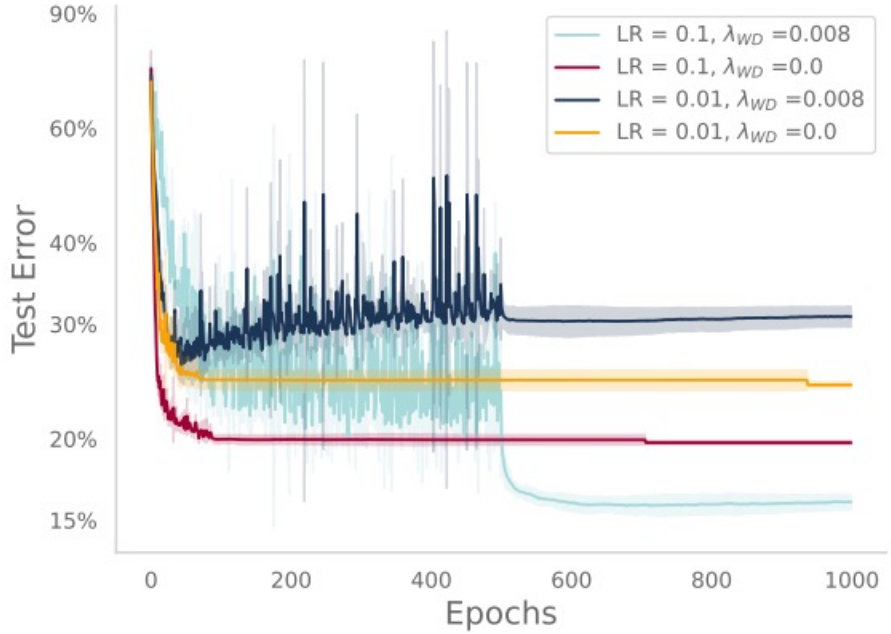
Why Do We Need Weight Decay in Modern Deep Learning?

Maksym Andriushchenko*, Francesco D'Angelo*, Aditya Varre, Nicolas Flammarion (EPFL)

Question: *WD is widely used for training most state-of-the-art deep networks, including large language models. But why? Is it about better regularization or optimization?*

Regime 1: overparameterized deep nets

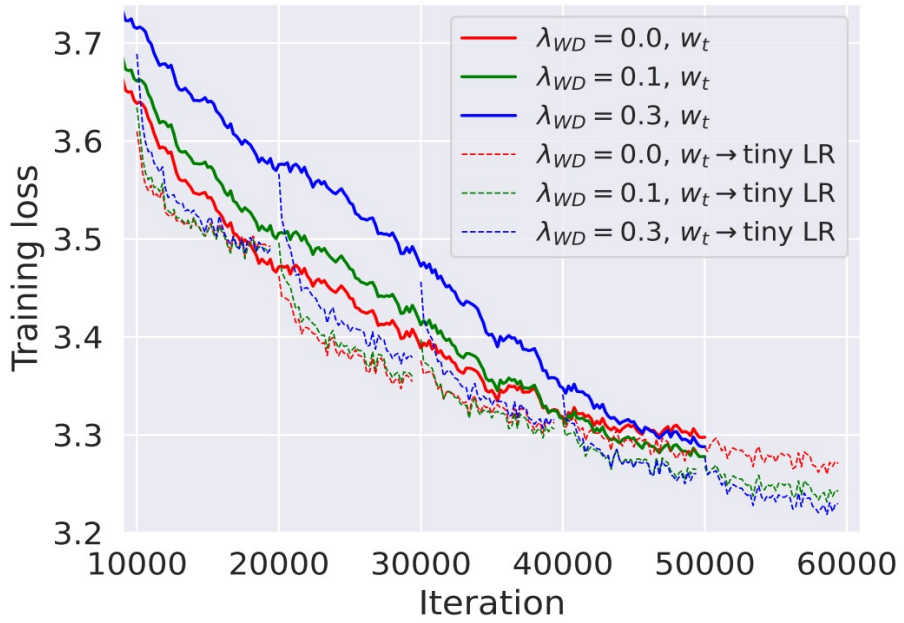
VGG on CIFAR-10



⚠️ WD helps only with large LRs *but not on its own*
✅ WD amplifies the implicit regularization of SGD via the loss stabilization mechanism

Regime 2: one-pass training of LLMs

GPT-2-124M on OpenWebText



⚠️ a higher loss with WD can still be a better starting point
✅ WD balances better the bias-variance optimization trade-off (+ prevents divergences for bfloat16 training!)

So that we are on the same page: L2 regularization vs. weight decay

Algorithm 1 SGD with L_2 regularization and SGD with decoupled weight decay (SGDW), both with momentum

given initial learning rate $\alpha \in \mathbb{R}$, momentum factor $\beta_1 \in \mathbb{R}$, weight decay/L_2 regularization factor $\lambda \in \mathbb{R}$

initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$

repeat

$t \leftarrow t + 1$

$\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient

$\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$

$\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, be used for warm restarts

$\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + \eta_t \alpha \boldsymbol{g}_t$

$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \boldsymbol{m}_t - \eta_t \lambda \boldsymbol{\theta}_{t-1}$

until *stopping criterion is met*

return optimized parameters $\boldsymbol{\theta}_t$

So that we are on the same page: L2 regularization vs. weight decay

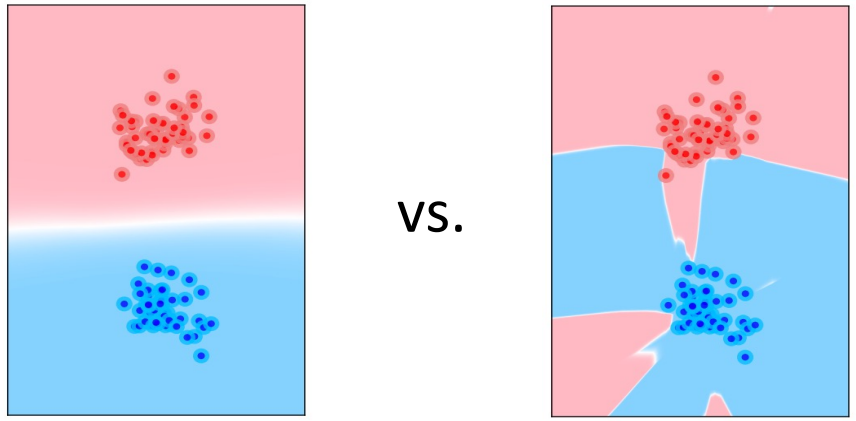
Algorithm 2 Adam with L_2 regularization and Adam with decoupled weight decay (AdamW)

