OPT2023: 15th Annual Workshop on Optimization for Machine Learning

Understanding the Role of Optimization in Double Descent

Chris Yuhao Liu YLIU298@UCSC.EDU
Department of Computer Science and Engineering
University of California, Santa Cruz
Santa Cruz, CA, USA

Jeffrey Flanigan JMFLANIG@UCSC.EDU
Department of Computer Science and Engineering
University of California, Santa Cruz
Santa Cruz, CA, USA

Abstract

The phenomenon of model-wise double descent, where the test error peaks and then reduces as the model size increases, is an interesting topic that has attracted the attention of researchers due to the striking observed gap between theory and practice [2]. Additionally, while double descent has been observed in various tasks and architectures, the peak of double descent can sometimes be noticeably absent or diminished, even without explicit regularization, such as weight decay and early stopping. In this paper, we investigate this intriguing phenomenon from the optimization perspective and propose a simple optimization-based explanation for why double descent sometimes occurs weakly or not at all. To the best of our knowledge, we are the first to demonstrate that many disparate factors contributing to model-wise double descent (initialization, normalization, batch size, learning rate, optimization algorithm) are unified from the viewpoint of optimization: model-wise double descent is observed if and only if the optimizer can find a sufficiently low-loss minimum. These factors directly affect the condition number of the optimization problem or the optimizer and thus affect the final minimum found by the optimizer, reducing or increasing the height of the double descent peak. We conduct a series of controlled experiments on random feature models and two-layer neural networks under various optimization settings, demonstrating this optimization-based unified view. Our results suggest the following implication: Double descent is unlikely to be a problem for real-world machine learning setups. Additionally, our results help explain the gap between weak double descent peaks in practice and strong peaks observable in carefully designed setups.

1. Introduction

The phenomenon of double descent [2, 40], where the generalization error first decreases, increases, and then decreases again as the model size P surpasses the dataset size N (the interpolation threshold), has attracted a lot of attention from the machine learning community. This unexpected behavior calls into question the traditional understanding of generalization in both under- and over-parameterized regimes. Over the past few years, the double descent phenomenon has been observed in various models [5, 16, 40, 51] and learning paradigms [8, 9, 21, 40, 50]. Previous research has contributed to our understanding of the double descent phenomenon in various contexts and from different perspectives, including bias-variance trade-off tools [11, 57], VC theory [34], condition numbers [32, 52] (the ratio between the maximum and the minimum singular values of the data matrix for linear regression), and aspects of optimization [17, 18, 29]. Although double descent is ubiquitous

across many machine learning and deep learning setups, it is not always observed [1, 4, 10, 24, 54], and depends heavily on various factors [12, 19, 30, 35, 40, 47, 48].

Many factors contribute to double descent occurring or not occurring in any given deep learning setup. These include the initialization, learning rate, scale of features, normalization, batch size, and choice of optimization algorithm. For the first time, we demonstrate that the effect of all these disparate factors is unified into a single phenomenon from the viewpoint of optimization: double descent is observed if and only if the given optimizer[1] can find a sufficiently low-loss minimum. For hyper-parameters affecting the underlying optimization problem, the condition number (the ratio between the largest and the smallest singular values of the feature matrix) is affected, which in turn affects the minimum being found by the optimizer. For hyper-parameters affecting the optimizer directly, those that lead to a worse minimum reduce the peak of double descent. Although, in hindsight, these results are intuitive and almost obvious, we are unaware of any prior work connecting these phenomena in this simple manner.

Our results on simple random feature models and two-layer neural networks indicate that classifiers in practice are unlikely to exhibit the peaking phenomenon. First, inductive biases and hyperparameters are usually chosen carefully using a validation set to prevent overfitting so that no model is over-trained. Second, even if a model overfits, many iterations are required for critically parameterized models to exhibit a strong double descent curve (Section 6). Therefore, realistic setups already act as a "natural" mitigator of double descent. While no existing theoretical framework explains all our results, we believe our results are particularly useful for suggesting avenues for further theoretical study.

2. Related Work

There has been some previous work investigating when the double descent phenomena occur and when it doesn't [1, 4, 10, 19, 24, 30, 41, 54]. Previous work has shown that the number of iterations (early-stopping) can affect the double descent peak, and sometimes eliminate it [3, 40, 57]. A recent study also argues that the double descent occurs due to label noise [20], but we show that double descent can be observed with clean datasets. However, we are unaware of any work investigating the effect of normalization, learning rate, batch size, choice of optimization algorithm, or the other hyperparameters we investigate on the observed peak in double descent.

Apart from the double descent phenomenon in generalization error, prior work, from the optimization perspective, has also identified a similar descent-like phenomenon in the condition number for simple linear regression problems. Specifically, for least-squares linear regression, prior theoretical and experimental work has shown that the condition number increases near the peak of the double descent, where the *condition number* here is the ratio between the maximum and the minimum singular values of the data matrix for linear regression. Poggio et al. [46] is the first to show that the condition number for a random matrix peaks when the number of rows equals the number of columns for linear regression. Rangamani et al. [49] demonstrated that the condition number regulates the stability of the least squares solution, which is why the error peaks at $P = N$. Kuzborskij et al. [32] argued that the condition number of the feature matrix is determined only by the minimal singular value under the assumption that the maximum singular value is constant. Their theory included dependence of the excess risk on the minimal eigenvalue. Mei and Montanari [38] hypothesized

1. We use the term "optimizer" to refer to an optimization algorithm configured with the corresponding hyperparameters.

that the peaking at $P = N$ can be explained by the explosion of the variance, which is related to the condition number.

To the best of our knowledge, previous studies have theoretically identified the peak in the condition number of random matrices and observed it in some real datasets but have not established a connection to the double descent curve. Our work connects the previous finding on the peaking of the condition number and its effect on a double descent curve. In Figure 1, the condition number is largest at $P = N$ (the double descent peak), which agrees with theoretical results. We find that this makes optimization harder at $P = N$, so models are less likely to converge, which causes the peak of double descent to be reduced or disappear, but reappear with longer training (Section 6). This finding extends the previous ones because we show the interplay of the condition numbers in features and other elements in optimization. Our observation on condition number also extends the previous work in that we can control the magnitude of double descent by controlling the condition number, as shown in Figures 1, 6 and 7. Our work is also related to the mitigation of double descent, which we briefly discuss in Appendix B.

3. Experimental Setup

We present empirical evidence on random feature models trained on MNIST [33] and include additional results for both random feature models and two-layer neural networks on Fashion-MNIST [56] and CIFAR-10 [31] in Appendix C. Full experimental details are given in Appendix A. We also emphasize that we choose a setup that avoids known factors that could mitigate the double descent phenomenon as much as possible. Specifically, we do not use early stopping, which is known to remove the double descent phenomenon [3, 40, 57] . On the contrary, in the default setup, we train all models for a sufficient number of epochs (1000) so that most models converge in the first $1/10$ of the epochs. To further exclude the possibility of stopping too early, even in this scenario (which is not likely), we extend it to 10,000 epochs in Section 6. We also do not use any norm-based weight penalty to restrict the parameter space [41] or add dropout layers during training. See Appendix A for a detailed experimental setup.

4. Poor Conditioning Reduces Double Descent

It is well-known from optimization theory that better conditioning leads to faster convergence for gradient descent optimization [42]. We observe that the height of the double descent peak negatively correlates with the condition number of the random feature matrices. In Figure 1, we see that, at $P = N$, the peak in condition number is always present, regardless of the peak in double descent. However, the height of the condition number matters because the setting with a lower condition number (lighter color in row 1) corresponds to a more prominent double descent peak (lighter color in row 2). Here, the poor conditioning of the optimization problem makes it more difficult for the model around $P/N = 1$ and the optimization algorithm to converge to a sufficiently low-loss minimum, as illustrated in the third row in Figure 1. Previous studies also have shown the condition number peaks at the peak of the double descent (which occurs at $P = N$) [7, 23, 32, 46, 49]. We observe this to be the case in Figure 1, confirming the findings of these previous studies.

