Broken Neural Scaling Laws

Ethan Caballero
Mila, McGill University
`ethan.victor.caballero@gmail.com`
`ethan.caballero@mila.quebec`

Kshitij Gupta
Mila, University of Montreal

Irina Rish
Mila, University of Montreal

David Krueger
University of Cambridge

Abstract

We present a smoothly broken power law functional form that accurately models and extrapolates the scaling behaviors of deep neural networks (i.e. how the evaluation metric of interest varies as the amount of compute used for training, number of model parameters, training dataset size, or upstream performance varies) for each task within a large and diverse set of upstream and downstream tasks, in zero-shot, prompted, and fine-tuned settings. This set includes large-scale vision and unsupervised language tasks, diffusion generative modeling of images, arithmetic, and reinforcement learning. When compared to other functional forms for neural scaling behavior, this functional form yields extrapolations of scaling behavior that are considerably more accurate on this set. Moreover, this functional form accurately models and extrapolates scaling behavior that other functional forms are incapable of expressing such as the non-monotonic transitions present in the scaling behavior of phenomena such as double descent and the delayed, sharp inflection points present in the scaling behavior of tasks such as arithmetic. Lastly, we use this functional form to glean insights about the limit of the predictability of scaling behavior. Code is available at `https://github.com/ethancaballero/broken_neural_scaling_laws`

1 Introduction

The amount of compute used for training, number of model parameters, and training dataset size of the most capable artificial neural networks keeps increasing and will probably keep rapidly increasing for the foreseeable future. However, no organization currently has direct access to these larger resources of the future; and it has been empirically verified many times that methods which perform best at smaller scales often are no longer the best performing methods at larger scales (e.g., one of such examples can be seen in Figure 2 (right) of Tolstikhin et al. (2021)). To work on, identify, and steer the methods that are most probable to stand the test-of-time as these larger resources come online, one needs a way to predict how all relevant performance evaluation metrics of artificial neural networks vary in all relevant settings as scale increases.

Neural scaling laws Cortes et al. (1994); Hestness et al. (2017); Rosenfeld et al. (2019); Kaplan et al. (2020); Zhai et al. (2021); Abnar et al. (2021); Alabdulmohsin et al. (2022); Brown et al. (2020) aim to predict the behavior of large-scale models from smaller, cheaper experiments, allowing to focus on the best-scaling architectures and algorithms. The upstream/in-distribution test loss typically (but not always!) falls off as a power law with increasing data, model size and compute. However, the downstream/out-of-distribution performance, and other evaluation metrics of interest (even upstream/in-distribution evaluation metrics) are often less predictable, sometimes exhibiting inflection points (on a linear-linear plot) and non-monotonic behaviors. Discovering *universal scaling* *laws* that accurately model a wide range of potentially unexpected behaviors is clearly important not only for identifying that which scales best, but also for AI safety, as predicting the emergence of novel capabilities at scale could prove crucial to responsibly developing and deploying increasingly advanced AI systems. The functional forms of scaling laws evaluated in previous work are not up to this challenge.

One salient defect is that they can only represent monotonic functions. They thus fail to model the striking phenomena of double-descent (Nakkiran et al., 2021), where increased scale temporarily decreases test performance before ultimately leading to further improvements. They also lack the expressive power to model inflection points (on a linear-linear plot), which can be observed empirically for many downstream tasks, and even some upstream tasks, such as our N-digit arithmetic task, or the modular arithmetic task introduced by Power et al. (2022) in their work on “grokking".

To overcome the above limitations, we present *broken neural scaling laws (BNSL)* - a functional form that generalizes power laws (linear in log-log plot) to “smoothly broken" power laws, i.e. a smoothly connected piecewise (approximately) linear function in a log-log plot. An extensive empirical evaluation demonstrates that BNSL accurately model scaling behaviors across a large and diverse set of both upstream and downstream tasks, in zero-shot, prompted, and fine-tuned settings. This set includes large-scale unsupervised language and vision tasks, arithmetic, and reinforcement learning. This functional form yields more considerably accurate extrapolations of scaling behavior then those achieved in any previous work. It captures well the non-monotonic transitions present in the scaling behavior of phenomena such as double descent and the delayed, sharp transitions present in the scaling behavior of tasks such as arithmetic.

