Tuning Adam(W): Default β_2 May Be Too Large

Matias D. Cattaneo[*]
Princeton University
cattaneo@princeton.edu

Boris Shigida[*]
Princeton University
bs1624@princeton.edu

Abstract

The default momentum hyperparameters of Adam(W), with or without decoupled weight decay, are usually set at $\beta_1 = 0.9$ and $\beta_2 = 0.999$. We provide an empirical evaluation of these hyperparameter choices on several language and vision tasks, focusing on how β_2 influences the training dynamics and performance outcomes. We observe that β_2 so close to one leads to unstable training because of loss spikes, and provide evidence that taking $\beta_1 \approx \beta_2$ mitigates this problem in modern and practical settings. Moreover, for medium and large batch sizes, we find that a much smaller β_2 is often better for generalization: our findings suggest a choice of β_2 in the range $[0.9, 0.95]$. Contrary to common practice, we observe instances where it is better to take a large β_1 and $\beta_2 \ll \beta_1$. We also find that Adam(W) with smaller β_2 can train faster (if the batch size is large). Our results suggest that ample justification is needed for setting a large β_2. In particular, for fair comparisons of Adam(W) with other optimizers, β_2 should either be carefully tuned or at least taken much smaller than $\beta_2 = 0.999$.

1 Introduction

Modifications of Adam (Kingma and Ba, 2014) such as AdamW (Loshchilov and Hutter, 2019) and AdaFactor (Shazeer and Stern, 2018) have become the standard optimizers for important deep learning tasks like training large language models (Brown et al., 2020; Anil, A. M. Dai, et al., 2023; Touvron et al., 2023; Dubey et al., 2024). In addition to the learning rate, Adam has three hyperparameters: two momentum hyperparameters (β_1, β_2) (usually called the "betas"), and a numerical stability hyperparameter ϵ. The choice of their values is a crucial part of preparing the (pre-)training pipeline, and practitioners often employ commonly accepted heuristics when setting these hyperparameters.

Kingma and Ba (2014) recommend setting $\beta_1 = 0.9$ and $\beta_2 = 0.999$. They report a grid search of $\beta_1 \in \{0, 0.9\}$ and $\beta_2 \in \{0.99, 0.999, 0.9999\}$ on a variational autoencoder training problem, comparing training speeds after 10 and 100 epochs. Based on this recommendation, the values $(\beta_1, \beta_2) = (0.9, 0.999)$ have become the default in many libraries such as PyTorch and Optax. It is conventional wisdom that adaptive gradient methods work well with their default hyperparameters (Sivaprasad et al., 2020). This practical success appears to be the reason why practitioners rely on default values (with often half-hearted tuning).

Apart from the practical usage of Adam-based algorithms when training new models, researchers often use default hyperparameter values when comparing Adam with other optimizers experimentally. In fact, when a claim is made about the relative performance of a proposed algorithm with respect to Adam, "Adam" usually implicitly means "Adam with default hyperparameters", and often minimal or no tuning takes place for the comparison. As an illustration, it is common to write "SGD with momentum outperforms Adam" regarding a subset of deep learning tasks, whereas SGD with momentum is essentially a special case of Adam (by taking large ϵ), and therefore Adam understood generally cannot possibly underperform relative to SGD with momentum (Choi, 2019; Savarese et al., 2021). Of course, what is meant is "SGD with momentum outperforms Adam with default (or minimally modified) hyperparameters", but such usage hides potential important insights about the possible utility of non-default hyperparameter values.

The lack of tuning, or minimal tuning, of competing algorithms is common practice: X. Chen et al. (2023) largely take default hyperparameters when comparing AdamW to Lion, Anil, Gupta, et al. (2020)

[*]Equal contribution

compare second-order optimizers to Adam and do not tune the betas for larger transformers, Bernstein et al. (2018) do not tune β_2 when comparing Adam with signSGD, Dozat (2016) use default settings when comparing different optimizers, Gupta, Koren, and Singer (2018) did not report hyperparameter tuning when comparing Shampoo to Adam, J. Ma and Yarats (2018) did not tune β_2 when comparing their quasi-hyperbolic momentum algorithm with Adam, L. Liu et al. (2019) take the defaults when comparing Rectified Adam with Adam, Reddi, Kale, and Kumar (2018) do not tune β_2 when comparing AMSGrad to Adam (with only large values $\beta_2 \in \{0.99, 0.999\}$ considered), Taniguchi et al. (2024) do not tune the hyperparameters of Adam when comparing ADOPT to Adam, Zhuang et al. (2020) do not tune β_2 when comparing AdaBelief with Adam (although they perform a large search tuning other hyperparameters), Shazeer and Stern (2018) do not tune β_2 when comparing Adafactor with Adam (with only two values $\beta_2 \in \{0.9, 0.999\}$), to name a few.

If a certain problem of Adam is identified (such as visible training instability), often custom interventions are used, but a simple approach of re-tuning the hyperparameters is either overlooked or not reported. For example, in Chowdhery et al. (2023, Section 5.1, "Training Instability"), the loss spikes are mitigated by restarting the training from an earlier checkpoint and skipping the fraction of data that was given to the model when the spike occured. It is possible that just tuning the hyperparameters and, in particular, the β_2 schedule can also serve as a mitigation strategy. Relatedly, Taniguchi et al. (2024) report "Adam causes loss spikes in the early stages of training and fails to converge, while ADOPT is always able to train stably" when training GPT-2 on OpenWeb with small batch sizes; this may be a problem of tuning rather than Adam per se (we find that Adam with $\beta_1 = \beta_2 = 0.9$ is also always able to train stably).

Recently, β_2 has been adjusted to be smaller (specifically $\beta_2 = 0.95$) when training large models with AdamW (Brown et al., 2020; S. Zhang et al., 2022; Zeng et al., 2022; Biderman et al., 2023; Touvron et al., 2023; Dubey et al., 2024) but the reason for this choice is not reported, and we are not aware of any comprehensive studies (released publicly) on which this decision may be based.

In addition, when comparing optimizers, different performance metrics may be of importance, such as training stability, training speed and generalization properties. An algorithm can theoretically train faster but achieve worse performance on the test set; or it can train faster, achieve better test performance but have a high risk of irrecoverable loss spikes. It is a highly under-explored question how setting smaller β_2 impacts different training outcomes.

This paper investigates experimentally how β_2 affects test accuracy, training speed, and the presence of loss spikes on a number of language and vision tasks. Our findings are summarized as follows:

- For Adam with and without weight decay, there is a high risk of loss spikes, including irrecoverable ones, when setting β_2 as close to 1. Decreasing β_2 even more seriously than in recent large training runs (e.g., setting $\beta_1 \approx \beta_2$) eliminates loss spikes. Our experimental results confirm the findings by C. Ma, Wu, and Weinan (2022) for simple multi-layers perceptrons on FashionMNIST and Resnet-18 on CIFAR-10 trained by full-batch Adam.
- How the test performance (best achieved before overfitting, or best after a fixed number of iterations) depends on β_2 is different for different batch sizes. For small batch sizes, the higher β_2 the better (ignoring the risk of loss spikes which may lead to divergence). For moderate batch sizes, the dependence becomes inverse U-shaped with optimal β_2 between 0.9 and 0.99. Finally, for large batches, the smaller β_2 the better. The default $\beta_2 = 0.999$ can only be the best for very small batch sizes, provided the training survives the loss spikes that appear frequently in this case.
- A similar trend as for the test accuracy is seen for speed of training: for small batch sizes, the higher β_2 the faster; for moderate batch sizes, gradually increasing β_2 from 0.9 to 0.999 first increases then decreases the training speed (if $\beta_1 = 0.9$); for very large batches, the higher β_2 the slower training.
- There is no law forcing β_2 to be much larger than β_1. In fact, setting $\beta_2 = \beta_1$ is often better than $\beta_2 > \beta_1$; sometimes, the best test accuracy we find is achieved with β_2 much smaller than β_1. For example, for a CNN trained on CIFAR-10, setting $(\beta_1, \beta_2) = (0.99, 0.9)$ in full-batch Adam achieves better test accuracy than $\beta_1 = 0.99$ and any $\beta_2 > 0.9$, or than $\beta_1 = 0.9$ and any $\beta_2 \geq \beta_1$; for ResNet-50 on CIFAR-10, if full-batch Adam is used with $\beta_1 = 0.99$, setting $\beta_2 = 0.8$ is better for test accuracy than any higher β_2.

In short, setting $\beta_2 = 0.999$ is rarely optimal in terms of test accuracy, leads to at least occasional loss spikes, and for large batches can lead to slower training than if β_2 is taken smaller. Our experimental results suggest $(\beta_1, \beta_2) = (0.9, 0.95)$ is a much more reasonable default choice than $(\beta_1, \beta_2) = (0.9, 0.999)$, not only for large language models but for modern architectures in general.

