FORGETTER with forgetful hyperparameters and recurring sleeps can continue to learn beyond normal overtfitting limits

Rui Yamamoto and **Keiji Miura**
Kwansei Gakuin University
1 Gakuen Uegahara, Sanda, Hyogo 669-1330, JAPAN
miura@kwansei.ac.jp

Abstract

LLMs suffer from considerable computational costs in training. A more biologically plausible curriculum learning may help to decrease the learning costs. Here we propose a FORGETTER training algorithm, in which a model forgets the variables for optimization after a sleep and the hyperparameters are set toward forgetting memory: rather large weight decay and learning rates as well as small but optimized batch sizes. By limiting minGemma model to 512 input length and speeding up the development cycle, we compared normal and FORGETTER learning algorithms by using more than a thousand different models. Specifically, we found and utilized the "120-rule" that the models with about 120 (Query) heads in total, irrespective of the head number per layer, outperform. The improvement by using the FORGETTER algorithm is far bigger than that by optimizing the model structure. Specifically, FORGETTER models can learn beyond the data size where the normal learning overfits. The FORGETTER also works for CIFAR10 image classification. These results suggest that forgetting can be beneficial for pretraining deep neural networks by avoiding overfitting.

1 Introduction

Although ChatGPT's performance was amazing enough to revolutionize what are human jobs (OpenAI, 2023; Rothman, 2024), they require considerable computational resources (Tunstall et al., 2022). While many approaches for making the training more efficient have been proposed (Goodfellow et al., 2016; Atienza, 2020; Chollet, 2021; Geron, 2022), recent LLMs are too huge to explore their hyperparameters and learning algorithms exhaustively.

As human babies do not need as many resources to learn a language apparently, a more efficient learning method may remain to be discovered (Ford, 2018; Warstadt et al., 2023). It is true that nowadays a handy Trainer class in PyTorch provides a normal training routine, which enables us to explore various model structures and other hyperparameters quite easily. However, it may be also promising to explore unconventional learning procedures (Smith, 2017; Zhao et al., 2024), possibly learned from the biological brain.

In the same vein, BabyLM data is very suitable to mimic how human babies earn language abilities including grammars (Warstadt et al., 2023; Mahowald et al., 2024) with rather small language models (Raschka, 2024; Lu et al., 2025; Tunador). Small language models allow us to explore more hyperparameters related to learning (Warstadt et al., 2023). Although many models succeeded to learn BabyLM10M/100M in the past contests, the biological plausibility of the training algorithm (Konishi et al., 2023; Lillicrap et al., 2020) was not necessarily pursued.

We believe that two ingredients are important for biological plausibility. First of all, animals are not perfect and inevitable to forget. For example, animals cannot keep huge training data and memory traces without fail. Thus, a biologically plausible model should allow animals to forget to reasonable extent. Furthermore, animals sleep, which also forces them to forget. If these inevitable forgetting is beneficial or not remains an open question. The observations on the effect of sleep on learning in neuroscience (Norimoto et al., 2018) should be incorporated for training LLMs. (The algorithm that works for big data with LLMs is also promising as the computational model of the brain.)

Here we proposed the FORGETTER model, that is trained with sleep and forgetful hyperparameters. We demonstrate that it can learn beyond the data size where the normal learning overfits.

In Section 2, as a methodology, we explain the base model we used and the novel FORGETTER training method, in which we insert sleep (variables for optimization are initialized) between epochs.

Proceedings of the First BabyLM Workshop, pages 91–99

This may be regarded as a procedural learning. In Section 3, as results for Baby10M, we show our FORGETTER model outperforms the normally trained models. Specifically, you can continue and repeat epochs for FORGETTER models, where couples of epochs suffice to excel the normal training. In Section 4, as results for Baby100M, we demonstrate that the benefit of FORGETTER is inherited to Baby100M. It can again learn beyond the normal limits. In Section 5, we demonstrates that the FORGETTER also works for CIFAR10 image classification. In Section 6, as Summary and Discussion, we summarized the results and discussed the strengths of the proposed methods. In Limitation, complement to the discussion of strengths in the main text, we discuss the limit of the proposed methods. Specifically if a FORGETTER model is always better than a normal model is an important question. We discuss how general can the benefit of the FORGETTER be.

2 Methods

All the computation was done by the custom-written Python codes on 7 PCs with NVIDIA RTX 3090, 4080, 4090 or A6000 GPU. All the codes to reproduce this paper's results and the list of validated loss for varieties of model structures trained with normal or FORGETTER algorithms are available at GitHub (https://github.com/keiji-miura/FORGETTER-BabyLM).

