SGD with large step sizes learns sparse features




Maksym Andriushchenko | Aditya Varre | Loucas Pillaud-Vivien | Nicolas Flammarion

École Polytechnique Fédérale de Lausanne

ICLR 2023 Workshop on Sparsity in Neural Networks + under review

12 April 2023

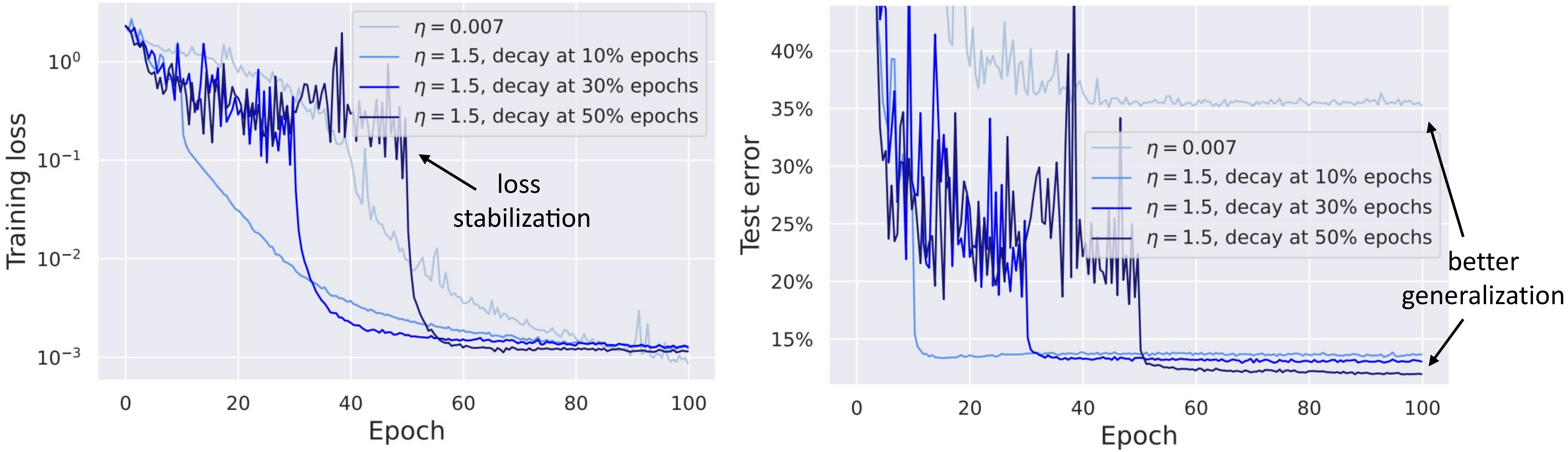
Deep Learning and Optimization Seminar, Westlake University

