OPT2023: 15th Annual Workshop on Optimization for Machine Learning

# Why Adam Outperforms Gradient Descent on Language Models: A Heavy-Tailed Class Imbalance Problem

**Robin Yadav** ROBINY12@STUDENT.UBC.CA
**Frederik Kunstner** KUNSTNER@CS.UBC.CA
**Mark Schmidt**[†] SCHMIDTM@CS.UBC.CA
*University of British Columbia, Vancouver, Canada*
*Canada CIFAR AI Chair (Amii)*[†]
**Alberto Bietti** ABIETTI@FLATIRONINSTITUTE.ORG
*Flatiron Institute, New York, United States*

## Abstract

We show that the heavy-tailed class imbalance found in language modeling tasks leads to difficulties in optimization dynamics. When training with gradient descent, the loss associated with low frequency classes decreases slower than the loss associated with high frequency classes. Under the heavy-tailed class imbalance found in language modeling tasks, most samples are from classes of low relative frequency. This leads to overall slow decreasing on the average loss with gradient descent. Sign-based optimizers such as Adam and sign descent do not suffer from this problem, and lead to decrease on all classes. We give evidence of this behavior on training for a transformer on language data, a linear model on synthetic data whose only property is a heavy-tailed class distribution, and a convolutional network on a modified MNIST dataset made to exhibit heavy-tailed class imbalance.

## 1. Introduction

The recent progress in ML for language modeling, exemplified by the success of models such as GPT3 (Brown et al., 2020) and its successors has been enabled by optimization heuristics such as Adam (Kingma and Ba, 2015) and its variants. On those problems, gradient descent (GD) is orders of magnitude slower than Adam, and leads to unacceptable training times as those models already take weeks, if not months, of training with Adam. Despite this large gap in performance, we have a poor understanding of why Adam works so much better. This makes it difficult to develop new methods that improve on it, as we do not have a clear understanding of the problem we are trying to fix.

Recent work has focused on developing assumptions under which Adam can outperform (S)GD (e.g., Zhang et al., 2020a; Zhang et al., 2020b; Crawshaw et al., 2022), but a justification for those assumptions based on properties of the data or model has been difficult to establish. Our goal is to identify such a property that leads to a large gap between the performance of GD and Adam.

Our main finding is that a large contribution to the gap in performance can be traced to heavy-tailed class imbalance. Language data is imbalanced, as some words (or tokens) are much more frequent than others, following a power-law (the $i$th most frequent word has frequency $\propto 1/i$). We provide experimental evidence that, when trained with GD, the training loss for examples of low-frequency classes decreases slower than for frequent classes, while Adam does not suffer from this problem. Due to the heavy-tailed nature of the data, *most samples* belong to classes with few samples, leading to slow progress on the overall training loss, as shown in Figure 1.

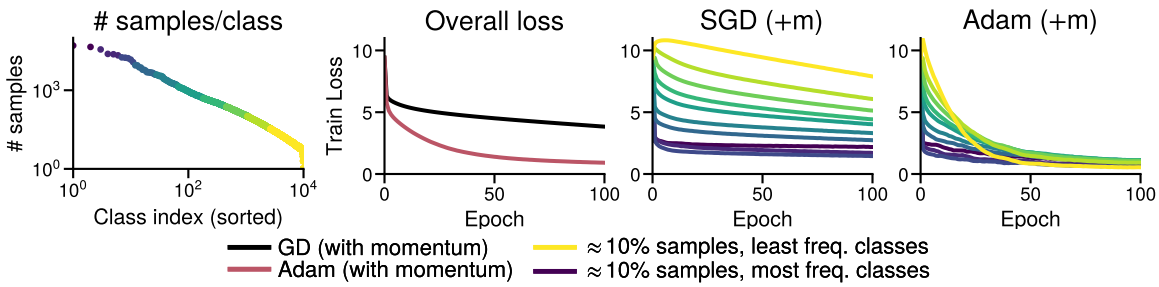



Figure 1: **Gradient descent does not make progress on low-frequency classes, while Adam does.** Training a 2-layer transformer on the PTB dataset with SGD and Adam. **Left:** Distribution of the classes and subsets of the data sorted by class frequency, each corresponding to $\approx 10\%$ of the samples. **Right:** Overall training loss and loss for each subset. SGD makes little to no progress on low-frequency classes, while Adam makes progress on all subsets.

We present empirical evidence for the observed behavior in the next sections and discuss connections with prior work in Section 3. Our key observations and the take-away message of this paper are

1. **Gradient descent fits lower-frequency classes much more slowly than Adam. and under heavy-tailed class imbalance, most examples belong to "low-frequency" classes,** providing a mechanism for why Adam makes more progress than GD on language tasks (Figure 1).
2. **This behavior is reproducible with linear models on synthetic data or CNNs on image data** when the dataset exhibits heavy-tailed class imbalance, showing that this property can cause a large performance gap between Adam and GD (Figures 2 and 3).
3. **These dynamics are not due to stochasticity and appear in deterministic (full batch) training.** All experiments beyond those presented in Figure 1 use the entire dataset at each iteration.
4. **Sign descent recovers similar dynamics to Adam while normalizing the magnitude and momentum have a smaller effect,** as observed by running additional experiments. (Figure 4).

Our experiments highlight a property of the data that has a strong influence on the optimization dynamics. They complement the growing body of literature on the challanges associated with neural networks architectures that lead to poor optimization performance, such as issues of normalization, initialization and rescaling (Liu et al., 2020; Noci et al., 2022; Zhai et al., 2023; He et al., 2023), or links between vanishing gradients and Hessians (Bengio et al., 1994; Orvieto et al., 2022).

**Limitations.** Our experiments focus on properties of the dynamics of the training loss. We plan to verify our observation on the validation loss on larger models and datasets in an extended paper.

**Experimental setup.** We use a constant step-size tuned for best training performance by grid search. Details on the models, datasets and training procedures are given in Appendix A.

## 2. Impact of class imbalance on optimization

The last layer of a language model for next-token prediction is often a linear map from the internal embedding representation to the (logits of a) probability distribution over the vocabulary. The size of the vocabulary is typically large, on the order of $10^4$ elements in the examples considered here. As the distribution of words in the corpus used to generate training examples is heavy-tailed, the relative frequency of the classes is also heavy-tailed, shown in Figure 1.