given $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}, \lambda \in \mathbb{R}$
initialize time step $t \leftarrow 0$, parameter vector $\boldsymbol{\theta}_{t=0} \in \mathbb{R}^n$, first moment vector $\boldsymbol{m}_{t=0} \leftarrow \boldsymbol{0}$, second moment vector $\boldsymbol{v}_{t=0} \leftarrow \boldsymbol{0}$, schedule multiplier $\eta_{t=0} \in \mathbb{R}$
repeat
  $t \leftarrow t + 1$
  $\nabla f_t(\boldsymbol{\theta}_{t-1}) \leftarrow \text{SelectBatch}(\boldsymbol{\theta}_{t-1})$ ▷ select batch and return the corresponding gradient
  $\boldsymbol{g}_t \leftarrow \nabla f_t(\boldsymbol{\theta}_{t-1}) + \lambda \boldsymbol{\theta}_{t-1}$
  $\boldsymbol{m}_t \leftarrow \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1)\boldsymbol{g}_t$ ▷ here and below all operations are element-wise
  $\boldsymbol{v}_t \leftarrow \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2)\boldsymbol{g}_t^2$
  $\hat{\boldsymbol{m}}_t \leftarrow \boldsymbol{m}_t / (1 - \beta_1^t)$ ▷ β_1 is taken to the power of t
  $\hat{\boldsymbol{v}}_t \leftarrow \boldsymbol{v}_t / (1 - \beta_2^t)$ ▷ β_2 is taken to the power of t
  $\eta_t \leftarrow \text{SetScheduleMultiplier}(t)$ ▷ can be fixed, decay, or also be used for warm restarts
  $\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \eta_t \left(\alpha \hat{\boldsymbol{m}}_t / (\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon) + \lambda \boldsymbol{\theta}_{t-1} \right)$
until *stopping criterion is met*
return optimized parameters $\boldsymbol{\theta}_t$

We focus on the **decoupled weight decay** since it's more popular (especially for LLMs)

Big picture: understanding generalization in deep learning

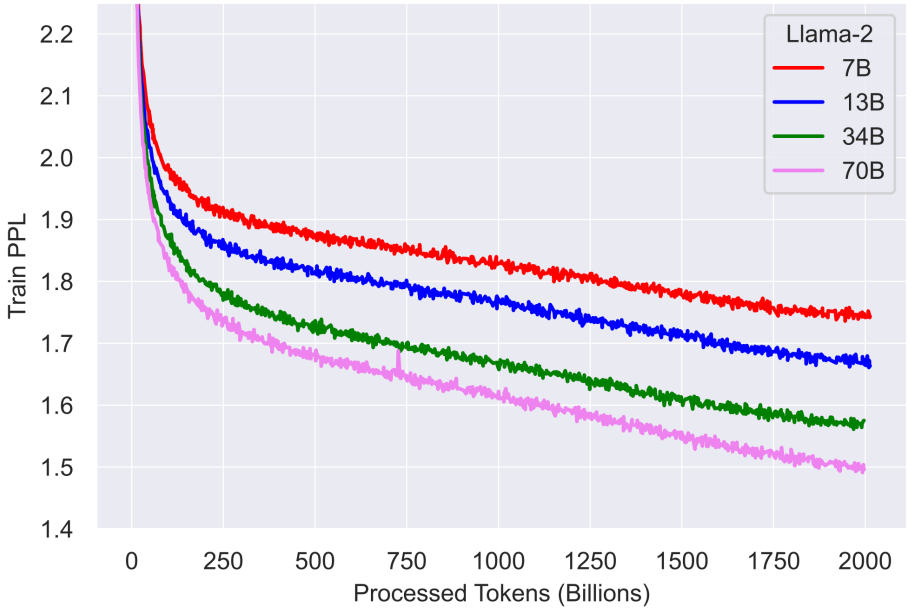
- **Overparameterized DL (CIFAR-10 / ImageNet)**: different global minima can generalize very differently
- Then the role of regularization (implicit or explicit) is to get to a better minimum



vs.

Bad Global Minima Exist and SGD Can Reach Them, NeurIPS'19

- ***Effectively* underparameterized DL (LLMs)**: training loss / perplexity already correlates very well with generalization!
- In almost all cases, we just need to minimize the training loss
- Why does everyone use weight decay for LLMs?
- GPT-3 paper: *"All models use weight decay of 0.1 to provide a small amount of regularization"*

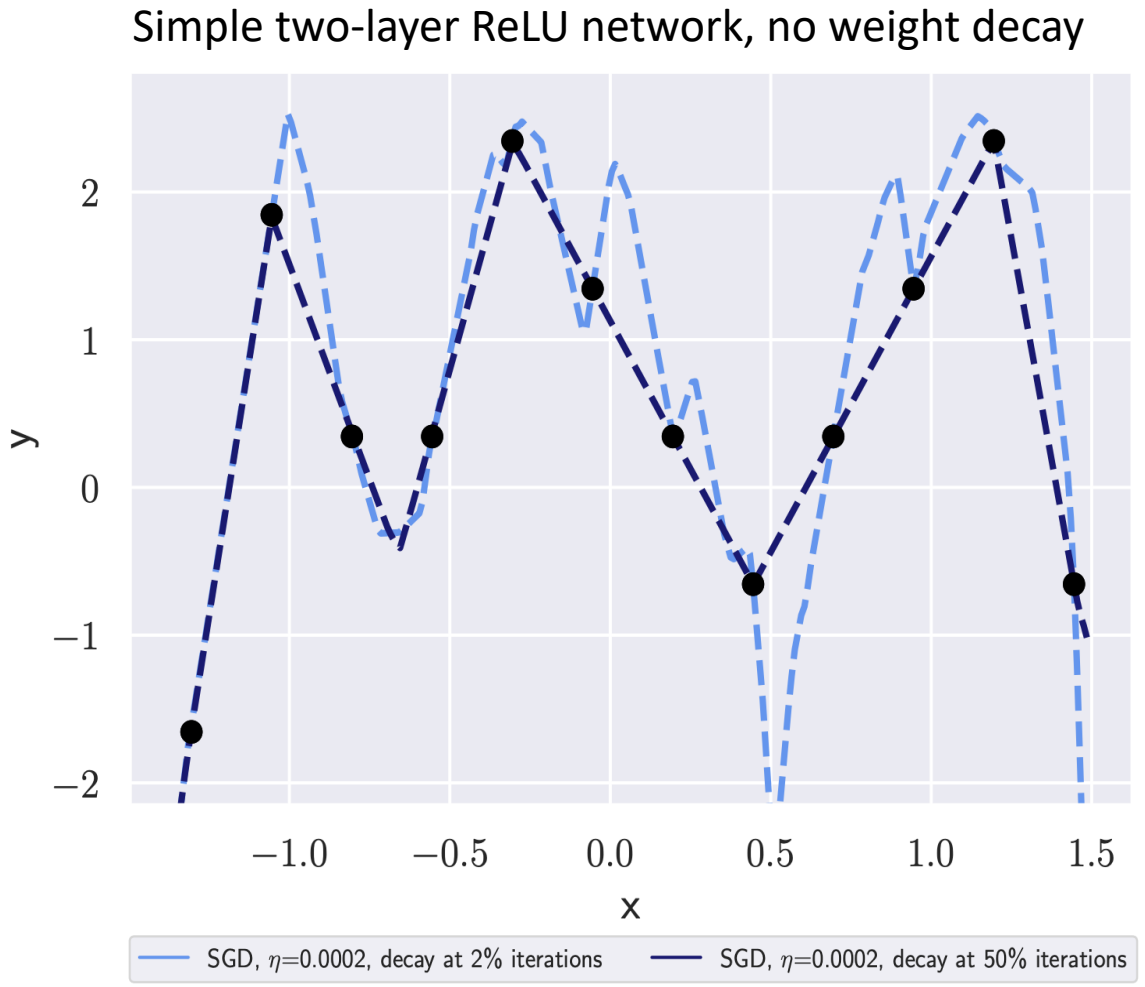


Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023

Let's start from the overparameterized case first!