EPFL

Let's start from a well-known observation

Longer schedules of large step size SGD lead to better generalization

Setting: ResNet-18 on CIFAR-10, standard mini-batch SGD, no data augmentation

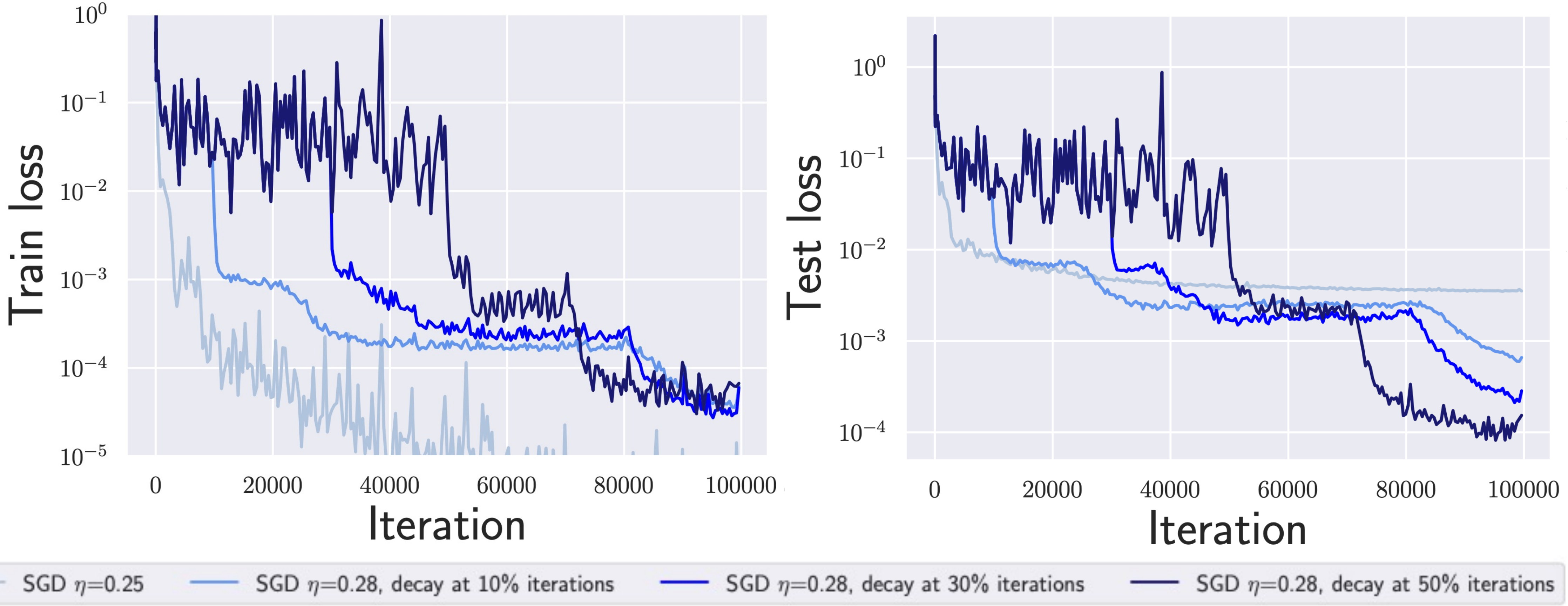


This raises multiple questions:

1. Why does the training loss stabilizes?
2. What kind of hidden dynamics is happening in this phase?
3. How is it related to sparsity of the predictor?

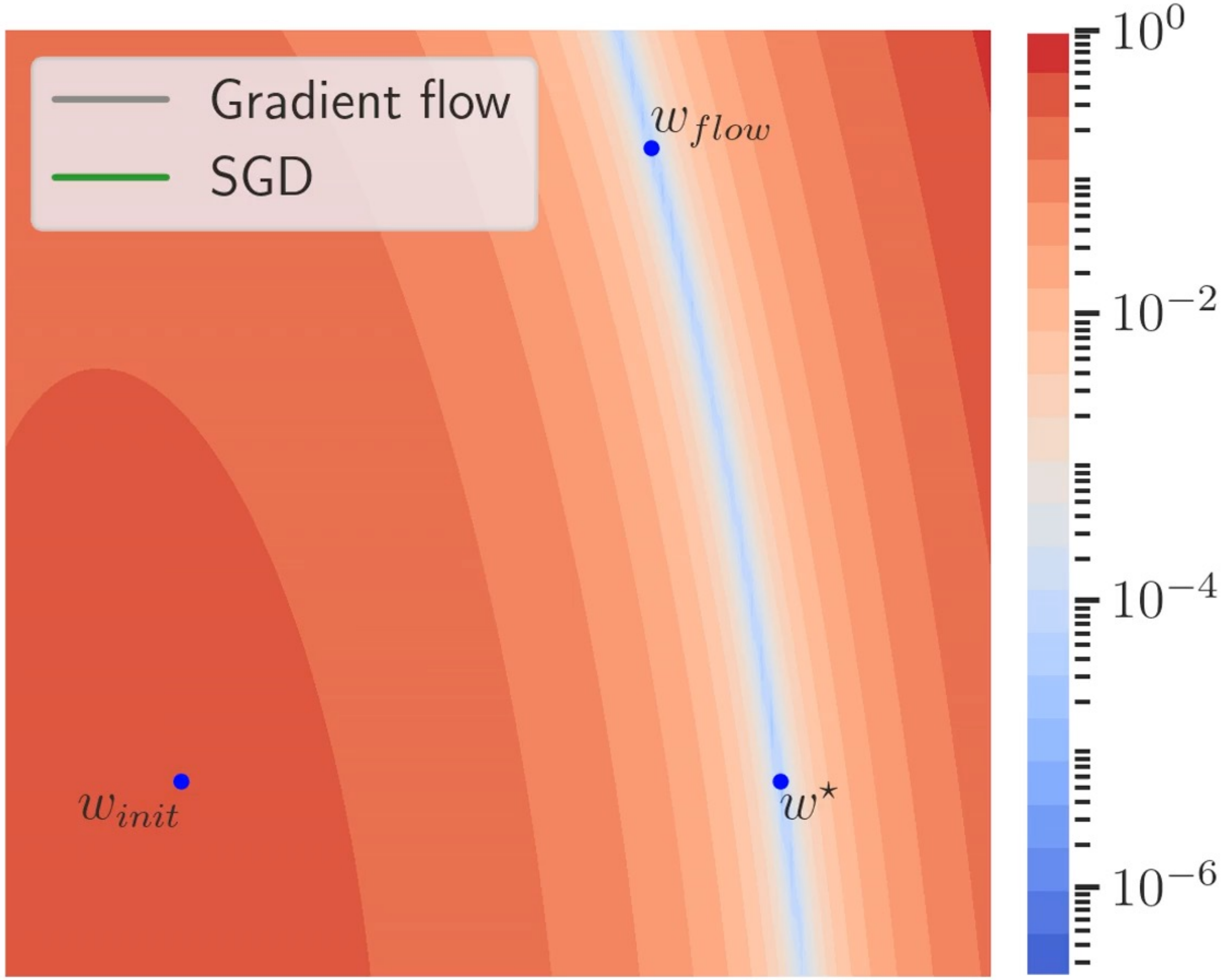
Is this a phenomenon inherent to deep networks? No!