2 The Functional Form of Broken Neural Scaling Laws

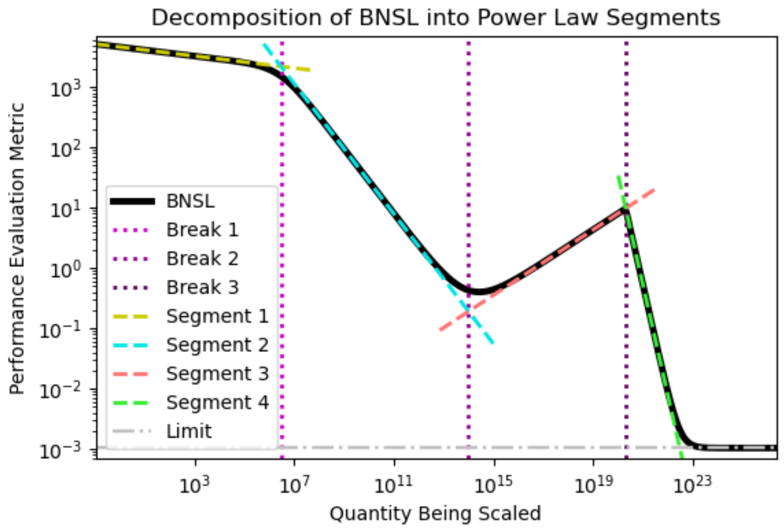


Figure 1: A Broken Neural Scaling Law (BNSL) (dark black solid line) with 3 breaks (where purple dotted lines intersect with dark black solid line) decomposed into the individual power laws (dashed lines that are yellow, blue, red, and green) that it is composed of overlaid on top of it. The 1st and 2nd break are very smooth; the 3rd break is very sharp. See Section 2 for more details.

The general functional form of a broken neural scaling law (BNSL) is given as follows:

$$y = a + \left(bx^{-c_0}\right) \prod_{i=1}^{n} \left(1 + \left(\frac{x}{d_i}\right)^{1/f_i}\right)^{-c_i * f_i}, \tag{1}$$

where y represents the performance evaluation metric (e.g. prediction error, cross entropy, BLEU score percentage, F1 score percentage, reward, Elo rating, or FID score) (downstream or upstream) and x represents a quantity that is being scaled (e.g. number of model parameters, amount of compute used for training, training dataset size, or upstream performance). The remaining parameters

$a, b, c_0, c_1...c_n, d_1...d_n, f_1...f_n$ are unknown constants that must be estimated by fitting the above functional form to the (x, y) data points. (In our experiments, SciPy curve-fitting library (Virtanen et al., 2020) was used.)

The constants in equation 1 are interpreted as follows. Constant n represents the number of (smooth) "breaks" (i.e. transitions) between $n+1$ consecutive approximately linear (on a log-log plot) segments, for a total of $n + 1$ approximately linear segments (on a log-log plot). Constant a represents the limit as to how far the value of y (performance evaluation metric) can be reduced (or maximized) even if x (the quantity being scaled) goes to infinity. Constant b represents the offset of functional form on a log-log plot (analogous to the intercept b in $y = mx + b$ on a linear-linear plot). Constant c_0 represents the slope of the first approximately linear region on a log-log plot. Constant c_i represents the difference in slope of the (i)th approximately linear region and $(i + 1)$th approximately linear region on a log-log plot. Constant d_i represents where on the x-axis the break between the (i)th and the $(i + 1)$th approximately linear region (on a log-log plot) occurs. Constant f_i represents the sharpness of break between the (i)th and the $(i + 1)$th approximately linear region on a log-log plot; smaller (nonnegative) values of f_i yield a sharper break and intervals (before and after the (i)th break) that are more linear on a log-log plot; larger values of f_i yield a smoother break and intervals (before and after the (i)th break) that are less linear on a log-log plot.

Note that, while an intuition for using such approximately piece-wise linear (in log-log plot) function was that, with enough segments, it could fit well any smooth univariate scaling function, it remained unclear whether BNSL would also *extrapolate* well; as we demonstrate below, it does extrapolate quite accurately in all our experiments. Additionally, we find that the number of breaks needed to accurately model an entire scaling behavior is often quite small.

3 Related Work

To the best of our knowledge, Cortes et al. (1994) was the first paper to model the scaling of multilayer neural network's performance as a power law (also known as a scaling law) (plus a constant) of the form $y = ax^b + c$ in which x refers to training dataset size and y refers to test error; we refer to that functional form as M2. Hestness et al. (2017) showed that the functional form, M2, holds over many orders of magnitude. Rosenfeld et al. (2019) demonstrated that the same functional form, M2, applies when x refers to model size (number of parameters). Kaplan et al. (2020) brought "neural" scaling laws to the mainstream and demonstrated that the same functional form, M2, applies when x refers to the amount of compute used for training. Abnar et al. (2021) proposed to use the same functional form, M2, to relate downstream performance to upstream performance. Zhai et al. (2021) introduced the functional form $y = a(x + d)^b + c$, (referred to by us as M3) where d represents the scale at which the performance starts to improve beyond the random guess loss (a constant) and transitions to a power law scaling regime. Alabdulmohsin et al. (2022) proposed functional form $(y - \epsilon_\infty)/((\epsilon_0 - y)^a) = bx^c$, (referred to by us as M4) where ϵ_∞ is irreducible entropy of the data distribution and ϵ_0 is random guess performance, for relating scale to performance and released a scaling laws benchmark dataset that we use in our experiments.