Notation. Adam (Kingma and Ba, 2014) with decoupled weight decay (Loshchilov and Hutter, 2019) is an algorithm with the update rule

$$\begin{aligned} \boldsymbol{m}_{t+1} &= \beta_1 \boldsymbol{m}_t + (1-\beta_1)\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t), \\ \boldsymbol{v}_{t+1} &= \beta_2 \boldsymbol{v}_t + (1-\beta_2)[\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t)]^2, \\ \boldsymbol{\theta}_{t+1} &= (1-\eta\lambda)\boldsymbol{\theta}_t - \eta \frac{\boldsymbol{m}_{t+1}/(1-\beta_1^{t+1})}{\sqrt{\boldsymbol{v}_{t+1}/(1-\beta_2^{t+1})}+\epsilon}, \end{aligned}$$

where $\boldsymbol{\theta}_t$ is the vector of network parameters, η is the learning rate, β_1, β_2 are momentum hyperparameters, ϵ is the numerical stability hyperparameter, $\lambda > 0$ is the weight decay hyperparameter, $\nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_t)$ is the loss gradient; the square, square root and division are component-wise.

2 Larger β_2 Hurts Generalization Except for Very Small Batches

2.1 Training Transformer-XL from Scratch on WikiText-2

We train Transformer-XL (Z. Dai et al., 2019) on WikiText-2 (Merity et al., 2017) with different batch sizes (all powers of two from 2^7 to 2^{14}) and learning rates in the set $\{10^{-3}, 10^{-3.5}, 10^{-4}\}$. The implementation follows Z. Dai et al. (2019); J. Zhang et al. (2020) as in Kunstner et al. (2023). We fix the default value $\beta_1 = 0.9$, and sweep β_2 on the log-scale between 0.9 and the typical default value 0.999. As shown in Fig. 1, the model quickly overfits as training loss continues to go to zero. Therefore, we train for sufficiently many epochs to let the model overfit, and plot the minimal validation perplexity achieved, depending on β_2.

The results are shown in Fig. 2. We observe that larger β_2 usually hurts generalization (increases minimal validation perplexity), except for very small batch sizes, in which case the relationship is U-shaped, with optimal β_2 between 0.93 and 0.98. Note also that taking a very large β_2 like the default 0.999 is never optimal in these experiments. Table 1 shows the relative improvements in validation perplexity of the tuned β_2 versus the default $\beta_2 = 0.999$, for different batch sizes and learning rates. We see that the improvements are very substantial, getting up to 12.87%. Note also that for many experiments the best β_2 we found is equal to $\beta_1 = 0.9$. It is likely that taking $\beta_2 < \beta_1$ is even better for validation error, but we did not include those values into our sweeps.

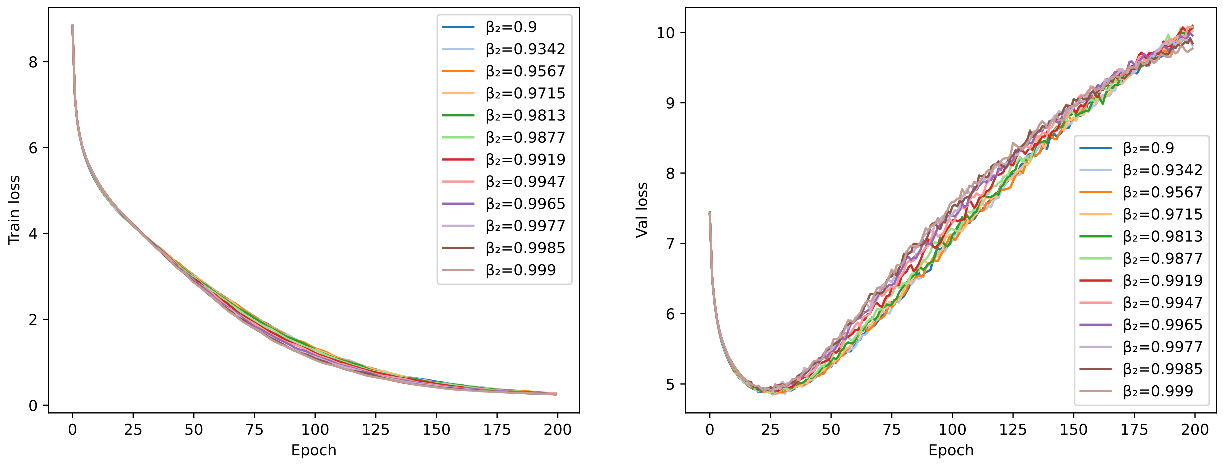

Figure 1: Transformer-XL on WikiText-2: training loss continues to decrease but validation loss starts to increase after about 30 epochs because of overfitting.

2.2 Fine-Tuning DistilBERT on SQuAD

We see the same trend on another language task: fine-tuning a pretrained DistilBERT (Sanh et al., 2019) model on the question-answering dataset SQuAD (Rajpurkar et al., 2016). We use the HuggingFace implementation from the "Transformers" library (Wolf et al., 2019), following Kunstner et al. (2023). As

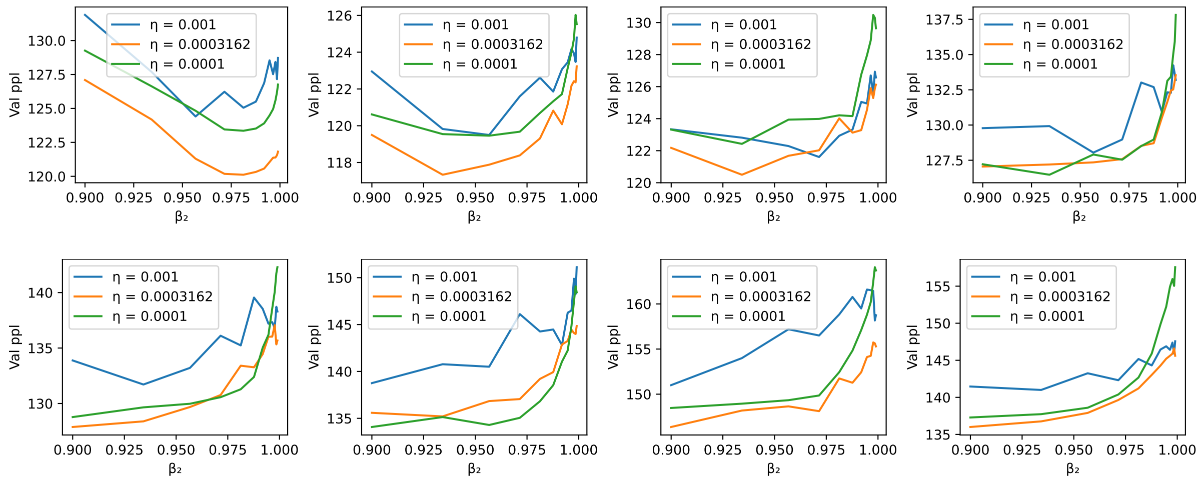

Figure 2: Minimal validation perplexity (before overfitting) of Transformer-XL on WikiText-2. Adam with batch sizes (left to right, top to bottom) 128, 256, 512, 1024, 2048, 4096, 8192 and 16384 (full-batch). Fixed hyperparameters: $\beta_1 = 0.9$, $\varepsilon = 10^{-6}$.

Table 1: Transformer-XL trained from scratch on WikiText-2: "optimal" hyperparameter values $\beta_2(\eta)$ we found, and relative improvements $\Delta(\eta)$ in validation perplexity for different learning rates η and batch sizes. Note that all "optimal" β_2's are smaller than even 0.99, let alone the default 0.999.

Batch Size	$\beta_2(10^{-3})$	$\Delta(10^{-3})$
128	0.957	3.35%
256	0.957	4.25%
512	0.972	3.92%
1024	0.957	3.87%
2048	0.934	4.76%
4096	0.900	8.18%
8192	0.900	4.88%
16384	0.934	4.45%

Batch Size	$\beta_2(10^{-3.5})$	$\Delta(10^{-3.5})$
128	0.972	1.33%
256	0.934	4.79%
512	0.934	4.46%
1024	0.934	4.68%
2048	0.900	5.75%
4096	0.934	8.22%
8192	0.900	5.75%
16384	0.934	7.00%

Batch Size	$\beta_2(10^{-4})$	$\Delta(10^{-4})$
128	0.981	2.69%
256	0.957	4.83%
512	0.934	5.57%
1024	0.934	8.23%
2048	0.900	9.50%
4096	0.900	9.68%
8192	0.900	9.31%
16384	0.900	12.87%

shown in Fig. 3, the model also quickly overfits, so we plot the maximal val_exact_match metric (percentage of answers exactly matching ground-truth on the validation set) achieved, depending on β_2. We observe (Fig. 4) that in all experiments larger β_2 hurt generalization (decrease this metric).

