Simplifying Adam: Bias Correction Debunked

Sam Laing
ELLIS Institute Tübingen, MPI-IS
Tübingen AI Center, Germany
slaing155@gmail.com

Antonio Orvieto
ELLIS Institute Tübingen, MPI-IS
Tübingen AI Center, Germany

Abstract

The Adam optimizer is a cornerstone of modern deep learning, yet the empirical necessity of each of its individual components is often taken for granted. This paper presents a focused investigation into the role of bias-correction, a feature whose contribution remains poorly understood. Through a series of systematic ablations on vision and language modelling tasks, we demonstrate that the conventional wisdom surrounding bias correction is misleading. In particular, we demonstrate that in the optimal hyper-parameter configuration, the inclusion of bias correction leads to no improvement in final test performance. Moreover, unless appropriate learning rate scheduling is implemented, the inclusion of bias correction can sometimes be detrimental to performance. We further reinterpret bias correction as a form of implicit learning rate scheduling whose behaviour is strongly dependent on the choice of smoothing hyperparameters $\beta_1, \beta_2 \in [0, 1)$. Our findings challenge the universal inclusion of this component.

1 Introduction

The Adam optimizer Kingma and Ba [2017] (with decoupled weight decay Loshchilov and Hutter [2019]) has established itself as the de facto standard in deep learning. Due to its robust empirical performance and ease of implementation, it is commonly adopted as the default choice when training neural networks both large and small and across a wide range of tasks. Practitioners often use Adam without extensive hyper-parameter tuning or consideration of each of the individual components of the optimizer. One such component is *bias correction*.

A single step of the Adam optimizer, with bias correction, is given by

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &= \frac{1}{1-\beta_1^t} m_t \ , \ \hat{v}_t = \frac{1}{1-\beta_2^t} v_t \\ \theta_t &= \theta_{t-1} - \eta_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \end{aligned} \tag{1}$$

where g_t is the stochastic gradient, m_t, v_t are the respective first and second moments (which are typically initialized at 0), $(\beta_1, \beta_2) \in [0, 1)^2$ are the exponential decay rates, $\lambda \in \mathbb{R}_{\geq 0}$ weight decay and $(\eta_j)_{j \geq 1} \subset \mathbb{R}_{\geq 0}$ is the learning rate schedule.

Background and Related Work While bias correction is universally included in practical implementations of Adam(Bradbury et al. [2018], Paszke et al. [2019], Abadi et al. [2015]), it is inconsistently treated in the theoretical literature. In many analyses, the term is explicitly incorporated (Balles and Hennig [2020], Li et al. [2023]), but is sometimes ignored for simplification

EurIPS 2025 Workshop on Principles of Generative Modeling (PriGM).

(Bernstein and Newhouse [2024]) . To the best of our knowledge, it has not been carefully ablated in empirical studies across a range of hyperparameter configurations and tasks. Nevertheless, it plays a non-trivial role in shaping the optimizer's behaviour and, when included, tends to complicate convergence analyses and add interpretive nuance. For example, Défossez et al. [2022] discuss its influence in the context of second-moment estimation. Explicitly, the authors omit the correction term for the first moment m_t, but not v_t arguing that this "simplifies the analyses". Moreover, nearly all existing analyses assume near default settings of $\beta_1 = 0.9, \beta_2 = 0.999$ or 0.99, leaving unexplored how the effects of bias correction might change across the (β_1, β_2)-landscape. This gap in the literature is particularly relevant in light of recent findings, such as Orvieto and Gower [2025], which demonstrate that the setting $\beta_1 = \beta_2$ generally achieves optimal performance when pretraining large language models.

