	STS-B	AVG
BERT-base									
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**
BERT-large									
CONSTANT-0.4	87.43/87.68	93.03	**88.84**	**83.25**	**94.48**	**93.64**	**63.53**	90.82	86.97
LINEAR-0.4-0.25	**87.69/87.9**	**93.33**	**89.23**	**83.14**	**94.59**	**93.86**	**64.07**	**91.21**	**87.22**

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 best-performing schedule, $P > 0.05$, t-test.

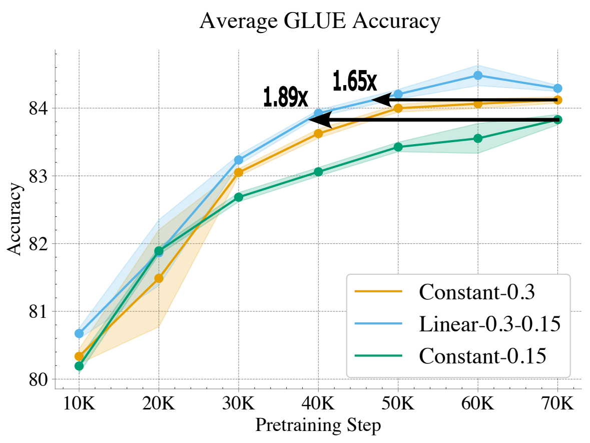

Figure 1: Average GLUE accuracy evaluated over the course of pretraining for `BERT-base`. The horizontal lines correspond to the difference in steps required for `linear-0.3-0.15` to achieve the best constant schedule performance.

`linear-0.3-0.15` is a Pareto improvement over both constant baselines; for each pretraining step evaluated, `linear-0.3-0.15` matches or exceeds the baseline with no increase in training time (Figure 1). For `BERT-large`, `linear-0.4-0.25` is also a Pareto improvement over `constant-0.4` (Appendix E). Appendix F contains further details on evaluating model speedups.

3.3 High to low, not low to high

To better understand how masking rate scheduling affects training dynamics, we investigate whether the scheduler must always gradually decrease the masking rate, in line with an interpretation based on simulated annealing (Kirkpatrick et al., 1983). If we find that either decreasing or increasing lead to similar improvements, then we instead would attribute the success of our method to just the range of masking rates covered. We find that

SCHEDULE	AVG GLUE ACCURACY
CONSTANT-0.15	83.83
LINEAR-0.15-0.3	83.71
LINEAR-0.3-0.15	**84.29**

Table 2: Average GLUE accuracy for increasing/decreasing schedules with the same range of masking rates. Bold indicates no significant difference from the highest-performing schedule, $P > 0.05$, t-test.

the reversed schedule `linear-0.15-0.3` performs significantly worse than the decreasing schedule `linear-0.3-0.15` on GLUE for `BERT-base`, and in fact has performance comparable to the `constant-0.15` baseline (Table 2).

3.4 Masking and loss are both necessary for improved performance

Is the added signal from a dynamic masking rate necessary, or does the removal of information from the inputs determine the majority of the gain? Here, we distinguish two possible sources of benefit from our schedule: benefits from smoothing the loss surface; and benefits from adding training examples by increasing the number of masked words to predict. To test whether the latter is necessary, we pretrain a `BERT-base` model linearly scheduling the masking rate from 30% to 15%, but we only compute the loss on a subset of the masked tokens such that the loss is defined over 15% of the input tokens (referenced as `subset-linear-0.3-0.15`). We find that `subset-linear-0.3-0.15` under-performs both `linear-0.3-0.15` and `constant-0.15` (Table 3). This result suggests that obfuscating the input sequence according to a dynamic masking rate does not by itself improve modeling performance, and thus the increased signal is also necessary.

Schedule	Avg GLUE Accuracy
constant-0.15	83.83
subset-linear-0.3-0.15	83.71
linear-0.3-0.15	**84.29**

Table 3: Average GLUE score for scheduling masking rate while holding constant the number of tokens used in training. Bold results show no significant difference (t-tested $p < 0.05$) from the highest-performing schedule.