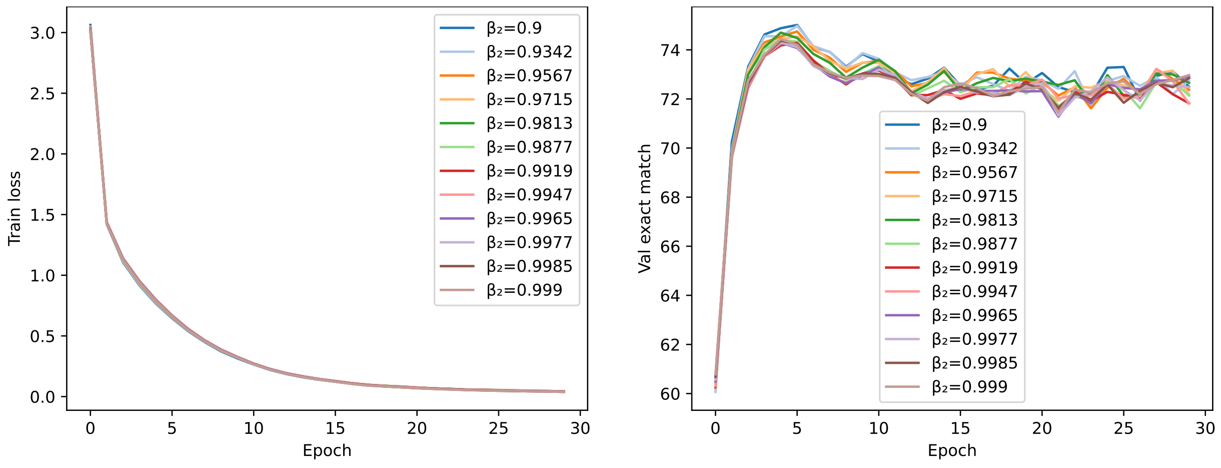

Figure 3: Pretrained DistilBERT fine-tuned on SQuAD: the training loss continues to decrease but the percentage of answers exactly matching ground-truth on the validation set decreases because of overfitting.

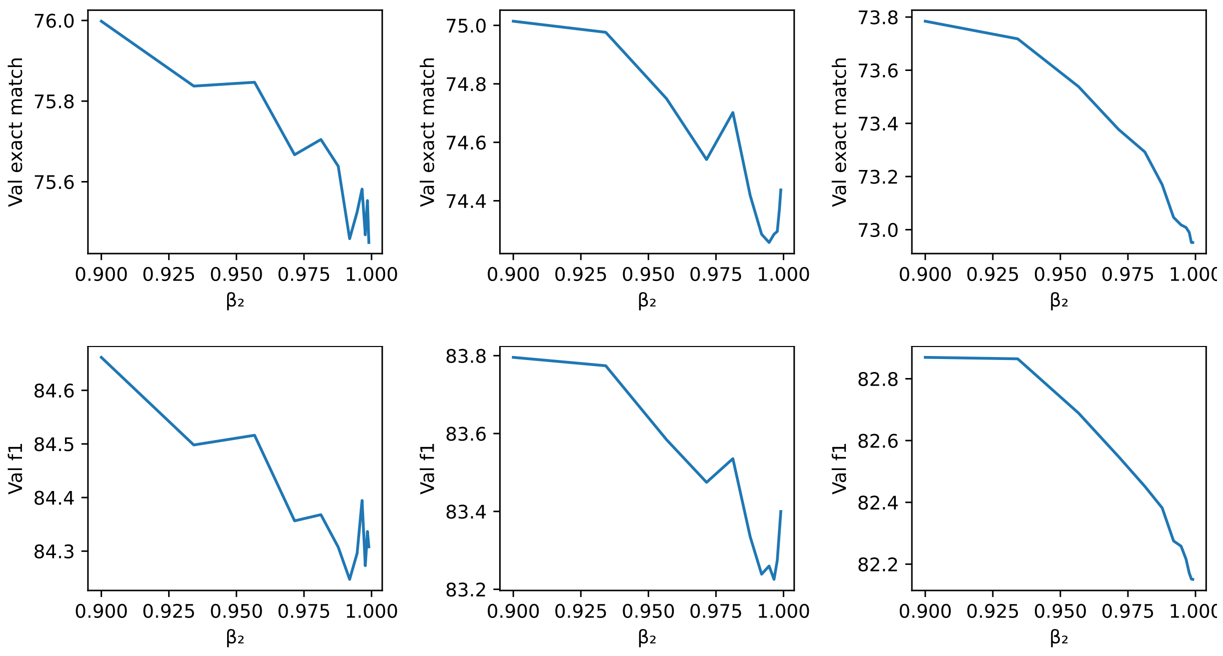

Figure 4: DistilBert fine-tuning on SQuAD. Top: percentage of answers exactly matching ground truth on the validation set; bottom: F1 score. Adam with batch sizes (left to right) 512, 2048, 8192, learning rate 10^{-4}, $\beta_1 = 0.9$, $\varepsilon = 10^{-6}$.

Table 2: Pretrained DistilBERT fine-tuned on SQuAD: "optimal" hyperparameter values $\beta_2(\eta)$ we found, and relative improvements $\Delta(\eta)$ in validation perplexity for learning rate η and different batch sizes.

Batch Size	$\beta_2(10^{-4})$	$\Delta(10^{-4})$
512	0.900	0.73%
2048	0.900	0.78%
8192	0.900	1.14%

2.3 Training GPT-2 from Scratch on FineWeb

We also train the GPT-2 (124M) model Radford et al., 2019 on the FineWeb-10B (10 billion tokens) dataset Penedo et al., 2024 from scratch. Our implementation is based on the nanoGPT repository and Xie, Mohamadi, and Z. Li (2024). The algorithm we use is AdamW with weight decay parameter 0.1. The learning rate is warmed up linearly from zero to $1.8 \cdot 10^{-3}$ for 2000 iterations, then decayed to $1.8 \cdot 10^{-4}$ using cosine schedule. We only do one pass over the training set, and plot in Fig. 5 the best validation loss achieved depending on β_2 for a few different batch sizes. We observe the same trend as we did in Section 2.1: for small batch sizes, larger β_2 can be optimal, but for medium-to-large batch sizes the best β_2 is much smaller than the default value 0.999. Note that we effectively constrain the number of iterations and sweep β_2's rather than waiting for the model to overfit, which would be computationally expensive. Thus, the results may be influenced by how β_2 affects the speed of training, which we discuss in more detail in Section 4.

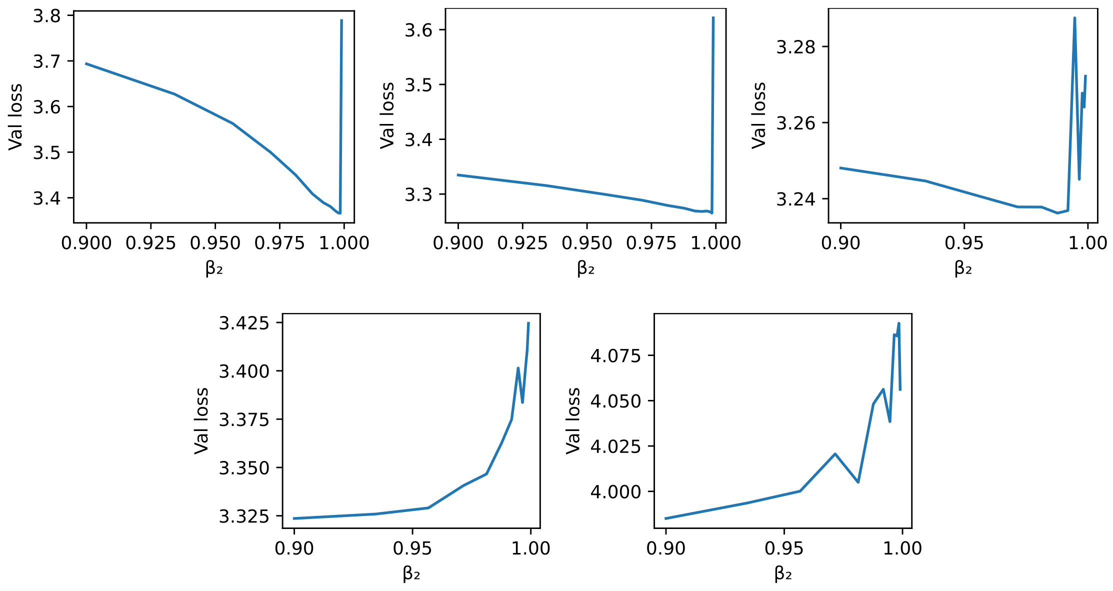

Figure 5: GPT-2 (124M) trained from scratch on FineWeb-10B: validation loss achieved after one pass over the training set. The batch sizes are (left to right, top to bottom) 2^{15}, 2^{17}, 2^{19}, 2^{21}, 2^{23}, sequence length 1024. AdamW with $\beta_1 = 0.9$, $\varepsilon = 10^{-6}$, weight decay 0.1.

2.4 Training CNN on CIFAR-10

To investigate large-batch training on vision tasks, we train a small CNN on CIFAR-10 with full-batch Adam. The implementation is based on Cattaneo, Klusowski, and Shigida (2024), and the CNN is taken from there. Throughout CIFAR-10 experiments, we use cross-entropy loss, subtract per-pixel mean and divide by standard deviation, and use data-augmentation from Lee et al. (2015), (padding followed by random crop and random horizontal flip).

Since full-batch training is slow and the model is small, we train for around 30K iterations to give the model enough time to achieve near-perfect train accuracy. If the default hyperparameter $\beta_1 = 0.9$ is taken, large β_2 incur very large loss spikes, to the point where $\beta_2 \approx 0.999$ fails to get to interpolating solution: the spikes are too frequent and most of the training is spent recovering from them (Fig. 6). However, we observe that large β_2 achieve better validation accuracy in this highly spiky regime, with the range of validation accuracies achieved 83.2% to 84% (Fig. 7).