2.1 DATA

Baby10M and Baby100M dataset were used for (pre)training of next token prediction. The both Baby10M and Baby100M data were tokenized by the GPT2 Tokenizer. We saved the tokenized data to a single file (separately for Baby10M/Baby100M or training/validation) to speed up I/O during training, in which consecutive 512 tokens were cut out at a random starting point for training data. There, we used different random starting points for different 512 tokens within a batch.

2.2 BASE MODELS

The minGemma model (Tunador; Gemma Team, 2024) was entirely used in this paper as a text generation model. We trained the minGemma model from scratch by using either the BabyLM10M or the BabyLM100M. In this paper, we solely compared the normal and FORGETTER models. The difference between the normal models and FORGETTER models are in the training algorithms.

Hyperparam.	**Baby10M**	**Baby100M**
Tokenizer	GPT2	GPT2
Input Size	512	512
Drop Out	No (p=0)	No (p=0)
Weight Decay	1.0	0.25
Batch Size	12	28
Learning Rate:		
- Normal	1.35×10^{-3}	10^{-3}
- FORGETTER	10^{-3}	0.8×10^{-3}
N Steps/Epoch:		
- Normal	19600	144000
- FORGETTER	10000	70000

Table 1: Hyperparameters. Fixed setting (GPT2 Tokenizer, input length=512, no drop-out) speeded up the development cycle, which enabled us to explore different training methods and varieties of model structures.

2.2.1 model representation

We represent a model by the combination of the numbers such as "L24-6(3)-648×4-240". (These numbers are what we occasionally changes to explore better results.) The meaning of the numbers are the number of layers, the number of Query heads, the number of Key/Value heads (this must divide the number of Query heads), the hidden dimension at the input layer of the feed forward layer, the hidden dimension at the hidden layer of the feed forward layer, the head dimension at attention. Note that, as our minGemma model was Gemma-based (Gemma Team, 2024), we not only explored the number of Query heads but also the number of Key/Value heads in a grouped attention.

2.3 TRAINING ALGORITHM

2.3.1 Normal training algorithm

The normal model was trained by using the PyTorch Trainer class with a single epoch where the learning rate linearly decays to zero. The number of steps in an epoch was optimized so that more steps caused overtraining.

2.3.2 FORGETTER with sleep 1 (light sleep)

The FORGETTER models was trained by repeating the Pytorch Trainer with multiple epochs. There, in each epoch, the learning rate linearly decays to zero. Between epochs, the model was not initialized (specifically the weights were kept) while the variables for AdamW were initialized. That

is, only the optimizer was reinitialized between epochs. This is why we say FORGETTER models "sleep", after which the optimizer is initialized. When we simply mention "sleep", we mean this sleep 1.

2.3.3 FORGETTER with sleep 2 (deep sleep)

Because sleep is a rather vague concept, we can consider another definition for sleep. For "sleep2", not only the optimizers but also all the state variables in the model except weights are initialized. We implemented this simply by loading the pre-dumped weights to a newly constructed minGemma model in PyTorch.

In principle, at the transition of epochs, you can use either sleep 1 or sleep 2. However, we found sleep 2 can be most effective at last. That is, after the repetition of sleep 1, in the final epoch sleep 2 can drop the validated loss largely. We sometimes call this phenomenon "last big drop" by sleep 2.

In this paper, specifically, once (repeated) Sleep 1 overfits (=validated loss increases), it is switched to Sleep 2 (with the model in the previous epoch recovered), although Sleep 2 also overfits soon (at the second time or so) typically.

2.4 FIXED HYPERPARAMETERS for FORGETTING

Here we briefly describe three hyperparameters that are set toward forgetting memory: rather large weight decay and learning rates as well as small but optimized batch sizes.

2.4.1 Weight decay is as large as 1 or 1/4 for Baby10M or Baby100M

Conventionally, a small value of weight decay like as small as 0.01 has been used (see PyTorch document for example). However, we found that a rather large value of the weight decay was beneficial irrespective of the model structures and other hyperparameters. While the optimal weight decay strongly depends on the training data size, it does not strongly depend on the other hyperparameters like model structures, apparently. Therefore, we set weight decay to 1.0 for Baby10M and 0.25 for Baby100M.

