How rare events shape the learning curves of hierarchical data

Anonymous Author(s)
Affiliation
Address
email

Abstract

The learning curves of deep learning methods often behave as a power of the dataset size. The theoretical understanding of the corresponding exponent yields fundamental insights about the learning problem. However, it is still limited to extremely simple datasets and idealised learning scenarios, such as the lazy regime where the network acts as a kernel method. Recent works study how deep networks learn synthetic classification tasks generated by probabilistic context-free grammars: generative processes which model the hierarchical and compositional structure of language and images. Previous studies assumed composition rules to be equally likely, leading to non-power-law behavior for classification. In realistic dataset, instead, some rules may be much rarer than others. By assuming that the probabilities of these rules follow a Zipf law with exponent a, we show that the classification performance of deep neural networks decays as a power $\alpha = a/(1+a)$ of the number of training examples, with a large multiplicative constant that depends on the hierarchical structure of the data.

1 Introduction

The improvement in performance of many machine-learning models with the amount of resources, including number of model parameters and training examples, has been shown to follow a simple power-law behaviour across several orders of magnitude [1, 2]. These power laws, known as *neural scaling laws*, are used in practice as a guideline for scaling up resources [3, 4]. Furthermore, they offer the possibility of explaining complex learning systems with only a few exponents.

Among scaling laws, the *learning curve* describes the improvement of test performance with the number of training examples. A simple approach, based on pure memorisation, leads to power-law learning curves after assuming Zipf distributed data [5, 6, 7]. In practice, however, data are rarely seen twice as this viewpoint assumes. Alternatively, power-law learning curves can occur in simple machine learning methods such as kernel regression. It occurs if the coefficient of the true function in the kernel basis decay as a power-law of rank, as expected from bounds [8, 9] or direct estimation of exponents [10, 11, 12, 13, 14, 15, 16], a hypothesis confirmed emprically [10, 11, 12]. However, these approaches are restricted to kernel-based approximations of deep learning methods, whose limited power cannot explain the successes of modern, large language and vision models.

In this respect, recent studies have identified hierarchical generative models such as probabilistic context-free grammars as model datasets that explain the difference in performance and data efficiency between deep learning methods and kernels or other shallow methods, while still simple enough to allow for some analytical understanding [17, 18, 19, 20, 21, 22, 23]. For these tasks, the input data are generated from their class labels according to a hierarchy of production rules, mapping high-level features to tuples of lower-level features. If productions rules are evenly distributed, learning curves are not power-law for classification problems [19], although they are for next-token prediction [22].

Submitted to 38th Conference on Neural Information Processing Systems (NeurIPS 2024). Do not distribute.

In this work, we combine one such data model—the Random Hierarchy Model (RHM) of [19]—with the hypothesis that the production rules of real datasets (e.g. the words in a text corpus) are Zipf distributed. For classification, we find that the performance of deep neural networks decays as a power $\alpha = a/(1+a)$ of the number of training examples, with a large multiplicative constant that depends on the hierarchical structure of the data.

2 Notation and setup

Hierarchical generative model. We consider synthetic datasets that model the hierarchical and compositional structure of real data such as images and text. These datasets are generated via a probabilistic context-free grammar (PCFG) [24]: a collection of symbols and rules that prescribe how to generate input data starting from their label. The PCFGs we consider consist of

- L finite vocabularies of hidden (nonterminal) symbols $(\mathcal{V}_\ell)_{\ell=1,\dots,L}$;
- A finite vocabulary of observable (terminal) symbols $\mathcal{V} \equiv \mathcal{V}_0$;
- L sets of *production rules* describing how one symbols of $\mathcal{V}_\ell$ generates a tuple of (lower-level) symbols of $\mathcal{V}_{\ell-1}$, for $\ell = 1, \dots, L$,

$$\mu^{(\ell)} \to \mu_1^{(\ell-1)}, \dots, \mu_s^{(\ell-1)}, \quad \text{for } \mu^{(\ell)} \in \mathcal{V}_\ell, \mu_i^{(\ell-1)} \in \mathcal{V}_{\ell-1}, \tag{1}$$