Hernandez et al. (2021) described a smoothly broken power law functional form (consisting of 5 constants after reducing redundant variables) in equation 6.1 of their paper, when relating scale and downstream performance. While this functional form can be summed with an additional constant representing unimprovable performance to obtain a functional form that is mathematically equivalent to our BNSL with a single break, it is important to note that (i) Hernandez et al. (2021) describes this form only in the specific context, when exploring how fine-tuning combined with transfer learning scales as a function of the model size - thus, their functional form contains a break only with respect to the number of model parameters but not with respect to the dataset size (which we do explore); (ii) Hernandez et al. (2021) mentioned this equation in passing and as a result did not try to fit or verify this functional form on any data; (iii) they arrived at it simply via combining the scaling law for transfer (that was the focus of their work) with a scaling law for pretraining data; (iv) they did not identify it as a smoothly broken power law, or note any qualitative advantages of this functional form; (v) they did not discuss the family of functional forms with multiple breaks.

Finally, we would like to mention that smoothly broken power law functional forms, equivalent to equation 1, are commonly used in the astrophysics literature (e.g. dam (2017)) as they happen to model well a variety of physical phenomena. This inspired us to investigate their applicability to a wide range of deep neural scaling phenomena as well.

4 Empirical Results: Fits and Extrapolations of Functional Forms

We now show the fits and/or extrapolations of various functional forms. In all plots here and in the quite large appendix, black points are points used for fitting a functional form, green points are held-out points used for evaluating extrapolation of a functional form fit to the black points, and a red line is BNSL that has been fit to the black points. All the extrapolation evaluations reported in the tables are reported in terms of root mean squared log error (RMSLE[1]) ± root standard log error. See Appendix C.9 for definition of root standard log error. Unless stated otherwise, every BNSL we fit only has one break. Please refer to Appendix Section C.11 for further experimental details on fitting BNSL. All non-math task results (e.g. vision, language, rl, and double descent) are in the appendix.

In the tables and elsewhere, M1 refers to functional form $y = ax^b$, M2 refers to functional form $y = ax^b + c$, M3 refers to functional form $y = a(x^{-1} + d)^{-b} + c$, and M4 refers to functional form $(y - \epsilon_\infty)/((\epsilon_0 - y)^a) = bx^c$.

4.1 Inflection Points

We show that BNSL is capable of modeling and extrapolating the scaling behavior of tasks that have an inflection point on a linear-linear plot such as the task of arithmetic (4-digit addition). Here we model and extrapolate the scaling behavior of a transformer model (Vaswani et al. (2017)) with respect to the dataset size on the 4-digit addition task. Other functional forms are mathematically incapable of expressing inflection points on a linear-linear plot (as shown in Section B) and as a result, are mathematically incapable of expressing and modeling inflection points (on a linear-linear plot) that are present in the scaling behavior of 4-digit addition. In Figure 2 left, we show that BNSL expresses and accurately models the inflection point present in the scaling behavior of 4-digit addition and as a result accurately extrapolates the scaling behavior of 4 digit addition. For further details about the hyperparameters please refer to the Appendix Section C.10. Fit of M3 to 4-digit addition task is also available in Figure 4 of the Appendix for comparison purposes. We also show a fit of BNSL to Large-Scale BIG-Bench Srivastava et al. (2022) 3-shot arithmetic task with number of parameters on the x-axis in Figure 3.