Weight decay is the speed to forget weights. So it is convenient from the viewpoint of biological plausibility that the weight decay as large as 1.0 or 0.25 is optimal for pretraining. It seems that animals or babies can forget rather a lot and still achieve the best learning performance, fortunately. (Note that optimality here is regarding the next token prediction.) So we consider that the weight decay value we use throughout the paper is consistent with the idea of forgetful learning.

2.4.2 Batch size is as small as 12 or 28 for Baby10M or Baby100M

The batch size was fixed to 12 for Baby10M or 28 for Baby100M. This is because we believe that, while the optimal batch size strongly depends on the training data size, it does not strongly depend on other the hyperparameters like model structures.

Although 32 can work as well for Baby100M, you need more VRAM in that case. So we chose 28. But it is actually hard to judge which one is better under the high trial-to-trial variability in validated losses.

Batch sizes can be regarded as a memory for recently encountered data. 12 or 28 is rather small and not like 512 or 1024, which are typically used numbers in deep learning. So it is convenient from the viewpoint of biological plausibility that the batch size as small as 12 or 28 is optimal for pretraining. It seems that animals or babies need to memorize only small number of data to achieve the best performance, fortunately. So we consider the batch size is again consistent with the idea of forgetful (=small memory) learning.

2.4.3 Learning rates is as large as 0.001

Within each epoch the learning rate linearly decays to zero. The initial (maximum) learning rate in an epoch was fixed to about 0.001. The value we used may be rather large, compared with the conventinal one like 0.0001 or smaller (see PyTorch Document, for example). However, we found that a rather large value of the learning rate is beneficial irrespective of the model structures and other hyperparameters. That is, we believe that, while the optimal learning rate strongly depends on the training data size, it does not strongly depend on other hyperparameters like model structures. Therefore we set learning rate to 10^{-3}. (To be precise, we used the range from 0.8×10^{-3} to 1.35×10^{-3} as in Table 1.)

A learning rate can be regarded as a rate to forget past encounters. 0.001 is rather big. So it is convenient from the viewpoint of biological plausibility that the learning rate as large as 0.001 is optimal for pretraining. Therefore, animals or babies can forget rather a lot and achieve the best performance, fortunately. So we consider that the learning rate value we use is again consistent with the idea of

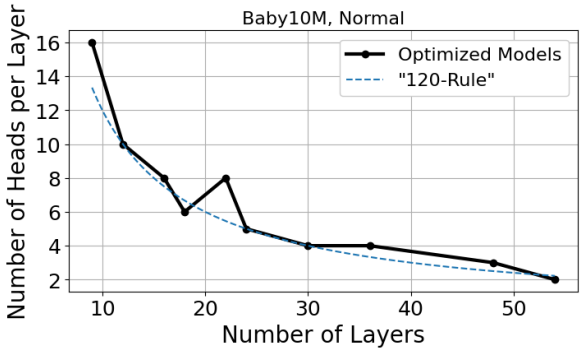

Figure 1: Number of (Query) heads per layer for best models for a given number of layers. These optimal models obey "120-rule" and have about 120 heads in total, irrespectively of the number of layers.

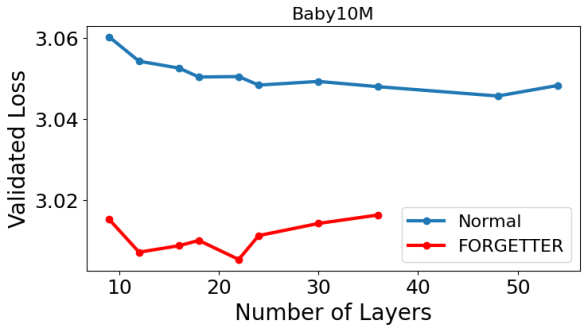

Figure 2: Validated losses for best models for a given number of layers trained with normal or FORGETTER algorithm for Baby10M.

forgetful (=small memory) learning.

3 Results for Baby10M

3.1 120 rule for searching model structure

First, good model structure was searched with the normal training algorithm for Baby10M. The model structures were explored in order to obtain as lowest validated loss as possible. (In this paper, the optimality is always about the validated loss for the next token prediction.)

To see the impact of the number of layers, we searched optimal model structures for a given number of layers (Figure 1). That is, we plotted the number of (Query) heads for the models whose structures are optimized for a given number of layers (by simple grid search for head number, hidden dim etc). Figure 1 demonstrated that, Surprisingly, the optimal models always had about 120 (Query) heads in total irrespective of the number of layers.