A Discussion of the "Proof" for Bias Correction One can express the moments m_t, v_t at time step $t \in \mathbb{N}$ in closed-form as follows:

$$m_t = (1-\beta_1)\sum_{j=1}^{t} \beta_1^{t-j} g_j \quad , \quad v_t = (1-\beta_2)\sum_{j=1}^{t} \beta_2^{t-j} g_j^2 \tag{2}$$

The following argument is used by the original authors to justify the inclusion of the bias correction step

$$\begin{aligned}
\mathbb{E}[m_t] &= \mathbb{E}\left[(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} g_i\right] \\
&= \mathbb{E}\left[(1-\beta_1) g_t \sum_{i=1}^{t} \beta_1^{t-i}\right] \quad \text{(assuming } \mathbb{E}[g_t] \approx E[g_i] \text{ for } i < t\text{)} \\
&= (1-\beta_1^t)\mathbb{E}[g_t]
\end{aligned}$$

It is therefore argued that dividing by $1-\beta_1^t$ removes the expected "bias" in the exponential moving average. An analogous argument is used to justify the bias correction factor for v_t.
While the assumption that $\mathbb{E}[g_t] \approx \mathbb{E}[g_i]$ for $i < t$ at early training steps simplifies the analysis, it usually does not hold in practice. In particular, unless a very gradual warm-up of learning rate is applied, it is unlikely that stepping somewhere in a complex loss landscape would not cause a significant change of the expected value of the gradient.

Contributions In this work, we challenge the conventional understanding of bias correction. Through controlled ablations in the language and vision settings, we show that:

- The inclusion of bias correction induces an *implicit learning rate schedule* by altering the effective learning rate.
- For the $\beta_1 = \beta_2$ setting (LLM-optimal), bias correction provides no benefit and can even degrade performance unless appropriate learning rate scheduling is implemented.
- For default parameters where performance is suboptimal, its removal is detrimental, explaining the source of conventional wisdom.

2 Experiments

Language Model Training Details In the case of pretraining language models, we focus exclusively on transformer-based models. Specifically, we make use of an enhancement (Ajroldi [2024]) of nanoGPT (Karpathy [2022]), which incorporates improvements such as RMSNorm Zhang and Sennrich [2019] in place of batch/layer normalization, SwiGLU Shazeer [2020], a FlashAttention Dao et al. [2022] mechanism and Rotary Positional Embeddings (Su et al. [2023]).
All language models were trained on the SlimPajama Shen et al. [2024] dataset. We pretrain a 160M parameter, 12 layer model with hidden size 768 on 2.5B tokens, varying the learning rate and investigating a number of different (β_1, β_2) settings. In all runs, we apply decoupled weight decayLoshchilov and Hutter [2017] and global gradient clipping Zhang et al. [2020] for optimal