We see that the double descent peaks disappear when the feature matrix is 1) **unnormalized**, or when the random matrix/features have a smaller scale due to the 2) **scaling of the features** (e.g., scaling the feature matrix by a small constant) or 3) **the initialization of the random matrix** (e.g.,

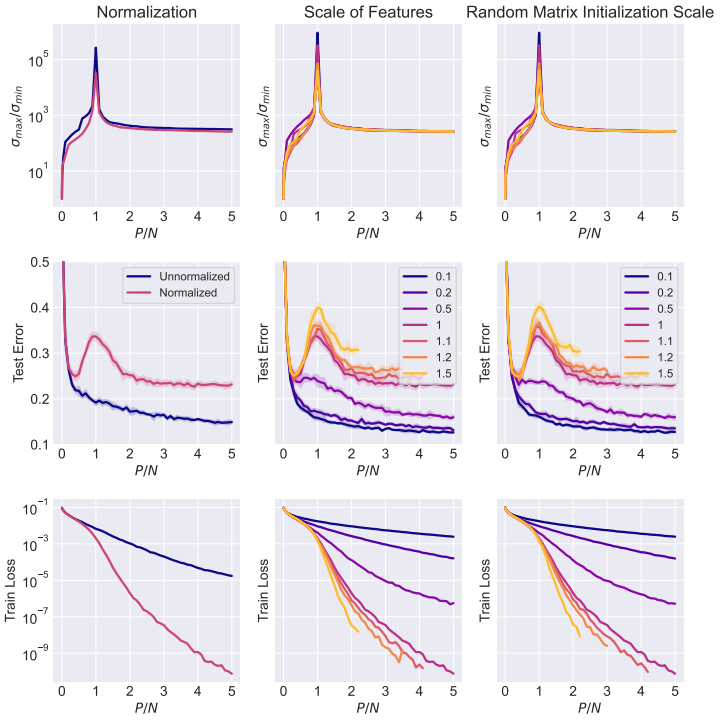


Figure 1: Condition number, test error, and training loss on random feature models with varying properties features on an RFM with ReLU features. Darker colors represent **higher condition number** in row 1, **weaker double descent** in row 2, and **high training loss** in row 3. **Row 1**: Normalizing the data and increasing the scale of the features and random matrix yield better (smaller) condition numbers at $P = N$. **Row 2**: Double descent does not occur in unnormalized, small-scale input features and small random matrix initialization. Double descent is observed more strongly when a lower condition number is at the peak. **Row 3**: A higher double descent peak corresponds to a lower training loss at $P = N$.

from a normal distribution with a small variance) (because they all change the input features to the linear classifier). We plot the training loss and observe that the setting where double descent occurs has a training loss much smaller than one in which double descent does not occur (Figure 1, bottom). The condition number is largest at $P = N$ (the double descent peak) in Figure 1, which agrees with prior theoretical results (see Section 2). We find that this makes optimization harder at $P = N$, so models do not converge, which causes the peak of double descent to be reduced or disappear, but reappear with longer training (Section 6). We observe that double descent tends to occur in better-conditioned matrices because a lower minimum is found by the optimizer.

5. Slow-Convergence Leads to Disappearance of Double Descent

Our results suggest that a slow-convergence setting[2] often reduces or removes the peaking phenomenon, and fast-convergence setting (i.e., finding a lower minimum) restores the peaking phenomenon. We observe this pattern in hyperparameters that affect the optimizer, such as **learning rate** (constant and decay), **batch size**, and **optimization algorithm**. All three factors can be considered as affecting the convergence of the optimizer directly, where a higher minimum found by the optimizer corresponds to a less prominent peak.

We observe that a faster-convergence optimization algorithm that finds a lower loss minimum exhibits double descent more strongly, and slow-convergence settings may not exhibit it at all (last two columns of Figure 2). We observe that the peak height negatively correlates with the training loss at $P = N$. For curves with Cholesky decomposition, the error rate approaches random guessing

2. We use the term "slow-convergence setting" to refer to situations where the model converges slowly given the optimization problem, the optimization algorithm, its associated hyperparameters, and all other hyperparameters.

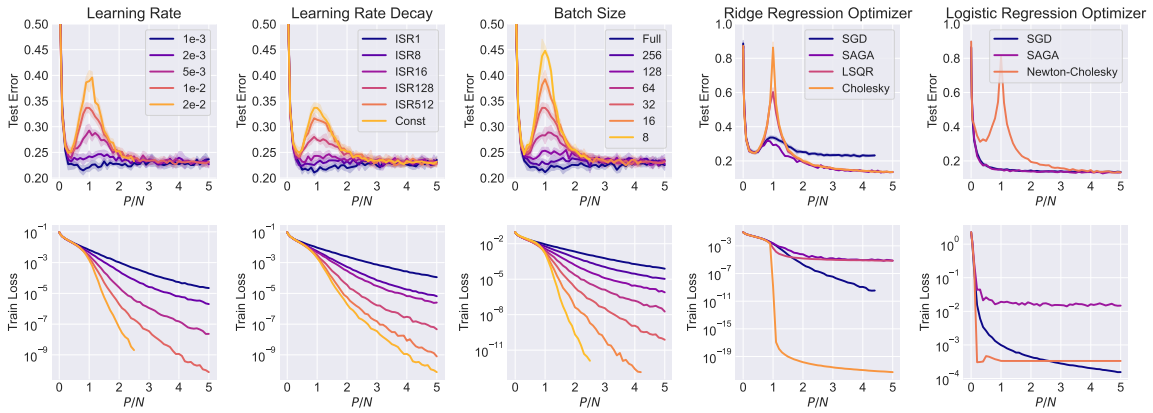


Figure 2: Test error and training loss on random feature models with varying learning rates, batch sizes, and optimization algorithms. Double descent occurs with a sufficiently large and steady learning rate, a small enough batch size, or a low-enough minimum is obtained by the optimizer.

in both ridge regression and logistic regression. Similar observations can be made on SAGA and SGD, where the double descent peak occurs mildly with a slight increase in test error.

We observe that low learning rates, which are insufficient to reach a low-loss minimum, reduce or eliminate the double descent peak (first column of Figure 2). When the overparameterization ratio P/N is greater than 3, using a lower learning rate has almost no impact on the test error, but it eliminates the peak without any form of explicit regularization. The same result holds for learning rate decay. We observe that faster learning rate decay has a similar effect to a small constant learning rate, and decaying every iteration removes the peak entirely. Both imply that the emergence of double descent requires maintaining a stable and large enough learning rate during training, corresponding to a faster convergence optimization setting that lands at a lower minimum loss.

We observe that large batch sizes reduce or eliminate the double descent phenomenon. In Figure 2, the peaking phenomenon disappears on the generalization curve for batch sizes of 500 (full-batch) and 256. Modifying the batch size changes the number of updates the optimization algorithm performs. Thus, large batch sizes take fewer steps.