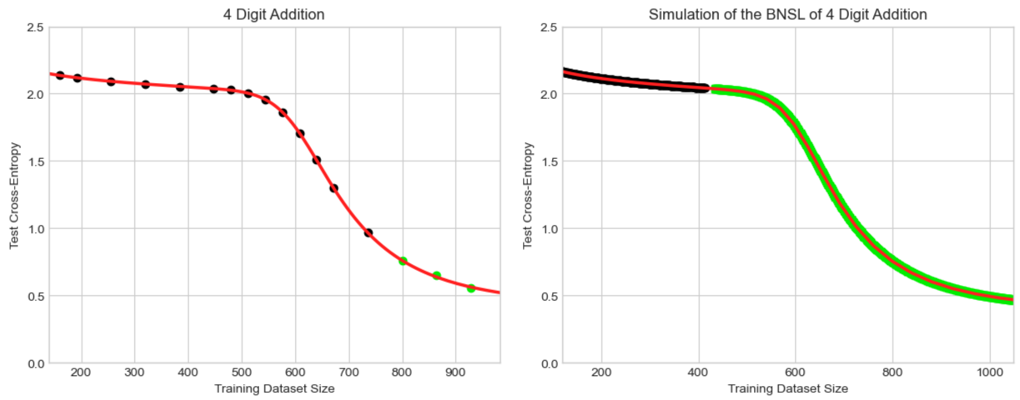


Figure 2: 4 Digit Addition. Note that these plots are linear-linear. Each point in the left plot is the mean of greater than 100 seeds at that dataset size. In the left plot, each point is gathered from a model trained to do the task of 4-digit addition. In the right plot, each point is gathered from a noiseless simulation of the BNSL of the task of 4-digit addition.

[1]RMSLE = $\sqrt{(\sum_{i=1}^{n}(log(y_i) - log(\hat{y}_i))^2)/n}$

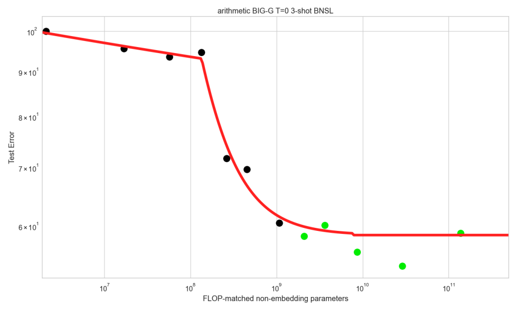


Figure 3: Large-Scale BIG-Bench Arithmetic. Extrapolation yielded by BNSL on BIG-Bench 3-shot Arithmetic task with number of parameters on the x-axis. This experimental data is obtained from the Srivastava et al. (2022) release.

4.2 The Limit of the Predictability of Scaling Behavior

We use BNSL to glean insights about the limit of the predictability of scaling behavior. Recent papers (Ganguli et al., 2022; Wei et al., 2022) have advertised many tasks as having “unpredictable" scaling behavior, the most famous of which is the task of arithmetic. In the previous section and in Figure 2 left, we successfully predicted (i.e. extrapolated) the scaling behavior of 4-digit addition (arithmetic). However, we are only able to accurately extrapolate the scaling behavior if given some points from training runs with a training dataset size of at least 720, and the break in which the scaling behavior of 4-digit addition transitions from one power law to another steeper power-law happens at around training dataset size of 415. Ideally, one would like to be able to extrapolate the entire scaling behavior by fitting only points from before the break. In Figure 2 right, we use a noiseless simulation of the BNSL of 4-digit addition to show what would happen if one had infinitely many training runs / seeds to average out all the noisy deviation between runs such that one could recover (i.e. learn via a curve-fitting library such as SciPy Virtanen et al. (2020)) the learned constant of the BNSL as well as possible. When using this noiseless simulation, we find that we are only able to accurately extrapolate the scaling behavior if given some points from training runs with a training dataset size of at least 415, which is very close to the break. This implies that very near to the break there is limit as to how small the supremum of the x-axis of the points used for fitting can be if one wants to perfectly extrapolate the scaling behavior, even if one has infinitely many seeds / training runs.

5 Conclusions

We have presented a smoothly broken power law functional form that accurately models the scaling behaviors of artificial neural networks for each task from a very large and diverse set of upstream and downstream tasks. These tasks include large-scale vision tasks, large-scale unsupervised language tasks, arithmetic, and reinforcement learning. This functional form yields extrapolations of scaling behavior that are considerably more accurate than other functional forms for modeling the scaling behavior of artificial neural networks. Additionally, this functional form accurately models many scaling behaviors that other functional forms are mathematically incapable of expressing such as non-monotonic transitions present in the scaling behavior of phenomena such as double descent and delayed, sharp inflection points present in the scaling behavior of tasks such as arithmetic. Finally, we used this functional form to obtain insights about the limit of the predictability of scaling behavior.

References

(2017). Direct detection of a break in the teraelectronvolt cosmic-ray spectrum of electrons and positrons. *Nature*, 552(7683):63–66.

Abnar, S., Dehghani, M., Neyshabur, B., and Sedghi, H. (2021). Exploring the limits of large scale pre-training.

Alabdulmohsin, I. M. I., Neyshabur, B., and Zhai, X. (2022). Revisiting neural scaling laws in language and vision. In *NeurIPS 2022*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. (2020). Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR.

Cortes, C., Jackel, L. D., Solla, S. A., Vapnik, V., and Denker, J. S. (1994). Learning curves: Asymptotic values and rate of convergence. In *Advances in Neural Information Processing Systems*, pages 327–334.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Fei-Fei, L., Fergus, R., and Perona, P. (2004). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE.