Schedule	Avg BLiMP Accuracy
linear-0.3-0.15	**82.70**
constant-0.15	**82.44**
constant-0.3	82.13

Table 4: Average accuracy across BLiMP tasks. Bold indicates mean + standard error matches best average.

3.5 Improvement in grammar capabilities

In order to better understand scheduling's effects on the linguistic capabilities of MLMs, we evaluated our models on the BLiMP benchmark (Warstadt et al., 2020); this benchmark tests understanding of syntax, morphology, and semantics.

We find the average BLiMP accuracy of `linear-0.3-0.15` significantly improves over `constant-0.3` and matches `constant-0.15` (Table 4). These results suggest that a dynamic schedule enables the linguistic capabilities of a lower masking rate.

3.6 Improvement in the pretraining objective

How does a decreasing schedule affect a model's language modeling ability? When evaluating models at a 15% masking rate, we find that `linear-0.3-0.15` and `constant-0.3` have the same average MLM loss of 1.56. However, `constant-0.15` performs significantly worse, with a best MLM loss of 1.59.

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

4 Related work

Masked Language Modeling Since ELMo (Peters et al., 2018), self-supervised pretraining has become the dominant paradigm for many NLP tasks, and BERT has been established as a basic standard for transfer learning. Many works have changed the BERT model architecture while retaining 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 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 both architectural changes and span masking are compatible with our masking rate scheduling.

ELECTRA (Clark et al., 2020) proposes an alternate denoising objective to masking; using a separate "generator" encoder language model, they replace a subset of tokens in the input sequence. While the gradual improvement of the generator may implicitly parallel a masking rate schedule, explicit scheduling may still be beneficial since accuracy can be sensitive to masking rate (Appendix G). Additionally, the generator is trained using an MLM objective, and as such could benefit from masking rate scheduling.

There has also been previous work exploring whether the standard 15% masking rate is optimal. Wettig et al. (2022) empirically investigate the optimal fixed masking rate and demonstrate that for larger BERT models higher masking rates are more performant.

Most closely related to our method is Yang et al. (2023), which also examines dynamic masking rates for MLM pretraining. Although there is significant overlap in the proposed methodologies, their work sets the final masking rate to be close to 0%, while we found that maintaining a higher final masking rate of 15% was necessary for performance improvements. Additionally, our analysis differs significantly from theirs. While both their work and ours evaluate downstream performance improvements, Yang et al. (2023) also investigates how dynamic masking rates affect performance when the training duration is extended and study nonrandom token masks. Our analysis, by contrast, focuses on why masking rate scheduling improves performance. To this end, we investigating whether dynamic masking rates must follow a decaying scheduling (Section 3.3), whether the observed gains are due to the additional training signal or the added noise (Section 3.4), the impact of differing masking rate schedules on grammatical capabilities (Section 3.5), and the impact of dynamic masking rates on the pre-training objective itself (Section 3.6).

Hyperparameter scheduling Although learning rate is the most commonly-scheduled hyperparameter (Loshchilov and Hutter, 2017; Smith, 2017; Howard and Ruder, 2018), other hyperparameter schedules are common. Our approach is also not the first to schedule a hyperparameter that removes information content from the model; prior work has suggested scheduling dropout (Morerio et al., 2017; Zhou et al., 2020) and input resolution (Howard and Gugger, 2020). Scheduling has also been applied to hyperparameters that control the training signal to the model such as batch size (Smith et al., 2018) and sequence length (Li et al., 2022). Masking rate combines both of these properties, making it a particularly good candidate for scheduling.

5 Discussion and Conclusions

In addition to our method's improvement on the average final downstream performance, we find that scheduling is a Pareto improvement for *all examined pretraining durations* over the typical constant masking rate baselines on GLUE. Our analysis suggests that this benefit comes from the combined advantages of higher and lower masking rates. We also demonstrate that our approach generalizes to other pretraining objectives (Appendix H).