6. Training Longer Recovers Double Descent

We show that for hyper-parameter setups that do not exhibit double descent, we can recover this phenomenon simply by running the optimization procedure longer to reach a lower training loss. In Figure 3, we increase the number of iterations by a factor of 10 so that the training loss of a slow-convergence setting is aligned with or approaches the default (a much lower one). We observe that the double descent peak is recovered for all factors. Precisely, in Figure 3 for the learning rate (column 4) and batch size (column 6), both the test error and training loss curves with ten times the original epochs exactly overlap with curves produced by the faster-converging optimization setup (i.e., the larger learning rate of $1e^{-2}$ or the smaller batch size of 32). Our results suggest that the optimization length is a simple but strong indicator of why double descent is observed in some realistic settings.

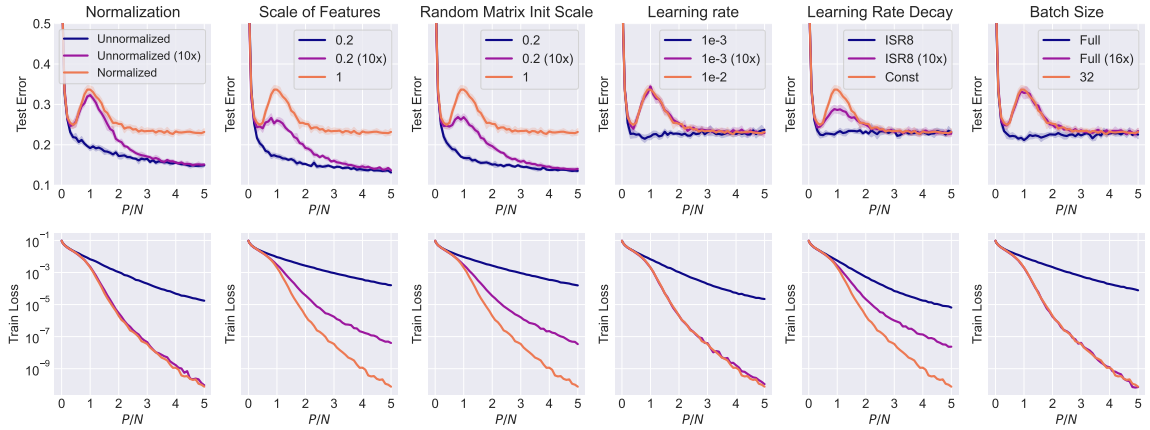


Figure 3: Test error and training loss on random feature models with varying learning rates, batch sizes, and optimization algorithms at 10x iterations. The peaking phenomenon is recovered as long as a sufficient number of gradient updates is applied, even for ill-conditioned features.

In practical settings, proper regularization often prevents models from reaching 0 training error. Even though no explicit regularization is applied, we usually do not continue training for a large number of epochs when the model has already overfit the training set. In our experiments, for double descent to recover Figure 3, models are usually trained 200-400 times longer after converging to 0 training error, which is not realistic for deep and large models used in practice. This also implies that the peak's emergence rate largely depends on the specific optimization setup, where optimization settings that converge slowly do not exhibit the peaking behavior.

7. Experiments with Two-Layer Neural Networks

We present results on two-layer neural networks in Figure 9 in the Appendix. Despite the fact that two-layer neural networks are fully non-linear models, we observe that the results on two-layer neural networks are consistent with our previous findings on random feature models.

8. Conclusion

In dissecting the occurrence of double descent in machine learning, our study elucidates a unifying underlying phenomenon tied to optimization: double descent occurs when the optimizer can achieve a low-loss minimum. While seemingly disconnected, factors like initialization, learning rate, normalization, batch size, and optimizer choice work together to influence the overall optimization trajectory, thereby affecting the optimizer's path to a minimum. Our findings not only simplify the understanding of these variables but also shed light on the practical likelihood of double descent, which is minimized due to careful hyperparameter selections and inherent inductive biases in real-world setups. Although current theoretical frameworks fall short of encapsulating our observations entirely, our results shed light on promising directions for deeper theoretical scrutiny into the interplay between optimization and double descent, thereby paving the way for a deeper understanding of the intriguing phenomenon.

References

[1] Jimmy Ba, Murat A. Erdogdu, Taiji Suzuki, Denny Wu, and Tianzong Zhang. Generalization of two-layer neural networks: An asymptotic viewpoint. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=H1gBsgBYwH.

[2] Mikhail Belkin, Daniel J. Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias-variance trade-off. *Proceedings of the National Academy of Sciences*, 116:15849–15854, 2018.

[3] Anthony Bodin and Nicolas Macris. Model, sample, and epoch-wise descents: exact solution of gradient flow in the random feature model. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:239616354.

[4] Sebastian Buschjäger and Katharina Morik. There is no double-descent in random forests. *ArXiv preprint*, abs/2111.04409, 2021. URL https://arxiv.org/abs/2111.04409.

[5] Xiangyu Chang, Yingcong Li, Samet Oymak, and Christos Thrampoulidis. Provable benefits of overparameterization in model compression: From double descent to pruning neural networks. In *AAAI Conference on Artificial Intelligence*, 2020.

[6] John Chen, Qihan Wang, and Anastasios Kyrillidis. Mitigating deep double descent by concatenating inputs. *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021.

[7] Zhijun Chen and Hayden Schaeffer. Conditioning of random feature matrices: Double descent and generalization error. *ArXiv preprint*, abs/2110.11477, 2021. URL https://arxiv.org/abs/2110.11477.

[8] Andrew Cotter, Aditya Krishna Menon, Harikrishna Narasimhan, Ankit Singh Rawat, Sashank J. Reddi, and Yichen Zhou. Distilling double descent. *ArXiv preprint*, abs/2102.06849, 2021. URL https://arxiv.org/abs/2102.06849.

[9] Yehuda Dar and Richard Baraniuk. Double double descent: On generalization errors in transfer learning between linear regression tasks. *ArXiv preprint*, abs/2006.07002, 2020. URL https://arxiv.org/abs/2006.07002.

[10] Yehuda Dar, Paul Mayer, Lorenzo Luzi, and Richard G. Baraniuk. Subspace fitting meets regression: The effects of supervision and orthonormality constraints on double descent of generalization errors. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2366–2375. PMLR, 2020. URL http://proceedings.mlr.press/v119/dar20a.html.

[11] Stéphane d'Ascoli, Maria Refinetti, Giulio Biroli, and Florent Krzakala. Double trouble in double descent : Bias and variance(s) in the lazy regime. In *International Conference on Machine Learning*, 2020.

[12] Stéphane d'Ascoli, Levent Sagun, and Giulio Biroli. Triple descent and the two kinds of overfitting: where and why do they appear? *Journal of Statistical Mechanics: Theory and Experiment*, 2021, 2020.

[13] Aaron Defazio, Francis R. Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 1646–1654, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/ede7e2b6d13a41ddf9f4bdef84fdc737-Abstract.html.

[14] Oussama Dhifallah and Yue M. Lu. A precise performance analysis of learning with random features. *ArXiv preprint*, abs/2008.11904, 2020. URL https://arxiv.org/abs/2008.11904.

[15] Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K. Fletcher. Generalization error of generalized linear models in high dimensions. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 2892–2901. PMLR, 2020. URL http://proceedings.mlr.press/v119/emami20a.html.

[16] Nayara Fonseca and Veronica Guidetti. Similarity and generalization: From noise to corruption. *ArXiv preprint*, abs/2201.12803, 2022. URL https://arxiv.org/abs/2201.12803.

[17] Matteo Gamba, Erik Englesson, Marten Bjorkman, and Hossein Azizpour. Deep double descent via smooth interpolation. *ArXiv preprint*, abs/2209.10080, 2022. URL https://arxiv.org/abs/2209.10080.

[18] Matteo Gamba, Hossein Azizpour, and Marten Bjorkman. On the lipschitz constant of deep networks and double descent. *ArXiv preprint*, abs/2301.12309, 2023. URL https://arxiv.org/abs/2301.12309.