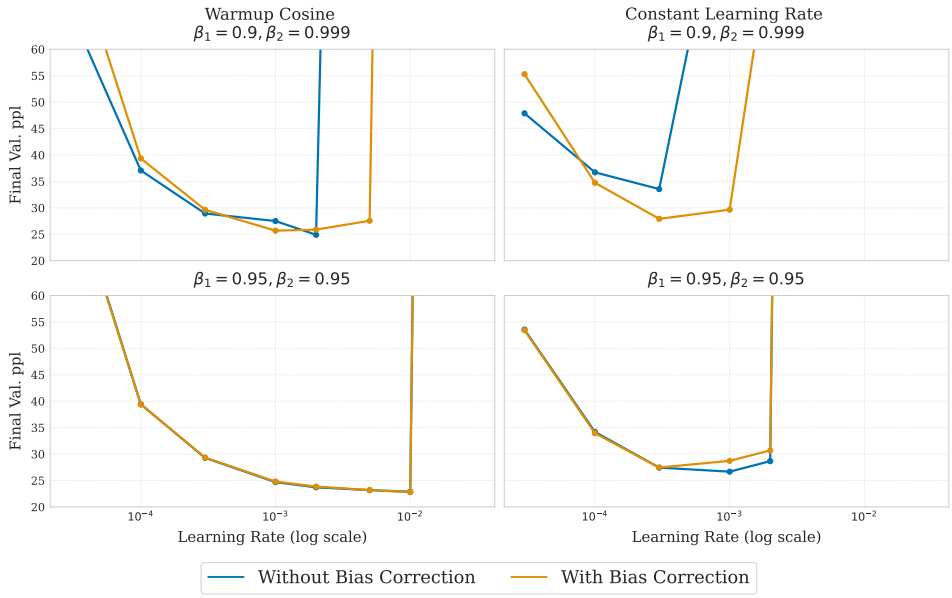

Figure 1: Sensitivity to learning rate for AdamW with and without bias correction (orange and blue respectively). The plots show final validation perplexity (y-axis) across a range of learning rates (x-axis, log-scale). Results are averaged over 3 random seeds.
With warm-up cosine scheduling, removing bias correction increases sensitivity for default hyperparameters $(\beta_1, \beta_2) = (0.9, 0.999)$ but with identical optimal performance. For the LM-optimal setting $(\beta_1, \beta_2) = (0.95, 0.95)$, performance is identical.
With a fixed learning rate, the inclusion of bias correction has a more pronounced effect. In the default torch setting $(0.9, 0.999)$, excluding bias correction has a detrimental effect whereas for the LM-optimal setting $(0.95, 0.95)$, bias correction slightly degrades optimal performance.

performance[1] . The batch size is held fixed at 256.
We further compare the performance when a warm-up cosine schedule (linear warm-up for the first 10% of steps followed by cosine decay to zero) is applied vs a fixed learning rate throughout training. We compare the performance when training in the torch default setting $(\beta_1, \beta_2) = (0.9, 0.999)$ and the more language model pretraining-optimal setting $\beta_1 = \beta_2 = 0.95$ which was shown in Orvieto and Gower [2025] to yield near best performance across a large sweep. Figure 1 displays the final validation perplexity across a sweep of learning rates for the two settings. In Appendix B, we extend our investigation of the $\beta_1 = \beta_2$ case, systematically evaluating different values of this shared parameter (see Figure 6 and an explanation in Figure 7).

Vision Models To support our intuition, we perform further experiments in the vision setting. Details, experimental results and discussion are all provided in the appendix A. In particular, we display our empirical findings in Figures 3, 5, 4.

3 Bias Correction as Implicit Learning Rate Scheduling

The bias-corrected Adam step direction (ignoring ε) factorizes as:

$$\frac{\hat{m}_t}{\sqrt{\hat{v}_t}} = \frac{\frac{1}{1-\beta_1^t} m_t}{\sqrt{\frac{1}{1-\beta_2^t} v_t}} = \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t} \frac{m_t}{\sqrt{v_t}} = \rho(t; \beta_1, \beta_2) \frac{m_t}{\sqrt{v_t}} \tag{3}$$

Where we define the *bias-correction factor* $\rho(t; \beta_1, \beta_2) := \frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}$. This term modulates the *effective learning rate* $\rho(t; \beta_1, \beta_2) \cdot \eta_t$ over time in a manner which depends heavily on the values of β_1, β_2.

[1]The dynamics without decoupled weight decay/gradient clipping are nearly identical but with generally worse performance throughout runs.