Motivation: let's revisit the classical textbook picture about overfitting

- Starting point: our work [SGD with Large Step Sizes Learns Sparse Features](#) (ICML 2023)
- The nice interpolation is obtained via **the implicit regularization of SGD**
- So Occam's razor is already there **even without** any explicit regularization!
- Why do we need **weight decay** (or any other explicit regularization) then?



Implicit regularization vs. weight decay on linear models

- Let's go back to **ML 101**: linear models!

- We need weight decay to get soft-margin SVM

Hard-margin SVM

$$\begin{aligned} \underset{\mathbf{w},\, b}{\text{minimize}} \quad & \|\mathbf{w}\|_2^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^\top \mathbf{x}_i - b) \geq 1 \quad \forall i \in \{1, \ldots, n\} \end{aligned}$$

Soft-margin SVM

$$\lambda \|\mathbf{w}\|^2 + \left[\frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^\mathsf{T} \mathbf{x}_i - b)\right) \right]$$

- But even without WD, we get to the same solution due to the implicit bias of gradient descent!

- More formally in **Soudry et al.**: using logistic loss on separable data, the predictor converges to the direction of the max-margin solution

The Implicit Bias of Gradient Descent on Separable Data

Daniel Soudry DANIEL.SOUDRY@GMAIL.COM
Elad Hoffer ELAD.HOFFER@GMAIL.COM
Mor Shpigel Nacson MOR.SHPIGEL@GMAIL.COM
Department of Electrical Engineering,Technion
Haifa, 320003, Israel

Suriya Gunasekar SURIYA@TTIC.EDU
Nathan Srebro NATI@TTIC.EDU
Toyota Technological Institute at Chicago
Chicago, Illinois 60637, USA

Editor: Leon Bottou

Abstract

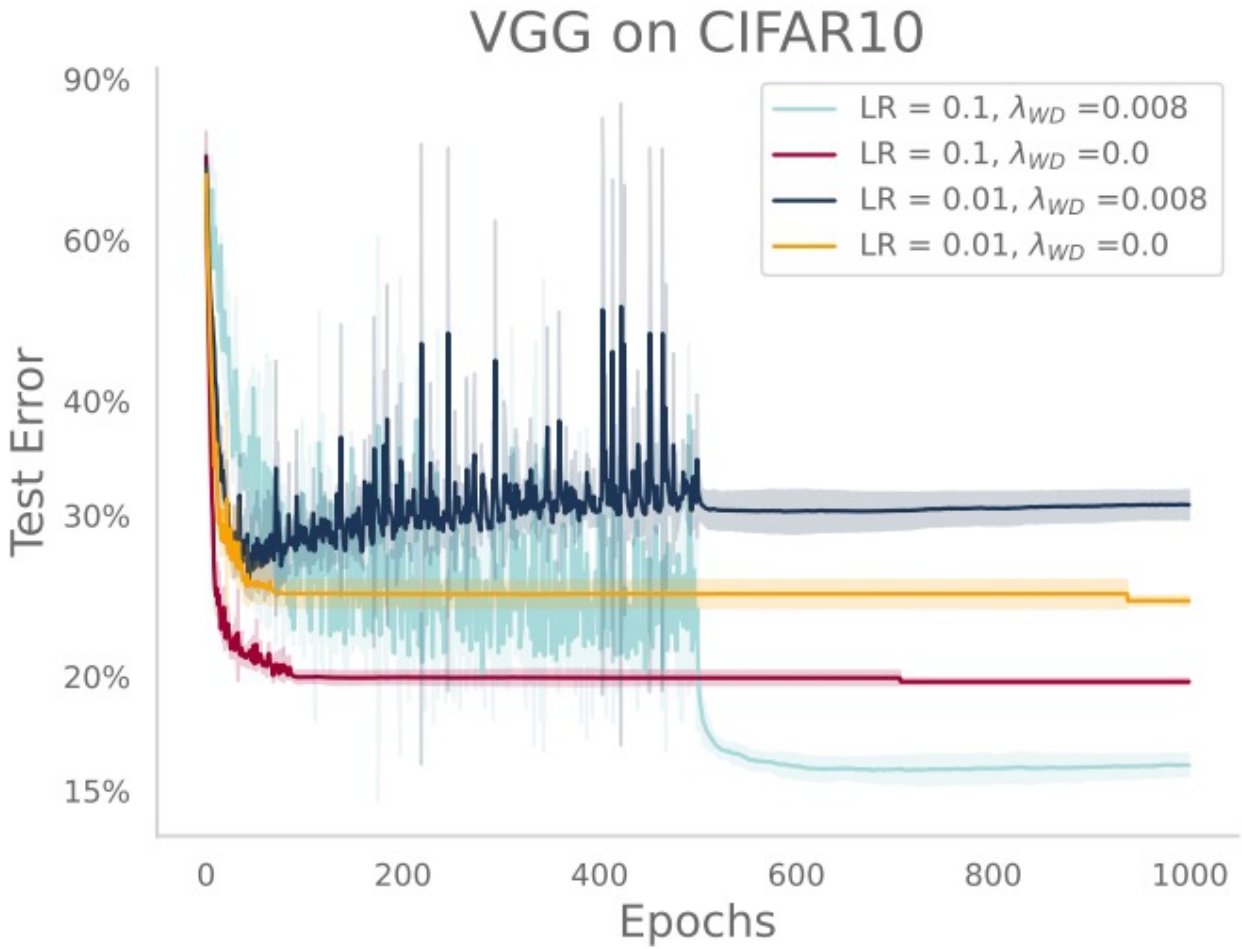
We examine gradient descent on unregularized logistic regression problems, with homogeneous linear predictors on linearly separable datasets. We show the predictor converges to the direction of the max-margin (hard margin SVM) solution. The result also generalizes to other monotone decreasing loss functions with an infimum at infinity, to multi-class problems, and to training a weight layer in a deep network in a certain restricted setting. Furthermore, we show this convergence is very slow, and only logarithmic in the convergence of the loss itself. This can help explain the benefit of continuing to optimize the logistic or cross-entropy loss even after the training error is zero and the training loss is extremely small, and, as we show, even if the validation loss increases. Our methodology can also aid in understanding implicit regularization in more complex models and with other optimization methods.