Although the result is variable even for the same hyperparameters and model structures, we tried more than a hundred models per layer for BabyLM10M. Therefore we believe the rule is true as an overall tendency. For example, regarding the validated loss for the normal training, it is rather easy to obtain <3.05 for 48-layers models but not for 18-layers models.

The blue line in Figure 2 denotes the validated losses for the same models as in Figure 1 for normal training. The deeper models performed well in general and the 48-layers model showed the best performance (3.0457).

54-layers models were worse, possibly because they have limited options on head numbers per layer. For example, 3 heads per layer is too much (3 x 54 = 162 heads in total), but 2 heads per layer is too little (2 x 54 = 108 heads in total).

3.2 NOTE: model search range can be limited

We succeeded not only to fix considerable hyperparameters without losing performance as in Table1, but also to limit the model structure search range.

The head dimension, the dimension projected immediately before the attention, was only explored from 96 to 352, because optimal values for a given number of layers were always between 192 to 288 for Baby10M trained with normal algorithms (i.e., in the well-explored category).

As the optiml number of Query heads per model was always around 120, we only needed to try limited ranges. For example, when we explored twelve-layer models, the models with ten or twelve Query heads tended to perform very well while too many or too little heads did not perform well. Sometimes we call this observational fact "120-rule" for short.

The hidden dimension for the token representation tended to be optimized around 700. So we only chose some value close to that within the multiples of the number of Query heads. For Baby10M trained with normal algorithms (i.e., a well-explored category), the optimal models for a given number of layers had from 576 to 832 dimensions to represent a single token.

The dimensions of the hidden layer of the feed forward layer is always fixed to the four times that of the input layer of the same feed forward block. Although we have changed from $\times 4$ to $\times 3$, $\times 5$, $\times 6$, $\times 8$, we could not observe significant improvements. (Consider this "$\times 4$" as a fixed parameter.)

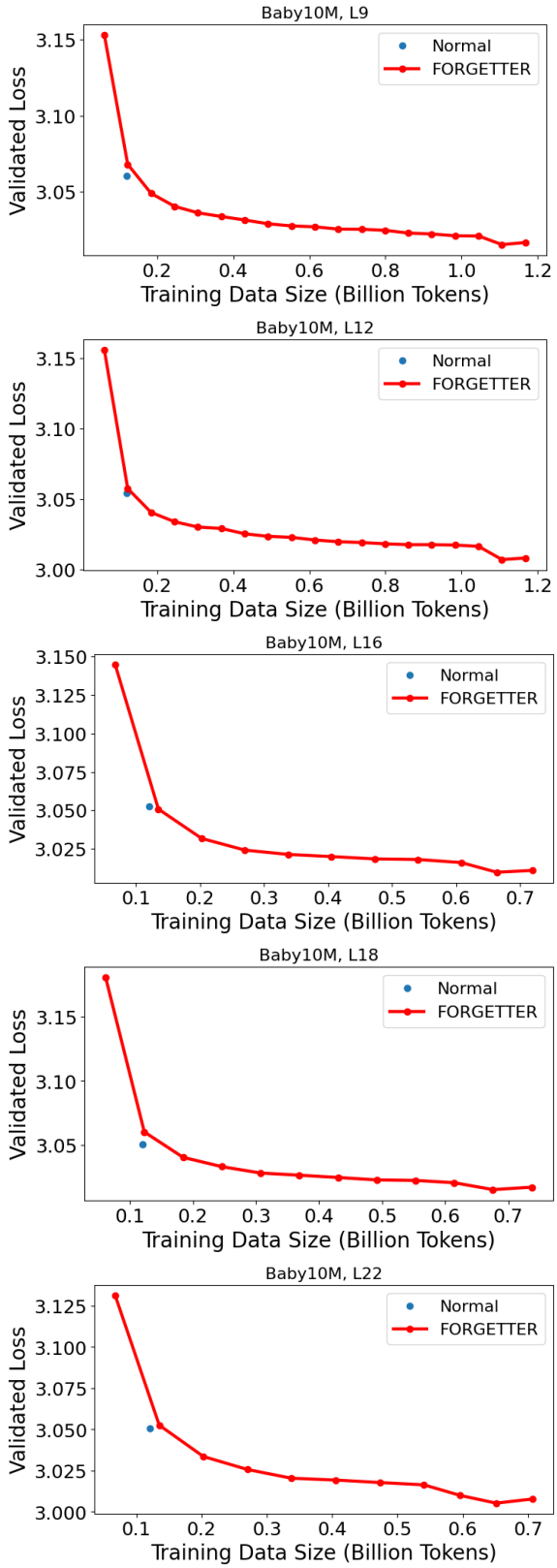

Figure 3: Five examples of training course of FORGETTER algorithm for Baby10M for models with 9, 12, 16, 18 and 22 layers. Validated losses for FORGETTERs are plotted as time series in red. The validated losses for normal models are denoted by a blue point.