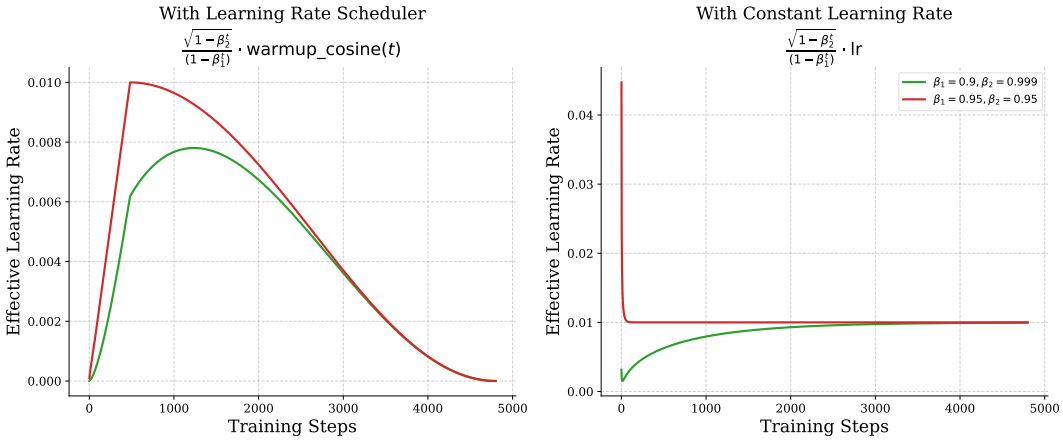

Figure 2: Comparison of the effective learning rate when bias correction is applied for $(\beta_1, \beta_2) = (0.9, 0.999)$ (green) and $(\beta_1, \beta_2) = (0.95, 0.95)$ (red) under both warm-up cosine scheduling (left) and a constant learning rate (right).
With warm-up cosine scheduling, the bias correction factor is effectively absorbed for the LM-optimal setting $(0.95, 0.95)$ (the true warmup cosine schedule is indistinguishable from the red curve), whereas for the default setting $(0.9, 0.999)$ it substantially modifies the effective learning rate, lowering the peak value. Without scheduling, the torch default configuration exhibits a very gradual warm-up on effect on the effective learning rate, while the LM-optimal setting produces an initial spike that quickly decays to the nominal learning rate.

While prior work John [2021] Défossez et al. [2022] has recognized this interpretation of the bias-correction factor, its empirical behaviour has only been considered for a limited subset of (β_1, β_2) Kingma and Ba [2017] – typically those near the default settings ($\beta_1 = 0.9, \beta_2 = 0.999$).
Figure 2 reveals how $\rho(t; \beta_1, \beta_2)$ behaves differently across configurations:

- **Default setting** $(0.9, 0.999)$: Creates a very gradual warm-up effect, slowly increasing the effective learning rate across thousands of iterations.
- **LM-optimal setting** $(0.95, 0.95)$: Produces a large initial spike that quickly decays to baseline

The interaction with explicit scheduling is crucial. Warm-up cosine scheduling completely absorbs the $\rho(t)$ spike for $(0.95, 0.95)$, but the bias correction factor non-negligibly alters the effective learning rate for $(0.9, 0.999)$. This results in a dampened peak learning rate.
This explains our empirical results in Figure 1:

- With scheduling, performance differences vanish for $\beta_1 = \beta_2$ as the spike is absorbed
- Without scheduling, the $(0.95, 0.95)$ spike causes instability in the early training steps, degrading performance
- Default parameters benefit from bias correction's implicit warm-up when no scheduling is used.

4 Conclusion

We have demonstrated a clear and actionable finding: when pretraining language models with a proper learning rate schedule and optimal hyperparameters, Adam achieves the same validation performance with or without the inclusion of bias correction. Bias correction is not a true performance enhancer; but merely an implicit, and often clumsy, learning rate warm-up. Since explicit learning rate scheduling is always required to achieve optimal results, we therefore advocate for its removal from both implementation and theoretical analysis. This ultimately yields a simpler, more interpretable optimizer without sacrificing performance. It also removes a potential confounder in convergence studies and large scale training.

Acknowledgements

The authors acknowledge the financial support of the Hector Foundation, and are thankful for the compute resources made available by MPI-IS and the Tübingen AI ecosystem. We additionally thank Philipp Hennig for the comments and remarks on our evaluations

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

Niccolò Ajroldi. plainlm: Language model pretraining in pytorch. `https://github.com/Niccolo-Ajroldi/plainLM`, 2024.

Lukas Balles and Philipp Hennig. Dissecting adam: The sign, magnitude and variance of stochastic gradients, 2020. URL `https://arxiv.org/abs/1705.07774`.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology, 2024. URL `https://arxiv.org/abs/2409.20325`.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/jax-ml/jax`.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022. URL `https://arxiv.org/abs/2205.14135`.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL `https://arxiv.org/abs/2010.11929`.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of adam and adagrad, 2022. URL `https://arxiv.org/abs/2003.02395`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

John St John. Adamd: Improved bias-correction in adam, 2021. URL `https://arxiv.org/abs/2110.10828`.