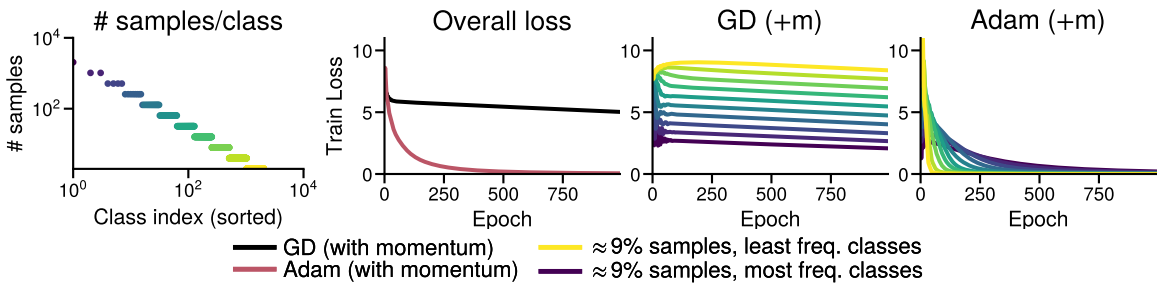



Figure 2: **The class imbalance dynamics of Figure 1 are reproducible with linear models.** A linear softmax regression on synthetic data. The input data are sampled from a high dimensional Gaussian $\mathcal{N}(0, I)$. The target classes are heavy-tailed (Histogram, left) and independent of the inputs, but the model can still fit the data as it is overparameterized.

To investigate the optimization dynamics across class frequency, we order the classes according to frequency and split the training set into 10 subsets, each containing $\approx 10\%$ of the data. The first group contains the most frequent tokens while the last group contains the least frequent ones. In the setup of Figure 1, the first group contains 2 classes with 34k and 38k samples, while the last contains $>$7k classes with a median of 8 samples each, a difference of $\approx 4\text{k}\times$ in relative frequency.

Figure 1 shows the loss per subset when training a 2-layer transformer on PTB with SGD or Adam. SGD makes slow progress on low-frequency classes, even increasing the loss for the least frequent ones. Adam reduces the loss on all subsets, driving the loss of the least frequent samples to 0 first.

### 2.1. Reproducing class imbalance issues on synthetic data with linear models

To highlight that heavy-tailed class distribution alone can lead to the difficulties observed in Figure 1, we reproduce this behavior on a toy problem. We create a classification dataset where the relative frequency of the classes approximates a power-law, as shown in Figure 2, while the inputs are sampled from a high dimensional Gaussian, independently of the class label.

While there is nothing to *learn* in this dataset, a linear model can still separate the data if the dimension is large enough. The training dynamics of (deterministic) GD and Adam on a softmax regression for this dataset behave similarly to the 2-layer transformer, compare Figures 1 and 2.

This example is not meant to suggest that language tasks are equivalent to training on random data. Rather, it illustrates that a problem that might look innocuous at first glance is hard to optimize with GD due to heavy-tailed class imbalance, while the performance of Adam is not negatively impacted.

### 2.2. Reproducing class imbalance issues on image classification with CNNs

To show that class imbalance can also harm the performance of GD on structured data, we reproduce the observed optimization dynamics with convolutional neural networks (CNN) by augmenting the MNIST dataset, which is generally considered "trivial" to learn, to be heavy-tailed.

We start with the MNIST dataset consisting of 50k examples across 10 classes and create $\approx$50k new images across $\approx$10k classes, each with 5 examples. The new examples are copies of existing images with an added "barcode"; a binary number encoded in a corner of the image by setting pixels to black and white. The class of the new image is a combination of the original class and this barcode.

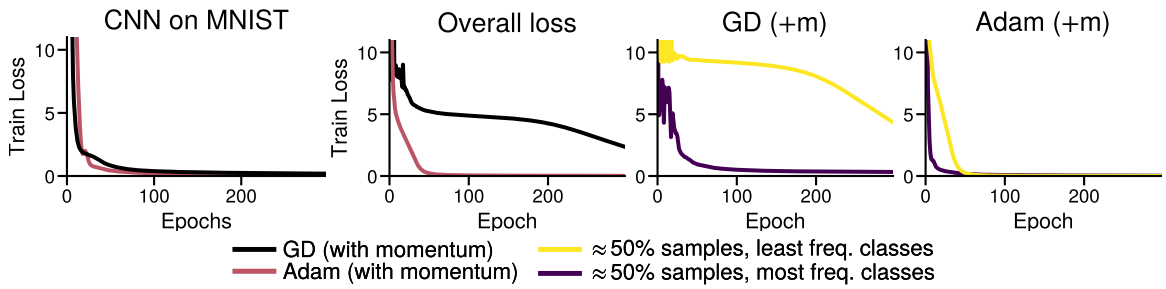



Figure 3: **Class imbalance impacts the optimization dynamics on CNNs on image data. Left:** Performance on the MNIST dataset. **Right:** Performance on a modified MNIST with two groups of classes. The first group consists of the 10 original MNIST classes, while the second consists of ≈10k added classes with 5 examples each.

We train a CNN on the original MNIST and this imbalanced MNIST, shown in Figure 3. GD and Adam can both drive the training loss down on the original MNIST dataset. But on the imbalanced variant, GD makes almost no progress on half of the data corresponding to the new low-frequency classes and the overall training loss stalls while Adam makes progress on both groups.

### 2.3. Interactions between optimizer and class imbalance: magnitude vs. direction

To identify which "component" of Adam makes it less sensitive to the class imbalance problem, we compare additional optimizers across class frequencies. Variants of normalized GD can perform better on separable logistic regression (Nacson et al., 2019), while the main benefits of Adam have been attributed to a similarity to sign descent (see e.g. Tieleman and Hinton, 2012; Balles and Hennig, 2018; Kunstner et al., 2023). We include normalized GD and sign descent, and use each optimizer with and without momentum (using a fixed $\beta = 0.9$ or $\beta_1 = 0.9$).

We present results for training the last layer of a simplified one-layer transformer, freezing the embedding and attention layers at initialization, in Figure 4. We see that normalization alone does not improve on GD, while sign descent behaves similarly to Adam. Momentum improves performance but has less impact on the difference across class frequencies than changing the update direction.