Hyperparameter	Normal	FORGETTER
N Layers	48	22
N Q Heads/Layer	3	8
N K/V Heads/Layer	1	4
Hidden Dimension	648	672
FFI Dimension	648×4	672×4
Head Dimension	288	192
Validated Loss	3.0457	3.0053
BLiMP Score	0.6958	0.7257

Table 2: Optimal model structures for Baby10M when input token length is 512.

3.3 Normal vs FORGETTER

The results in Figure 2 demonstrate that FORGETTER models are always much better than normal models. Although the 48-layers model was the best for the normal training, the 22-layers model turned out to be the best for the FORGETTER training.

Note that the difference between normal and FORGETTER models are much larger than that by model structures. This means that the model structure search is not that fruitful. Rather, changing the learning curriculum to FORGETTER is much more efficient way to improve the performance.

In fact, if you look at the time course of the training, the FORGETTER model excels the normal model within a couple of iterations as shown in Figure3. The sleep interval of FORGETTER, that is optimized to minimize the validated loss for next token prediction task, was about half of that of normal models. This is shown as the number of steps per epoch in Table 1. (In normal models, there is no sleep and the entire training consists of only a single interval or epoch.) Therefore the computational time for the two epochs for FORGETTER models is roughly equivalent to that for normal models. At that time, their performances are almost equal. However, FORGETTER models can continue to learn beyond the normal limit as in Figure3. Note that the number of steps per epoch for the normal model (=19600) was already optimized. That means if you used longer steps (more training data) per epoch, the model would overfit and its validated loss deteriorates. Thus, it is interesting that FORGETTER can continue to learn beyond the normal overfitting limits.

Surprisingly, there is a drop in the end of the training (Figure 3). This was caused by sleep 2 (deep sleep). This drop is commonly observed among the models with different layers. The im-

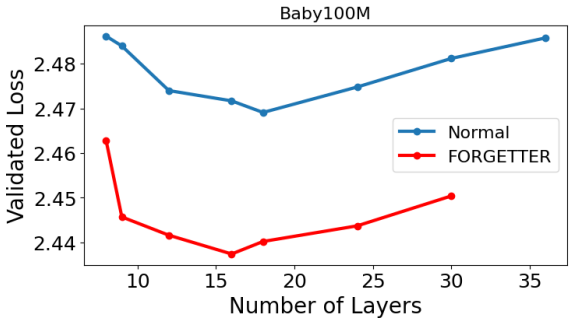

Figure 4: Validated losses for best models for a given number of layers trained with normal or FORGETTER algorithm for Baby100M.

pact of the last big drop is much larger than that by the difference of the model structures. Again, this means that the model structure search is not that fruitful. Rather, changing the learning curriculum to the FORGETTER (with sleep 2 in the end) is much more efficient way to improve the performance.

Remember that our FORGETTER algorithm is not just a learning rate scheduler, but, it randomly initializes state variables after each sleep. The last drop demonstrates how the initialization after the sleep is effective. As the effect of last drop (sleep 2 in the end) is rather variable, the resulting validated loss for the FORGETTER fluctuates across layers (the red line in Figure 2).

Table 2 summarized the best normal and FORGETTER models for Baby10M, where we also computed the BLiMP Score. (We believe that BLiMP Score is almost in one-to-one correspondence to the validated loss for the next token prediction. This is because we have not observed the contradictory results before.) The BLiMP score is (inversely) related to the validated loss for the next token prediction in the current case. The absolute value of BLiMP Score is rather limited because the models were trained with only Baby10M dataset.

4 Results for Baby100M

We trained the minGemma models with variable structures with normal or FORGETTER training algorithm for Baby100M, specifically, to see if the FORGETTER is also effective for Baby100M. The results in Figure 4 demonstrate that the FORGETTER models are always much better than the normal models, again.