[19] Yufei Gu, Xiaoqing Zheng, and Tomaso Aste. Unraveling the enigma of double descent: An in-depth analysis through the lens of learned feature space. *ArXiv preprint*, abs/2310.13572, 2023. URL https://arxiv.org/abs/2310.13572.

[20] Arunav Gupta, Rohit Mishra, William Luu, and Mehdi Bouassami. On feature scaling of recursive feature machines. *ArXiv preprint*, abs/2303.15745, 2023. URL https://arxiv.org/abs/2303.15745.

[21] Hamed Hassani and Adel Javanmard. The curse of overparametrization in adversarial training: Precise analysis of robust generalization for random features regression. *ArXiv preprint*, abs/2201.05149, 2022. URL https://arxiv.org/abs/2201.05149.

[22] Trevor J. Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of statistics*, 50 2:949–986, 2019.

[23] Hanwen Huang and Qinglong Yang. Large scale analysis of generalization error in learning using margin based classification methods. *Journal of Statistical Mechanics: Theory and Experiment*, 2020, 2020.

[24] Lang Huang, Chao Zhang, and Hongyang Zhang. Self-adaptive training: beyond empirical risk minimization. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e0ab531ec312161511493b002f9be2ee-Abstract.html.

[25] Lang Huang, Chaoning Zhang, and Hongyang Zhang. Self-adaptive training: Bridging the supervised and self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2021.

[26] Ningyuan Teresa Huang, David W. Hogg, and Soledad Villar. Dimensionality reduction, regularization, and generalization in overparameterized regressions. *SIAM J. Math. Data Sci.*, 4:126–152, 2020.

[27] Like Hui and Mikhail Belkin. Evaluation of neural architectures trained with square loss vs cross-entropy in classification tasks. *ArXiv preprint*, abs/2006.07322, 2020. URL https://arxiv.org/abs/2006.07322.

[28] Kelvin K. Kan, James G. Nagy, and Lars Ruthotto. Avoiding the double descent phenomenon of random feature models using hybrid regularization. *ArXiv preprint*, abs/2012.06667, 2020. URL https://arxiv.org/abs/2012.06667.

[29] Ganesh Ramachandra Kini and Christos Thrampoulidis. Analytic study of double descent in binary classification: The impact of loss. *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2527–2532, 2020.

[30] Jesse H. Krijthe and M. Loog. The peaking phenomenon in semi-supervised learning. In *International Workshop on Structural and Syntactic Pattern Recognition*, 2016.

[31] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[32] Ilja Kuzborskij, Csaba Szepesvari, Omar Rivasplata, Amal Rannen-Triki, and Razvan Pascanu. On the role of optimization in double descent: A least squares study. In *Neural Information Processing Systems*, 2021.

[33] Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2, 2010.

[34] Eng Hock Lee and Vladimir Cherkassky. Vc theoretical explanation of double descent. *ArXiv preprint*, abs/2205.15549, 2022. URL https://arxiv.org/abs/2205.15549.

[35] Licong Lin and Edgar Dobriban. What causes the test error? going beyond bias-variance via anova. *J. Mach. Learn. Res.*, 22:155:1–155:82, 2020.

[36] Bruno Loureiro, C'edric Gerbelot, Maria Refinetti, Gabriele Sicuro, and Florent Krzakala. Fluctuations, bias, variance & ensemble of learners: Exact asymptotics for convex losses in high-dimension. *ArXiv preprint*, abs/2201.13383, 2022. URL https://arxiv.org/abs/2201.13383.

[37] Lorenzo Luzi, Yehuda Dar, and Richard Baraniuk. Double descent and other interpolation phenomena in gans. *ArXiv preprint*, abs/2106.04003, 2021. URL https://arxiv.org/abs/2106.04003.

[38] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75, 2019.

[39] Diganta Misra. Mish: A self regularized non-monotonic activation function. In *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020. URL https://www.bmvc2020-conference.com/assets/papers/0928.pdf.

[40] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=B1g5sA4twr.

[41] Preetum Nakkiran, Prayaag Venkat, Sham M. Kakade, and Tengyu Ma. Optimal regularization can mitigate double descent. *ArXiv preprint*, abs/2003.01897, 2020. URL https://arxiv.org/abs/2003.01897.

[42] Yurii Nesterov. Lectures on convex optimization. 2018. URL https://api.semanticscholar.org/CorpusID:125291746.

[43] Christopher C. Paige and Michael A. Saunders. Lsqr: An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Softw.*, 8:43–71, 1982.

[44] Pratik V. Patil, Jin-Hong Du, and Arun K. Kuchibhotla. Bagging in overparameterized learning: Risk characterization and risk monotonization. 2022.

[45] Pratik V. Patil, Arun K. Kuchibhotla, Yuting Wei, and Alessandro Rinaldo. Mitigating multiple descents: A model-agnostic framework for risk monotonization. *ArXiv preprint*, abs/2205.12937, 2022. URL https://arxiv.org/abs/2205.12937.

[46] Tomaso A. Poggio, Gil Kur, and Andy Banburski. Double descent in the condition number. *ArXiv preprint*, abs/1912.06190, 2019. URL https://arxiv.org/abs/1912.06190.

[47] Victor Qu'etu and Enzo Tartaglione. Can we avoid double descent in deep neural networks? 2023.

[48] Victor Qu'etu and Enzo Tartaglione. Dodging the sparse double descent. *ArXiv preprint*, abs/2303.01213, 2023. URL https://arxiv.org/abs/2303.01213.

[49] Akshay Rangamani, Lorenzo Rosasco, and Tomaso A. Poggio. For interpolating kernel machines, minimizing the norm of the erm solution maximizes stability. *Analysis and Applications*, 2020.

[50] Leslie Rice, Eric Wong, and J. Zico Kolter. Overfitting in adversarially robust deep learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 8093–8104. PMLR, 2020. URL http://proceedings.mlr.press/v119/rice20a.html.

[51] Mojtaba Sahraee-Ardakan, Tung Mai, Anup B. Rao, Ryan A. Rossi, Sundeep Rangan, and Alyson K. Fletcher. Asymptotics of ridge regression in convolutional models. In *International Conference on Machine Learning*, 2021.

[52] Rylan Schaeffer, Mikail Khona, Zachary Robertson, Akhilan Boopathy, Kateryna Pistunova, Jason W. Rocks, Ila Rani Fiete, and Oluwasanmi Koyejo. Double descent demystified: Identifying, interpreting & ablating the sources of a deep learning puzzle. *ArXiv preprint*, abs/2303.14151, 2023. URL https://arxiv.org/abs/2303.14151.

[53] Vasu Singla, Sahil Singla, David Jacobs, and Soheil Feizi. Low curvature activations reduce overfitting in adversarial training. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16403–16413, 2021.

[54] Jianxin Wang and José Bento. Optimal activation functions for the random features regression model. *ArXiv preprint*, abs/2206.01332, 2022. URL https://arxiv.org/abs/2206.01332.

[55] Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *ArXiv preprint*, abs/2002.08791, 2020. URL https://arxiv.org/abs/2002.08791.

[56] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *ArXiv preprint*, abs/1708.07747, 2017. URL https://arxiv.org/abs/1708.07747.

[57] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. *ArXiv preprint*, abs/2002.11328, 2020. URL https://arxiv.org/abs/2002.11328.

Appendix A. Experimental Setup and Hyperparameters

In this section, we described the detailed setup of the dataset, models, and training.