Our method of beginning with a larger masking ratio and decaying, which we found necessary (Section 3.3), 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 through the addition of noise early in training. However, we found that the increasing noise early in training is not the only source of advantage. We also benefit from increasing the signal by predicting more masked tokens (Section 3.4).

Overall, our work demonstrates that masking rate scheduling is a simple and reliable way to improve the quality and efficiency of MLM pretraining.

Limitations

In this work, we restrict ourselves to English-only pretraining and finetuning. For other languages with free word order, there may be less information about the overall sentence structure when masking at a higher rate because the position of a word provides less information. As such our technique may not generalize or be suitable for other languages.

Additionally, we only investigate masking rate scheduling in the encoder setting. Further applying our method to encoder-decoder settings where the model is partially trained with a reconstruction loss, such as T5, is a direction for future research.

Finally, we only evaluate models on the GLUE benchmark. While our evaluation is in line with previous work, a more comprehensive set of tasks could provide a better evaluation.

Acknowledgments

While performing this work, Naomi Saphra was employed by New York University and Matthew Leavitt was employed by MosaicML. This work was supported by Hyundai Motor Company (under the project Uncertainty in Neural Sequence Modeling) and the Samsung Advanced Institute of Technology (under the project Next Generation Deep Learning: From Pattern Recognition to AI).

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.

Luca Di Liello, Matteo Gabburo, and Alessandro Moschitti. 2022. Effective pretraining objectives for transformer-based autoencoders. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5533–5547, Abu Dhabi, United Arab Emirates. 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)*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

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.

Dongjie Yang, Zhuosheng Zhang, and Hai Zhao. 2023. Learning better masking for better language model pre-training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7255–7267, Toronto, Canada. 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` and `BERT-large` model as implemented in HuggingFace (Wolf et al., 2020), which have 110 million and 345 million parameters respectively. To manage the training of models we use the Composer library (Tang et al., 2022). All training is conducted on 8 NVIDIA A100 GPUs. `BERT-base` and `BERT-large` take approximately 10 hours and 24 hours to train respectively.

Pretraining. For our `BERT-base` 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). For `BERT-large` experiments, we perform 2 trials of MLM pretraining for 2 epochs of the C4 dataset. For all models, 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 decoupled 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 performs worse than schedule Y", 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 (Table 5), 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

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-0.2	**84.46/84.95**	90.64	**88.24**	76.73	92.59	**91.63**	**56.45**	89.6	83.92
CONSTANT-0.25	84.28/84.79	90.61	**88.3**	76.27	92.54	**92.06**	**56.74**	**89.84**	83.94
CONSTANT-0.3	**84.5**/84.83	**90.82**	**88.31**	76.56	**92.79**	**92.18**	**57.24**	**89.85**	**84.12**
CONSTANT-0.35	**84.4/84.99**	**90.84**	**88.31**	**77.81**	**92.86**	**91.67**	55.62	**89.88**	**84.04**

Table 5: Downstream performance for different constant schedule configurations. For each model, we report the average accuracy for each task in GLUE. Bold indicates no significant difference from the highest-performing schedule, $P > 0.05$, t-test.

Schedule	MNLI-m/mm	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Avg
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 6: Downstream performance for different linear schedule configurations. For each model, we report the average accuracy for each task in GLUE. Bold indicates no significant difference from the highest-performing schedule, $P > 0.05$, t-test.

the optimal linear schedule. Furthermore, decreasing masking rate schedules consistently outperform constant masking rate schedules (Table 6).

For computational reasons, we did not perform the corresponding sweep over scheduling rates for `BERT-large`. Instead, we follow the recommendation of Wettig et al. (2022) and use a 40% masking rate as the best constant masking rate. We then propose `linear-0.4-0.15` as our dynamic schedule following the optimal setting of a 15% decreasing dynamic schedule observed from our sweep over hyperparameters for `BERT-base`.