Ganguli, D., Hernandez, D., Lovitt, L., Askell, A., Bai, Y., Chen, A., Conerly, T., Dassarma, N., Drain, D., Elhage, N., et al. (2022). Predictability and surprise in large generative models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1747–1764.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. (2021). Scaling Laws for Transfer. *arXiv e-prints*, page arXiv:2102.01293.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. (2017). Deep Learning Scaling is Predictable, Empirically. *arXiv e-prints*, page arXiv:1712.00409.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling Laws for Neural Language Models. *arXiv e-prints*, page arXiv:2001.08361.

Karpathy, A. (2020). mingpt. `https://github.com/karpathy/minGPT`.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. (2020). Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. (2021). Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003.

Neumann, O. and Gros, C. (2022). Scaling laws for a multi-agent reinforcement learning model. *arXiv preprint arXiv:2210.00849*.

Nichol, A. Q. and Dhariwal, P. (2021). Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR.

Power, A., Burda, Y., Edwards, H., Babuschkin, I., and Misra, V. (2022). Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*.

Rosenfeld, J. S., Rosenfeld, A., Belinkov, Y., and Shavit, N. (2019). A constructive prediction of the generalization error across scales. *CoRR*, abs/1909.12673.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. (2022). Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852.

Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al. (2021). Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. (2022). Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-ucsd birds 200.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. (2021). Scaling vision transformers. *CoRR*, abs/2106.04560.

A Appendix

B Theoretical Limitations of Previously Proposed Scaling Laws

Our use of BNSLs is inspired by the observation that scaling is not always well predicted by a simple power law; nor are many of the modifications which have been applied in previous works sufficient to capture the qualitative properties of empirical scaling curves. Here we show mathematically two qualitative defects of these functional forms:

1. They are strictly monotonic (first-order derivative does not change its sign) and thus unable to fit double descent phenomena.
2. They cannot express inflection points (second-order derivative does not change its sign), which are frequently observed empirically. An exception to this is M4, proposed by Alabdulmohsin et al. (2022).

Note that these functional forms *can* exhibit inflection points on the log-log axes which are commonly used for plotting scaling data (as it was observed in several prior works). However, for inflection points on a *linear-linear* plot, the extra expressiveness of broken neural scaling laws appears to be necessary (and sufficient). Figure 5 and Figure 2, provide examples of BNSLs producing non-monotonic behavior and inflection points, respectively, establishing the capacity of this functional form to model these phenomena that occur in real scaling behavior.

name	$f(x)$	$f'(x)$	$f''(x)$
M1	ax^b	abx^{b-1}	$ab(b-1)x^{b-2}$
M2	$ax^b + c$	abx^{b-1}	$ab(b-1)x^{b-2}$
M3	$a(x^{-1}+d)^{-b}+c$	$\frac{ab}{x(1+dx)(d+1/x)^b}$	$abx^{(b-2)}(1+dx)^{(-2-b)}(b-1-2dx)$

Table 1: Previously proposed functional forms M1, M2, M3 and their (first and second order) derivatives. See Equation 2 for M4.

M1, M2, M3 functional forms cannot model non-monotonic behavior or inflection points: First, recall that expressions of the form m^n can only take the value 0 if $m = 0$. We now examine the expressions for the first and second derivatives of M1, M2, M3, provided in Table 1, and observe that they are all continuous and do not have roots over the relevant ranges of their variables, i.e. $x > 0$ in general and $b < 0$ in the case of M3 (we require $x > 0$ because model size, dataset size, and compute are always non-negative). This implies that, for any valid settings of the parameters a, b, c, d, x, these functional forms are monotonic (as the first derivative never changes sign), and that they lack inflection points (since an inflection point must have $f''(x) = 0$).

M4 functional form cannot model non-monotonic behavior. The case of M4 is a bit different, since the relationship between y and x in this case is expressed as an inverse function, i.e.

$$x = g(y) = \left(\frac{y - \epsilon_\infty}{b(\epsilon_0 - y)^a}\right)^{1/c} \tag{2}$$

However, non-monotonicity of y as an inverse function $y = g^{-1}(x)$ is ruled out, since that would imply two different values of $x = g(y)$ can be obtained for the single value of y – this is impossible, since $f(y)$ maps each y deterministically to a single value of x. As a result, M4 cannot express non-monotonic functions.