A.1. Datasets

We perform all experiments on MNIST [33], Fashion-MNIST [56], and CIFAR-10 [31]. We select small subsets of size N from the full training set as our training data and evaluate the generalization error using the complete test set. We trained models for RFMs with a random feature size of up to $P/N = 5$, while for two-layer NNs, we utilize models up to $P/N = 5 \cdot C$ parameters, where C is fixed at 10 in our setting. For two-layer NNs, the interpolation threshold is at $N \cdot C$ instead of N [2]. By default, we normalize the image pixel features to follow a normal distribution by applying the transformation $\frac{\mathbf{X}-\mu}{s} \cdot \gamma$, where μ, s, γ are the mean, standard deviation, and the scaling factor, respectively.

A.2. Training

Models. For random feature models, we use static first-layer weights $\overline{\mathbf{W}}_0 \in \mathbb{R}^{d \times P}$ and trainable second-layer weights $\mathbf{W}_1 \in \mathbb{R}^{P \times C}$, where P and C represent the projected feature dimension and the number of classes, respectively. We initialize both weight matrices from $\overline{\mathbf{W}}_0 \sim \mathcal{N}\left(0, \frac{k_0}{\sqrt{D}}\right)$ and $\mathbf{W}_1 \sim \mathcal{N}\left(0, \frac{k_1}{\sqrt{P}}\right)$, where k_0 is a scaling factor for the standard deviation of the weight matrix, and D represents the input feature dimension. k_1 is always 1 in all experiments. Bias terms are always set to 0, and the ReLU nonlinearity is used by default. Our setup for two-layer neural networks resembles that of a random feature model. The only difference is that the first layer weights are trained.

Optimization. In our learning rate decay experiments, we adopt an inverse square root schedule similar to Nakkiran et al. [40], but we multiply the initial learning rate by the factor $\frac{1}{\sqrt{\lfloor t/l \rfloor + 1}}$, where t is the current iteration and l is an interval parameter. By controlling the interval l, we modify the decay frequency during the training trajectory. In optimizer experiments, we select numerical solvers, Cholesky and QR [43], for ridge regression and Newton-Cholesky for logistic regression because they obtain solutions with much lower loss than SGD. We also chose SAGA [13] as an alternative gradient-based algorithm that (empirically) converges slower than SGD with momentum for comparison. A small regularization constant 1e-8 is used for numerical stability. In the 10x iteration experiments, we ensure that the number of gradient updates matches that in a mini-batch case. Given a full-batch setting, we calculate the extended number of iterations by $\mathcal{T}_{\text{new}} = \lceil N/b \rceil \cdot \mathcal{T}$, which is equal to the number of gradient updates in a mini-batch setting with batch size b and $\mathcal{T}$ iterations.

Producing double descents. Consistent with Belkin et al. [2] and Nakkiran et al. [40], we employed the same set of hyperparameters for all model sizes and trained them using SGD with a fixed number of epochs and constant step size. The default hyperparameters are selected based on two training error constraints: 1) the largest model has to attain 0 training error within the first $1/10$ iterations, and 2) (at least) all models with $P \geq N$ have to converge to 0 training error before the final iteration. These constraints, derived from empirical observations, are fast in convergence and effective in generating the double descent phenomenon for both RFM and two-layer NN trained with SGD on

MNIST and Fashion-MNIST. After some exploration, we use SGD with a Nesterov momentum of 0.95, a mini-batch size of 32, and a constant step size of 1e-2 for 1000 epochs. For two-layer NNs, we increase the step size to 5e-2 and the number of epochs to 1500. By default, we utilize the standard MSE loss. When a specific hyperparameter is studied, all other parameters follow the default ones. We primarily focus on the MSE loss because it has been heavily studied in the literature [2, 14, 15, 22, 38]. We follow the previous implementation and do not include softmax at the network output Belkin et al. [2], Hui and Belkin [27]. In our experiments, we strictly control the confounding variables with potential effects on the peak in all experiments, such as label noise, which could exacerbate the curve [19, 40], and all forms of weight decay and early stopping, which are known to flatten the peak [41]. This ensures the robustness of our results. All experiments are repeated at least five times.

Appendix B. Related Work on Double Descent Mitigation

While we do not intend to mitigate the double descent, our work investigates conditions under which the peak of double descent does or does not occur. Related, various techniques have been demonstrated to successfully reduce the peak, including ℓ_2 regularization [12, 28, 35, 38, 40, 41, 47, 48], ensemble methods [12, 36, 44, 55], cross-validation [45], dimensionality reduction [26], input concatenation [6], and the type of non-linear random features [14]. Several works claim that double descent is not observed in certain settings, even without the explicit mitigation techniques mentioned above. These settings include self-adaptive training [24, 25], level of supervision [10, 37], random forest models [4], a two-layer neural network with certain initialization of the first layer weight [1], and special activation functions [53, 54]. We highlight that our focus in this paper is not to propose a technique to mitigate double descent. Instead, we find that the conditioning of the optimization problem and specific setups significantly affects the magnitude of the peaking phenomenon, and models in slower-convergence settings do not exhibit the peak. However, this observation might be helpful as a simple technique to mitigate double descent in practice, which we leave to future research. We also emphasize the importance of carefully examining the effect of optimization in producing double descent. This is because optimization is involved in almost all settings mentioned above. Therefore, it is crucial to identify whether the absence of double descent is due to optimization before examining other more complicated factors.

Appendix C. Additional Experiments

Here, we present the exact figures as in the paper's main text but on additional datasets (Fashion-MNIST and CIFAR-10) and two-layer neural networks. Our findings in the paper's main text generalize to these datasets and models.

C.1. Time Evolution of A Double Descent Curve

We present the temporal evolution of a double descent curve through gradient descent iterations, demonstrating that the peaks form only post-model convergence. This aligns with the theoretical findings and synthetic experiments by Bodin and Macris [3], indicating that the peaking phenomenon at $P \approx N$ appears only beyond a certain iteration, with a monotonic curve prevailing prior. Figure Figure 4 illustrates the iteration-wise change in training/test error and loss. We discern that models at $P \approx N$ converge on the training set after 50 epochs. Moreover, a training span of fewer than

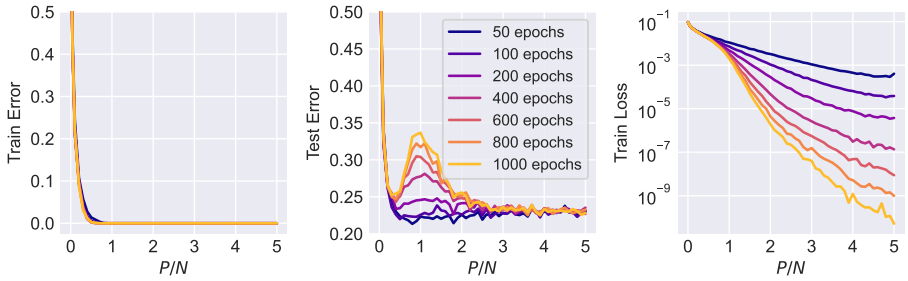
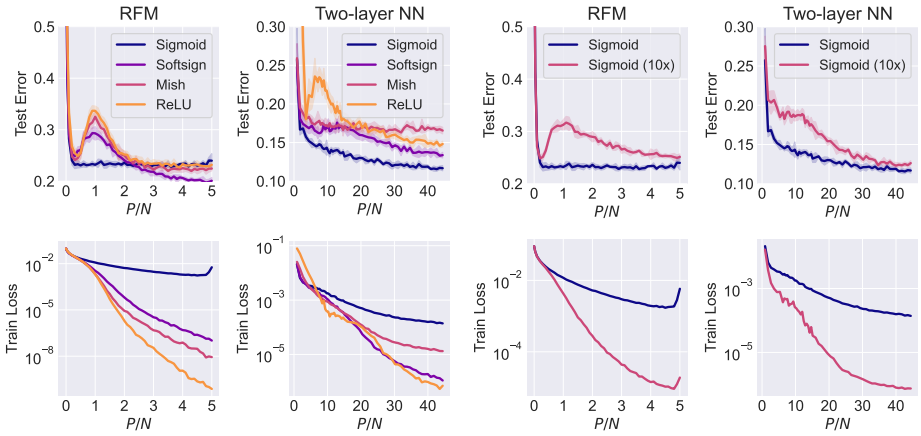


Figure 4: The time evolution of a double descent curve. At the interpolation threshold $P = N$, all models have achieved zero training error, and the peak starts to emerge only after zero training error is obtained.