Keywords: gradient descent, implicit regularization, generalization, margin, logistic regression

Weight Decay for Overparameterized Deep Networks

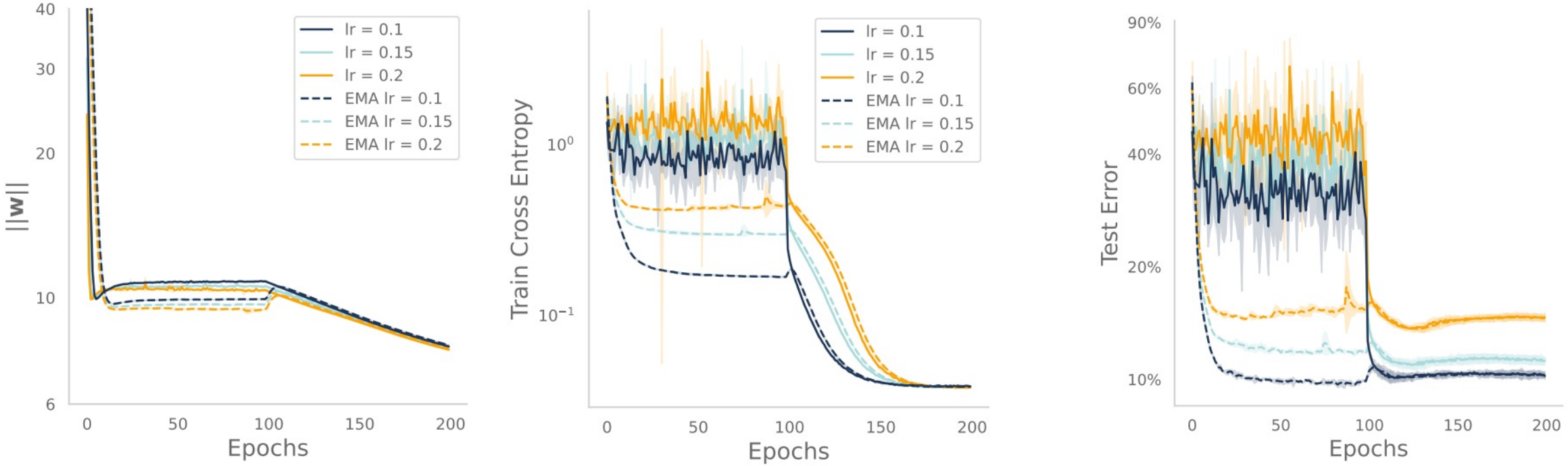
Simplest deep networks first

- We revisit a well-studied setting: VGG without BatchNorm (where the weight norm means something!) on CIFAR-10
- Interestingly, WD doesn't improve the test error on its own (even contrary in this case)!
- **Instead, WD improves only with large learning rates**
- Weight decay is not useful on its own but only in a combination with the **implicit regularization of SGD**



Setting: standard VGG, plain SGD,
no data augmentation, LR decay after 500 epochs

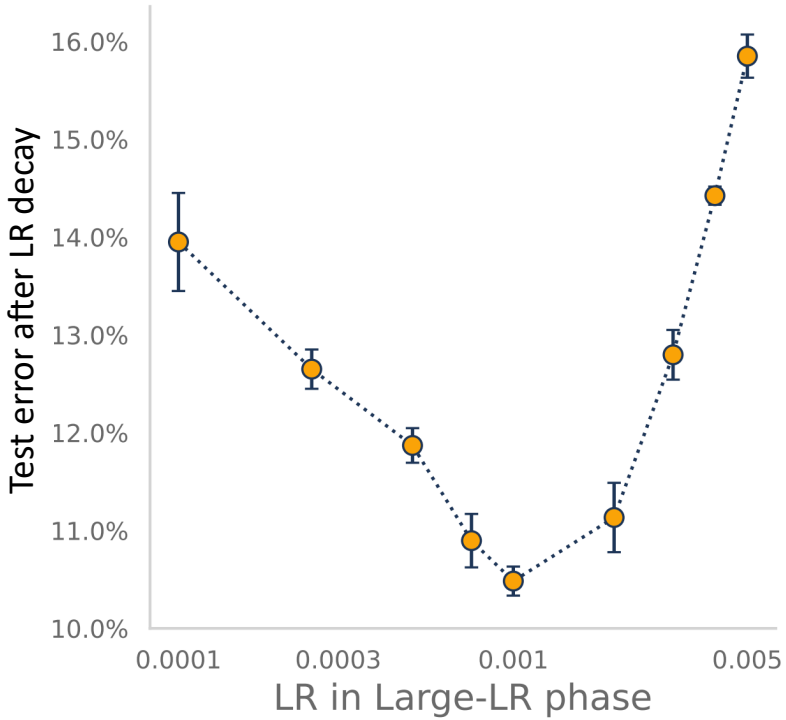
The weight norm is not at all predictive of generalization

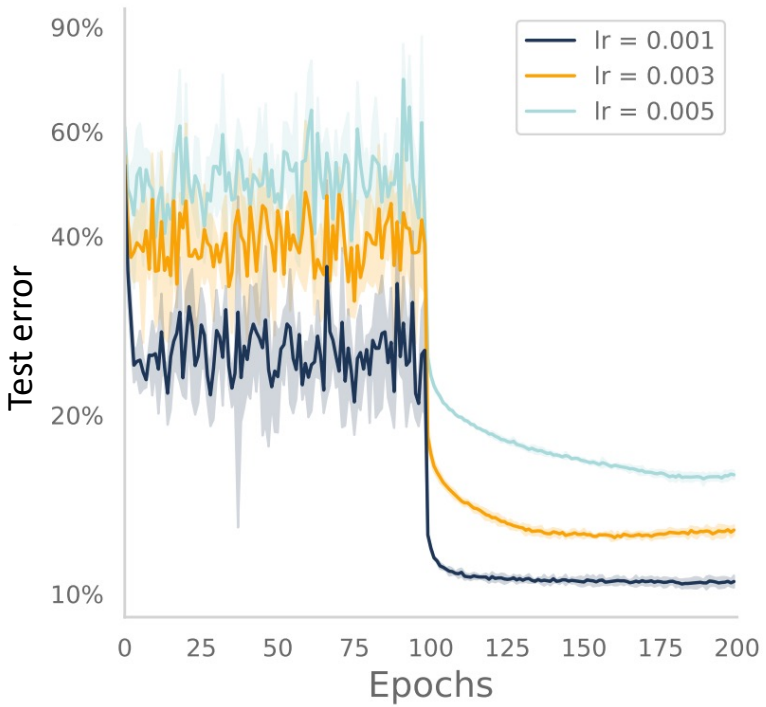


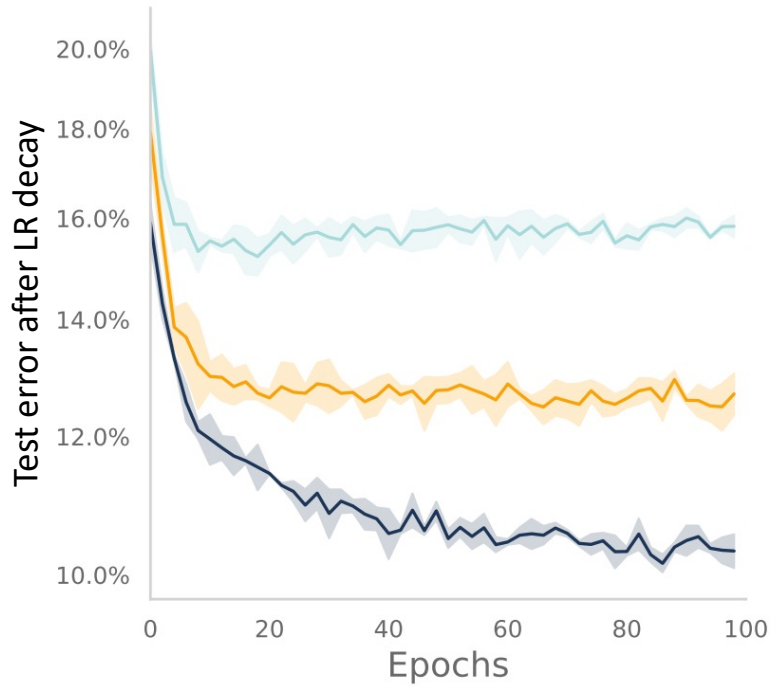
Setting: standard ResNet-18, CIFAR-10, plain SGD (EMA = exponential moving average)