M4 functional form can model inflection points. It is easy to see that if $y = g^{-1}(x)$ had an inflection point, then $x = g(y)$ would have it as well. This is because an inflection point is defined as a point x where $f(x)$ changes from concave to convex, which implies that $g(y)$ changes from convex to concave, since the inverse of a convex function is concave; the root(s) of $g''(y)$ are the point(s) at which this change occurs. Using Wolfram Alpha[2] and matplotlib (Hunter, 2007), we observe that M4 is able to express inflection points, e.g. $(a, b, c, \epsilon_0, \epsilon_\infty, x, y) = (1, 1, -2, 3/4, 1/4, 1/\sqrt{3}, 5/8)$, or $(a, b, c, \epsilon_0, \epsilon_\infty, x, y) = (2, 1, -3, 2/3, 1/3, (-5/6 + \sqrt{3}/2)^{1/3}, 1/\sqrt{3})$.

[2] https://www.wolframalpha.com/

C Additional Empirical Results: Additional Fits and Extrapolations of Functional Forms

C.1 Fit of M3 (functional form that can't express inflection point) to 4 digit addition

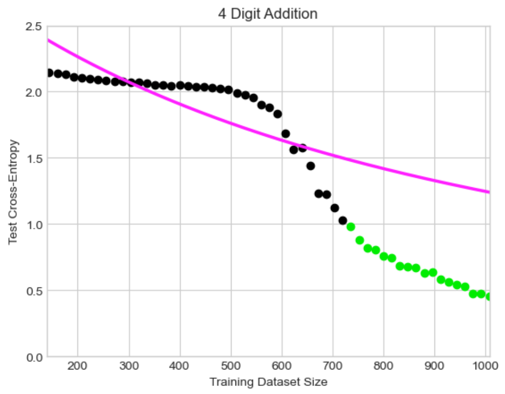


Figure 4: 4 Digit Addition. Fit of M3 (functional form that can't express inflection point) to 4 digit addition.

C.2 Table summarizing results on large-scale scaling laws benchmark of Alabdulmohsin et al. (2022)

Domain	M1 ↑	M2 ↑	M3 ↑	M4 ↑	BNSL ↑
Downstream Image Classification	2.78%	5.56%	13.89%	13.89%	**63.89%**
Language	10%	10%	25%	0%	**55%**

Table 2: Percentage of tasks by domain where each functional form is the best for extrapolation of scaling behavior. Numbers for M1, M2, M3, and M4 were obtained via correspondence with authors of Alabdulmohsin et al. (2022). See Sections C.3 and C.4 for more details.

C.3 Vision

Using the scaling laws benchmark of Alabdulmohsin et al. (2022), we evaluate how well various functional forms extrapolate performance on vision tasks as training dataset size increases. In this vision subset of the benchmark, the tasks that are evaluated are error rate on each of various few-shot downstream image classification (IC) tasks; the downstream tasks are: Birds 200 Welinder et al. (2010), Caltech101 Fei-Fei et al. (2004), CIFAR-100 Krizhevsky et al. (2009), and ImageNet Deng et al. (2009). The following architectures of various sizes are pretrained on subsets of JFT-300M: big-transfer residual neural networks (BiT) Kolesnikov et al. (2020), MLP mixers (MiX) Tolstikhin et al. (2021), and vision transformers (ViT) Dosovitskiy et al. (2020). As can be seen in Tables 2 and 3, BNSL yields extrapolations with the lowest RMSLE (Root Mean Squared Logarithmic Error) for 63.89% of tasks of any of the functional forms, while the next best functional form performs the best on only 13.89% of the tasks.

To view all plots of the BNSL on each of these tasks, see figures 9, 10, 11, 12 in Appendix C.12. To view all plots of M1, M2, M3, and M4 on each of these tasks, see Appendix A.4 of Alabdulmohsin et al. (2022).

C.4 Language

Using the scaling laws benchmark of Alabdulmohsin et al. (2022), we evaluate how well various functional forms extrapolate performance on language tasks as the training dataset size increases. In this language subset of the benchmark, the tasks that are evaluated are error rates on each of the various downstream tasks from the BIG-Bench (BB) Srivastava et al. (2022) benchmark and upstream