for some integer size $s \geq 1$. We further assume that *i)* the label set $\mathcal{V}_L$ has size n_c and all the other vocabularies have size v; *ii)* Each hidden symbol of level $\ell = 1, \dots, L$ enters in m distinct production rules and each of these rules can be picked with probability $f_i^{(\ell)}$, with $i = 1, \dots, m$ and $\sum_i f_i^{(\ell)} = 1$; *iii)* two different high-level symbols cannot generate the same lower-level s-tuple, i.e. low-level s-tuples determine the corresponding higher-level symbol *unambiguously*.

In the Random Hierarchy Model (RHM) of [19], the production rules are sampled uniformly among all the possible sets of rules compatible with the constraints above, and their probabilities $f_i^{(\ell)}$ are set to $1/m$ for all i's and ℓ's. Here, mimicking the power-law distribution of word frequencies [25], we set the production rule distribution to be uniform in all but one layer ℓ, where it follows a Zipf law [5, 6], $f_k^{(\ell)} \propto k^{-(1+a)}$.

Once the model is specified, input data are generated by picking a class label y (or level-L symbol) uniformly at random, then picking a production rule emanating from that label and replacing the label with the right-hand side of the production rule. Repeating the process L times yields the input sequence $x = (x_1, \dots, x_d)$, with $d = s^L$. Each feature x_i is represented as a v-dimensional one-hot vector $(x_{i,\mu})_{\mu=1,\dots,v}$, with $x_{i,\mu} = 1$ if x_i encodes for the μ-th element of the vocabulary and 0 otherwise. The probability of the input conditioned on the label, $\mathbb{P}\{X_1 = x_1, \dots, X_d = x_d | Y = y\}$, is given by multyplying the probabilities of all the prodiction rules involved.

Learning Setup. We focus on deep convolutional networks (CNNs) trained for classification by gradient descent over the empirical cross-entropy loss,

$$\mathcal{L}(\mathcal{X}_P) = -\frac{1}{P} \sum_{(\boldsymbol{x},y)\in\mathcal{X}_P} \log\left(p_\theta(y|x_1, \dots, x_d)\right), \tag{2}$$

where $\mathcal{X}_P$ is a set of P training examples drawn from the joint input-label distribution and p_θ denotes the parametric approximation of the label proability,

$$p_\theta(y|x_1, \dots, x_d) \approx \mathbb{P}\{Y = y | X_1 = x_1, \dots, X_d = x_d\}. \tag{3}$$

Numerical experiments are performed in PyTorch [26], with the code attached as Supplemental Material. Details of the machine learning models, training hyperparameters and computer resources are presented in App.A.

3 Input-label correlations and sample complexity

Correlations versus sampling noise. The analysis of [19] shows that, in the uniform production rules case, the sample complexity of deep neural networks trained on RHM data is controlled by the

correlations between the class label and the s-tuples of low-level features. Indeed, these correlations can be used to group together tuples corresponding to the same higher-level variable, allowing to learn the generative model bottom up. This correlation is measured via the probability for a datum to belong to class y conditioned on displaying the s-tuple $\boldsymbol{\mu}$ in the j-th input patch,

$$p_j^L(y|\boldsymbol{\mu}) \coloneqq \mathbb{P}\left\{Y = y | X_{(j-1)s+1} = \mu_1, \dots, X_{js} = \mu_s\right\}_L , \tag{4}$$

with the subscript L indicating explicitly the depth of the generative model. As shown in [19], these correlations can be thought of as random variables over different realisations of the RHM, with mean $1/n_c$ and variance $v/(n_c^2 m^L)$. The mean coincides with the uniform probability over classes, thus it is uninformative. Removing the mean results in a 'signal' that can be used to solve the task. However, when measuring correlations from a set of P training examples, these are affected by a sampling noise of zero mean and variance $v/(n_c P)$. Comparing the sizes of noise and signal results in the sample complexity $P^* = n_c m^L$ actually observed.