Details on the synthetic datasets, additional token distributions and results are given in Appendix B.

## 3. Discussion

Our experiments highlight a link between properties of the data and the optimization dynamics; GD struggles to drive the training loss down in the presence of heavy-tailed class imbalance, leading to slow convergence, while it does not seem to affect the performance of Adam or sign descent. The heavy-tailed class distribution gives an explanation of why Adam and related sign-based methods exhibit a larger performance improvement over GD in language modeling than on vision problems.

The impact of token imbalance alone on the training dynamics of language models likely runs deeper than the observations made here, as we focus on class imbalance at the classification stage. It is likely that the heavy-tailed distribution of tokens also leads to difficulties elsewhere, for example in training the embedding layer, as rows are only used in samples containing the associated token.

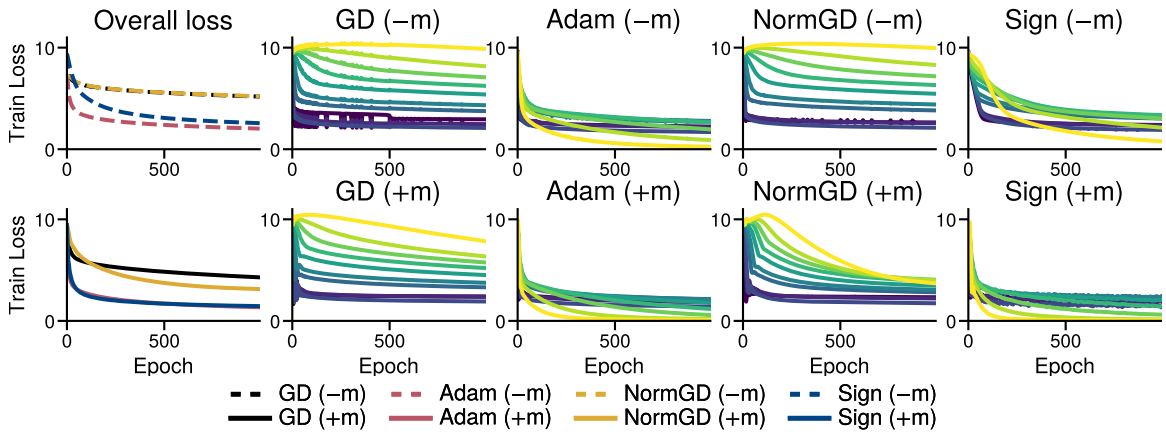



Figure 4: **Sign descent, as Adam, does not suffer from class imbalance.** Training the last layer of a simplified one-layer transformer with GD, Adam, normalized GD, and sign descent, with and without momentum (+m, bottom/−m, top). Sign descent recovers similar dynamics to Adam while momentum or normalizing the magnitude has smaller effects.

We hope that our toy example showing that this behavior appears when training linear models on synthetic data can provide a path toward a theoretical analysis and a deeper explanation of the mechanisms at play. We conclude by discussing connections with prior work.

### 3.1. Class imbalance

The difficulty of fitting imbalanced datasets with GD, and that it primarily minizes the error of higher-frequency classes, has previously been documented by Anand et al. (1993), Ye et al. (2021), and Francazi et al. (2023). Anand et al. and Francazi et al. argue that following the gradient decreases the loss *on average* but need not lead to progress on low-frequency classes, as their impact on the gradient is small. They propose to normalize the gradients of each class before averaging, and provide a theoretical justification that this leads to progress on all classes. Although different from the sign-like operation of Adam, both might have similar effects.

However, class imbalance alone is insufficient to lead to poor performance on average. Consider the extreme example of an imbalanced binary classification with $10k\times$ more examples of one class. GD might not drive down the error of the minority class, but this would only have a limited impact on the *average* loss. The key feature we identify is that this slow convergence is exhacerbated by the scale and heavy-tailed nature of the imbalance; the problem is not only that some classes have fewer samples, but that *most samples* belong to classes with few samples. The frequency of tokens in language tasks is roughly Zipfian (the $i$th most frequent token has a frequency $\propto 1/i$) over a large vocabulary, for example 32k, 50k and 256k tokens for LLaMA, GPT-2 and PALM, respectively (Touvron et al., 2023; Radford et al., 2019; Chowdhery et al., 2022). The slow convergence of GD is more noticeable here than on typical synthetic datasets such as the long-tailed variants of CIFAR-100 and Imagenet (Cui et al., 2019; Liu et al., 2019), which have lighter tails and fewer classes.

Beyond training performance, the class imbalance and long-tail learning literature typically focus on generalization issues brought by class imbalance. While some techniques, such as testing on balanced datasets to promote a uniform error rate across classes, need not be applicable to language tasks, others modifications might prove helpful. Reweighting, resampling or otherwise changing

the objective function (e.g., Chawla et al., 2002; Zhou and Liu, 2006; Menon et al., 2021), as was done for Word2Vec (Mikolov et al., 2013), could improve the performance of GD in the heavy-tailed setting. Especially if the model is sufficiently overparameterized, resampling or reweighting could improve the optimization dynamics without significantly changing the optimal solutions.

### 3.2. Theoretical justifications for the improved performance of Adam

The work of Anand et al. (1993) and Francazi et al. (2023) show that per-class normalization can decrease the loss for all classes. While progress on all classes seems intuitively preferable, the argument does not show an improvement in convergence over GD. Adam, RMSProp (Tieleman and Hinton, 2012) and variants suffer from similar issues. Although they are by now the default in many applications, the reason for their improvement over GD is poorly understood, and worst-case guarantees based on standard assumptions give worse guarantees for Adam than GD (e.g., Défossez et al., 2022). The limited explanations provided by existing theory has motivated works investigating alternative assumptions and properties of the problem that could explain why Adam outperform GD.

Related to the heavy-tail behavior explored here, Zhang et al. (2020b) provide evidence that the larger performance gap between GD and Adam on language tasks, compared to vision tasks, coincides with heavier tails in the noise of the stochastic gradients. They show that Adam-like modifications are more robust to heavy-tailed noise than SGD, and can lead to better performance. However, Kunstner et al. (2023) showed that the performance gap remains when noise is removed, indicating that the difference in relative performance is deterministic in nature. Our work provides an alternative view of the difficulties induced by the heavy-tailed nature of language data through class imbalance.