The same observations can be seen even on the **simplest diagonal linear networks** $h(x) = \langle x, u \odot v \rangle$ for a sparse regression problem



$\Rightarrow$ we can try to understand the phenomenon **theoretically** by leveraging prior works on this toy model

How can the training loss stabilize around some level set?



Setting: a 2-D slice of the 60-D training loss surface of a diagonal linear network

General picture:

- Initially, the training loss decreases
- Then, due to the noise, SGD cannot enter the narrow valley and keep oscillating (⇒ *no convergence but also no divergence*)
- We formalize it in the paper with a proposition that describes how this can occur **provably** for a 1D diagonal linear net
- In addition, it's apparent that SGD slowly moves to a certain direction. **Can we better understand that?**

Modelling SGD with a Stochastic Differential Equation (Part I)

- **Observation:** the noise intensity of SGD is proportional to the training loss
  $\Rightarrow$ when the loss stabilizes, we can assume **constant noise intensity**

- Thus, we can model the **large step size SGD phase** with the following **constant-noise SDE**:

step size

constant noise intensity
due to loss stabilization

Constant-noise SDE: $$\mathrm{d}\theta_t = -\nabla_\theta \mathcal{L}(\theta_t)\mathrm{d}t + \sqrt{\eta\delta}\,\underbrace{\phi_{\theta_t}(X)^\top}_{\text{the Jacobian of the network } [\nabla_\theta h_\theta(x_i)^\top]_{i=1}^n \in \mathbb{R}^{n\times p}}\mathrm{d}B_t$$

Brownian motion in $\mathbb{R}^n$,
i.e., Gaussian noise

- We check empirically that this **SDE fully agrees with SGD** in terms of the generalization improvements and other key metrics (*we'll see these experiments later*)

- This SDE can be seen as the effective **slow dynamics** (due to the gradient + the noise term) that drives the θ_t while they bounce rapidly due to the noise (**fast dynamics**)

Modelling SGD with a Stochastic Differential Equation (Part II)

Constant-noise SDE: $\mathrm{d}\theta_t = -\nabla_\theta \mathcal{L}(\theta_t)\mathrm{d}t + \sqrt{\eta\delta}\, \phi_{\theta_t}(X)^\top \mathrm{d}B_t$

- **Prior works**: for diagonal linear networks, Pillaud-Vivien et al. (COLT 2022) proved the **sparsity of the solution** using a similar SDE derived for **label noise SGD**
- **Our work**: we conjecture that for arbitrary deep networks, a **similar sparsifying effect** is taking place for **standard SGD with large step sizes** (no label noise needed)
  - **Observation**: for the Brownian motion $\mathrm{d}B_t \in \mathbb{R}^n$: $\phi_{\theta_t}(x_i)^\top dB_t = ||\phi_{\theta_t}(x_i)||_2 dW_t$ where $dW_t \in \mathbb{R}$ is a 1D Brownian motion (*basic property of the Gaussian distribution*)
  - Thus, the SDE resembles the **geometric Brownian motion** (Oksendal, 2013):

$$d\theta_t = \mu\theta_t dt + \delta\theta_t dW_t \quad \rightarrow \text{closed-form solution} \quad \theta_t = \theta_0 \exp((\mu - \delta^2/2)t + \sigma\, W_t)$$