To see if the number of layers matters, we plotted the validated losses for our best models for a

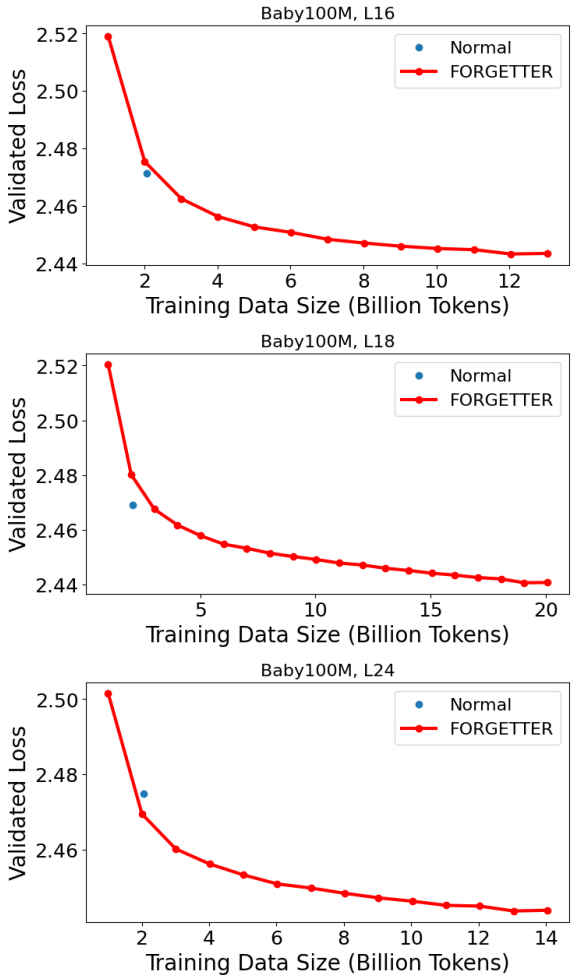

Figure 5: Three examples of training course of FORGETTER algorithm for Baby100M for models with 16 (top), 18 (middle), and 24 layers (bottom). Validated losses for FORGETTERs are plotted as time series in red. The validated losses for normal models are denoted by a blue point.

given number of layers in Figure 4. We observed a clear trough in the plot and the model with 18 layers is the best for both normal and FORGETTER training for Baby100M. (However, we should admit that our structure search may not be complete for Baby100M, whose computational time is rather long.)

Note that the difference between normal and FORGETTER models are much larger than that by model structures. This means that changing the learning curriculum to FORGETTER was the most efficient way to improve the performance.

Next, we looked at the time course during training (Figure 5). Again, the FORGETTER model excels the normal model within a couple of iterations as shown in Figure 5. The sleep interval for the FORGETTER, that is optimized to minimize

Hyperparameter	Normal	FORGETTER
N Layers	18	16
N Q Heads/Layer	8	9
N K/V Heads/Layer	4	3
Hidden Dimension	576	612
FFI Dimension	576×4	612×4
Head Dimension	256	224
Validated Loss	2.4691	2.4374
BLiMP Score	0.7669	0.7761

Table 3: Optimal model structures for Baby100M when input token length is 512.

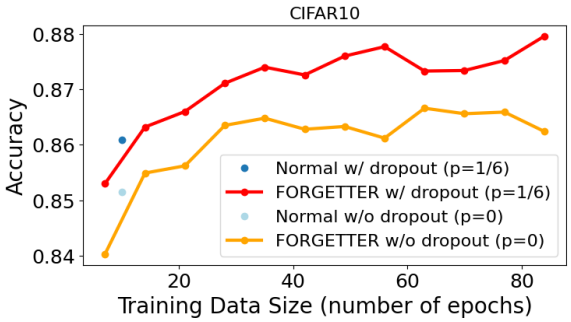

Figure 6: Accuracy of CIFAR10 classification for normal or FORGETTER algorithm with and without dropout. $p = 0$ represents the case without dropout.

the validated loss for next token prediction task, was about half of that of normal models. This is shown as the number of steps per epoch in Talbe 1. (In normal models, there is no sleep and the entire training consists of only a single interval or epoch.)

Therefore the computational time for the two epochs for FORGETTER models is roughly equivalent to that for normal models. At that time, their performances are almost equal. However, FORGETTER models can continue to learn beyond the training data size where normal models overfit as in Figure5. Note that the number of steps per epoch for the normal model (=144000) is already optimized, which means that if you used longer steps (more training data) per epoch, the model would overfit and its validated loss deteriorates.

The drop in the end of the training caused by sleep 2 (deep sleep) was also effective but mild for Baby100M (Figure 5).