Adam and related adaptive methods such as AdaGrad were informally justified as approximating second-order methods (Duchi et al., 2011; Kingma and Ba, 2015). However, they were not designed to guarantee that the preconditioner approximates second-order information, and this interpretation does not hold, even on simple problems (Kunstner et al., 2019). But this does not preclude the possibility that Adam works better for problems where the gradient *happens* to be a good proxy for the Hessian. Indeed, empirical evidence suggests that the direction used by Adam leads to better local progress (Pan and Li, 2023) and that the gradient and Hessian have similar trends across coordinates (e.g., Singh and Alistarh, 2020). Zhang et al. (2020a) provide additional evidence of this behavior and a relaxed smoothness assumption to justify why clipping and normalization outperform GD. This assumption was extended to justify element-wise clipping or sign-like methods by Crawshaw et al. (2022). However, we do not yet have a clear picture as to why this relationship arises. Heavy-tailed class imbalance provides grounds to further study the relationships between the gradient and Hessian, as both should depend on relative frequencies. However, the dynamics of the gradients and Hessian are non-trivial, even the linear model of Figure 2, as shown in Figure 10, Appendix B.6.

One of the motivations of RMSProp was to do a smoothed form of sign descent, as analyzed by Balles and Hennig (2018). Despite the success of other sign-based approaches such as LION (Chen et al., 2023), the primary benefit of sign-based methods advertised in the optimization literature is its cheap communication cost (Seide et al., 2014; Karimireddy et al., 2019; Safaryan and Richtárik, 2021). Heavy-tailed class imbalance provides a simple setting where sign-based methods outperform gradient methods. It is promising to investigate whether a provable improvement in performance can be shown for sign-like methods in this setting, using the $L_\infty$, $L_2$ and $L_1$ geometry arguments of Bernstein et al. (2018) and Balles et al. (2020), or the invariance argument of Zhuang et al. (2022).

**Acknowledgements**

We thank Greg d'Eon, Aaron Mishkin, and Victor Sanches Portella for useful discussions and comments on the manuscript. This research was partially supported by the Canada CIFAR AI Chair Program, the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grants RGPIN-2022-03669 and an Undergraduate Student Research Award, the Borealis AI Global Fellowship Award, and was enabled by the support provided by the BC DRI Group and the Digital Research Alliance of Canada (`alliancecan.ca`).

## References

Rangachari Anand, Kishan G. Mehrotra, Chilukuri K. Mohan, and Sanjay Ranka (1993). "An improved algorithm for neural network classification of imbalanced training sets". In: *IEEE Transactions on Neural Networks* 4.6, pp. 962–969.

Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). "Layer Normalization". In: *Advances in Neural Information Processing Systems, Deep Learning Symposium.*

Lukas Balles and Philipp Hennig (2018). "Dissecting Adam: The Sign, Magnitude and Variance of Stochastic Gradients". In: *International Conference on Machine Learning*. Vol. 80, pp. 413–422.

Lukas Balles, Fabian Pedregosa, and Nicolas Le Roux (2020). *The Geometry of Sign Gradient Descent*. arXiv/2002.08056.

Yoshua Bengio, Patrice Y. Simard, and Paolo Frasconi (1994). "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166.

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar (2018). "SIGNSGD: Compressed Optimisation for Non-Convex Problems". In: *International Conference on Machine Learning*. Vol. 80, pp. 559–568.

Tom B. Brown et al. (2020). "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*.

Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer (2002). "SMOTE: Synthetic Minority Over-sampling Technique". In: *Journal of Artificial Intelligence Research* 16, pp. 321–357.

Xiangning Chen et al. (2023). "Symbolic Discovery of Optimization Algorithms". In: *CoRR* abs/2302.06675.

Aakanksha Chowdhery et al. (2022). "PaLM: Scaling Language Modeling with Pathways". In: Preprint. arXiv/2204.02311.

Michael Crawshaw, Mingrui Liu, Francesco Orabona, Wei Zhang, and Zhenxun Zhuang (2022). "Robustness to Unbounded Smoothness of Generalized SignSGD". In: *Advances in Neural Information Processing Systems*.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge J. Belongie (2019). "Class-Balanced Loss Based on Effective Number of Samples". In: *Conference on Computer Vision and Pattern Recognition*, pp. 9268–9277.

Felix Dangel, Frederik Kunstner, and Philipp Hennig (2020). "BackPACK: Packing more into Backprop". In: *International Conference on Learning Representations*.

Alexandre Défossez, Leon Bottou, Francis Bach, and Nicolas Usunier (2022). "A Simple Convergence Proof of Adam and Adagrad". In: *Transactions on Machine Learning Research*.

John C. Duchi, Elad Hazan, and Yoram Singer (2011). "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12, pp. 2121–2159.

Emanuele Francazi, Marco Baity-Jesi, and Aurélien Lucchi (2023). "A Theoretical Analysis of the Learning Dynamics under Class Imbalance". In: *International Conference on Machine Learning*. Vol. 202, pp. 10285–10322.

Philip Gage (1994). "A new algorithm for data compression". In: *C Users Journal* 12.2, pp. 23–38.

Bobby He, James Martens, Guodong Zhang, Aleksandar Botev, Andrew Brock, Samuel L. Smith, and Yee Whye Teh (2023). "Deep Transformers without Shortcuts: Modifying Self-attention for Faithful Signal Propagation". In: *International Conference on Learning Representations*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Sai Praneeth Karimireddy, Quentin Rebjock, Sebastian U. Stich, and Martin Jaggi (2019). "Error Feedback Fixes SignSGD and other Gradient Compression Schemes". In: *International Conference on Machine Learning*. Vol. 97, pp. 3252–3261.

Diederik P. Kingma and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*.

Taku Kudo (2018). "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates". In: *Annual Meeting of the Association for Computational Linguistics*, pp. 66–75.

Frederik Kunstner, Jacques Chen, Jonathan Wilder Lavington, and Mark Schmidt (2023). "Noise is not the main factor behind the gap between SGD and Adam on transformers, but sign descent might be". In: *International Conference on Learning Representations*.

Frederik Kunstner, Philipp Hennig, and Lukas Balles (2019). "Limitations of the empirical Fisher approximation for natural gradient descent". In: *Advances in Neural Information Processing Systems*, pp. 4158–4169.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE*. Vol. 86. 11, pp. 2278–2324.