  - Thus, we expect the SDE to induce a similar **shrinkage effect** for each multiplicative factor to dW_t, i.e., $||\phi_{\theta_t}(x_i)||_2$ with strength proportional to the loss stabilization level δ

Notions of sparsity for arbitrary architectures

Constant-noise SDE: $\mathrm{d}\theta_t = -\nabla_\theta \mathcal{L}(\theta_t)\mathrm{d}t + \sqrt{\eta\delta}\,\phi_{\theta_t}(X)^\top \mathrm{d}B_t$

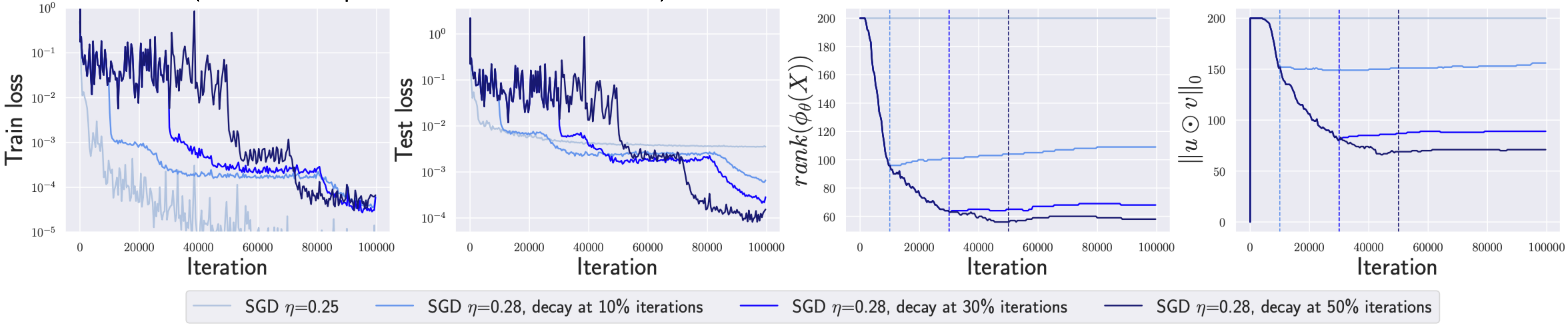
We empirically track **two quantities** related to the Jacobian $\phi_\theta(X) \in \mathbb{R}^{n\times p}$:

1. **Rank of the Jacobian** that reflects
   - how many columns collapsed completely to zero (e.g., if ReLU = 0 for all x_i)
   - how many columns are linearly dependent on others (e.g., if two ReLUs implement the same function, up to a constant rescaling)

2. **"Feature sparsity coefficient"**: the average number of distinct (we count highly-correlated neurons as one), non-zero activations
   - formally: $\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}_{g(x_i)_j>0}$ where $g(x_i) \in \mathbb{R}^m$ is the feature vector at some layer where we merge beforehand *highly correlated neurons*
   - this serves as a cheap proxy of the rank that **scales to deep networks**