3.1 Nonuniform production rules at the bottom layer

Rare productions rules will contribute little to correlations, thus a larger training set will be required to detect their effects and learn these rare rules. Assuming that these rules are learnt precisely when their effect on the correlation becomes detectable offers a perspective on the learning curve, which we will examine further below.

The probability of a low-level tuple conditioned on the class reads

$$\mathbb{P}\left\{\boldsymbol{X}_j = \boldsymbol{\mu} | Y = y\right\}_L = f^{(1)}_{k(\boldsymbol{\mu})} \mathbb{P}\left\{X_j = \mu_1(\boldsymbol{\mu}) | Y = y\right\}_{L-1} , \tag{5}$$

where $f^{(1)}_{k(\boldsymbol{\mu})}$ is the probability of the unique production rule that generates $\boldsymbol{\mu}$, and $\mu_1(\boldsymbol{\mu})$ the unique level-1 features that generates $\boldsymbol{\mu}$ via that production rule. Summing over y yields a similar result for the probability of $\boldsymbol{\mu}$: $\mathbb{P}\left\{\boldsymbol{X}_j = \boldsymbol{\mu}\right\}_L = f^{(1)}_{k(\boldsymbol{\mu})} \mathbb{P}\left\{X_j = \mu_1(\boldsymbol{\mu})\right\}_{L-1}$.

Since $p_j^L(y|\boldsymbol{\mu})$ is proportional to the ratio $\mathbb{P}\left\{\boldsymbol{X}_j = \boldsymbol{\mu} | Y = y\right\}_L / \mathbb{P}\left\{\boldsymbol{X}_j = \boldsymbol{\mu}\right\}_L$, it is independent of $f^{(1)}_{k(\boldsymbol{\mu})}$ and identical to the model with uniform production rules. However, the sampling noise is affected by the probability of the production rules, as the number of data with $\boldsymbol{X}_j = \boldsymbol{\mu}$ is proportional to $f^{(1)}_{k(\boldsymbol{\mu})}$. With respect to the case where all low-level tuples have the same probability, this effect is equivalent to replacing P by $P f^{(1)}_{k(\boldsymbol{\mu})} m$, which recovers in the uniform case $f^{(1)}_{k(\boldsymbol{\mu})} = 1/m$). Thus it results in a sampling noise of variance $v/(n_c f^{(1)}_{k(\boldsymbol{\mu})} P)$. Therefore, the sample size necessary to resolve the correlations of the tuple $\boldsymbol{\mu}$ with the label is $P^*(\boldsymbol{\mu}) = n_c m^{L-1}/(f^{(1)}_{k(\boldsymbol{\mu})})$.

Ranking all the low-level tuples by the probability of the corresponding production rules yields a sequence of m sample complexities $P_k^* = n_c m^{L-1}/(f_k^{(1)})$. To estimate the learning curve, we assume that, when $P > P_k^*$, the model can correctly classify data consisting of tuples with probability higher than f_k. Then the model could correctly classify if and only if all s^{L-1} input layer patches are resolvable. The resulting test error, defined as the probability of misclassification, reads

$$\varepsilon(P) = 1 - \left(\sum_{k | P_k^* < P} f_k^{(1)} \right)^{s^{L-1}} . \tag{6}$$

When $P \gg P_1^* \simeq n_c m^{L-1}$, this expression implies, as shown in App.B,

$$\varepsilon(P) \simeq s^{L-1} \left(\frac{P}{n_c m^{L-1}} \right)^{-a/(1+a)} . \tag{7}$$

These arguments are extended to nonuniform production rules in an arbitrary layer in App.C.

4 Empirical scaling laws of deep CNNs

We confirm the results of Eqs. 6,7 with the measurements of learning curves of CNNs learning the RHM for (i) varying Zipf exponent a in Fig.1.A, (ii) varying the layer where the production rules are following a Zipf law in Fig.1.B, (iii) varying the number of production rule per symbol m in Fig.1.C,D. The comparisons are excellent in all cases. Further evidence exploring more parameters are given in App.D.