Liyuan Liu, Xiaodong Liu, Jianfeng Gao, Weizhu Chen, and Jiawei Han (2020). "Understanding the Difficulty of Training Transformers". In: *Conference on Empirical Methods in Natural Language Processing*, pp. 5747–5763.

Ziwei Liu, Zhongqi Miao, Xiaohang Zhan, Jiayun Wang, Boqing Gong, and Stella X. Yu (2019). "Large-Scale Long-Tailed Recognition in an Open World". In: *Conference on Computer Vision and Pattern Recognition*, pp. 2537–2546.

Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar (2021). "Long-tail learning via logit adjustment". In: *International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher (2017). "Pointer Sentinel Mixture Models". In: *International Conference on Learning Representations*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*, pp. 3111–3119.

Mor Shpigel Nacson, Jason D. Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry (2019). "Convergence of Gradient Descent on Separable Data". In: *International Conference on Artificial Intelligence and Statistics*. Vol. 89, pp. 3420–3428.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurélien Lucchi (2022). "Signal Propagation in Transformers: Theoretical Perspectives and the Role of Rank Collapse". In: *Advances in Neural Information Processing Systems*.

Antonio Orvieto, Jonas Kohler, Dario Pavllo, Thomas Hofmann, and Aurélien Lucchi (2022). "Vanishing Curvature in Randomly Initialized Deep ReLU Networks". In: *International Conference on Artificial Intelligence and Statistics*. Vol. 151, pp. 7942–7975.

Yan Pan and Yuanzhi Li (2023). *Toward Understanding Why Adam Converges Faster Than SGD for Transformers*. NeurIPS 2022 Workshop on Optimization for Machine Learning. arXiv/2306.00204.

Adam Paszke et al. (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*, pp. 8024–8035.

F. Pedregosa et al. (2011). "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). *Language Models are Unsupervised Multitask Learners*. Tech. Report.

Mher Safaryan and Peter Richtárik (2021). "Stochastic Sign Descent Methods: New Algorithms and Better Theory". In: *International Conference on Machine Learning*. Vol. 139, pp. 9224–9234.

Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu (2014). "1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs". In: *Annual Conference of the International Speech Communication Association*, pp. 1058–1062.

Rico Sennrich, Barry Haddow, and Alexandra Birch (2016). "Neural Machine Translation of Rare Words with Subword Units". In: *Annual Meeting of the Association for Computational Linguistics*.

Sidak Pal Singh and Dan Alistarh (2020). "WoodFisher: Efficient Second-Order Approximation for Neural Network Compression". In: *Advances in Neural Information Processing Systems*.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958.

Tijmen Tieleman and Geoffrey Hinton (2012). *RMSPROP: Divide the gradient by a running average of its recent magnitude*. Lecture notes
http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.

Hugo Touvron et al. (2023). *LLaMA: Open and Efficient Foundation Language Models*. Preprint. arXiv/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Han-Jia Ye, De-Chuan Zhan, and Wei-Lun Chao (2021). "Procrustean Training for Imbalanced Deep Learning". In: *International Conference on Computer Vision*, pp. 92–102.

Shuangfei Zhai, Tatiana Likhomanenko, Etai Littwin, Dan Busbridge, Jason Ramapuram, Yizhe Zhang, Jiatao Gu, and Joshua M. Susskind (2023). "Stabilizing Transformer Training by Preventing Attention Entropy Collapse". In: *International Conference on Machine Learning*. Vol. 202, pp. 40770–40803.

Jingzhao Zhang, Tianxing He, Suvrit Sra, and Ali Jadbabaie (2020a). "Why Gradient Clipping Accelerates Training: A Theoretical Justification for Adaptivity". In: *International Conference on Learning Representations*.

Jingzhao Zhang, Sai Praneeth Karimireddy, Andreas Veit, Seungyeon Kim, Sashank J. Reddi, Sanjiv Kumar, and Suvrit Sra (2020b). "Why are Adaptive Methods Good for Attention Models?" In: *Advances in Neural Information Processing Systems*, pp. 15383–15393.

Zhi-Hua Zhou and Xu-Ying Liu (2006). "Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem". In: *IEEE Transactions on Knowledge and Data Engineering* 18.1, pp. 63–77.

Zhenxun Zhuang, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona (2022). "Understanding AdamW through Proximal Methods and Scale-Freeness". In: *Transactions of Machine Learning Research*.

# Appendix



## Appendix A. Implementation and reproducibility

The next few sections document the datasets, synthetic datasets, models, experimental setup and the software use to generate our experiments. Exact details can be checked in the accompagnying code,

`(code to be released)`

### A.1. Datasets

The MNIST dataset (LeCun et al., 1998) is used in our experiment on CNNs in Figure 3.

The PTB dataset is used for the language modeling task in Figure 1 and further experiments in Appendix B. We use the dataset for a language modeling task using a word-based tokenizer (`basic_english` provided by `torchtext`) using sequences of 35 tokens.

The WikiText-2 dataset (Merity et al., 2017) is used in Appendix B.1 to illustrate that other combinations of datasets and tokenizers lead to heavy-tailed distributions.

### A.2. Synthetic datasets

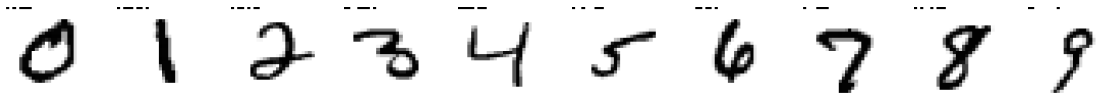
We give names to the synthetic datasets for easier references throughout the appendix. **HT-Random** is the dataset used in Figure 2 and Figure 10, **Barcode MNIST** is the dataset used in Figure 3, and **HT-Linear** and **HT-Cube** are additional datasets described and used in experiments in Appendix B.