Sparse feature learning for diagonal linear networks

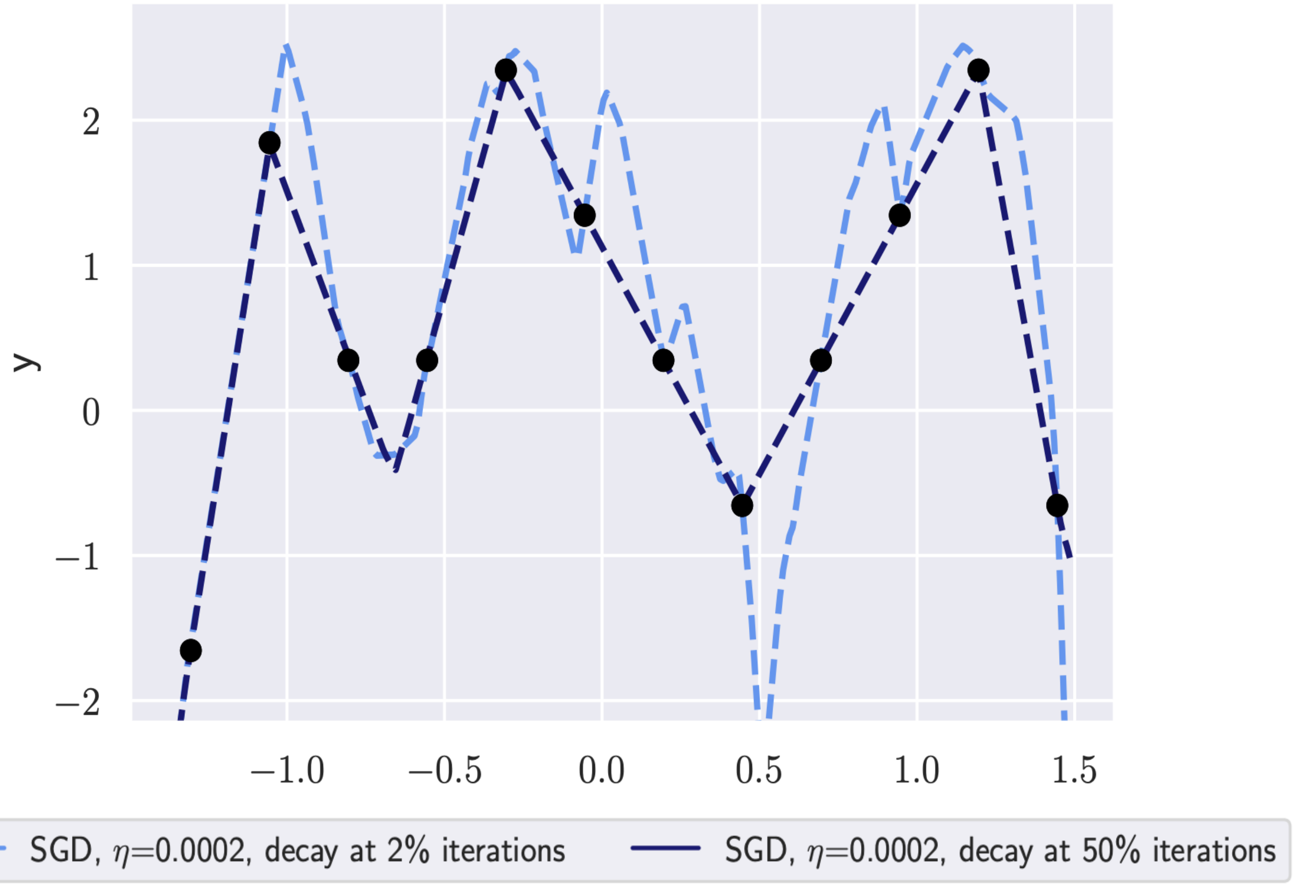
(the first two plots are the same as before) (new) (new)



- The last two plots clearly show that **sparsity is progressively achieved** in the large step size phase
- **Note**: for this task, sparsity is desirable because the ground truth vector w^* was selected to be sparse
- If there is no alignment between the ground truth and implicit bias, we don't expect to see improvements in generalization!

Sparse feature learning for a simple one-layer ReLU network

Illustration: a classical textbook picture about overfitting

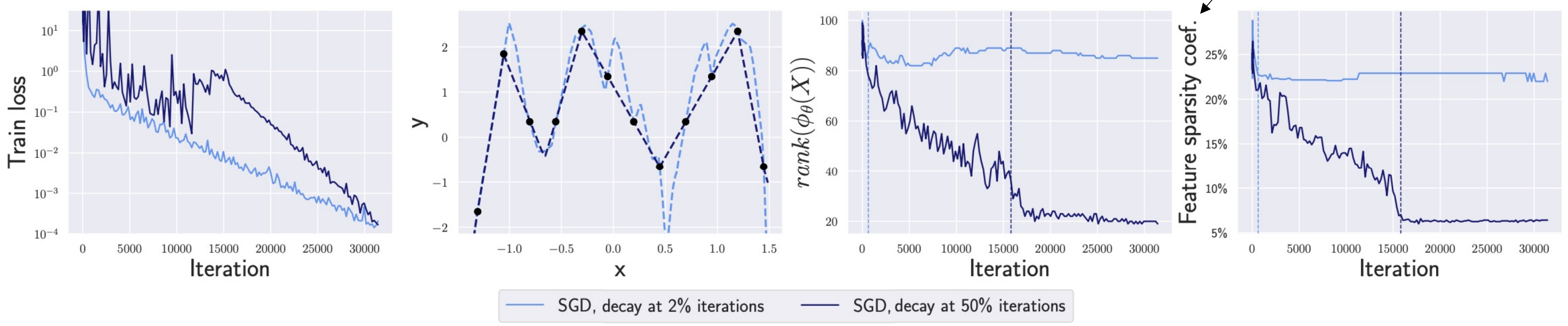


Here, however, the nice interpolation between the points
is **due to the implicit regularization effect** of large step sizes