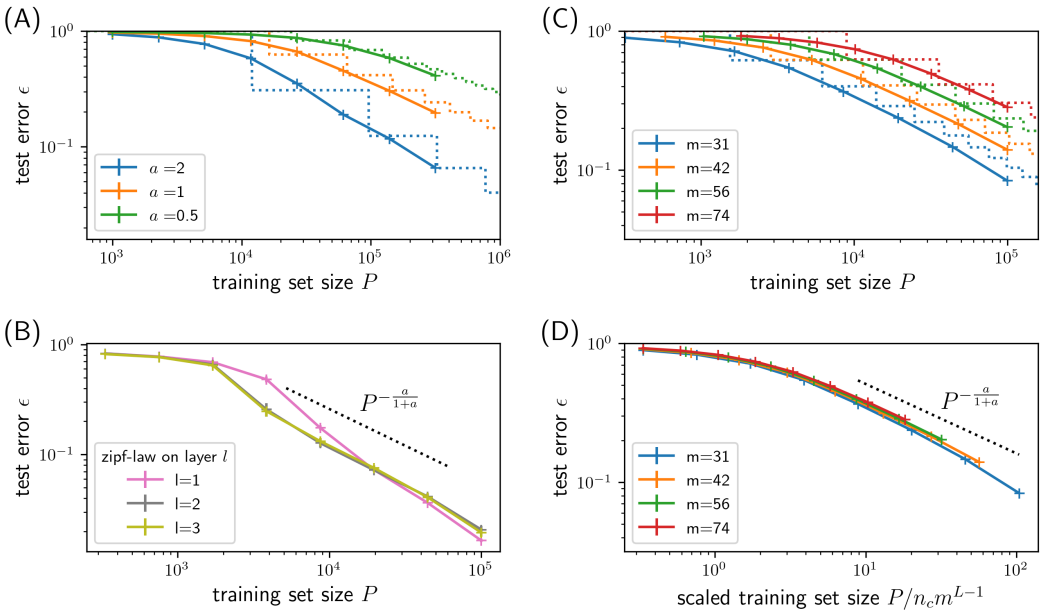

Figure 1: **(A):** Learning curves of CNN trained on $L = 2, m = 100$, zipf-law on input layer($l = 1$) varying zipf-law exponent a. Dotted lines are predictions from Eq.6. **(B):** Learning curves of CNN trained on $L = 3, m = 10, a = 2$, varying the layer implementing Zipf-law. Black dotted line is our scaling exponent from Eq.7. **(C),(D):** Learning curves of CNN trained on $L = 2, a = 1$, varying m. (D) is same plot with (C) except for rescaling x-axis. Dotted lines in (C) is theoretical prediction from Eq.6, Black dotted line in (D) is from Eq.7.

5 Conclusions

In the simplest context-free grammars where the generative tree is fixed, and production rules are uniform and random, the learning curve for classification of the top node is not power-law. Instead, it is characterized by a single sample scale, polynomial in the number of rules m and exponential in the depth L of the generative model. We have shown that if production rules are not uniform but power-law distributed with some exponent a, then the learning curve inherits that property and decays as $m^{(L-1)a/(a+1)} P^{-a/(a+1)}$. The exponent $a/(a+1)$ is also found in elementary toy models of Zipf law learning [3, 4]. Yet, in our case, as in real data, this behaviour is not simply caused by pure memorization, as a datum is never seen twice for large L [1]. Interestingly, we find that the pre-factor in front of the power-law behaviour of the learning curve can be very large, and depends on the combinatorial nature of the problem.

In real data, we expect other factors to shape scaling law exponents, including the shape of the generative tree. For example, the tree depth L does not need to be single-valued, as would occur e.g. for images acquired with different zoom levels. For next-token prediction, other effects associated with long-range correlations, present already for uniform production rules and fixed tree depth, also affect learning curve exponents [22]. It is plausible that among all these effects, the one leading to the smallest exponent would dominate the learning curve. An interesting question for the future is to understand which contribution dominates empirically in which datasets, and estimate parameters characterizing the structure of real data.