**HT-Random.** To generate the labels, we use "step"-based approximation to Zipf's law, creating

$$1 \text{ class with } 2^{10} \text{ samples}, \quad 2 \text{ classes with } 2^{9} \text{ samples}, \quad ..., \quad 2^{10} \text{ classes with } 2 \text{ samples}.$$

This leads to $c = 2\,047$ $(2^{11} - 1)$ classes split into 11 "groups" that each contain $1/11$th of the data. The difference in relative class frequency between the most and least common classes is $512\times$. The number of samples is $n = 22\,528$ $(11 \times 2^{11})$. We generate inputs independently of the class label, by sampling from a Gaussian, $\mathcal{N}(0, I)$. $d = 24\,576$ $(12 \times 2^{11})$ dimensions. A linear model can still achieve good training accuracy despite the lack of statistical relationship between inputs and outputs.

**Barcode MNIST dataset.** We start with the MNIST dataset consisting of 50k examples across 10 classes and create 51 200 ($5 \times 10 \times 2^{10}$) new images, such that the total dataset contains 101 200 examples, spread across 10 240 ($10 \times 2^{10}$) classes with 5 examples each. The new examples are copies of existing image with an added "barcode", a 10-bit number encoded in a corner of the image, as in the examples below. The class label is a combination of the original class and this barcode.



### A.3. Models

**The 2-layer transformer** used in Figure 1 follow the encoder model described by Vaswani et al. (2017), as provided in PyTorch (Paszke et al., 2019) and is summarized as

$$\text{Embedding} \to 2\times \text{ [Attention} \to \text{Linear} \to \text{ReLU} \to \text{Linear]} \to \text{Classifier.}$$

The model includes LayerNorm, dropout, and skip connections (He et al., 2016; Ba et al., 2016; Srivastava et al., 2014). The embedding dimension and width of the linear layers is 1 000 and the attention modules use 4 heads.

**The simplified transformer** used in Figure 4 uses only one attention layer,

$$\text{Embedding} \to \text{Attention} \to \text{Classifier.}$$

We remove LayerNorm, dropout, and the non-linearity induced by the [Linear $\to$ ReLU $\to$ Linear] part of the transformer module. We freeze the embedding and attention layers are frozen at initialization and only the last classification layer is trained. The model is then equivalent to a linear model using a fixed feature transformation, and the optimization problem is convex.

**The convolutional network** used in Figure 3 is a 2-layer convolution

$$\text{Conv} \to \text{Relu} \to \text{MaxPool} \to \text{Conv} \to \text{Relu} \to \text{MaxPool} \to \text{Linear}$$

**The linear model** used in Figures 2 and 10 uses a bias vector and the cross entropy loss.

### A.4. Training procedures

Our primary focus is on the performance of the optimizers on the training error, using as simple a training procedure as possible. We use a constant step-size throughout training, set by grid search. We start with a sparse grid of powers of 10 $[10^{-6}, 10^{-2}, ..., 10^{1}]$ and increase the density to half-powers around the best step-size. The step-size is selected to minimize the maximum over seeds of the training loss at the end of training. For some settings, this selection still produces runs that are "unstable"; the training loss is smallest at the end of training with this step-size, but the behavior prior training oscillates wildly. When this happens, we use the next smaller step-size, which has similar performance but is more stable throughout training.

The transformer experiment in Figure 1 uses minibatches of 512 sequences (of 35 tokens). Other experiments use the entire dataset to compute updates, using gradient accumulation (computing the gradient through multiple passes) to avoid memory issues.

## Appendix B. Additional results

### B.1. Class distribution for common datasets and tokenizers

Figure 5 provides additional examples of the heavy-tailed distribution of tokens using the basic english tokenizer in `torchtext` (Paszke et al., 2019), Byte-Pair Encoding (BPE, Sennrich et al., 2016; Gage, 1994) and Unigram (Kudo, 2018) on the PTB and WikiText-2 datasets.

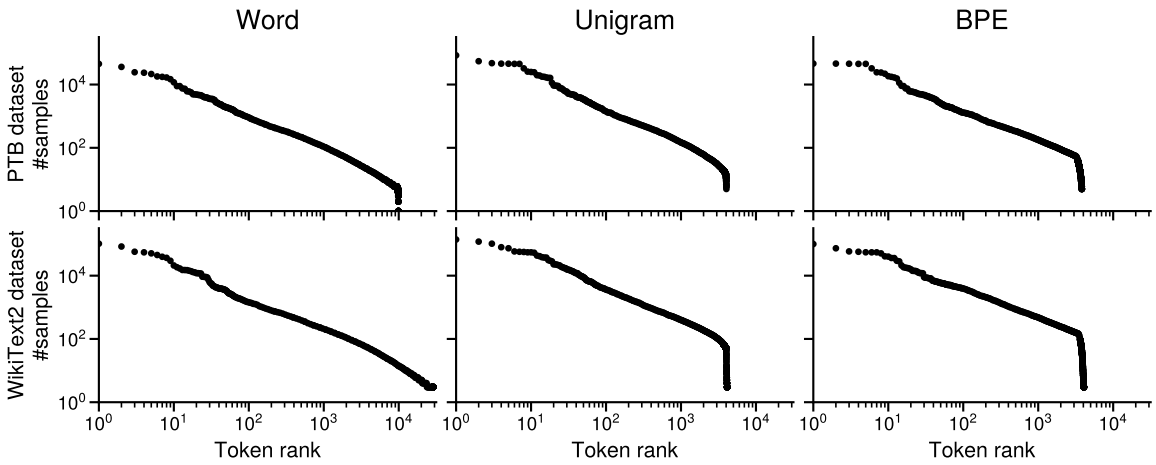



Figure 5: **Different tokenizers and datasets lead to heavy-tailed token distributions.** Comparison of word and subword tokenization (BPE, Unigram) on the PTB and WikiText2 datasets.

### B.2. All optimization dynamics on the HT-Random synthetic dataset (extension to Figure 2)

Figure 6 presents results of the optimization dynamics of all optimizers mentioned in the main text on the HT-Random synthetic dataset used in Figure 2.