- The weight norm and training loss are the same but the final test error is very different!
- However, the training dynamics is crucially affected: during the large-LR phase, WD enables **loss stabilization / equilibrium** at some level that depends on the LR and WD
- Let's try to understand the **regularization effect** of the loss stabilization phase

Too high loss stabilization is harmful







Setting: scale-invariant ResNet-18 on the sphere, CIFAR-10, plain SGD

- **Note:** zero training error is achieved for all runs
- **Main observation**: a clear U-shape wrt the learning rate (too large LR is harmful)
- **Last plot**: although it seems like nothing is happening, the noisy process drives us to a better solution (after LR decay)

Loss stabilization amplifies the noisy dynamics of SGD

- Observation: stochastic noise vanishes when the loss → 0 since the norm of the noise directly depends on the loss (the proposition is from Wojtowytsch (2021))

Proposition 3. *Assume* $\|\mathbf{w}\| \in [a, b]$, *for any* $x \in \mathcal{D}$, $\|\nabla h(\mathbf{w}, x)\| \in [m, M]$ *holds. For* n *sufficiently large, there exists constants* c_1, c_2 *such that*

$$c_1 \mathcal{L}(\mathbf{w}) \leq \mathbb{E}\left[\|g(\mathbf{w})\|^2\right] \leq c_2 \mathcal{L}(\mathbf{w})$$

- Thus, **loss stabilization** helps to prevent the stochastic noise from vanishing and drive the noisy dynamics for longer

- This noisy dynamics is beneficial for generalization but what's its effect?

Understanding the regularization effect of loss stabilization

- **Observation**: the shape of the covariance of the stochastic gradients, when the labels are injected with Gaussian noise, also matches the shape of the Hessian
- This crucial observation is used in several works (Damian et al. (2021); Pillaud-Vivien et al. (2022)) to show that the SGD trajectory closely tracks the solution of a **regularized problem**
- Inspired by these theoretical results, we conjecture a similar regularized process **but for the large-LR phase of SGD** and without label noise:

standard SGD + WD update

Conjecture 2. *Consider the algorithm in Eq. 2 with* $\mathbf{w}_0$ *initialized from a distribution* $\mu_0\left(\mathbb{R}^{(p)}\right)$. *For any input* x, *let* $\mathbf{w}_t, h(\mathbf{w}_t, x)$ *be the random variables that denote the iterate at time* t *and its functional value. The stochastic process* $(h(\mathbf{w}_t, x))_{t\in\mathbb{N}}$ *converges to the stationary distribution* $\mu^{\infty}_{\eta,\lambda}(x)$ *with mean* $\bar{\mu}_{\eta,\lambda}(x)$ *for which the following property holds,*

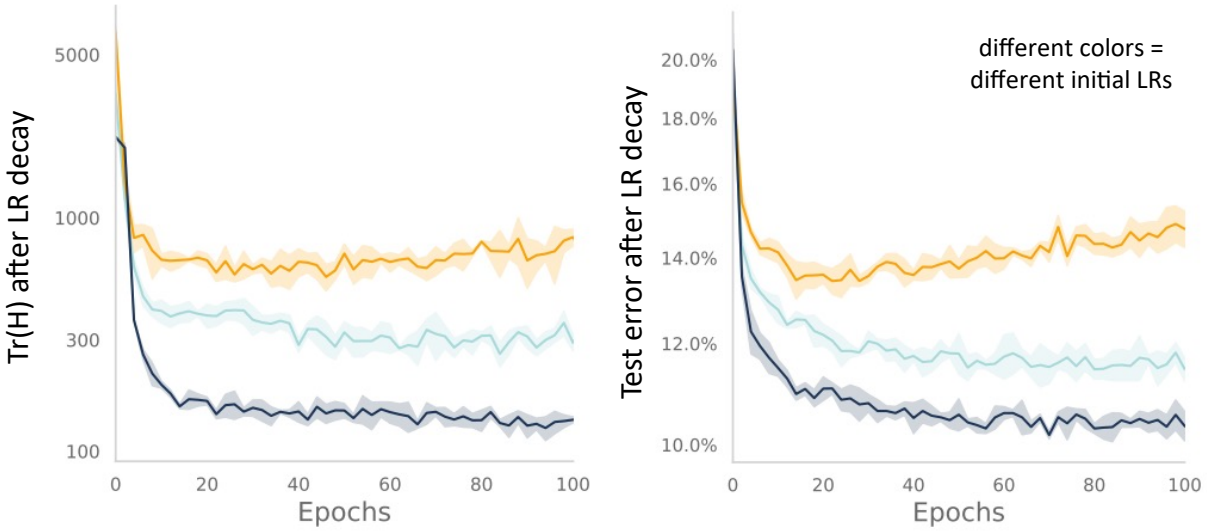
to get rid of stochasticity

$$\bar{\mu}_{\eta,\lambda}(x) = h\left(\mathbf{w}^*_{\eta,\lambda}, x\right), \text{ where } \mathbf{w}^*_{\eta,\lambda} := \arg\min_{\mathbf{w}\in\mathbb{R}^p} \mathcal{L}_\lambda(\mathbf{w}) + \eta\sigma^2_{\eta,\lambda}\,\mathrm{Tr}\left(\nabla^2\mathcal{L}(\mathbf{w})\right). \quad (5)$$

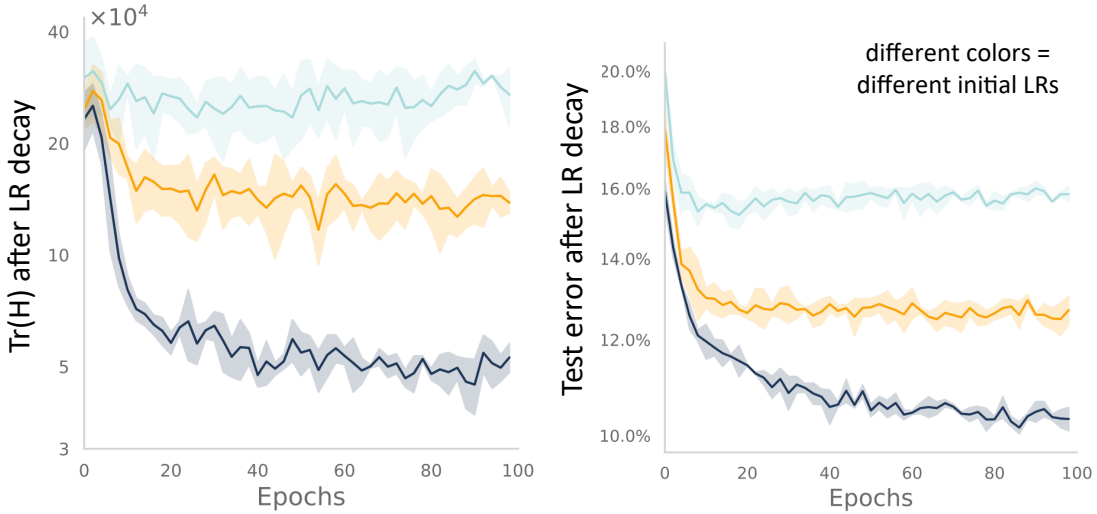
main point

Testing the conjecture empirically

Standard ResNets



Scale-invariant ResNets on the sphere



- **Tr(∇^2) after LR decay ($\approx$ taking the mean of the process) decreases along the trajectory** & closely mirrors the test error (not to say that Tr(∇^2) is *always* predictive of generalization!)
- We believe our conjecture holds *generally* since we see the same effect of WD for VGGs, standard ResNets, and scale-invariant ResNets on the sphere
- **Takeaway**: contrary to the classical understanding of WD, its regularization effect is more subtle and comes from the interaction with the implicit bias of (S)GD!