Andrej Karpathy. NanoGPT. `https://github.com/karpathy/nanoGPT`, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL `https://arxiv.org/abs/1412.6980`.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Haochuan Li, Alexander Rakhlin, and Ali Jadbabaie. Convergence of adam under relaxed assumptions, 2023. URL `https://arxiv.org/abs/2304.13972`.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017. URL https://arxiv.org/abs/1608.03983.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Antonio Orvieto and Robert Gower. In search of adam's secret sauce, 2025. URL https://arxiv.org/abs/2505.21829.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. URL https://arxiv.org/abs/1912.01703.

Noam Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020. URL https://arxiv.org/abs/2002.05202.

Zhiqiang Shen, Tianhua Tao, Liqun Ma, Willie Neiswanger, Zhengzhong Liu, Hongyi Wang, Bowen Tan, Joel Hestness, Natalia Vassilieva, Daria Soboleva, and Eric Xing. Slimpajama-dc: Understanding data combinations for llm training, 2024. URL https://arxiv.org/abs/2309.10818.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Biao Zhang and Rico Sennrich. Root mean square layer normalization, 2019. URL https://arxiv.org/abs/1910.07467.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J Reddi, Sanjiv Kumar, and Suvrit Sra. Why are adaptive methods good for attention models?, 2020. URL https://arxiv.org/abs/1912.03194.

A Additional Experiments in the Vision Setting

Vision Training Setup In order to support our claims, we investigate the effect of removing bias correction in the vision setting. We train a ResNet9[2] model on CIFAR-10 Krizhevsky and Hinton [2009] and both a vision transformer (ViT) Dosovitskiy et al. [2021] and ResNet He et al. [2015] model on TinyImagenet Hendrycks and Dietterich [2019]. Standard data augmentation was applied in all cases along with decoupled weight decay Loshchilov and Hutter [2017]. We compare the effect in the torch default setting $(\beta_1, \beta_2) = (0.9, 0.999)$ and the $\beta_1 = \beta_2 = 0.95$ setting. Again we consider both the case of warm-up cosine scheduling and constant learning rate. For each learning rate, we compute the average test accuracy from 3 random seeds.

Experimental Results Across the three model-dataset settings (Figure 3, Figure 5, 4, we observe no qualitative difference in test performance when removing bias correction. This is true for both the warm-up cosine and constant learning rate cases and for both (β_1, β_2) settings.

Takeaway These results further support our claims that bias correction can be safely removed from Adam without decreased performance, provided a solid training pipeline is adopted.

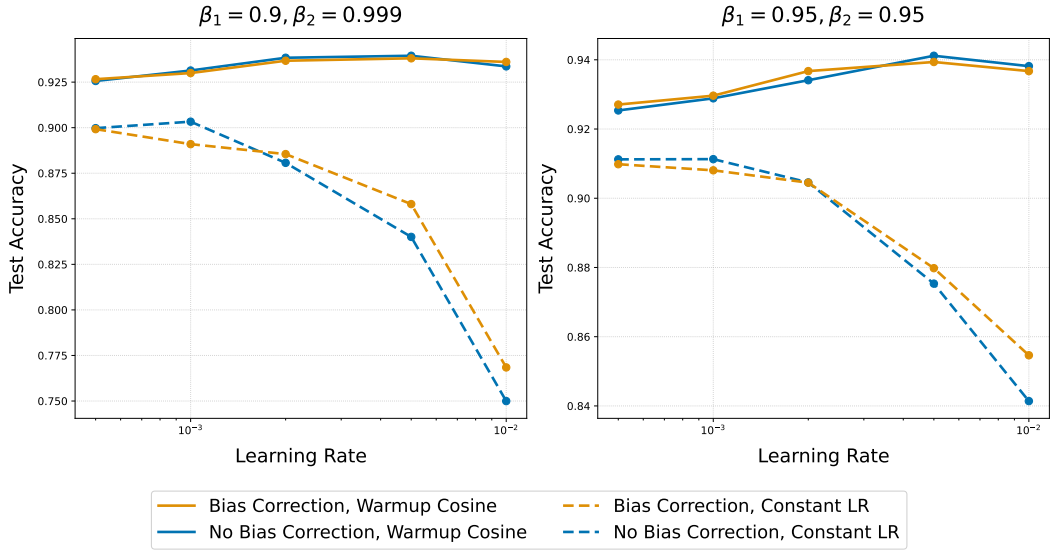

Figure 3: ResNet9 on CIFAR-10.