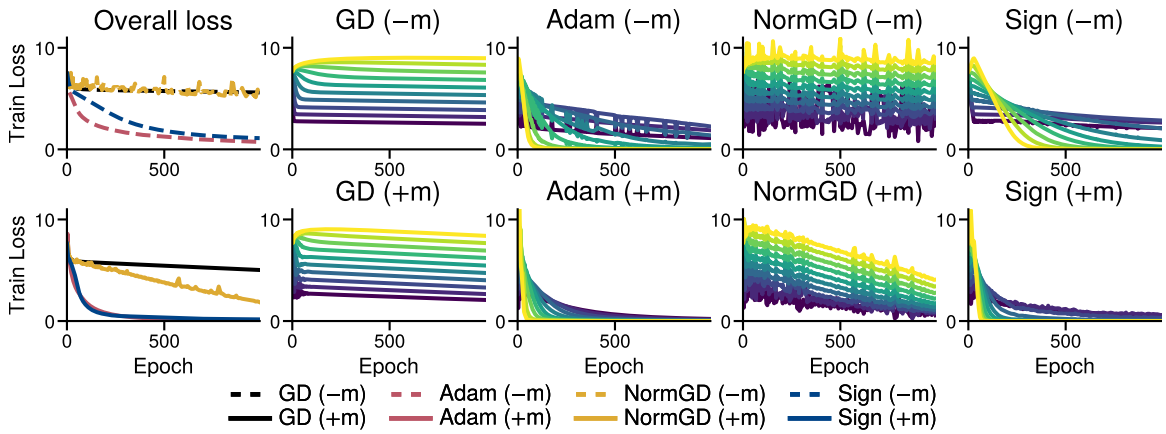



Figure 6: **Additional optimization dynamics on the synthetic imbalanced dataset HT-Random.**

### B.3. All optimization dynamics on Barcode MNIST (extension to Figure 3)

Figure 7 presents results of the optimization dynamics of all optimizers mentioned in the main text on the Barcode MNIST dataset with a CNN used in Figure 3

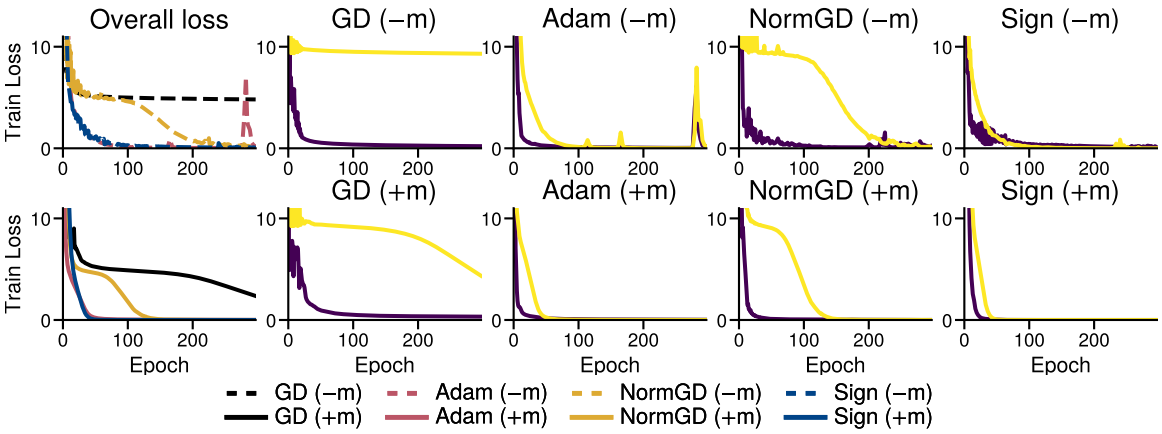



Figure 7: **Additional optimization dynamics on the Barcode MNIST dataset with a CNN.**

### B.4. Additional toy model: Linear regression with squared loss on HT-Linear

We create a linear regression dataset to "mimic" class imbalance in the regression setting, to have an example involving the squared-loss in Figure 8. Adam and sign descent make more progress on the "low-frequency samples" compared to GD (there are no classes as this is a regression problem; see description of the dataset below), but GD with momentum decreases the training error faster.

**The HT-Identity dataset** is similar to HT-Random dataset (Appendix A). We first create groups of classes (1 class with $10^3$ samples, ..., $10^3$ classes with 1 sample). To make it into a regression problem solvable by least squares, the input and output $x, y$ are set to the one-hot encoding of the class, such that the solution of a linear model with weights $W$ is at $W = I$. The heavy-tailed class imbalance induces a heavy-tailed distribution in the eigenvalues of the Hessian, which is block-diagonal. The magnitude of the eigenvalues correspond to the relative frequency of the corresponding class.

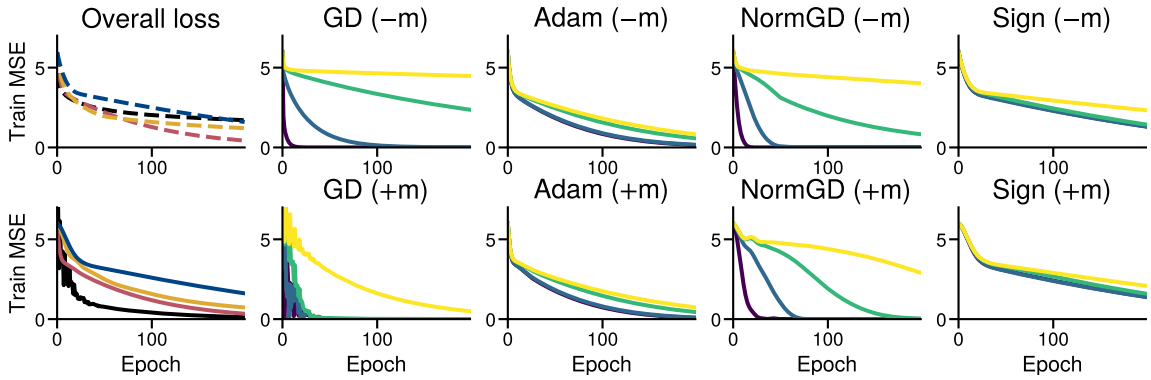



Figure 8: **Additional optimization dynamics on a linear regression problem with squared loss.** Data split by frequency into 4 groups, see text. Adam and sign descent make more progress on "low-frequency samples" than GD, but GD+momentum performs best on training error.

### B.5. Additional toy model: softmax regression on HT-Cube

We generate another synthetic classification dataset with heavy-tailed class imbalance using Scikit Learn's (Pedregosa et al., 2011) `make_classification` function. This dataset is similar to the dataset used for Figure 2, but the data is separable because it is structured, rather than fitting high dimensional noise. The results with all optimizers is shown in Figure 9.