Weight Decay for Large Language Models

Now let's switch to LLMs!

- GPT-3 paper: *"All models use weight decay of 0.1 to provide a small amount of regularization"*
- Subsequently, all major LLMs (Chinchilla, Llama, PALM) just followed exactly the same training recipe
- **Is regularization really what's happening?** We can't train GPT-3 ourselves :(are we out of the game?
- Let's try to see how far we can go with a **GPT-2 model** with 124M parameters on OpenWebText (only 1 A100 + 24h of training is sufficient)!
- The training and validation losses remain **very close** across different WD values (≈0 generalization gap) ⇒ **WD is not about regularization here**

GPT-2-124M on OpenWebText

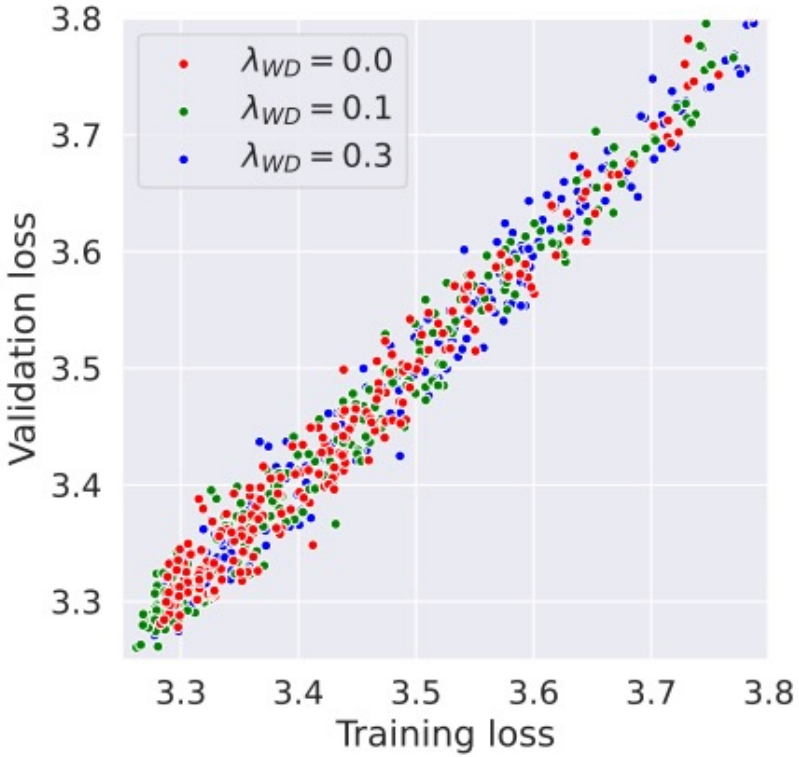


Figure 4: The validation loss is determined by the training loss and not influenced by λ_{WD}.

The Chinchilla observation

- Training Compute-Optimal Large Language Models (DeepMind, NeurIPS 2022) introduced a family of models called *"Chinchilla"*

- There is a very curious observation about the effect of weight decay

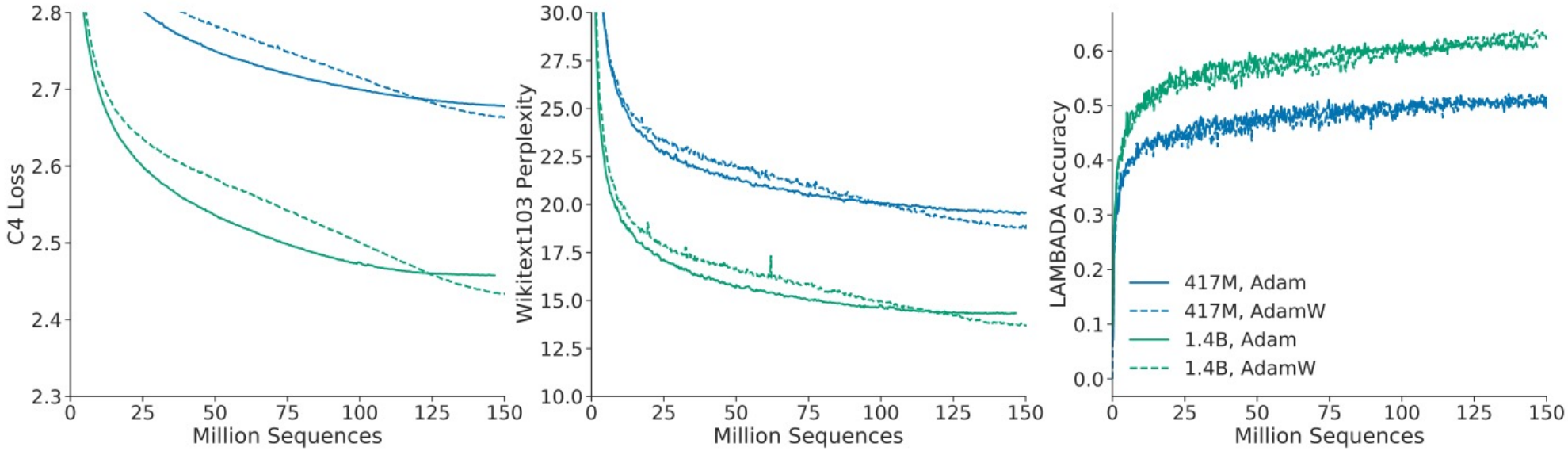
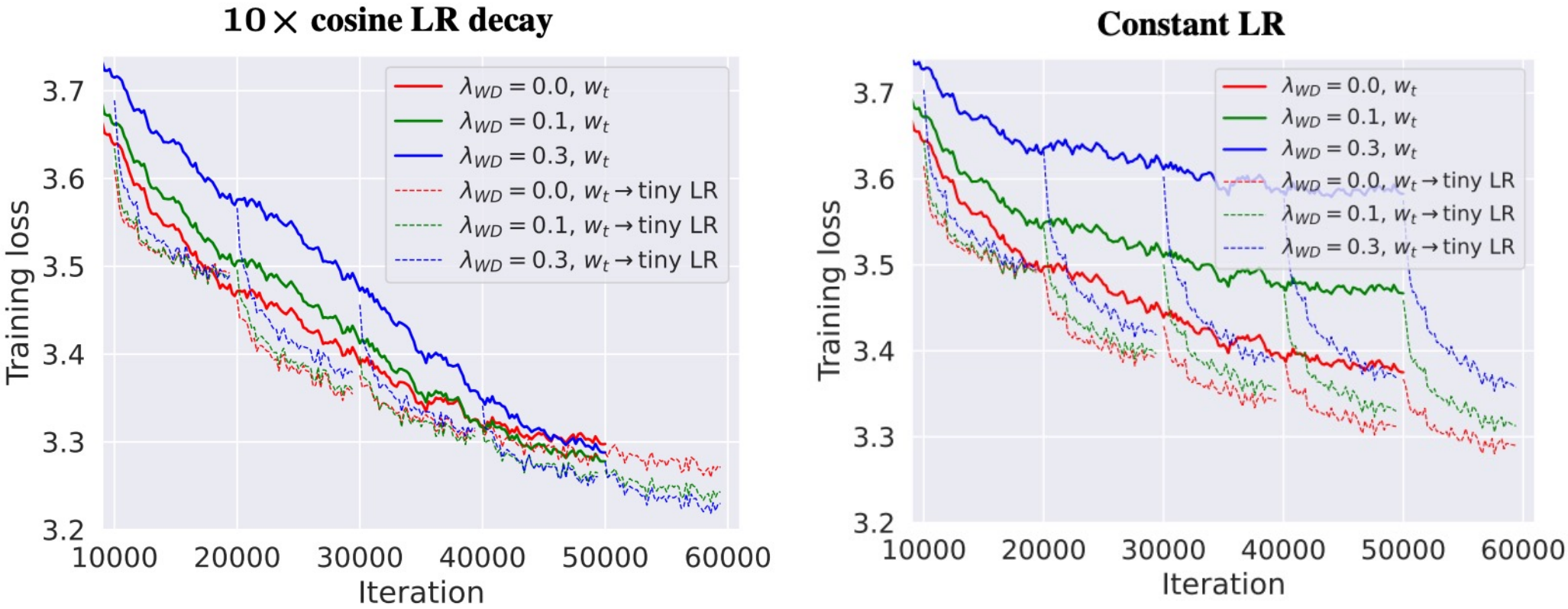