Table 3 summarizes the best normal and FORGETTER models for Baby100M, where we also computed the BLiMP Score. Note that the validated losses for Baby100M is much smaller than that of Baby10M. The BLiMP score is in one-to-one correspondence to the validated loss for the next token prediction in the current case.

The absolute value of our BLiMP Score is rather mild (cf. 47.7, 46.2, 78.2 and 79.1 for GPT2 Small, Medium, Large and XL, respectively.) This is partly because the input token length was limited to 512 entirely in this paper. Having shorter input token lengths is as if setting another task. It can impose the upper limit for performances. Although we believe that the comparison between normal and FORGETTER models gave a general result, trying longer input token lengths toward contest quality will be needed in the future work.

5 Results for CIFAR10

We examined if FORGETTER is also effective for CIFAR10 image classification by CNNs. The result in Figure 6 demonstrates that the FORGETTER with a CNN used in a tutorial (Sayah, 2022) can again continue to learn beyond the normal overfitting limit with or without dropout.

The optimal training data size for the normal training with linear learning rate decay was 10 epochs (=10 repeats of the entire training dataset) and longer training caused overfitting. Meanwhile, the optimal inter-sleep interval for the FORGETTER was 7 epochs. That is, the learning rate linearly decayed within every 7 epochs there (the learning rate is reset after 7 epochs). Thus, it is natural that the accuracy of the FORGETTER after one sleep or at 7 epochs is worse than that of the normal model, that learns for 10 epochs. But it exceeds after two sleeps or at 14 epochs. Then, it continues to avoid overfitting for a long time.

These observations are common with or without dropout. Although dropouts might not be strictly needed for pretraining, the impact of dropout is interesting in the sense it can somehow have a similar effect as forgetting. Forgetful hyperparameters and sleeps that initialize optimizers might enhance the redundancy and robustness of representations, which can be a similar role as dropout.

In fact, the dropout ($p = 1/6$) is not only effective but also synergistic with the FORGETTER curriculum learning, suggesting that the mechanism of FORGETTER may be independent from that of dropout. Specifically, the FORGETTER with dropout ($p = 1/6$) attained 88% (87.96%) even without data augmentation. Note that the same code with normal training attains 88% only with data augmentation (Sayah, 2022).

6 Summary and Discussion

We extensively explored the optimal model structure for Baby10M and found "120-rule" where optimal models always almost have 120 (Query) heads. This suggests that there is a specific number (=120) of information processing the model has to treat. And the optimal model tends to have this number of attention heads. We compared normal and FORGETTER models for Baby10M and found that FORGETTER models performed much better. This tendency also holds for Baby100M, in which the best performance was much better. The FORGETTER also worked for CIFAR10 image classification. Overall, FORGETTER models can continue to learn beyond the normal overfitting limits. These results suggest that forgetting can be beneficial for pretraining deep neural networks by avoiding overfitting.

The FORGETTER training can bring about Copernican Revolution on overfitting. It is beneficial if you can control to train beyond the normal overfitting limits.

Regular sleep intervals (number of steps per epoch) apparently worked, once interval lengths were carefully optimized as in Table 1. We could not get a significantly better result by using linearly increasing/decreasing sleep intervals. Also we could not get a significantly better result by using linear increasing/decreasing initial learning rates across epochs.

120 rule saves your computational cost for model structural search. We already found the similar rule for WikiText-103 dataset (not shown), although the magic number (120 for Baby10M) seems different depending on datasets. If this rule holds for general datasets, when you search for the best model for a new dataset, probably you can start finding the magic number of the rule for that dataset first. Once the prospected total number of heads can be estimated first, then, you can save your exploration cost quite a lot. Although "120-rule" by itself cannot select a unique best model, having another rule as well like "about 700 hidden-dim needed to represent a token in FFNs" could uniquely determine. Typically, language models have not only attention structures but also feed forward networks. The balance between the (input) dimensions of attention structures and FFNs may be the key for the best performance like two wheels.

Limitations

Generalizability is unclear. So far, other than Baby10M, Baby100M and CIFAR10, we have observed the significant benefits of the FORGETTER training algorithms only for WikiText-103 and WikiText-2 as training datasets (not shown). It is highly important to examine how general the benefit is by trying different (possibly large) training datasets.

Only GPT Tokenizer was used. We are not sure if there is a better one. Although we almost did not explore alternative tokenizers, we hope that the comparison results, such as normal versus FORGETTER models, are general to some extent. Also, it is not clear if an existing tokenizer like the GPT 2 Tokenizer is biologically plausible. Maybe tokenizer should also be learnt from BabyLM with limited vocabrary. We need further study.