Task	Model	M1 ↓	M2 ↓	M3 ↓	M4 ↓	BNSL ↓
Birds 200 10-shot	BiT/101/3	9.13e-2 ± 2.8e-3	9.13e-2 ± 2.8e-3	9.13e-2 ± 2.8e-3	2.95e-2 ± 1.3e-3	**1.76e-2 ± 1.1e-3**
Birds 200 10-shot	BiT/50/1	6.88e-2 ± 7.5e-4	6.88e-2 ± 7.5e-4	5.24e-2 ± 6.2e-4	2.66e-2 ± 5.3e-4	**1.39e-2 ± 3.9e-4**
Birds 200 10-shot	MiX/B/16	9.15e-2 ± 1.1e-3	9.15e-2 ± 1.1e-3	3.95e-2 ± 7.0e-4	4.62e-2 ± 8.2e-4	**3.16e-2 ± 7.0e-4**
Birds 200 10-shot	MiX/L/16	5.51e-2 ± 1.4e-3	5.51e-2 ± 1.4e-3	5.51e-2 ± 1.4e-3	5.15e-2 ± 1.7e-3	**3.46e-2 ± 1.3e-3**
Birds 200 10-shot	ViT/B/16	6.77e-2 ± 1.1e-3	6.77e-2 ± 1.1e-3	3.52e-2 ± 8.1e-4	**1.51e-2 ± 6.2e-4**	2.43e-2 ± 8.1e-4
Birds 200 10-shot	ViT/S/16	3.95e-2 ± 1.2e-3	3.95e-2 ± 1.2e-3	3.74e-2 ± 1.1e-3	**1.85e-2 ± 7.9e-4**	2.35e-2 ± 8.4e-4
Birds 200 25-shot	BiT/101/3	9.41e-2 ± 3.2e-3	9.41e-2 ± 3.2e-3	9.41e-2 ± 3.2e-3	6.38e-2 ± 2.0e-3	**3.94e-2 ± 1.6e-3**
Birds 200 25-shot	BiT/50/1	1.10e-1 ± 1.0e-3	7.29e-2 ± 8.0e-4	**1.52e-2 ± 4.9e-4**	1.97e-2 ± 5.6e-4	2.73e-2 ± 6.1e-4
Birds 200 25-shot	MiX/B/16	1.40e-1 ± 1.9e-3	1.40e-1 ± 1.9e-3	6.93e-2 ± 1.2e-3	2.11e-2 ± 6.9e-4	**1.83e-2 ± 6.2e-4**
Birds 200 25-shot	MiX/L/16	1.12e-1 ± 2.0e-3	1.12e-1 ± 2.0e-3	1.12e-1 ± 2.0e-3	5.44e-2 ± 1.8e-3	**5.04e-2 ± 1.7e-3**
Birds 200 25-shot	ViT/B/16	9.02e-2 ± 1.6e-3	9.02e-2 ± 1.6e-3	3.75e-2 ± 1.0e-3	**1.51e-2 ± 5.7e-4**	1.62e-2 ± 6.1e-4
Birds 200 25-shot	ViT/S/16	5.06e-2 ± 1.4e-3	5.06e-2 ± 1.4e-3	4.96e-2 ± 1.4e-3	4.02e-2 ± 1.2e-3	**2.02e-2 ± 8.5e-4**
Birds 200 5-shot	BiT/101/3	8.17e-2 ± 2.0e-3	8.17e-2 ± 2.0e-3	8.17e-2 ± 2.0e-3	3.38e-2 ± 1.3e-3	**2.47e-2 ± 1.1e-3**
Birds 200 5-shot	BiT/50/1	5.44e-2 ± 5.6e-4	5.44e-2 ± 5.6e-4	5.44e-2 ± 5.6e-4	2.59e-2 ± 5.4e-4	**1.34e-2 ± 3.7e-4**
Birds 200 5-shot	MiX/B/16	8.27e-2 ± 1.0e-3	8.27e-2 ± 1.0e-3	5.49e-2 ± 7.8e-4	2.14e-2 ± 5.3e-4	**1.60e-2 ± 4.7e-4**
Birds 200 5-shot	MiX/L/16	5.68e-2 ± 1.4e-3	5.68e-2 ± 1.4e-3	5.68e-2 ± 1.4e-3	3.20e-2 ± 9.7e-4	**1.85e-2 ± 6.4e-4**
Birds 200 5-shot	ViT/B/16	3.40e-2 ± 8.9e-4	3.40e-2 ± 8.9e-4	3.40e-2 ± 8.9e-4	1.65e-2 ± 6.7e-4	**1.36e-2 ± 5.8e-4**
Birds 200 5-shot	ViT/S/16	2.75e-2 ± 7.9e-4	2.75e-2 ± 7.9e-4	2.75e-2 ± 7.9e-4	1.20e-2 ± 5.2e-4	**1.00e-2 ± 4.8e-4**
CIFAR-100 10-shot	BiT/101/3	8.57e-2 ± 3.8e-3	8.57e-2 ± 3.8e-3	8.25e-2 ± 3.7e-3	4.77e-2 ± 3.0e-3	**3.41e-2 ± 2.7e-3**