Figure A7 | **Adam vs AdamW.** For a 417M (blue) and 1.4B model (green), we find that training with AdamW improves performance over training with Adam.

- But they don't provide any understanding about this phenomenon

Reproducing the Chinchilla phenomenon at a smaller scale



- We could reproduce this with the academic-friendly setting of GPT-2-124M
- Interestingly, we observe that a higher loss with WD can still be a better point **if we drop the LR**
- We see resemblance of loss stabilization which is, however, not useful in this setting

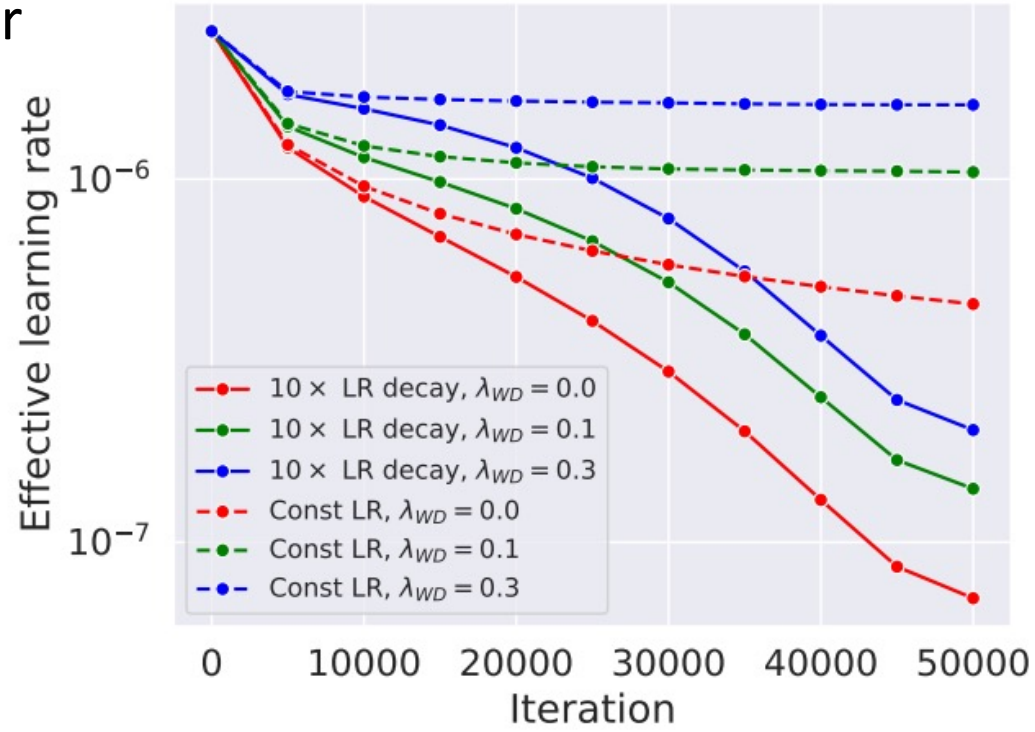
Bias-variance tradeoff in stochastic optimization

- Let's try to understand it via a simple model: linear regression with squared loss:

$$\text{Excess Risk} \lesssim \underbrace{(1-\eta\mu)^t \left\|x_0 - x_*\right\|^2}_{\text{bias}} + \underbrace{\eta\sigma^2}_{\text{variance}}$$

- **Our explanation**: WD better balances the bias-variance optimization trade-off via larger initial effective LRs

- However, since the noise term also scales with the LR, **we don't see the whole picture from the loss alone**



Effective LR: $\eta_{eff} \propto \frac{\eta_t}{||w_t||_2}$

(justified for scale-invariant networks)

- So it becomes even more important to reduce the LR which happens only closer to the middle/end with the cosine LR schedule!

Weight decay prevents divergences with bfloat16

- While previous works (e.g., BLOOM) documented that **float16** leads to divergences, usually switching to **bfloat16** resolves it (*remark: but all these training details are very poorly documented in the LLM community!*)

- However, you still need to combine **bfloat16** with WD (in many cases)