To avoid loss spikes, we also take $\beta_1 = 0.99$. Then, all training curves look smooth (Fig. 8). In this case, the best validation accuracy achieved decreases as β_2 grows, with the range of validation accuracies 83.2% at $\beta_2 \approx 0.9977$ to 85.5% at $\beta_2 = 0.9$ (Fig. 9). Note that the best validation accuracy is achieved at $0.9 = \beta_2 < \beta_1 = 0.99$, and this is the boundary of the range of β_2's we sweeped (it may be the case that much smaller β_2's are even better). Note also that in addition to training curves being smooth, the validation accuracies are significantly better than in the spiky regime.

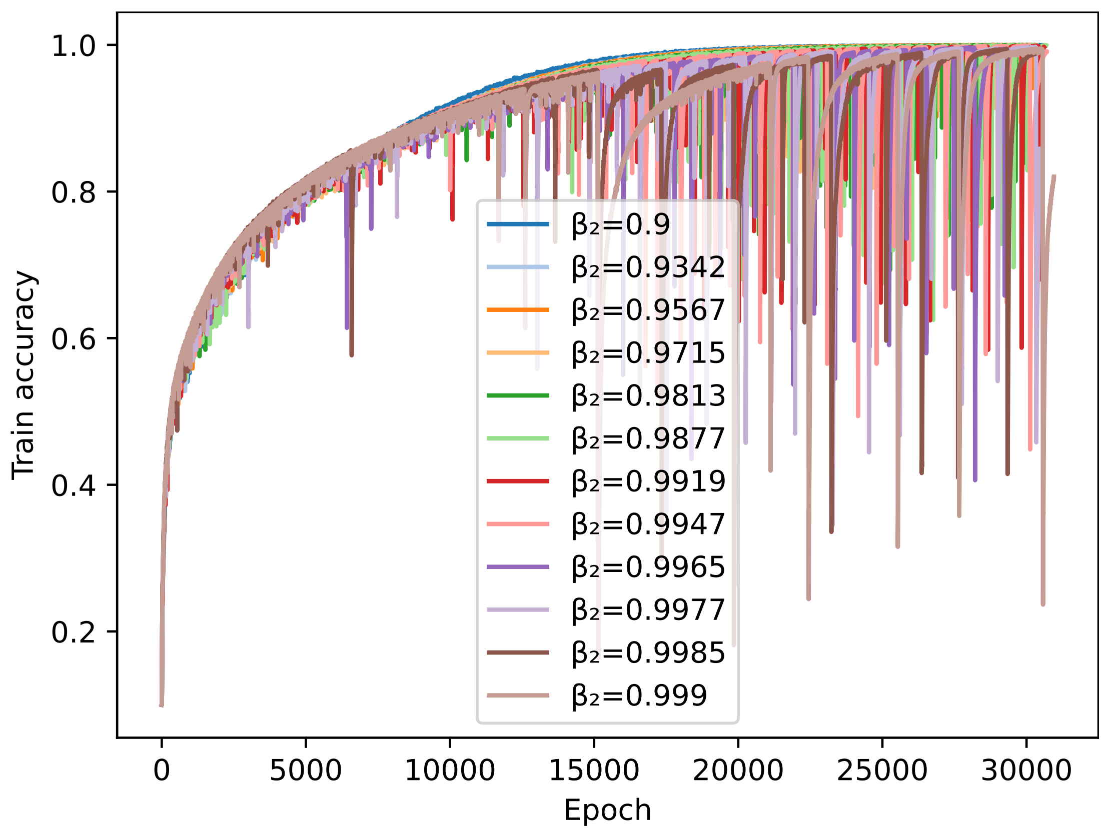

Figure 6: CNN trained on CIFAR-10 with full-batch Adam, $\beta_1 = 0.9$, $\varepsilon = 10^{-6}$. Train accuracy curves are very spiky.

2.5 Training ResNet-50 on CIFAR-10

To investigate further how β_2 affects generalization on a vision task for different batch sizes, we train ResNet-50 on CIFAR-10 with Adam, taking batch sizes 1024, 4096, 16384, and 50000. Because of the issue of loss spikes as described in Section 2.4, we take $\beta_1 = 0.99$. Since the best test accuracy in Section 2.4 was achieved at $\beta_2 = 0.9$ which was the boundary of the sweep, we include smaller values of β_2 (using the log-scale from 0.8 to 0.999). For batch sizes 1024 and 4096, the training with $\beta_2 < 0.9$ is divergent. For batch size 1024, higher β_2 slightly improves test accuracy, the dependence becomes inverse U-shaped as the batch size increases, and finally in full-batch training the smaller β_2 the better. For full-batch Adam, the best test accuracy is achieved at $\beta_2 = 0.8$, which is *again* the boundary of the sweep. The results are shown in Fig. 10.

From observations here and Section 2.4 we may conclude, in particular, that for large batch sizes experimenting with $\beta_2 < \beta_1$ can be beneficial for generalization on vision tasks.

3 Loss Spikes

It is known that the training of Adam (with or without decoupled weight decay) can be spiky Shazeer and Stern (2018); Chowdhery et al. (2023); C. Ma, Wu, and Weinan (2022). In particular, C. Ma, Wu, and Weinan (2022) demonstrate that when training ResNets on CIFAR-10 spiky training happens in the setting $\beta_2 \gg \beta_1$. We reproduce this, and verify that the same phenomenon is present when training modern architectures on all our language tasks. In addition, our observations are not constrained to large batches, and our training pipeline is closer to modern practice. In particular, we use both a learning rate decay schedule and weight decay when training GPT-2. We find that for $\beta_1 = 0.9$, $\beta_2 > 0.99$ large loss spikes are a very frequent problem, essentially guaranteed for moderate batch sizes. If β_2 is reduced, loss spikes are mitigated, and if $\beta_1 = \beta_2$ we do not observe large spikes at all. For a vision task, Fig. 6 should provide a convincing illustration. For a language task, see Fig. 11 (loss curves of GPT-2 trained on

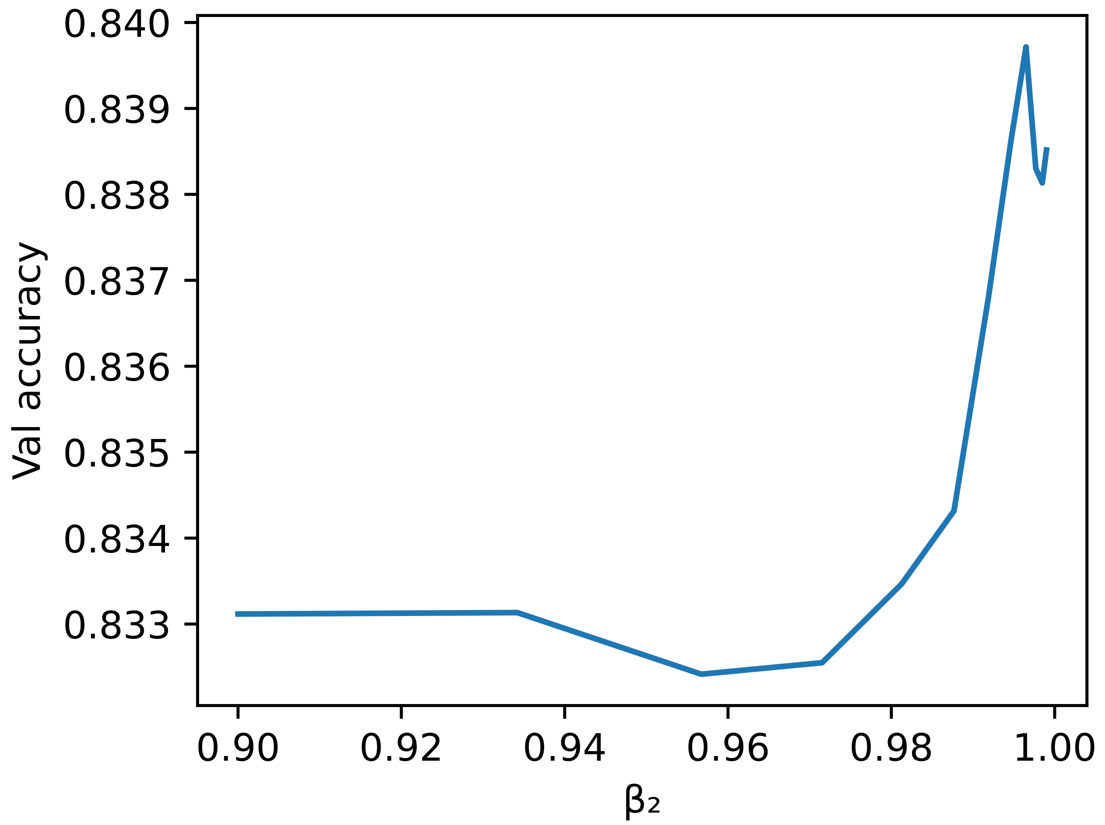

Figure 7: CNN trained on CIFAR-10 with full-batch Adam, $\beta_1 = 0.9$, $\varepsilon = 10^{-6}$, learning rate $2 \cdot 10^{-4}$. Best test accuracy achieved after about 30K epochs. The results are averaged across 6 runs with different initialization seeds.