[2]A lightweight ResNet implementation He et al. [2015] with $\sim$ 1M parameters.

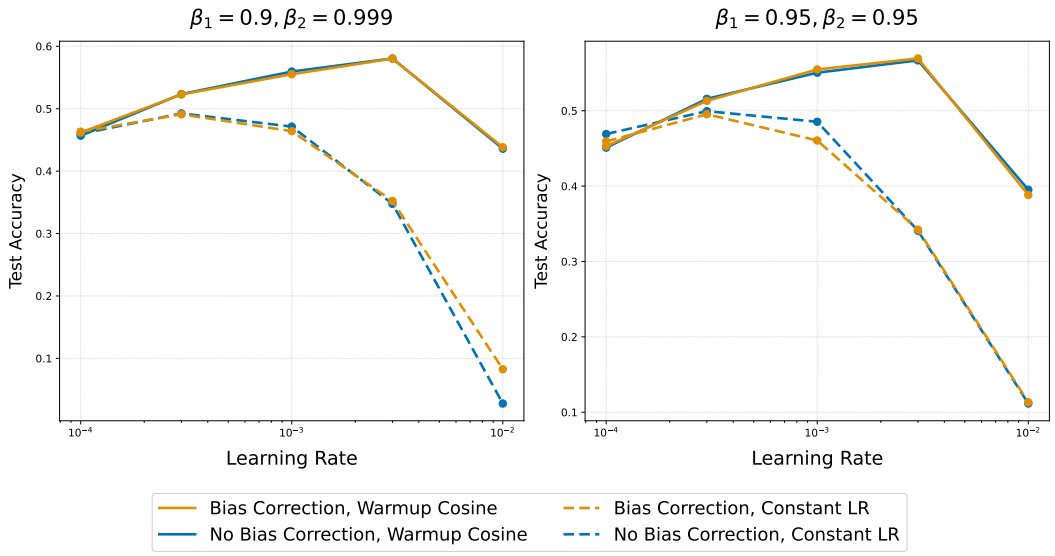

Figure 4: ResNet50 on Tiny ImageNet.