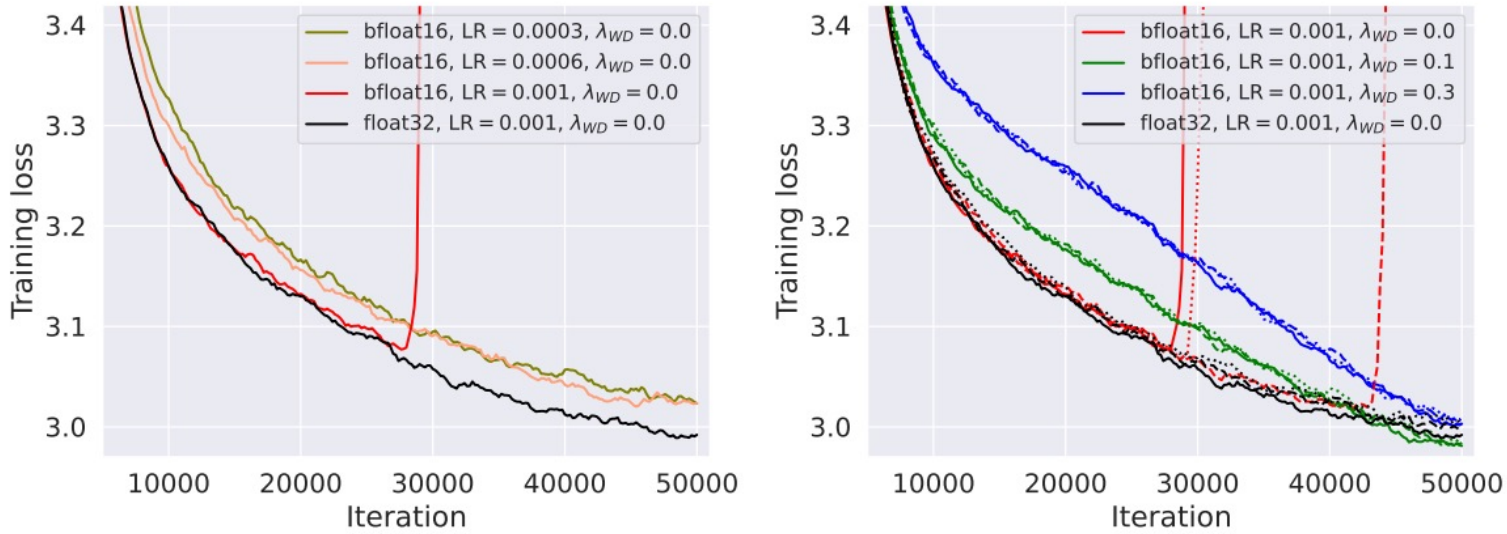
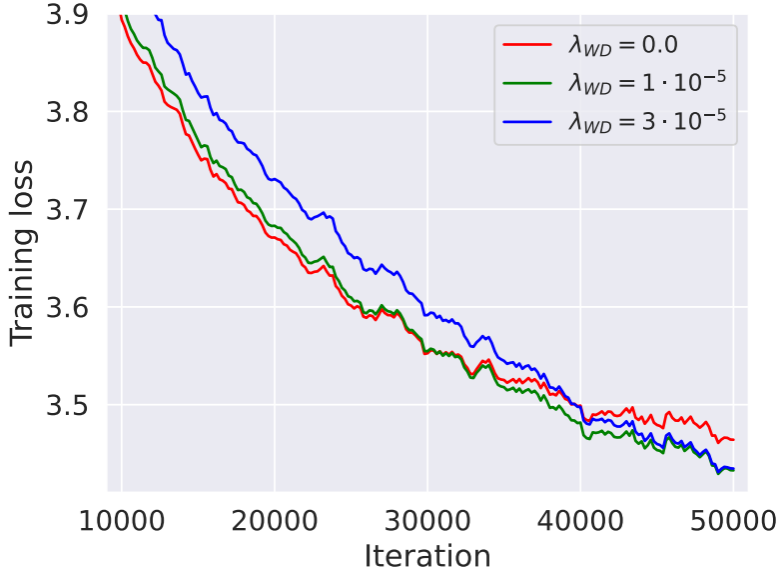


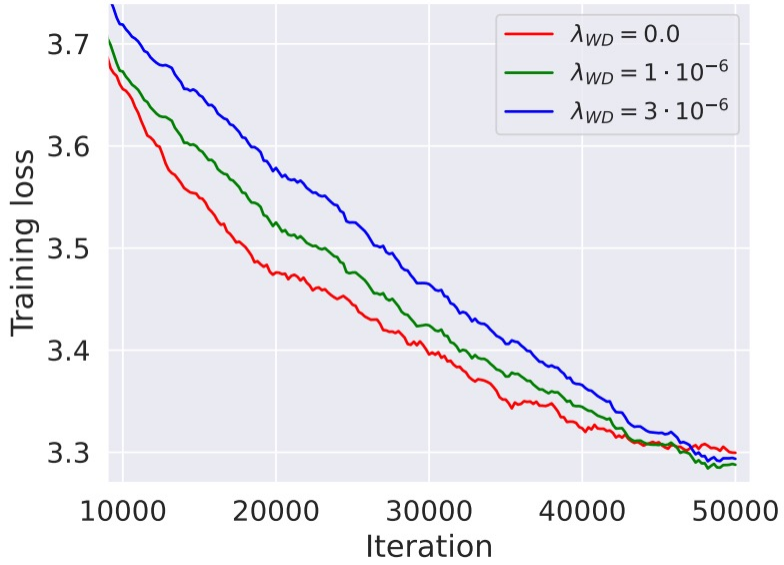
Figure 7: **GPT-2-124M on OpenWebText with context length 1024. Left:** The model trained with a moderate LR 0.001 diverges for `bfloat16` but not for `float32`; lowering the LR prevents the divergence but leads to a worse loss. **Right:** Weight decay prevents divergence for LR= 0.001 and enables `bfloat16` training (the three random seeds are denoted with —, - - -, ··· lines).

Ablation studies for weight decay in LLMs

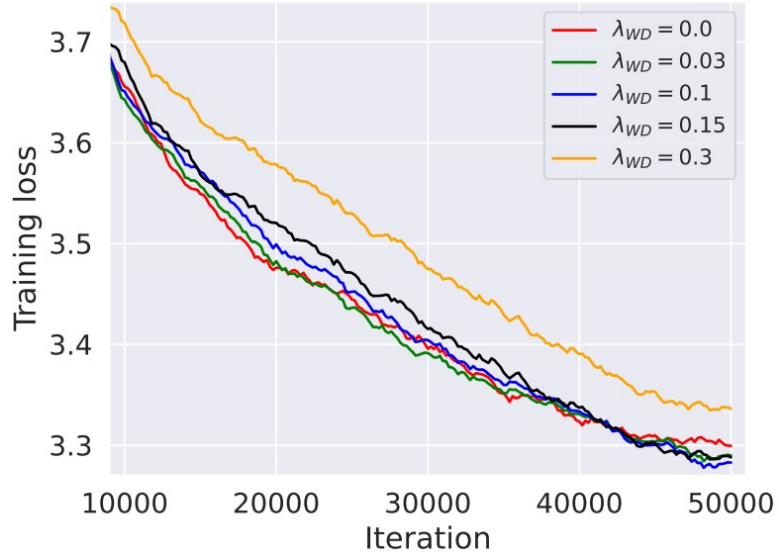
SGD + Momentum



Adam + L2 regularization



WD on all parameters



- Same picture for SGD+Momentum and Adam+L2 as for AdamW
- Omitting LayerNorm parameters from WD is important, but only for higher WD values
- Weight averaging can work as a nearly zero-cost proxy to see if we are *"variance-bounded"* (see Appendix)

Conclusions and Takeaways

Conclusions and takeaways

- It's totally remarkable that there are at least **three distinct mechanisms of WD**
  1. regularization when paired with stochastic noise,
  2. enhancing optimization of the training loss,
  3. ensuring stability of low-precision training.

- Interestingly, AdamW was introduced only as a better regularization method and now every LLM cites it **without critical reassessment of its effect**!

- **Our intuition**: decoupling WD is likely to be not necessary, it just eases the hyperparameter tuning

- In modern deep learning, WD is rarely useful as an explicit regularizer but instead its adoption is **due to its effect on the training dynamics**

Thanks for your attention!