References

[1] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Patwary, Mostofa Ali, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[1] The total number of data these models generate is doubly exponential in L, while the sample complexity simply exponential in L.

[2] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[3] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[4] OpenAI. GPT-4 Technical Report, March 2023. arXiv:2303.08774 [cs].

[5] Marcus Hutter. Learning curve theory, 2021.

[6] Eric J. Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling, 2024.

[7] Elvis Dohmatob, Yunzhen Feng, Pu Yang, Francois Charton, and Julia Kempe. A tale of tails: Model collapse as a change of scaling laws. *arXiv preprint arXiv:2402.07043*, 2024.

[8] Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7:331–368, 2007.

[9] F. Bach. The quest for adaptivity. *Machine Learning Research Blog*, 2021.

[10] Stefano Spigler, Mario Geiger, and Matthieu Wyart. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124001, December 2020. Publisher: IOP Publishing.

[11] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum Dependent Learning Curves in Kernel Regression and Wide Neural Networks. In *International Conference on Machine Learning*, pages 1024–1034. PMLR, November 2020. ISSN: 2640-3498.

[12] Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws, 2021.

[13] Alessandro Favero, Francesco Cagnetta, and Matthieu Wyart. Locality defeats the curse of dimensionality in convolutional teacher-student scenarios. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9456–9467. Curran Associates, Inc., 2021.

[14] Alexander Maloney, Daniel A Roberts, and James Sully. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

[15] Francesco Cagnetta, Alessandro Favero, and Matthieu Wyart. What can be learnt with wide convolutional neural networks? In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 3347–3379. PMLR, 23–29 Jul 2023.

[16] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. *arXiv preprint arXiv:2402.01092*, 2024.

[17] Eran Malach and Shai Shalev-Shwartz. A provably correct algorithm for deep learning that actually works. *Preprint at http://arxiv.org/abs/1803.09522*, 2018.

[18] Eran Malach and Shai Shalev-Shwartz. The implications of local correlation on learning some deep functions. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1322–1332. Curran Associates, Inc., 2020.

[19] Francesco Cagnetta, Leonardo Petrini, Umberto M Tomasini, Alessandro Favero, and Matthieu Wyart. How deep neural networks learn compositional data: The random hierarchy model. *Physical Review X*, 14(3):031001, 2024.

[20] Antonio Sclocchi, Alessandro Favero, and Matthieu Wyart. A phase transition in diffusion models reveals the hierarchical nature of data. *arXiv preprint arXiv:2402.16991*, 2024.

[21] Umberto Tomasini and Matthieu Wyart. How deep networks learn sparse and hierarchical data: the sparse random hierarchy model. *arXiv preprint arXiv:2404.10727*, 2024.

[22] Francesco Cagnetta and Matthieu Wyart. Towards a theory of how the structure of language is acquired by deep neural networks. *arXiv preprint arXiv:2406.00048*, 2024.

[23] Jerome Garnier-Brun, Marc Mézard, Emanuele Moscato, and Luca Saglietti. How transformers learn structured data: insights from hierarchical filtering. *arXiv preprint arXiv:2408.15138*, 2024.

[24] Grzegorz Rozenberg and Arto Salomaa. *Handbook of Formal Languages*. Springer, 1997.

[25] Álvaro Corral, Gemma Boleda, and Ramon Ferrer-i Cancho. Zipf's law for word frequencies: Word forms versus lemmas in long texts. *PLOS ONE*, 10(7):e0129031, July 2015.

[26] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019.

A Methods

A.1 RHM implementation

The code implementing the RHM is available online at https://github.com/pcsl-epfl/ hierarchy-learning/blob/master/datasets/ hierarchical.py. The inputs sampled from the RHM are represented as a one-hot encoding of low-level features so that each input consists of s L pixels and v channels (size s L × v). The input pixels are whitened over channels, i.e., each pixel has zero mean and unit variance over the channels.