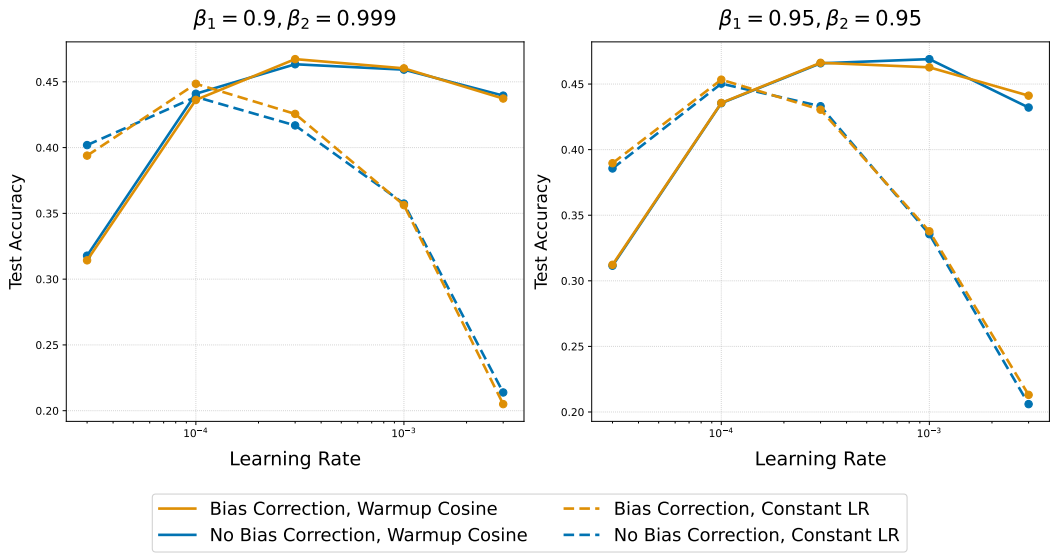

Figure 5: ViT on Tiny ImageNet

B A Closer Look at the $\beta_1 = \beta_2$ Setting

We further investigate the final validation performance when considering different values of $\beta = \beta_1 = \beta_2$. We conduct a sweep across values $\beta \in \{0.9, 0.95, 0.975, 0.9875\}$ with the other training parameters the same as before. Due to computational constraints, each point represents a single random seed.

Consider Figure 6.When warm-up cosine scheduling is applied, final validation performance is indistinguishable for every learning rate and every choice of β. On the other hand, when the learning rate is constant, the inclusion of bias correction always worsens optimal performance. Moreover increasing values of β result in an increasing discrepancy between performance.

We posit that the performance discrepancy is explained by the behaviour of the bias correction factor as shown in Figure 7. Indeed for higher values of β, the bias correction factor decays more slowly towards the baseline. This spike in effective learning rate in the early steps likely has a destabilizing effect, resulting in a worse trained model.