D Grammatical Understanding

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

BLiMP sub-tasks are organized into collections of super-tasks that 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 for `BERT-base` across all tasks in Section 3.5, finding that `linear-0.3-0.15` outperforms `constant-0.3` and has similar performance to `constant-0.15`. In table 7, 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 tasks.

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.

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

E BERT-Large Downstream Performance Throughout Pretraining

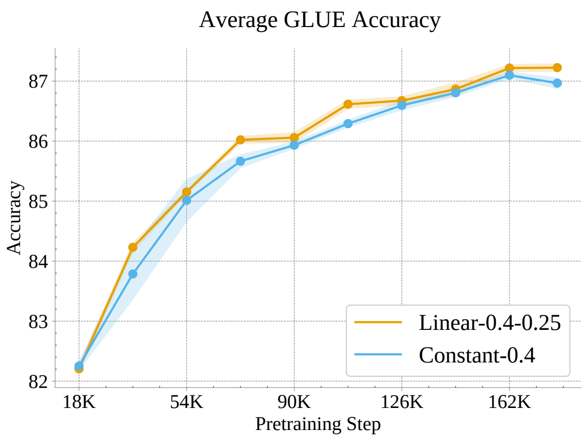

Figure 2: Average GLUE accuracy evaluated over the course of pretraining for BERT-large.

In this section we report the average GLUE performance from different pretraining checkpoints of `linear-0.4-0.25` and `constant-0.4` for `BERT-large` (Figure 2). We find that `linear-0.4-0.25` is a Pareto improvement over `constant-0.4` for each pretraining step evaluated. This means that `linear-0.4-0.25` exceeds or matches baseline performance for no increase in training time.

F Computing Scheduling Speedup

To compute the efficiency gain of linear scheduling, we evaluate all models on GLUE after every 10K pretraining 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. We show 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 because dynamic schedules and constant schedules have identical throughput.

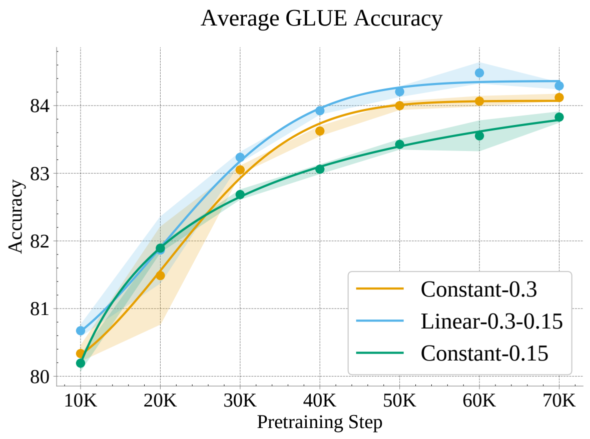

Figure 3: Pretraining step vs interpolated average GLUE accuracy for `BERT-base`.

G 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

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

Schedule	MNLI-m/mm	QNLI	QQP	RTE	SST-2	MRPC	CoLA	STS-B	Avg
rts-constant-0.15	83.06/83.46	90.64	88.22	**75.38**	**92.06**	91.21	**56.87**	89.92	83.42
rts-constant-0.3	83.09/**83.72**	**90.64**	88.27	**75.74**	**91.9**	91.15	55.41	90.02	83.33
rts-linear-0.3-0.15	**83.54/83.91**	**90.83**	**88.37**	74.15	**92.06**	**91.76**	**57.53**	**90.21**	**83.60**

Table 9: Downstream performance for different random substitution rate schedules. For each model, we report the average accuracy for each task in GLUE. Bold indicates no significant difference from best-performing schedule, *P > 0.05*, t-test.