200 epochs is inadequate for these models to manifest the peaking phenomenon. Despite the model maintaining a zero training error up to the 1000-th epoch, the peak in test error initiates and escalates as the training loss goes down. This insight corroborates preceding studies [3, 40], asserting that early stopping diminishes the peak. Yet, we contend that the use of early stopping in hyperparameter tuning might be excessive for this goal, given double descent necessitates a substantial number of iterations post-convergence to appear. Hence, from an optimization standpoint, mitigating double descent is feasible, provided a model is not over-trained.

C.2. Ill-Conditioned Non-Linear Features

For random feature models, another way to modify the input features is the choice of the non-linear function. Previous work shows that activation functions reduce the peaking phenomenon [14], but we show that we can recover the peak by simply increasing the number of iterations. In the left figure of Figure 5, we employ ReLU, mish [39], softsign, and sigmoid nonlinearities. ReLU and sigmoid show consistent behavior for both RFM and two-layer neural networks, even though activation functions operate differently on these two models (i.e., one with the input and the other with the intermediate embeddings). However, for mish and softsign, double descent is only observed in RFMs. In the second figure of Figure 5, we show that the absence of double descent on sigmoid follows the same pattern as our previous experiments in Figure 3. Non-linear features produced by sigmoid make optimization difficult, resulting in monotonicity, but we can recover the peaking phenomenon by scaling the number of iterations by 10.



(*a*) Varying nonlinearities (*b*) Increasing iterations

Figure 5: **Left**: Test error and training loss curves for RFMs trained and four different activation functions. Models with activation functions that converge a higher training loss tend to avoid double descent. Given the same optimization and hyperparameter setup, the sigmoid activation does not exhibit double descent in RFM and two-layer neural networks. **Right**: Test error and training loss curves for RFMs trained and sigmoid activation. 10x iterations recover double descent. This matches our findings on the impact of a slow-convergence setting on double descent.

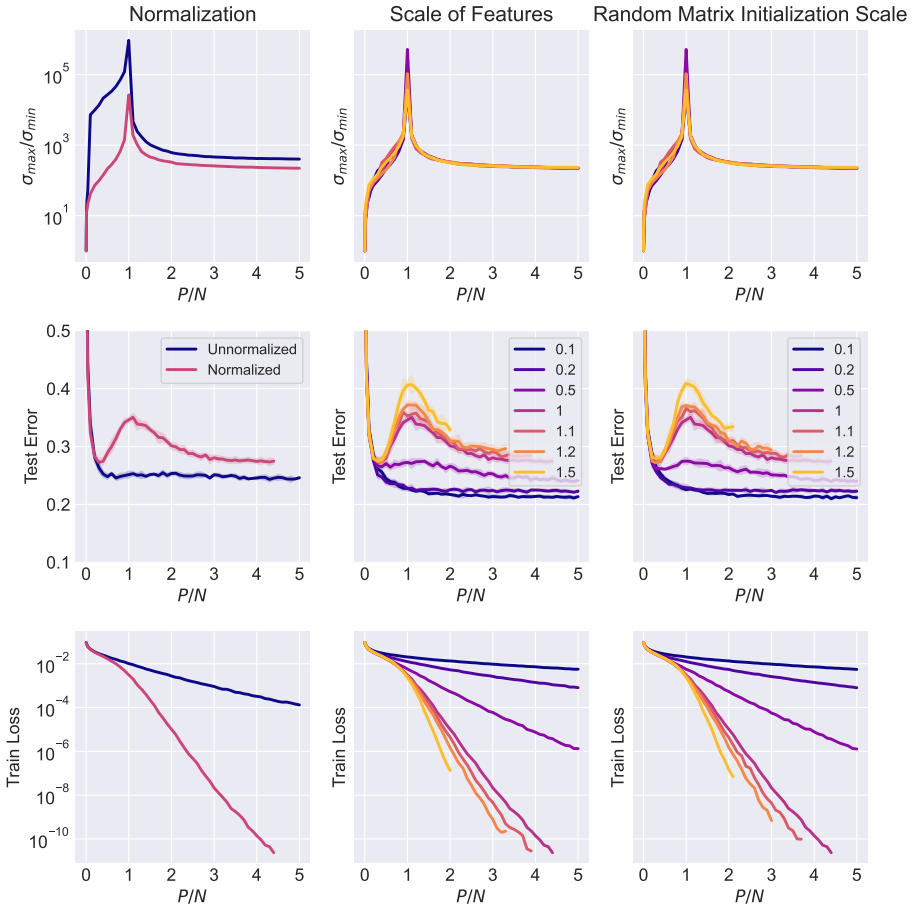


Figure 6: Additional results to support Figure 1 using the Fashion-MNIST dataset.