A.2 Deep CNNs

The deep CNNs we consider are made by stacking standard convolutional layers. To tailor the network to the structure of the data generative model, we fix both the stride and filter size of these layers to s. Since each layer reduces the spatial dimensionality by a factor s, the input size d must be an integer power of s and the CNNs depth equals log d/ log s. We use the Rectified Linear Unit (ReLU) (x) = max (0, x) as activation function.

A.3 Trainig Procedure

Training is performed within the PyTorch deep learning framework [67]. Neural networks are trained on P training points sampled uniformly at random from the RHM data, using Gradient Descent on the cross-entropy loss. Learning rate is initialised to 1 and follows a cosine annealing schedule over 3×10^4 epochs. Training stops when the training loss reaches 10^{-2} . The performance of the trained models is measured as the classification error on a test set. The size of the test set is set to 2×10^6. Reported results for a given value of RHM parameters are averaged over 10 jointly different instances of the RHM and network initialization for $m \leq 20$, and 3 instances for $m > 20$.

B Asymptotics of the learning curve $\varepsilon(P)$

Definition of $g(P)$:

We define $g(P)$ as the proportion of contributions from the first k' terms:

$$g(P) = \sum_{k=1}^{k'} f_k^{(1)} = \frac{\sum_{k=1}^{k'} k^{-1-a}}{\sum_{k=1}^{m} k^{-1-a}}, \tag{8}$$

where k' is the largest integer k such that $f_k^{(1)} > n_c m^{L-1}/P$.

Derivation of k':

Starting with the condition $P_k^* < P$:

Substituting $f_k^{(1)} = \frac{k^{-1-a}}{\sum_{j=1}^{m} j^{-1-a}}$, this inequality simplifies to:

Rearranging gives:

$$k^{-1-a} > \frac{n_c m^{L-1}}{P} \cdot \sum_{j=1}^{m} j^{-1-a}, \tag{9}$$

$$k' \sim \left(\frac{P}{n_c m^{L-1}} \right)^{\frac{1}{1+a}}. \tag{10}$$

Approximation of $g(P)$:

Now, substituting k' into the expression for $g(P)$:

$$g(P) = \frac{\sum_{k=1}^{k'} k^{-1-a}}{\sum_{k=1}^{m} k^{-1-a}}. \tag{11}$$

Approximation Using the Euler-Maclaurin Formula:

To approximate these sums, we use the Euler-Maclaurin formula, which relates sums to integrals:

$$\sum_{k=a}^{b} f(k) \approx \int_a^b f(x)\,dx + \frac{f(a)+f(b)}{2} + \sum_{j=1}^{n} \frac{B_{2j}}{(2j)!} f^{(2j-1)}(x)\Big|_a^b,$$

where B_{2j} are Bernoulli numbers. For large b, the correction terms diminish, allowing the sum to be closely approximated by the integral.

1. Numerator of $g(P)$:

Approximating the sum from 1 to k':

$$\sum_{k=1}^{k'} k^{-1-a} \approx \int_1^{k'} x^{-1-a}\,dx + \frac{k'^{-1-a} + 1^{-1-a}}{2} + \text{higher-order corrections}, \tag{12}$$

$$\approx \int_1^{k'} x^{-1-a}\,dx, \quad \text{(neglecting boundary terms for large } k'). \tag{13}$$

The boundary term $\frac{1^{-1-a}}{2} = \frac{1}{2}$ is negligible relative to the integral, especially when $a > 0$, because the integral scales with k', which grows large as P increases.

2. Denominator of $g(P)$:

For the sum from 1 to m, we approximate by extending the upper limit to infinity:

$$\sum_{k=1}^{m} k^{-1-a} \approx \int_1^{m} x^{-1-a}\,dx + \frac{m^{-1-a} + 1^{-1-a}}{2} + \text{correction terms}, \tag{14}$$

$$\approx \int_1^{\infty} x^{-1-a}\,dx, \quad \text{(extending to infinity is valid as } m \text{ is large}). \tag{15}$$

For large m, the tail of the integral beyond m contributes negligibly, as x^{-1-a} decays rapidly when $a > 0$. The impact on the result is minimal, making the approximation practically accurate.