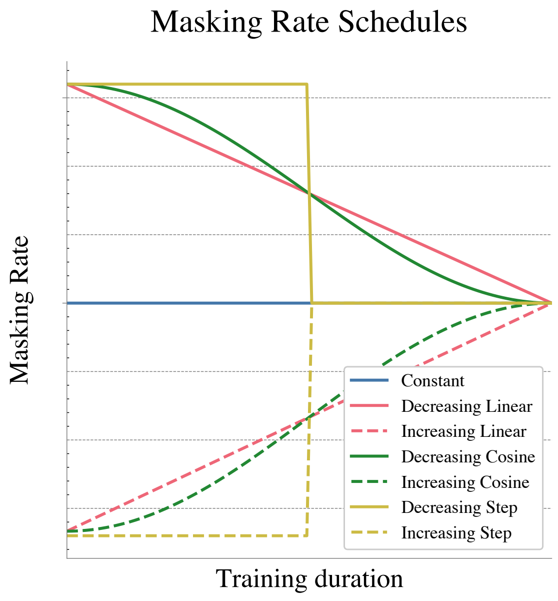

Figure 4: Various masking rate schedules we considered. Schedules can be constant, increasing or decreasing, and change following a linear, cosine, or step function.

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 curve. The masking rate is then defined as:

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

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-`$\{p_i\}$`-`$\{p_f\}$.

G.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 linear, cosine, and step-wise scheduling there is no statistically significant difference in average GLUE performance (Table 8). We find that `linear-0.3-0.15` outperforms `cosine-0.3-0.15` on 3 tasks, underperforms 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, underperforms 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.

H Generalization to Other Objectives

H.1 Set-Up

In order to further demonstrate the success of dynamically scheduling the pretraining objective for encoder transformers, we evaluate dynamically scheduling the token substitution in the Random Token Substitution (RTS) objective (Di Liello et al., 2022). In the RTS objective a subset of tokens, defined by the random token substitution rate, are randomly substituted with another token in the vocabulary. The model is then trained to classify whether a token was randomly substituted or is the original token. The random token substitution rate was originally set to be a constant 15%. In our work, we experiment both with a constant 30% and linearly decreased from 30% to 15% random token substitution rate.

All other hyperparameters and data choices are the same as the ones we used for MLM training of `BERT-base` (Appendix A).

H.2 Results

Improvement in final performance We examine the effect of scheduling the random token substitution rate on downstream GLUE performance (Table 9). As `rts-constant-0.15` is the better-performing constant schedule for RTS, we focus our comparison on this baseline. We find that `rts-linear-0.3-0.15` outperforms `rts-constant-0.15` on 6 out of the 8 tasks in GLUE, and only performs worse on 1 task, leading to an average improvement on GLUE of 0.18%. This result demonstrates that the improved gains from dynamically scheduling the pretraining objective for `BERT` style models also generalize to the RTS task.

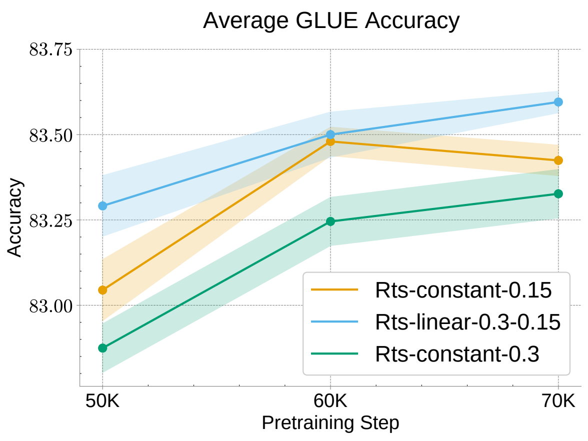

Figure 5: Pretraining step vs interpolated average GLUE accuracy for RTS with `BERT-base`.

Performance throughout pretraining We examine the effect at different points of pretraining of scheduling the random token substitution rate. Specifically, we compute the downstream GLUE accuracy for the different schedules at 50K, 60K, and 70K of training. We find that `rts-linear-0.3-0.15` is a Pareto improvement over both `rts-constant-0.3` and `rts-constant-0.15`, meaning linear scheduling performs better for each intermediate checkpoint evaluated (Figure 5).