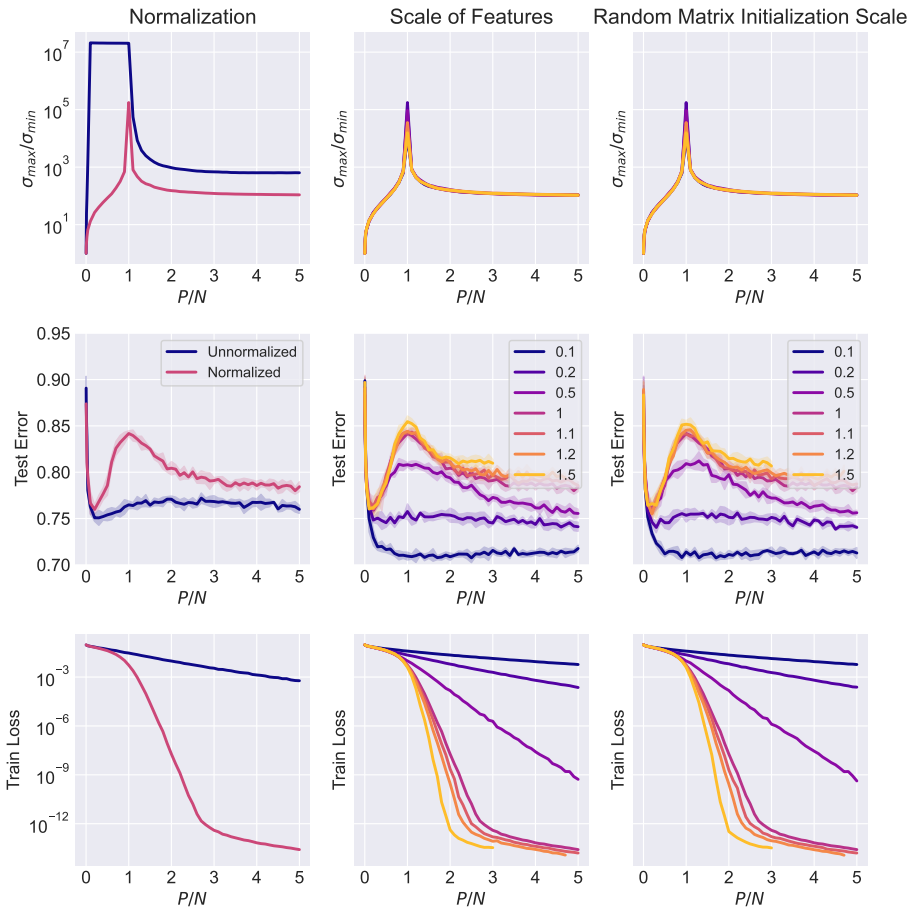
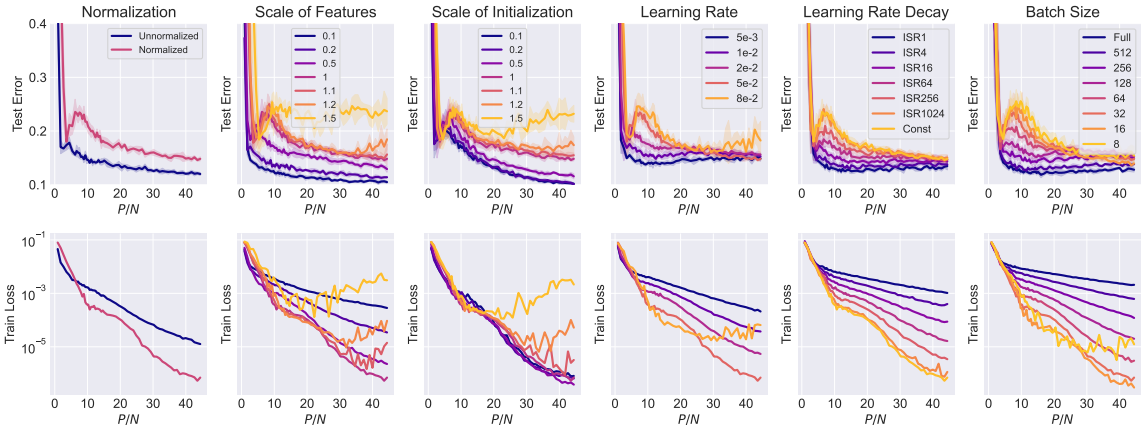
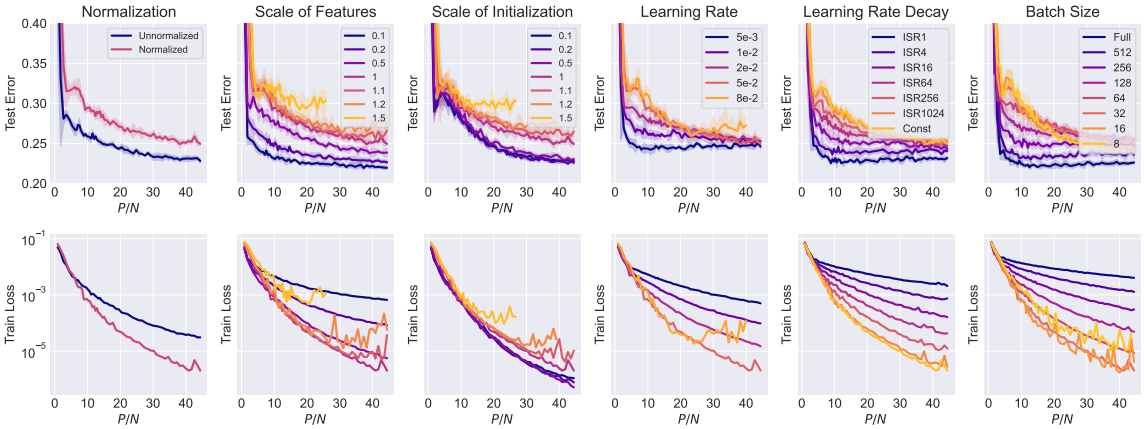


Figure 7: Additional results to support Figure 1 using the CIFAR-10 dataset.



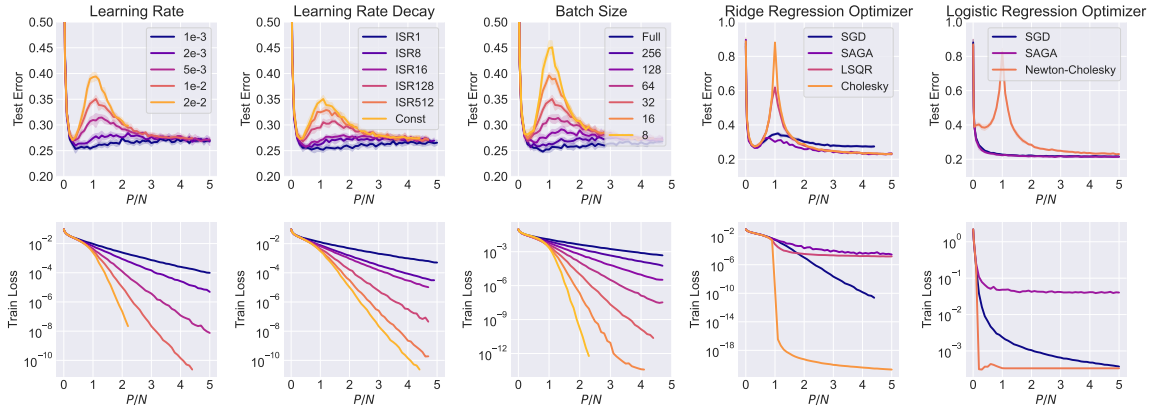
(*a*) Two-layer neural network on MNIST



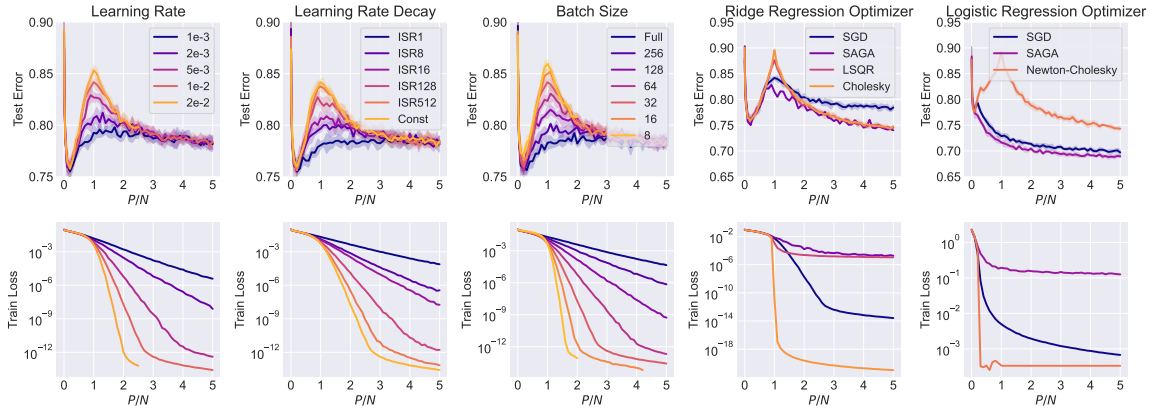
(*b*) Two-layer neural network on Fashion-MNIST

Figure 8: Additional results to support Figures 1 and 2. Test error and training loss curves by (using) normalization, varying scale of features or initialization, learning rate, batch size, and optimization algorithm of a two-layer neural network on MNIST and Fashion-MNIST. The peaking phenomenon becomes less prominent as the features become worse-conditioned or directly using a slow-convergence setting.