Evaluating the Integrals:

$$\int_1^{k'} x^{-1-a}\,dx = \left[\frac{x^{-a}}{-a}\right]_1^{k'} = \frac{1}{a}\left(1 - (k')^{-a}\right), \tag{16}$$

$$\int_1^{\infty} x^{-1-a}\,dx = \left[\frac{x^{-a}}{-a}\right]_1^{\infty} = \frac{1}{a}. \tag{17}$$

Since $m^{-a} \to 0$ for large m, the integral in the denominator simplifies to $\frac{1}{a}$.

Substituting into $g(P)$:

$$g(P) \approx \frac{\frac{1}{a}\left[1-(k')^{-a}\right]}{\frac{1}{a}} \tag{18}$$

$$= 1-(k')^{-a} \tag{19}$$

$$= 1-c\left(\frac{P}{n_c m^{L-1}}\right)^{-\frac{a}{1+a}}, \quad \text{for some constant } c. \tag{20}$$

Approximation of $\varepsilon(P)$:

Substituting $g(P)$ into the expression for $\varepsilon(P)$:

$$\varepsilon(P) = 1-\{g(P)\}^{s^{L-1}} \tag{21}$$

$$\approx 1-\left[1-c\left(\frac{P}{n_c m^{L-1}}\right)^{-\frac{a}{1+a}}\right]^{s^{L-1}}. \tag{22}$$

Using a Taylor expansion around $g(P) \approx 1$ for large P:

$$\approx s^{L-1}\cdot c\left(\frac{P}{n_c m^{L-1}}\right)^{-\frac{a}{1+a}}, \quad \text{for } P \gg n_c m^{L-1}. \tag{23}$$

$$\sim \left(\frac{P}{n_c m^{L-1}}\right)^{-\frac{a}{1+a}} \tag{24}$$

Thus, the final approximation is:

$$\varepsilon(P) \sim \left(\frac{P}{n_c m^{L-1}}\right)^{-\frac{a}{1+a}}. \tag{25}$$

C Nonuniform production rules on an arbitrary layer

When the nonuniform distribution of production rules affects an arbitrary layer $\ell \neq 1$, the probabilities of low-level tuples can be decomposed as sums of conditional probabilities over a specific choice of production rules. As a result, the conditional class probability $p_j^L(y|\mu\boldsymbol{\mu})$ can be written as a sum of contributions due to production rules of a given probability $f_k^{(\ell)}$. Let us assume that the correct classification of data containing production rules with probability $f_k^{(\ell)}$ requires the accurate resolution of the corresponding contribution to the label-tuples correlations. Then, we can apply again the argument of the previous section, and derive Eq. 6 for the behaviour of the test error.

D More empirical results

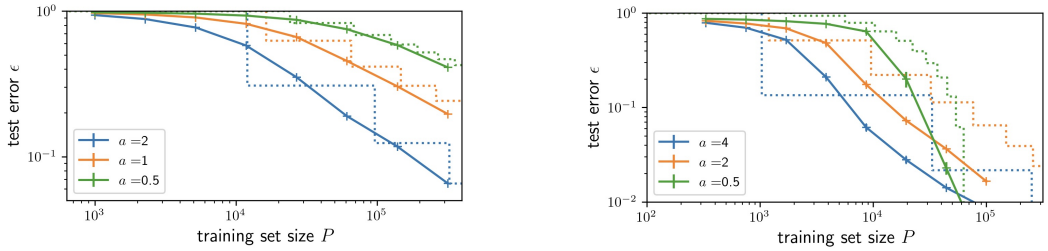

Figure 2: **Left:** Learning curves of CNN trained on $L = 2, m = 100$. Zipf-law on input layer($l = 1$) varying zipf-law exponent a. Dotted lines are predictions from Eq.6. **Right:** Learning curves of CNN trained on $L = 3, m = 10$ (Top) zipf-law on input layer($l = 1$) varying zipf-law exponent a. Dotted lines are from our theoretical prediction.