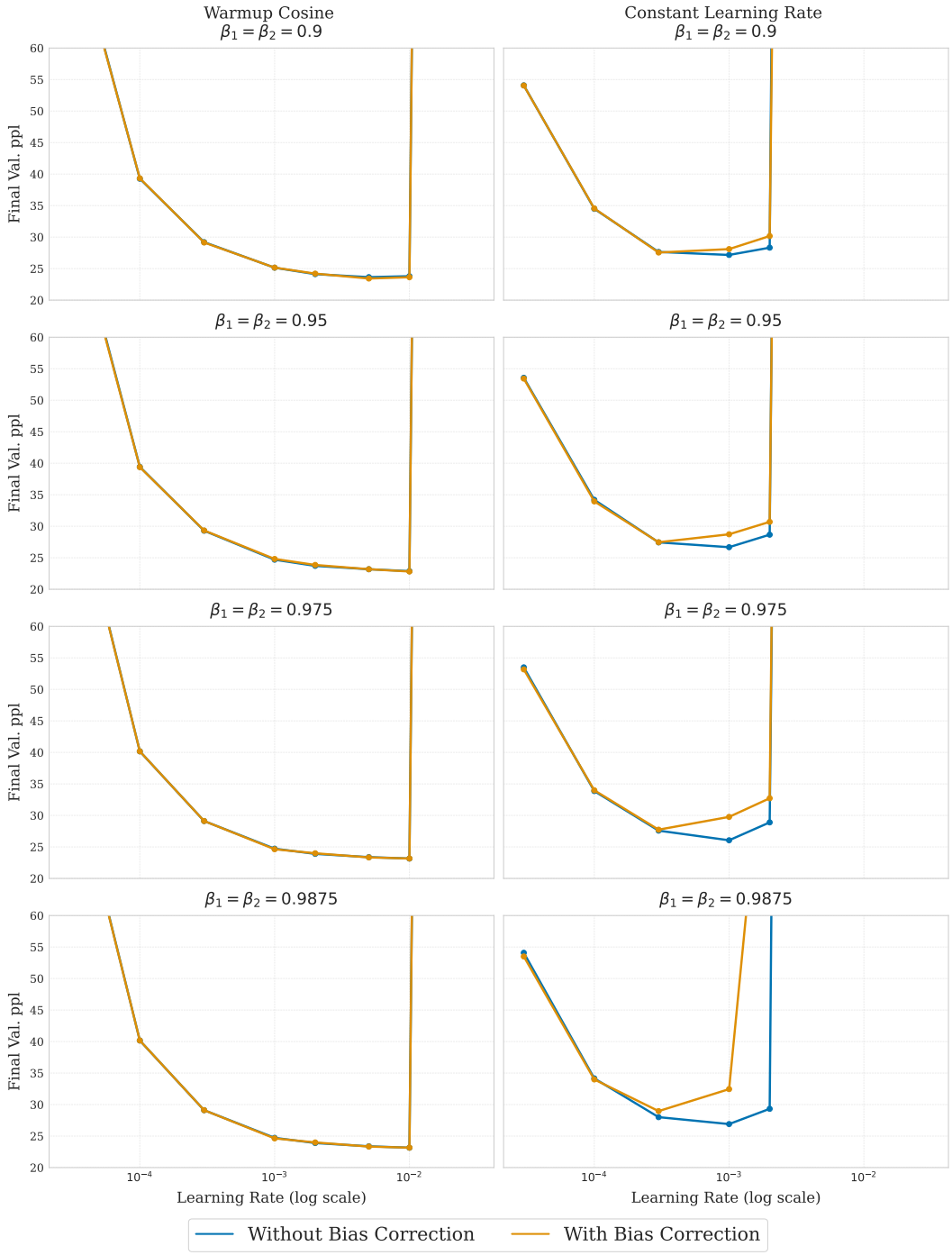

Figure 6: Sensitivity curves comparing performance with and without bias correction (orange and blue respectively) for the case that $\beta_1 = \beta_2$. The x-axis represents the learning rate (log scale) and the y-axis the final validation perplexity.
With warm-up cosine scheduling, the inclusion of bias correction does not effect final validation perplexity across all learning rates.
In contrast, with constant learning rate, bias correction actually denigrates performance, progressively more for larger values of $\beta_1 = \beta_2$ – with bias correction always performing worse.

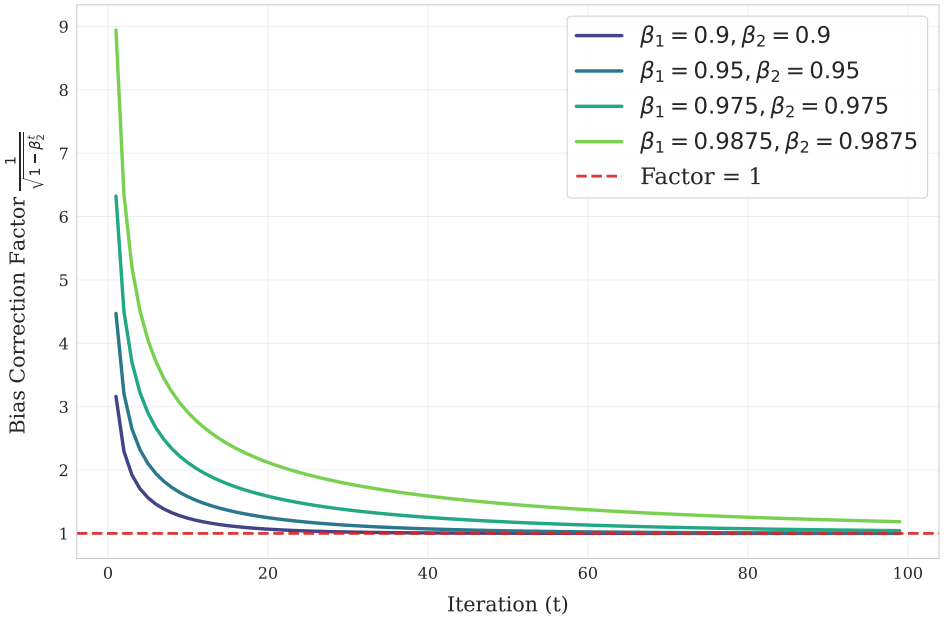

Figure 7: Plot of the *bias correction* factor the first 100 steps for $\beta = \beta_1 = \beta_2$
As we can see, the behaviour for different β values is similar, but larger values require more iterations to decay to 1.