We entirely used 512 as a input token length throughout the paper. Although this length is shorter than that of GPT2 (=1024), we observed that the effect on the performance is mild, compared with shorter input lengths such as 256 or 128. However, scalability to long text should be checked with large GPU resources.

Transfer learning (instruction learning) is nowadays important for LLMs. Then the effect of the FORGETTER, that was used for pretraining, on fine-tuning is interesting. Our evaluation was by next token prediction throughout this paper. Therefore it is interesting how the FORGETTER learning beyond the normal overfitting limits can affect the following instructive learning. Relatedly, distillation is nowadays important for small language models. The combination of FORGETTER with distillation is interesting but to be done.

Although we repeated training beyond the normal overfitting limit as a curriculum learning, we just repeated the same type of learning homogenously. It is possible a model is good at some topic but not in another. By sampling training data from the topics the model is not good at, you could accelerate the training (Müller et al., 2025).

Ethics Statement

This work complies with the ACL Ethics Policy.

Acknowledgements

KM is partially supported by JSPS KAKENHI Grant Number JP25K15283.

References

Rowel Atienza. 2020. *Advanced Deep Learning with TensorFlow 2 and Keras: Apply DL, GANs, VAEs, deep RL, unsupervised learning, object detection and segmentation, and more*. Packt.

Francois Chollet. 2021. *Deep Learning with Python 2nd Edition*. Manning.

Martin Ford. 2018. *Architects of Intelligence: The truth about AI from the people building it*. Packt Publishing.

Google DeepMind Gemma Team. 2024. Gemma 2: Improving open language models at a practical size.

Aurlien Geron. 2022. *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. The MIT Press.

M Konishi, KM Igarashi, and K Miura. 2023. Biologically plausible local synaptic learning rules robustly implement deep supervised learning. *Front Neurosci.*, 17:1160899.

TP Lillicrap, A Santoro, L Marris, CJ Akerman, and G Hinton. 2020. Backpropagation and the brain. *Nat Rev Neurosci.*, 21(6):335–346.

Zhenyan Lu, Xiang Li, Dongqi Cai, Rongjie Yi, Fangming Liu, Xiwen Zhang, Nicholas D. Lane, and Mengwei Xu. 2025. Small language models: Survey, measurements, and insights.

Kyle Mahowald, Anna A. Ivanova, Idan A. Blank, Nancy Kanwisher, Joshua B. Tenenbaum, and Evelina Fedorenko. 2024. Dissociating language and thought in large language models. *Trends in Cognitive Sciences*, 28(6):517–540.

Reuven Müller, Ying Xie, Linh Le, and Shaoen Wu. 2025. Dynamic knowledge elicitation: Leveraging student feedback for improved language model distillation. *Proceedings of International Joint Conference on Neural Networks 2025 (IJCNN2025)*.

H Norimoto, K Makino, M Gao, Y Shikano, K Okamoto, T Ishikawa, T Sasaki, H Hioki, S Fujisawa, and Y Ikegaya. 2018. Hippocampal ripples down-regulate synapses. *Science*, 359(6383):1524–1527.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

Sebastian Raschka. 2024. *Build a Large Language Model (From Scratch)*. Manning.

Denis Rothman. 2024. *Transformers for Natural Language Processing and Computer Vision - Third Edition: Explore Generative AI and Large Language Models with Hugging Face, ChatGPT, GPT-4V, and DALL-E 3*. Packt.

Fares Sayah. 2022. Cifar-10 images classification using cnns (88%).

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.

Evin Tunador. mingemma (github).

Lewis Tunstall, Leandro Von Werra, and Thomas Wolf. 2022. *Natural Language Processing with Transformers: Building Language Applications with Hugging Face*. O'Reilly.

Alex Warstadt, Aaron Mueller, Leshem Choshen, Ethan Wilcox, Chengxu Zhuang, Juan Ciro, Rafael Mosquera, Bhargavi Paranjabe, Adina Williams, Tal Linzen, and Ryan Cotterell. 2023. Findings of the BabyLM challenge: Sample-efficient pretraining on developmentally plausible corpora. In *Proceedings of the BabyLM Challenge at the 27th Conference on Computational Natural Language Learning*, pages 1–34, Singapore. Association for Computational Linguistics.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. 2024. Galore: memory-efficient llm training by gradient low-rank projection. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.