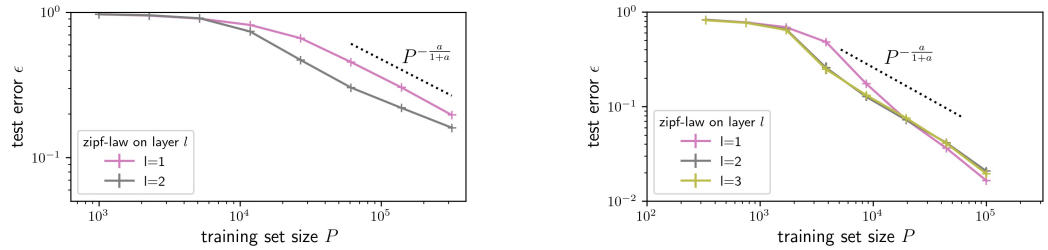

Figure 3: **Left:** Learning curves of CNN trained on $L = 2, m = 100$. Zipf-law with exponent $a = 1$ varying which layer to implement it. **Right:** Learning curves of CNN trained on $L = 3, m = 10$. Zipf-law with exponent $a = 2$ varying which layer to implement it.

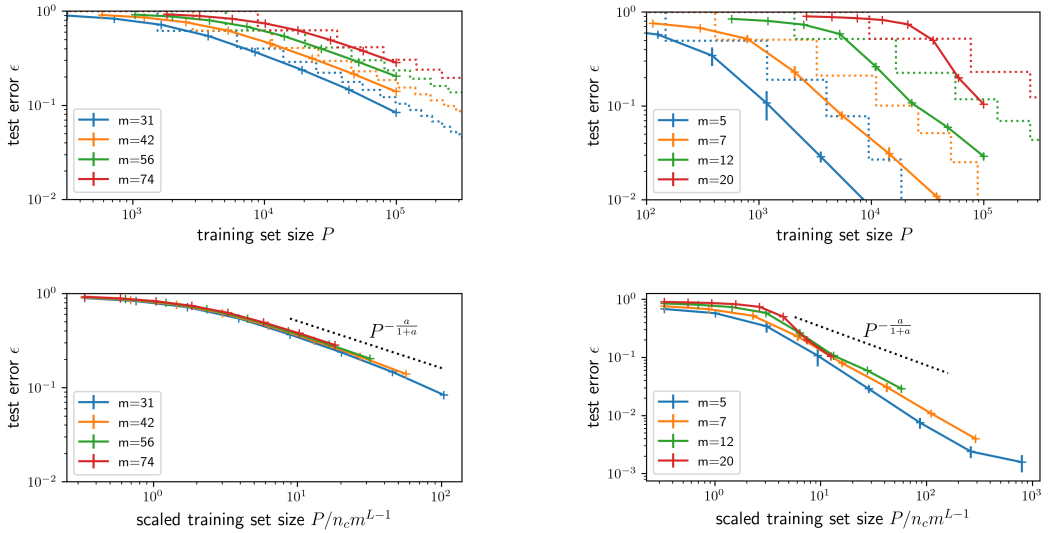

Figure 4: **Left:** Learning curves of CNN trained on $n_c = v = m, L = 2, s = 2$. (Top) zipf-law on last layer with $a = 1$ and varying m. Dotted lines are from our theoretical prediction Eq.6. (Bottom) same plot with x-axis was scaled by dividing $n_c m^{L-1}$ so that curves collapse, as expected from Eq.7. **Right:** Learning curves of CNN trained on $n_c = v = m, L = 3, s = 2$. (Top) zipf-law on last layer with $a = 2$ and varying m. Dotted lines are from our theoretical prediction Eq.6. (Bottom) same plot with x-axis resclaed so that curves collapse. Black dotted line is $P^{-\alpha} = P^{-\frac{a}{1+a}}$ from Eq.7.