**The HT-Cube dataset.** The labels are generated by sampling from a heavy-tailed distribution, sample a class $c$ according to a power-law $p(c = i) = 1/(e + i)$. To each class $c$ corresponds a corner of a 50-dimensional hypercube, denoted $\mu_c$, and the input data is sampled from $\mathcal{N}(\mu_c, I)$.

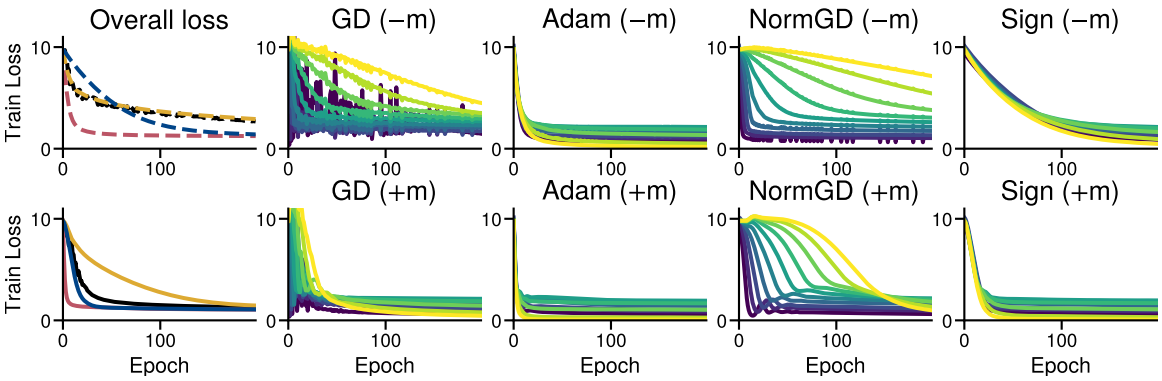



Figure 9: **Optimization dynamics on a softmax regression problem with synthetic data.** The setting is similar to that of Figure 2, but the data is separable because it is structured, rather than fitting high dimensional noise.

### B.6. Dynamics of the gradient and Hessian

Due to the scale of the problem, we use Backpack's implementation of the Monte-Carlo approximation to the diagonal of the Fisher information (Dangel et al., 2020), which is equivalent to the Hessian for linear models. Although the model is linear, computing the Hessian directly with $n$ samples, $d$ dimensions and $c$ classes requires $d \times n \times c$ operations. The values of $d, n, c$ are all large enough in the HT-Random dataset to be challenging (on the order of $10^4, 10^4, 10^3$, respectively).

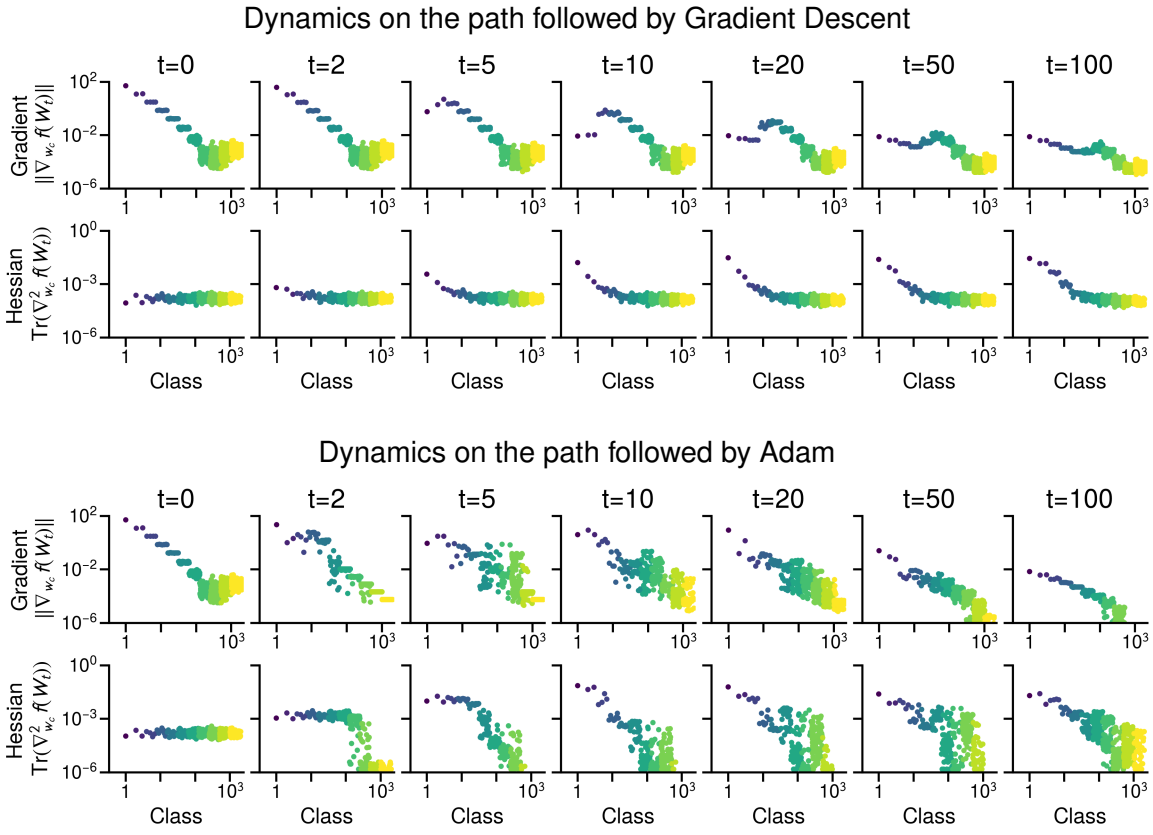



Figure 10: **Evolution of the gradient norm and Hessian trace through iterations, GD and Adam.** Both show the metrics for each weight vector $w_c$ used to map to the logits of class $c$ on the linear model of Figure 2. The gradient is initially largest for the weights of the majority class, and decays as each class gets fit. The Hessian and model are initially uniform, suggesting that even though the gradient is primarily influenced by high-frequency classes, it is a good descent direction, at least at the start. But as the model improves, the pattern of the relative class frequencies becomes visible in the diagonal of the Hessian, suggesting that smaller step-sizes for the weights associated with high-frequency classes are beneficial after a few steps of training.