C.3. Slow-Convergence Settings in Optimization

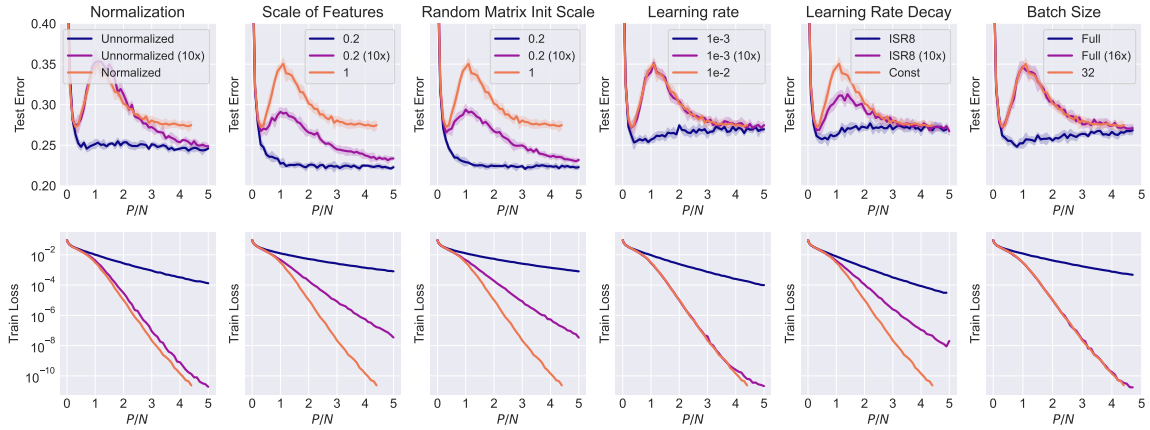


(*a*) RFM on Fashion-MNIST

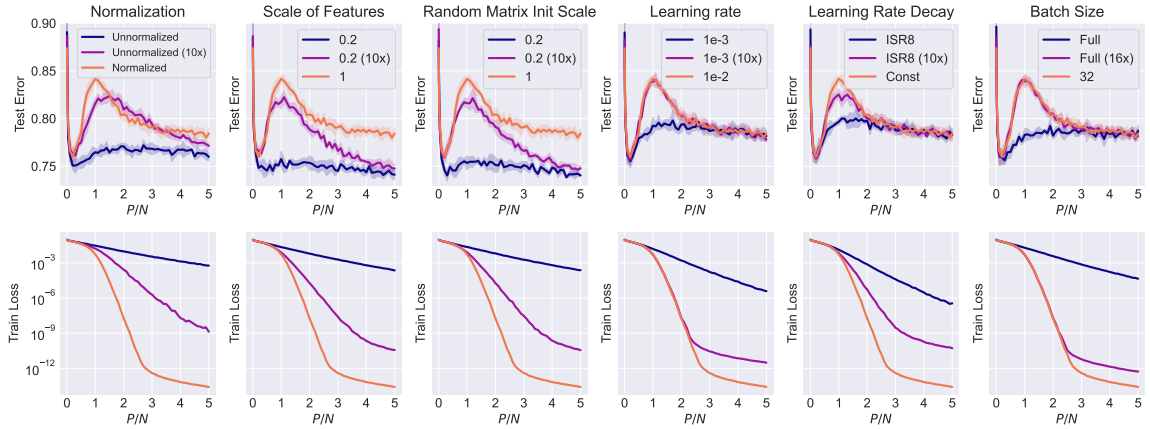


(*b*) RFM on CIFAR-10

Figure 9: Additional results to support Figure 2. Test error and training loss curves by varying learning rate, batch size, and optimization algorithm of a random feature model on Fashion-MNIST and CIFAR-10. The peaks are reduced on slow-convergence settings with too small a learning rate, too frequent learning rate decay, too large batch size, and slow-convergence settings.

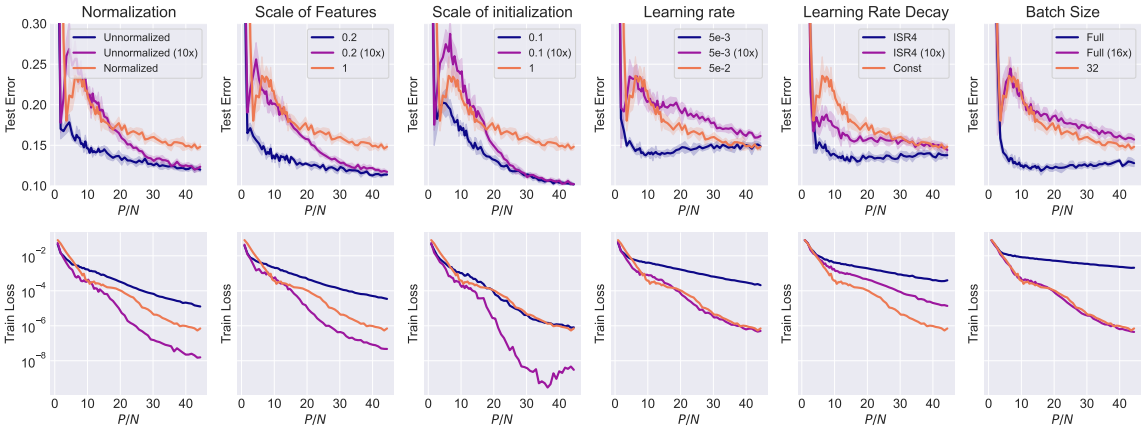


(*a*) RFM on Fashion-MNIST

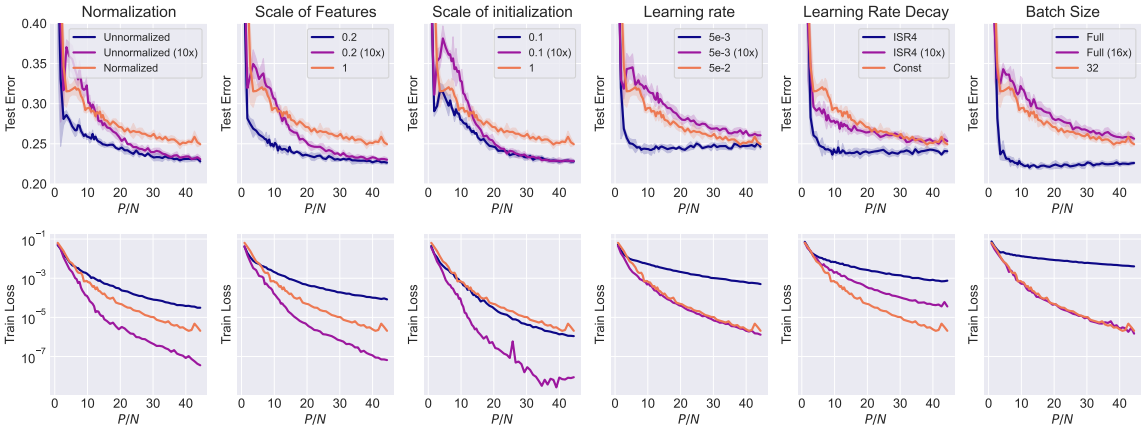


(*b*) RFM on CIFAR-10

Figure 10: Additional results to support Figure 3. Test error and training loss curves of slow-convergence settings with 10x iterations of RFMs on Fashion-MNIST and CIFAR-10. We are able to recover the peaking phenomenon in all cases by scaling the number of iterations by a factor of 10.



(*a*) Two-layer neural network on MNIST



(*b*) Two-layer neural network on Fashion-MNIST

Figure 11: Additional results to support Figure 3. Test error and training loss curves of slow-convergence settings with 10x iterations of two-layer neural networks on MNIST and Fashion-MNIST. We are able to recover the peaking phenomenon in all cases by scaling the number of iterations by a factor of 10.