FineWeb) as an example.

However, in practice it appears to be rare to attribute loss spikes to a large β_2 and attempt to fix the problem by decreasing it (see a more detailed discussion of this in the introduction). We stress that it is not fair to blame Adam itself for being unstable, rather than Adam with a specific hyperparameter choice. We recommend trying $\beta_1 \approx \beta_2$ as a sufficiently simple intervention to improve the stability of training.

4 Speed of Training

For CNN on CIFAR-10 trained with full-batch Adam, we plot the training loss curves and how many epochs are required to reach the training loss threshold 0.1. The results in Fig. 12 show that small β_2 improves training speed (measured by how fast the training loss decreases).

Similarly, we plot how many epochs are required to reach the training loss threshold 0.6 for all the Transformer-XL training on WikiText-2 tasks with learning rate 10^{-4} (only one learning rate to improve presentation; the trends for other learning rates are the same). The results in Fig. 13 show that the relationship of the number of epochs required depending on β_2 is different for different batch sizes, similarly to validation perplexity: for small batch sizes, it is decreasing (larger β_2 makes training faster), for moderate batch sizes it is inverse U-shaped, and finally for very large batches it becomes increasing (larger β_2 makes training slower).

5 A Note on How β_1 Affects Generalization

Similarly to the β_2 sweeps in Section 2, we train Transformer-XL on WikiText-2 with different batch sizes and learning rates, and plot the minimal validation perplexity achieved depending on β_1. We fix the default value $\beta_2 = 0.999$ to ensure that $\beta_1 < \beta_2$ which is common practice. The results are shown in Fig. 14. It

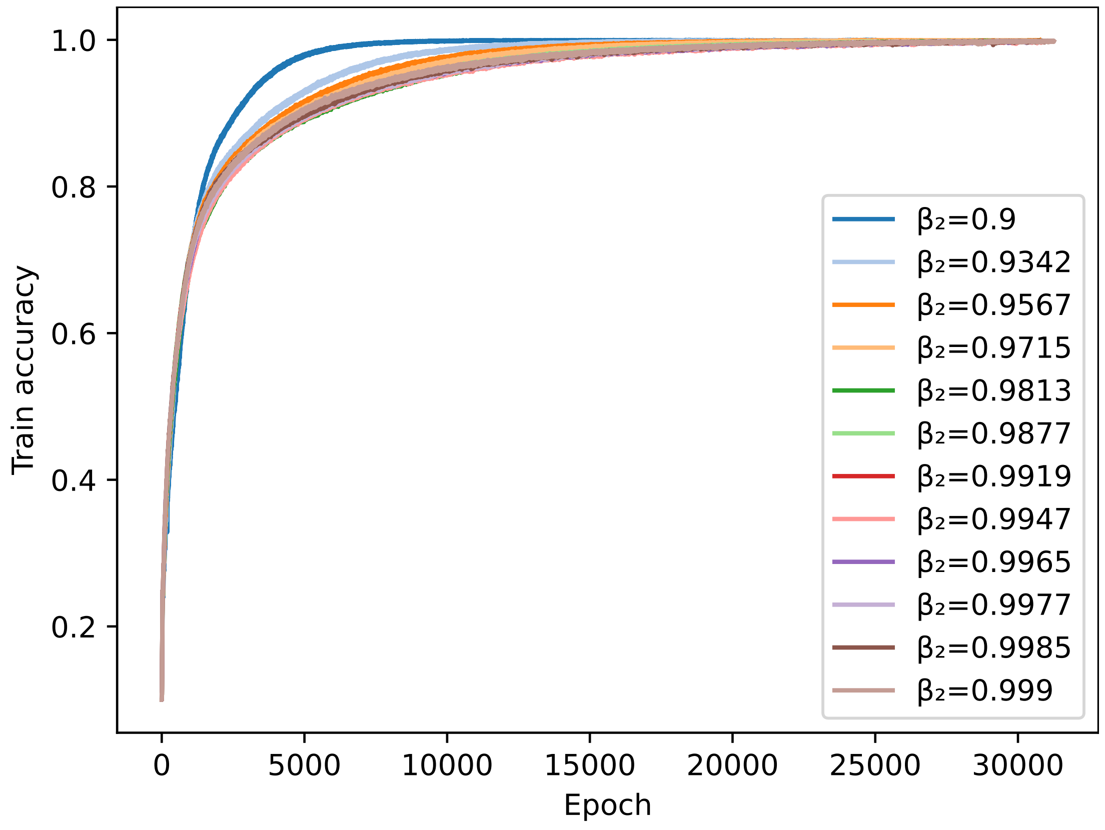

Figure 8: CNN trained on CIFAR-10 with full-batch Adam, $\beta_1 = 0.99$, $\varepsilon = 10^{-6}$. Train accuracy curves are smooth.

can be observed that for medium learning rates, validation perplexity has a U-shaped dependence on β_1, with optimal β_1 somewhere in between 0.9 and 0.99, but for very small learning rates the trend reverses. For very small learning rates and large-batch training, increasing β_1 improves generalization.

6 A Note on Sharpness and Edge of Stability

Cohen et al. (2022) notice that in a sense Adam trains at the edge of stability. They view Adam as momentum gradient descent with evolving preconditioner

$$\mathbf{P}_{t+1} = (1 - \beta_1^{t+1})\left[\operatorname{diag}\left(\sqrt{\frac{\boldsymbol{\nu}_{t+1}}{1 - \beta_2^{t+1}}}\right) + \epsilon \mathbf{I}\right].$$

They define "preconditioned sharpness" to be the top eigenvalue of the preconditioned Hessian $\lambda_1(\mathbf{P}_t^{-1}\mathbf{H}_t)$, where $\mathbf{H}_t$ is the Hessian of the loss, and observe that this quantity often oscillates around the stability threshold $\frac{2+2\beta_1}{(1-\beta_1)\eta}$, where η is the learning rate. (This fraction comes from the fact that if the preconditioner were constant, Adam would become a form of preconditioned gradient descent with EMA-style momentum, and this is the ordinary stability threshold of EMA-style heavy-ball momentum on the quadratic Taylor approximation of the loss; we refer to Cohen et al. (2022) for details.) They use large-batch training on CIFAR-10/100. We train a CNN on CIFAR-10, and reproduce this result in Fig. 15. We also plot ordinary sharpness $\lambda_1(\mathbf{H}_t)$ (top hessian eigenvalue), which first increases and then decreases. Recall from Fig. 6 that this is an extremely unstable regime of training.

Note, however, that if we take $\beta_1 = 0.99$ which is the "smooth" regime of training, preconditioned sharpness does not reach the stability threshold (Fig. 16). Note also that ordinary sharpness $\lambda_1(\mathbf{H}_t)$ (top hessian eigenvalue) is much lower for small β_2 (especially noticeable for $\beta_1 = 0.9$). This suggests that in this situation taking $\beta_2 < \beta_1$ may regularize training, moving the model parameters to flatter regions of the loss space.

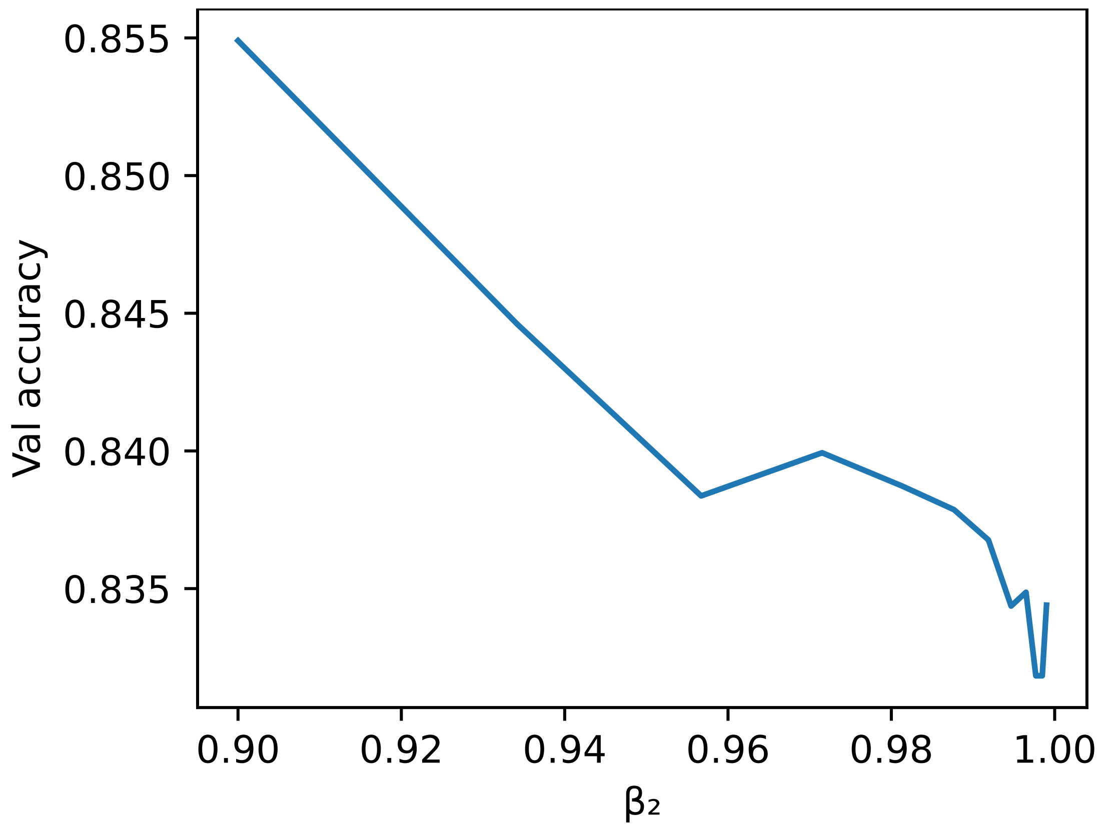

Figure 9: CNN trained on CIFAR-10 with full-batch Adam, $\beta_1 = 0.99$, $\varepsilon = 10^{-6}$. Best validation accuracy achieved after about 30K epochs. The results are averaged across 3 runs with different initialization seeds.