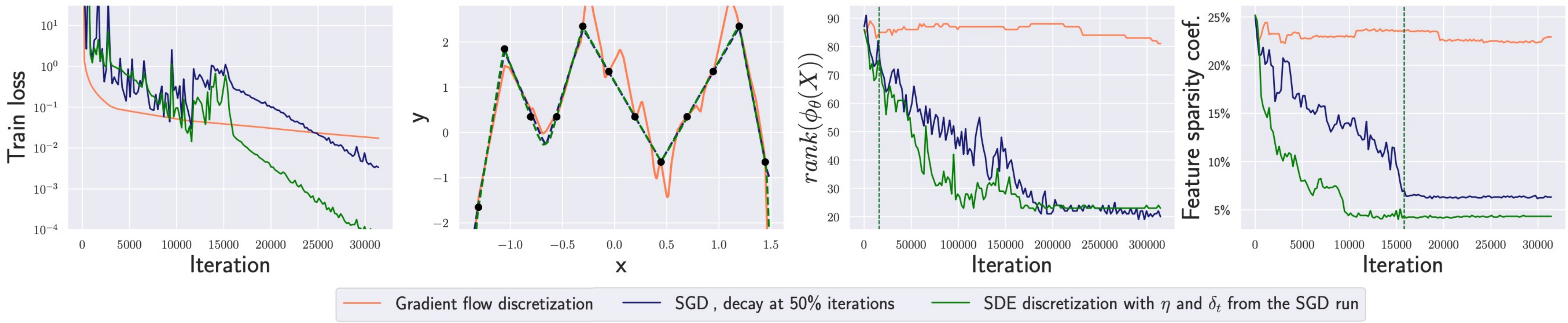
A simple one-layer ReLU network: sparsity metrics

$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}_{g(x_i)_j>0}$ for deduplicated features $g(x_i)$