7 Final Remarks

Based on the evidence provided, we conclude that tuning the hyperparameters of Adam (in particular β_2) is highly beneficial and often not done enough. In particular, in most cases the default (β_1, β_2) pair leads to unstable training and is very suboptimal in terms of test performance in modern tasks. We have observed that decreasing β_2 can increase test performance by up to 12.87% for large batches. Loss spikes, a very popular problem when training with AdamW, are mitigated when taking $\beta_1 \approx \beta_2$. Finally, decreasing β_2 with respect to β_1 is sometimes (for large batches) beneficial for the training speed as well. As a simple rule, we recommend taking β_2 in the range $[0.9, 0.95]$ if $\beta_1 = 0.9$. In the next version of this article, we plan to expand the grids and average the results across multiple runs to smooth out the plots. For reproducibility, we plan to make the code available.

Acknowledgments

Cattaneo gratefully acknowledges financial support from the National Science Foundation through DMS-2210561 and SES-2241575. We acknowledge the Princeton Research Computing resources, coordinated by the Princeton Institute for Computational Science and Engineering (PICSciE) and the Office of Information Technology's Research Computing.

References

Anil, Rohan, Andrew M Dai, et al. (2023). “Palm 2 technical report”. In: *arXiv preprint arXiv:2305.10403* (cit. on p. 1).

Anil, Rohan, Vineet Gupta, et al. (2020). “Scalable second order optimization for deep learning”. In: *arXiv preprint arXiv:2002.09018* (cit. on p. 1).

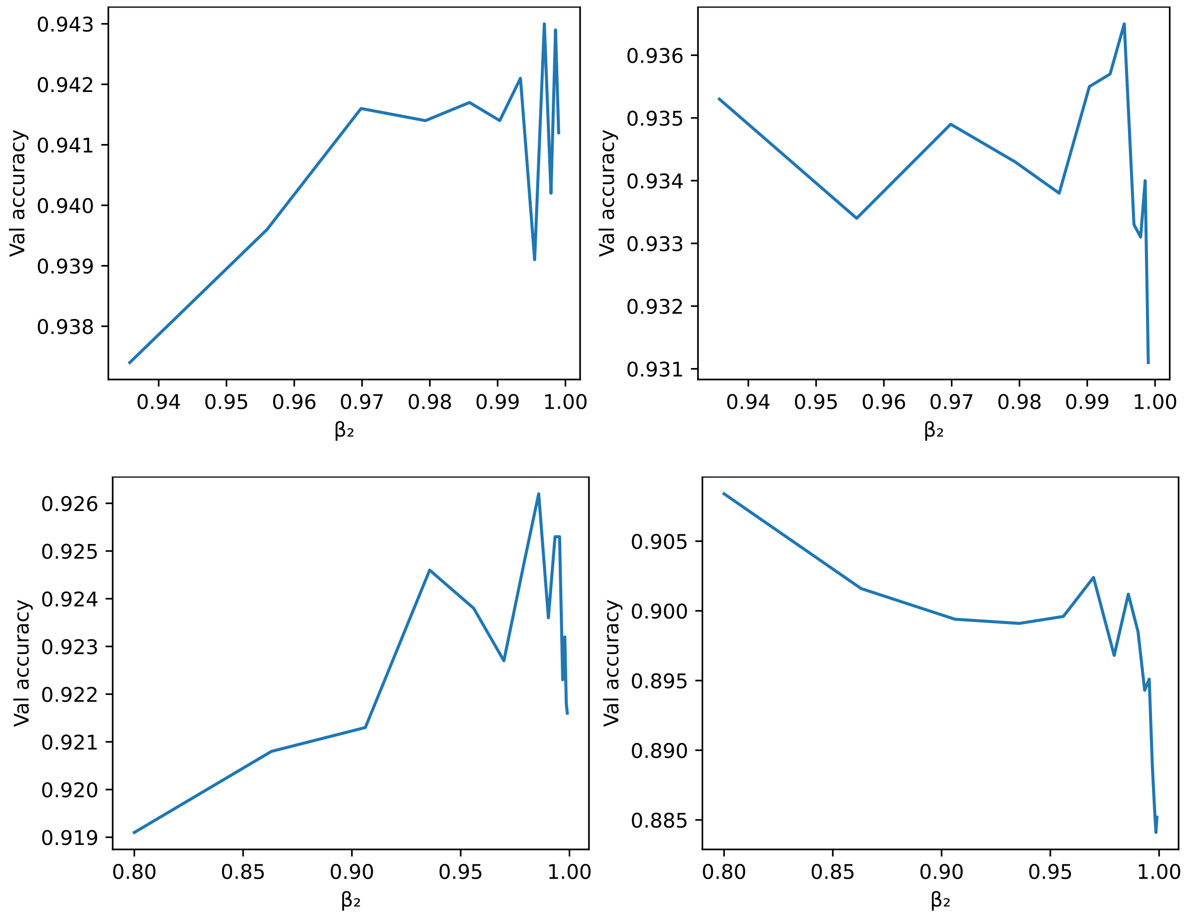

Figure 10: ResNet-50 trained on CIFAR-10 with Adam, $\beta_1 = 0.99$, $\varepsilon = 10^{-6}$, learning rate 10^{-3}. Batch sizes, left to right, top to bottom: 1024, 4096, 16384, 50000 (full-batch).

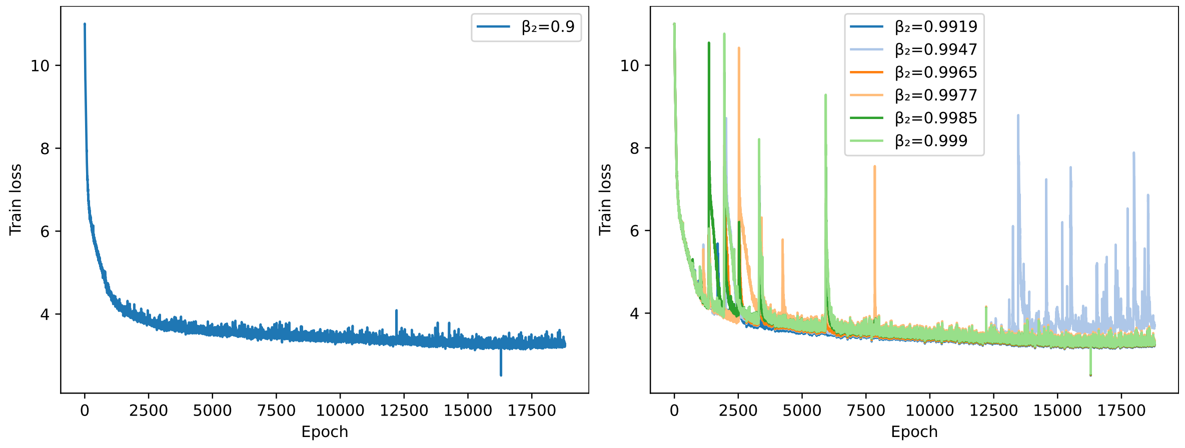

Figure 11: GPT-2 (124M) trained from scratch on FineWeb-10B: for β_2 close to β_1 the training is relatively smooth, and for $\beta_2 \gg \beta_1$ there are large loss spikes. Batch size 2^{19}, sequence length 1024. AdamW with $\beta_1 = 0.9$, $\varepsilon = 10^{-6}$.

Bernstein, Jeremy et al. (2018). "signSGD: Compressed optimisation for non-convex problems". In: *International Conference on Machine Learning*. PMLR, pp. 560–569 (cit. on p. 2).

Biderman, Stella et al. (2023). "Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by

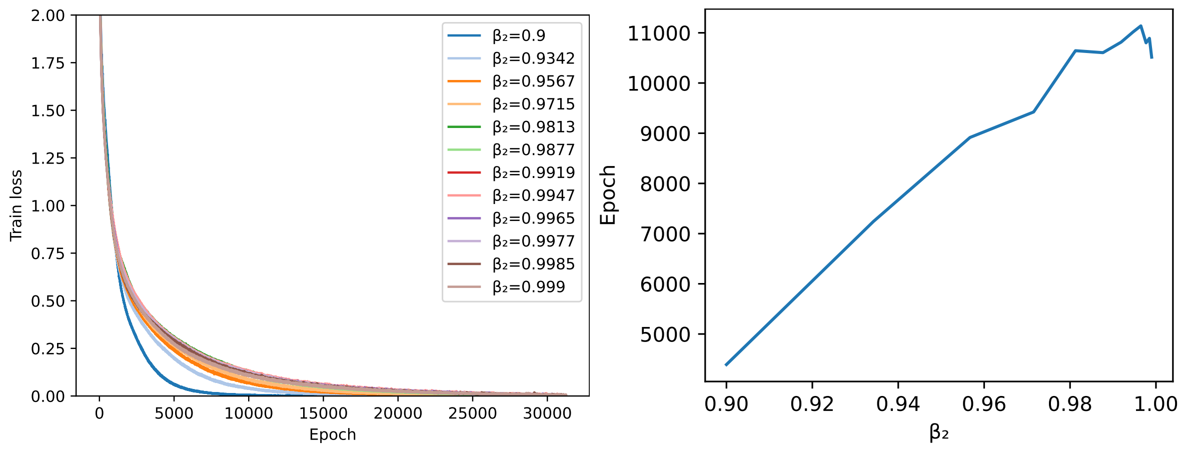

Figure 12: CNN trained on CIFAR-10 with full-batch Adam, $\beta_1 = 0.99$, $\varepsilon = 10^{-6}$, learning rate $2 \cdot 10^{-4}$. Left: training loss curves. Right: how many epochs were required to reach loss threshold 0.1 (averaged over three runs with different initialization seeds).

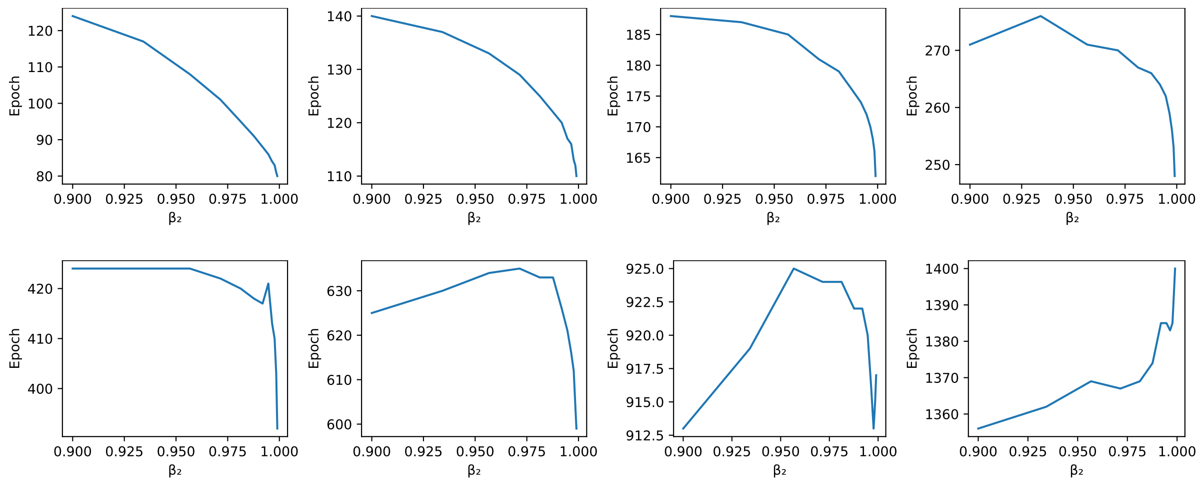

Figure 13: Transformer-XL on WikiText-2: how many epochs are required to reach the training loss threshold 0.6. Adam with batch sizes (left to right, top to bottom) 128, 256, 512, 1024, 2048, 4096, 8192 and 16384 (full-batch). Fixed hyperparameters: $\beta_1 = 0.9$, $\varepsilon = 10^{-6}$, learning rate 10^{-4}.

Andreas Krause et al. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 2397–2430 (cit. on p. 2).

Brown, Tom et al. (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., pp. 1877–1901 (cit. on pp. 1, 2).

Cattaneo, Matias D., Jason Matthew Klusowski, and Boris Shigida (2024). "On the Implicit Bias of Adam". In: *Proceedings of the 41st International Conference on Machine Learning*. Ed. by Ruslan Salakhutdinov et al. Vol. 235. Proceedings of Machine Learning Research. PMLR, pp. 5862–5906 (cit. on p. 6).

Chen, Xiangning et al. (2023). "Symbolic Discovery of Optimization Algorithms". In: *Thirty-seventh Conference on Neural Information Processing Systems* (cit. on p. 1).

Choi, D (2019). "On empirical comparisons of optimizers for deep learning". In: *arXiv preprint arXiv:1910.05446* (cit. on p. 1).

Chowdhery, Aakanksha et al. (2023). "Palm: Scaling language modeling with pathways". In: *Journal of Machine Learning Research* 24.240, pp. 1–113 (cit. on pp. 2, 7).

Cohen, Jeremy M et al. (2022). "Adaptive gradient methods at the edge of stability". In: *arXiv preprint arXiv:2207.14484* (cit. on p. 9).

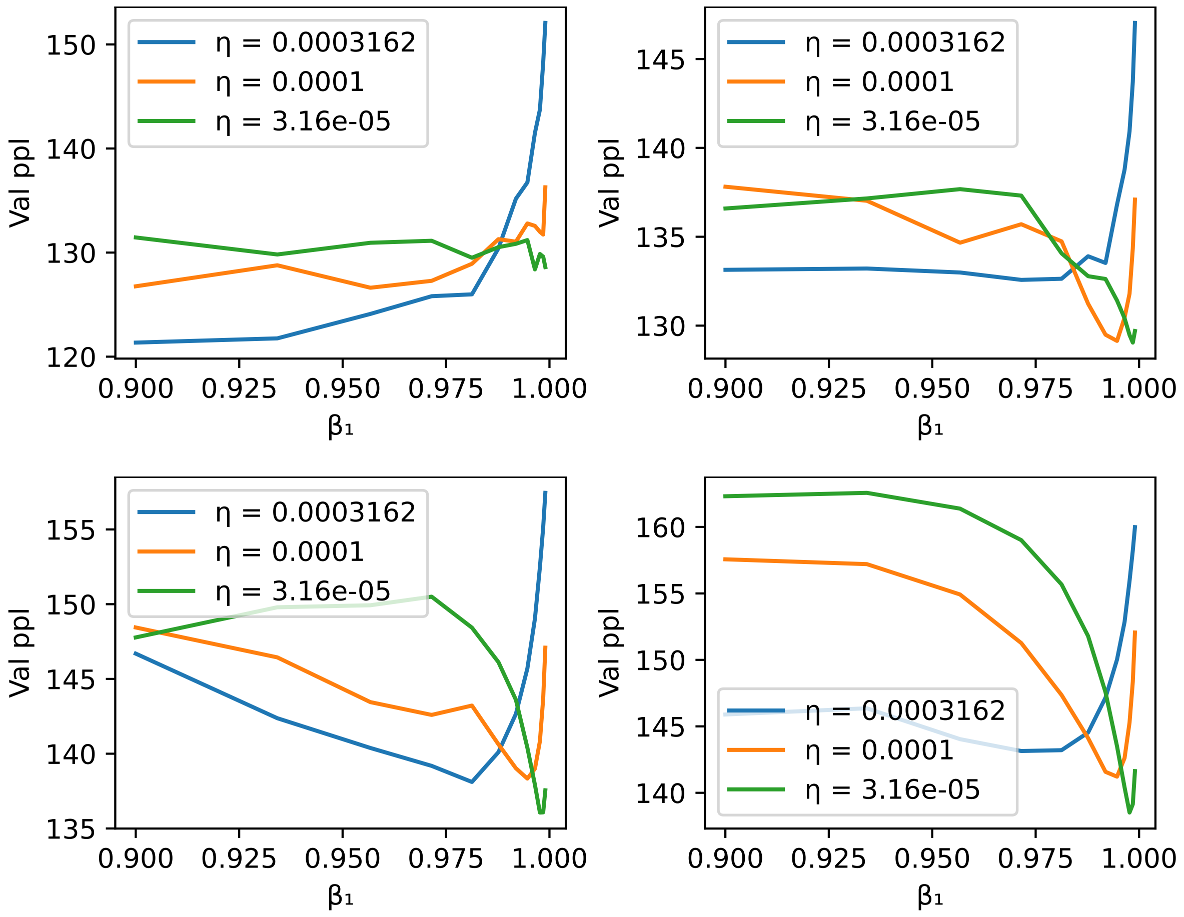

Figure 14: Minimal validation perplexity (before overfitting) of Transformer-XL on WikiText-2. Adam with batch sizes (left to right) 128, 1024, 4096, and 16384 (full-batch). Fixed hyperparameters: $\beta_2 = 0.999$, $\varepsilon = 10^{-6}$.