Sparse feature learning



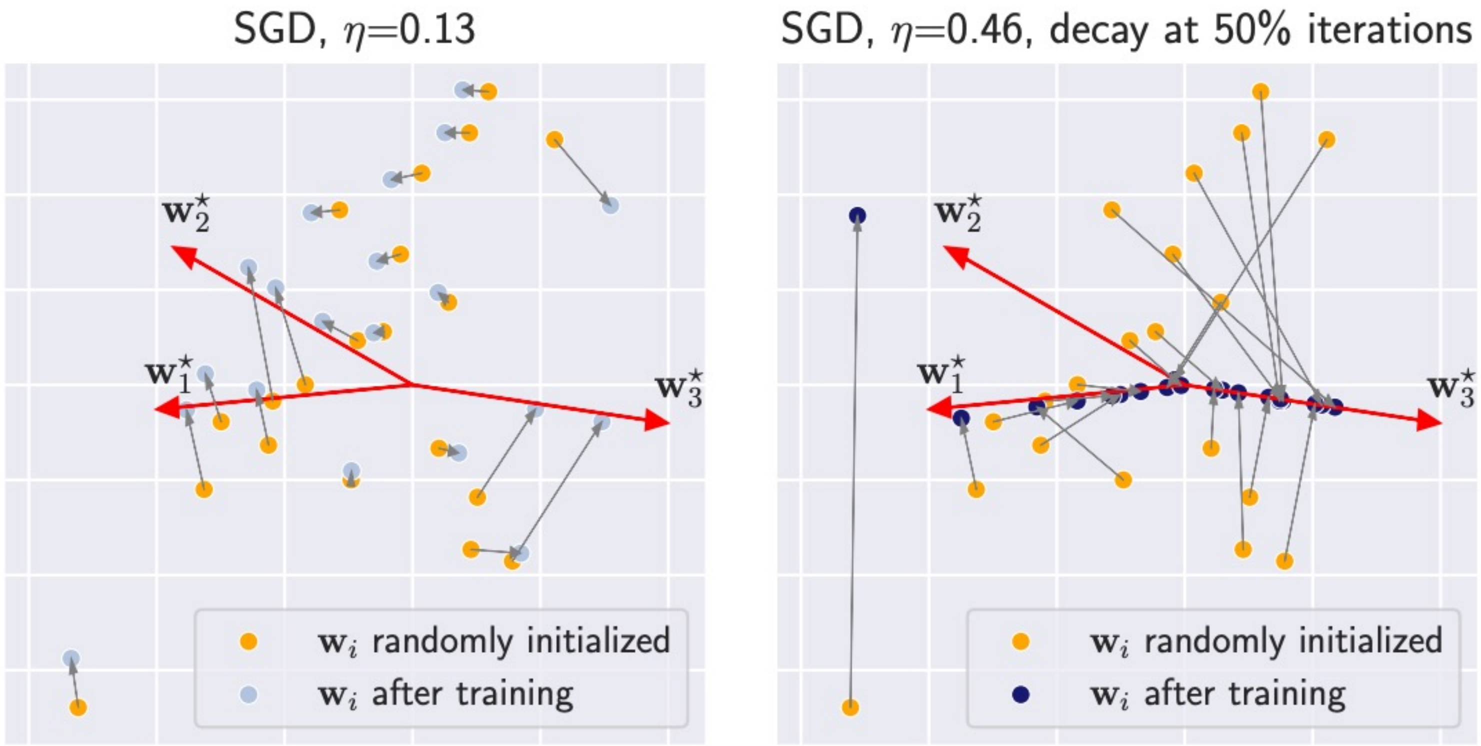
Validity of the SDE modeling



Dynamics of individual neurons in 2D

- How do the weight vectors corresponding to neurons move depending on the step size?

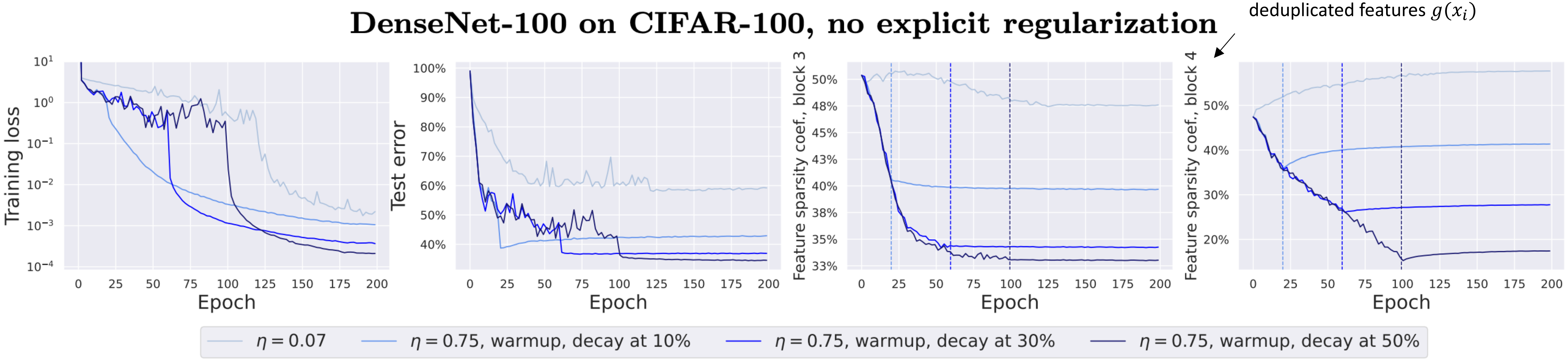
Setting: input dimension $d = 2$, teacher with 3 neurons $w_1^\star, w_2^\star, w_3^\star$, student with 20 neurons



- With small step sizes, the neurons barely move! i.e., **the network fits the data with effectively fixed random features** → not desirable
- **Sparse feature learning occurs only for large step sizes**

Sparse feature learning for deep networks (part I)

$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}_{g(x_i)_j>0}$ for deduplicated features $g(x_i)$