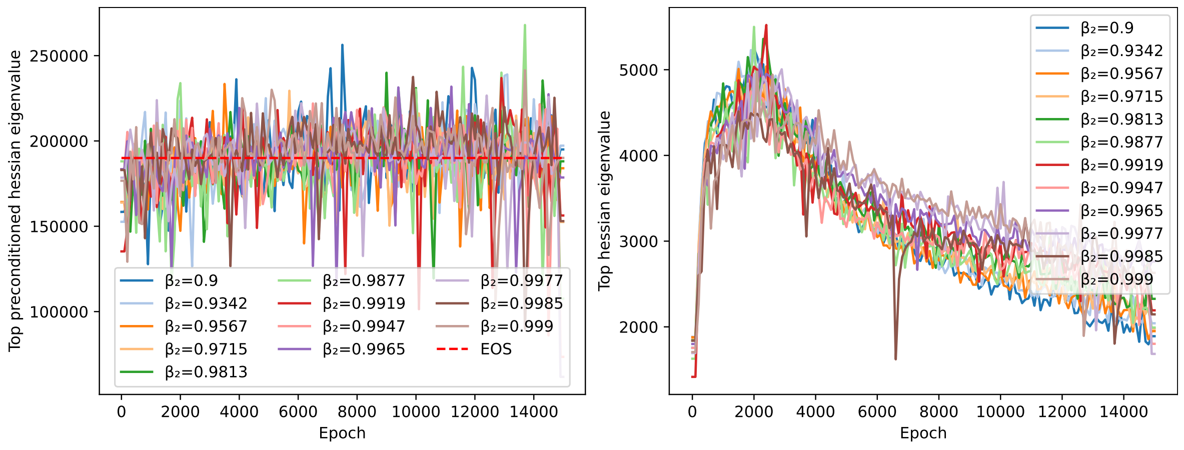

Figure 15: CNN trained on CIFAR-10 with full-batch Adam, $\beta_1 = 0.9$, $\epsilon = 10^{-6}$. Left: preconditioned sharpness $\lambda_1(\mathbf{P}_t^{-1}\mathbf{H}_t)$ oscillates around the stability threshold. Right: the plots of ordinary sharpness $\lambda_1(\mathbf{H}_t)$.

Dai, Zihang et al. (2019). “Transformer-XL: Attentive Language Models beyond a Fixed-Length Context”. In: *Annual Meeting of the Association for Computational Linguistics* (cit. on p. 3).
Dozat, Timothy (2016). “Incorporating nesterov momentum into adam”. In: (cit. on p. 2).

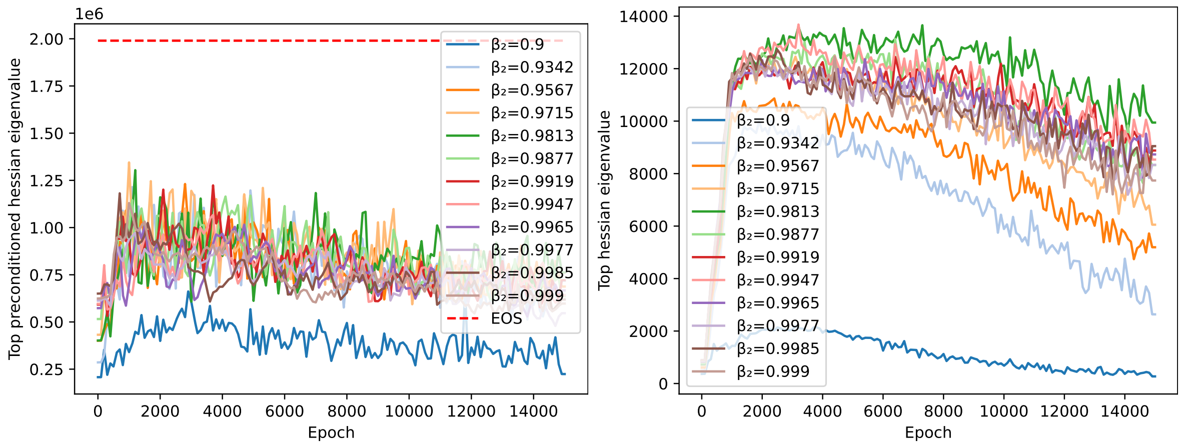

Figure 16: CNN trained on CIFAR-10 with full-batch Adam, $\beta_1 = 0.99$, $\epsilon = 10^{-6}$. Left: preconditioned sharpness $\lambda_1(\mathbf{P}_t^{-1}\mathbf{H}_t)$ does not reach the stability threshold. Right: the plots of ordinary sharpness $\lambda_1(\mathbf{H}_t)$.

Dubey, Abhimanyu et al. (2024). "The llama 3 herd of models". In: *arXiv preprint arXiv:2407.21783* (cit. on pp. 1, 2).

Gupta, Vineet, Tomer Koren, and Yoram Singer (2018). "Shampoo: Preconditioned stochastic tensor optimization". In: *International Conference on Machine Learning*. PMLR, pp. 1842–1850 (cit. on p. 2).

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (cit. on pp. 1, 3).

Kunstner, Frederik et al. (2023). "Noise Is Not the Main Factor Behind the Gap Between Sgd and Adam on Transformers, But Sign Descent Might Be". In: *The Eleventh International Conference on Learning Representations* (cit. on p. 3).

Lee, Chen-Yu et al. (2015). "Deeply-supervised nets". In: *Artificial intelligence and statistics*. Pmlr, pp. 562–570 (cit. on p. 6).

Liu, Liyuan et al. (2019). "On the variance of the adaptive learning rate and beyond". In: *arXiv preprint arXiv:1908.03265* (cit. on p. 2).

Loshchilov, Ilya and Frank Hutter (2019). *Decoupled Weight Decay Regularization* (cit. on pp. 1, 3).

Ma, Chao, Lei Wu, and E Weinan (2022). "A qualitative study of the dynamic behavior for adaptive gradient algorithms". In: *Mathematical and Scientific Machine Learning*. PMLR, pp. 671–692 (cit. on pp. 2, 7).

Ma, Jerry and Denis Yarats (2018). "Quasi-hyperbolic momentum and adam for deep learning". In: *arXiv preprint arXiv:1810.06801* (cit. on p. 2).

Merity, Stephen et al. (2017). "Pointer Sentinel Mixture Models". In: *International Conference on Learning Representations* (cit. on p. 3).

Penedo, Guilherme et al. (2024). *The FineWeb Datasets: Decanting the Web for the Finest Text Data at Scale* (cit. on p. 6).

Radford, Alec et al. (2019). "Language Models are Unsupervised Multitask Learners". In: (cit. on p. 6).

Rajpurkar, Pranav et al. (2016). "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Ed. by Jian Su, Kevin Duh, and Xavier Carreras. Austin, Texas: Association for Computational Linguistics, pp. 2383–2392 (cit. on p. 3).

Reddi, Sashank J., Satyen Kale, and Sanjiv Kumar (2018). "On the Convergence of Adam and Beyond". In: *International Conference on Learning Representations* (cit. on p. 2).

Sanh, Victor et al. (2019). "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *ArXiv* abs/1910.01108 (cit. on p. 3).

Savarese, Pedro et al. (2021). "Domain-independent dominance of adaptive methods". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16286–16295 (cit. on p. 1).

Shazeer, Noam and Mitchell Stern (2018). "Adafactor: Adaptive learning rates with sublinear memory cost". In: *International Conference on Machine Learning*. PMLR, pp. 4596–4604 (cit. on pp. 1, 2, 7).

Sivaprasad, Prabhu Teja et al. (2020). "Optimizer benchmarking needs to account for hyperparameter tuning". In: *International conference on machine learning*. PMLR, pp. 9036–9045 (cit. on p. 1).

Taniguchi, Shohei et al. (2024). "ADOPT: Modified Adam Can Converge with Any β_2 with the Optimal Rate". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems* (cit. on p. 2).

Touvron, Hugo et al. (2023). "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (cit. on pp. 1, 2).

Wolf, Thomas et al. (2019). "Transformers: State-of-the-Art Natural Language Processing". In: *Conference on Empirical Methods in Natural Language Processing* (cit. on p. 3).

Xie, Shuo, Mohamad Amin Mohamadi, and Zhiyuan Li (2024). *Adam Exploits ℓ_∞-geometry of Loss Landscape via Coordinate-wise Adaptivity* (cit. on p. 6).

Zeng, Aohan et al. (2022). "Glm-130b: An open bilingual pre-trained model". In: *arXiv preprint arXiv:2210.02414* (cit. on p. 2).

Zhang, Jingzhao et al. (2020). "Why are adaptive methods good for attention models?" In: *Advances in Neural Information Processing Systems* 33, pp. 15383–15393 (cit. on p. 3).

Zhang, Susan et al. (2022). "Opt: Open pre-trained transformer language models". In: *arXiv preprint arXiv:2205.01068* (cit. on p. 2).

Zhuang, Juntang et al. (2020). "Adabelief optimizer: Adapting stepsizes by the belief in observed gradients". In: *Advances in neural information processing systems* 33, pp. 18795–18806 (cit. on p. 2).