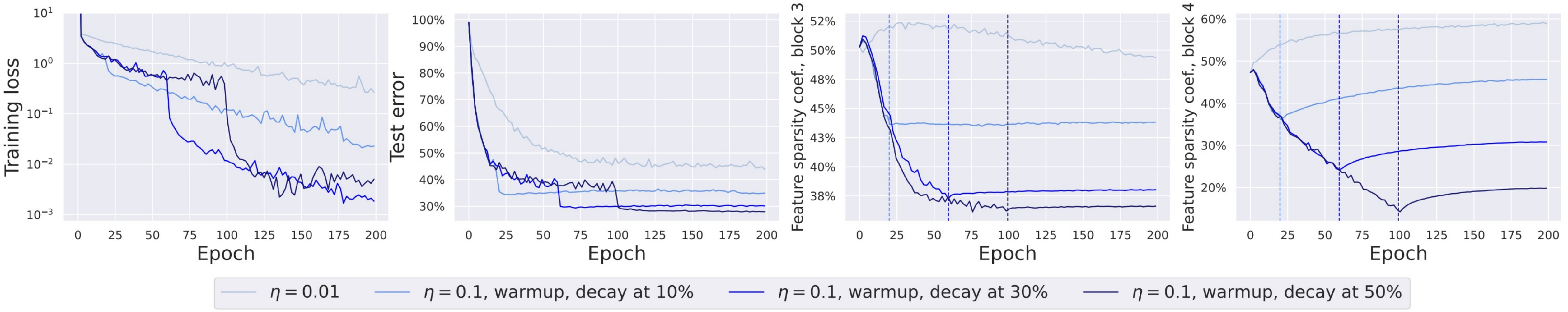
Main observations:

- Plot 1: the **training loss** stabilizes
- Plot 2: the **test error** noticeably depends on the length of the schedule
- Plots 3 & 4: the **feature sparsity coefficient** *at top layers* (blocks 3 and 4 out of 4 blocks in total) is minimized during the large step size phase

Sparse feature learning for deep networks (part 2)

$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}_{g(x_i)_j>0}$ for deduplicated features $g(x_i)$

DenseNet-100 on CIFAR-100, state-of-the-art setting



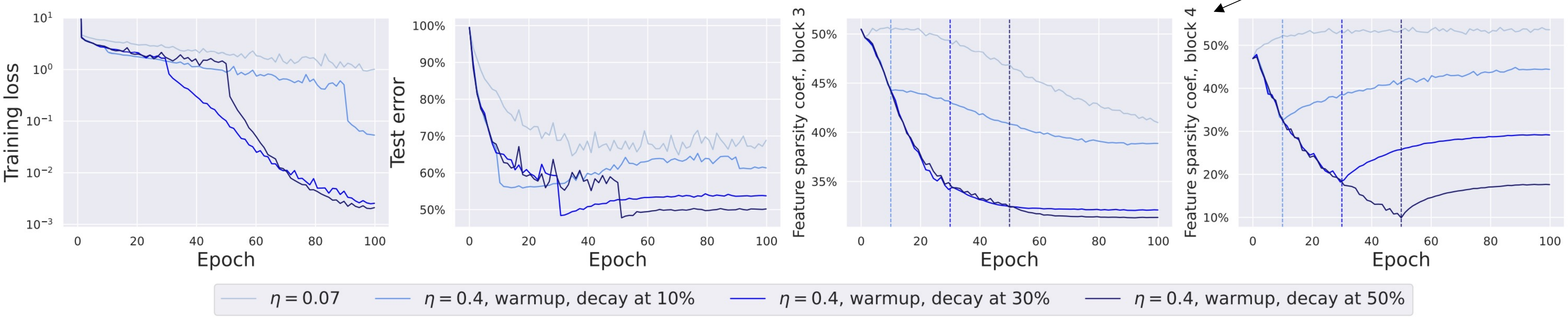
Main observations:

- Plot 1: the **training loss** stabilizes
- Plot 2: the **test error** noticeably depends on the length of the schedule
- Plots 3 & 4: the **feature sparsity coefficient** *at top layers* (blocks 3 and 4) is minimized during the large step size phase

Sparse feature learning for deep networks (part 3)

$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{m}\sum_{j=1}^{m}\mathbf{1}_{g(x_i)_j>0}$ for deduplicated features $g(x_i)$

DenseNet-100 on Tiny ImageNet, no explicit regularization



Main observations:

- Plot 1: the **training loss** stabilizes
- Plot 2: the **test error** noticeably depends on the length of the schedule
- Plots 3 & 4: the **feature sparsity coefficient** *at top layers* (blocks 3 and 4) is minimized during the large step size phase

Conclusions and takeaways

- **Our picture**: SGD noise drives the iterates to a sparse solution which we observe on many models (from **diagonal linear networks** to **ResNets on CIFAR-100**)
- Sparse features are very often (but surely not always) beneficial for generalization
- We can get them for free via the SGD dynamics **if we don't converge too early**
- Is stochasticity necessary in general? Surely not, the same dynamics is likely to be achieved via different means but **with SGD we get this effect "for free"** unlike, e.g., for gradient/Jacobian regularizers

Thanks for your attention!

Happy to answer your questions and chat more :)