CIFAR-100 10-shot	BiT/50/1	7.44e-2 ± 1.5e-3	1.24e-2 ± 5.8e-4	2.08e-2 ± 7.2e-4	**1.24e-2 ± 5.8e-4**	1.25e-2 ± 5.8e-4
CIFAR-100 10-shot	MiX/B/16	8.77e-2 ± 1.9e-3	8.77e-2 ± 1.9e-3	2.71e-2 ± 1.2e-3	2.37e-2 ± 9.9e-4	**2.36e-2 ± 9.4e-4**
CIFAR-100 10-shot	MiX/L/16	1.05e-1 ± 3.1e-3	1.05e-1 ± 3.1e-3	**4.85e-2 ± 2.6e-3**	4.97e-2 ± 1.6e-3	5.03e-2 ± 1.6e-3
CIFAR-100 10-shot	ViT/B/16	8.98e-2 ± 2.0e-3	8.98e-2 ± 2.0e-3	8.98e-2 ± 2.0e-3	4.98e-2 ± 1.7e-3	**3.71e-2 ± 1.4e-3**
CIFAR-100 10-shot	ViT/S/16	6.84e-2 ± 1.1e-3	**2.11e-2 ± 6.6e-4**	3.35e-2 ± 8.6e-4	2.54e-2 ± 7.5e-4	2.57e-2 ± 7.5e-4
CIFAR-100 25-shot	BiT/101/3	8.77e-2 ± 5.6e-3	8.77e-2 ± 5.6e-3	4.44e-2 ± 3.5e-3	3.40e-2 ± 2.7e-3	**2.49e-2 ± 2.2e-3**
CIFAR-100 25-shot	BiT/50/1	7.31e-2 ± 2.0e-3	2.35e-2 ± 1.5e-3	3.65e-2 ± 1.8e-3	2.35e-2 ± 1.5e-3	**1.89e-2 ± 1.1e-3**
CIFAR-100 25-shot	MiX/B/16	1.08e-1 ± 2.3e-3	4.75e-2 ± 1.6e-3	**2.10e-2 ± 9.4e-4**	2.24e-2 ± 9.9e-4	2.67e-2 ± 1.1e-3
CIFAR-100 25-shot	MiX/L/16	9.79e-2 ± 2.2e-3	9.79e-2 ± 2.2e-3	3.67e-2 ± 1.7e-3	**2.98e-2 ± 1.4e-3**	3.19e-2 ± 1.5e-3
CIFAR-100 25-shot	ViT/B/16	1.07e-1 ± 1.9e-3	1.07e-1 ± 1.9e-3	6.54e-2 ± 1.6e-3	4.80e-2 ± 1.4e-3	**2.97e-2 ± 1.9e-3**
CIFAR-100 25-shot	ViT/S/16	8.03e-2 ± 1.2e-3	**2.19e-2 ± 7.4e-4**	3.13e-2 ± 8.4e-4	2.27e-2 ± 7.1e-4	3.24e-2 ± 8.5e-4
CIFAR-100 5-shot	BiT/101/3	5.94e-2 ± 3.2e-3	5.94e-2 ± 3.2e-3	5.94e-2 ± 3.2e-3	3.30e-2 ± 2.4e-3	**2.35e-2 ± 2.0e-3**
CIFAR-100 5-shot	BiT/50/1	4.87e-2 ± 1.3e-3	4.87e-2 ± 1.3e-3	1.69e-2 ± 8.8e-4	1.87e-2 ± 8.9e-4	**1.45e-2 ± 8.7e-4**
CIFAR-100 5-shot	MiX/B/16	7.07e-2 ± 1.2e-3	7.07e-2 ± 1.2e-3	2.78e-2 ± 8.4e-4	1.76e-2 ± 6.6e-4	**1.70e-2 ± 6.3e-4**
CIFAR-100 5-shot	MiX/L/16	7.06e-2 ± 1.6e-3	7.06e-2 ± 1.6e-3	4.17e-2 ± 1.4e-3	3.32e-2 ± 1.2e-3	**3.03e-2 ± 1.1e-3**
CIFAR-100 5-shot	ViT/B/16	6.27e-2 ± 1.6e-3	6.27e-2 ± 1.6e-3	6.27e-2 ± 1.6e-3	4.30e-2 ± 1.3e-3	**2.86e-2 ± 1.1e-3**
CIFAR-100 5-shot	ViT/S/16	6.93e-2 ± 1.2e-3	**2.84e-2 ± 8.2e-4**	3.88e-2 ± 8.0e-4	3.16e-2 ± 7.5e-4	3.49e-2 ± 7.7e-4
Caltech101 10-shot	BiT/101/3	3.07e-1 ± 2.0e-2	3.07e-1 ± 2.0e-2	1.51e-1 ± 1.3e-2	1.00e-1 ± 1.1e-2	**4.97e-2 ± 5.8e-3**
Caltech101 10-shot	BiT/50/1	3.29e-1 ± 1.6e-2	7.68e-2 ± 5.0e-3	1.13e-1 ± 6.0e-3	6.01e-2 ± 4.4e-3	**1.77e-2 ± 2.5e-3**
Caltech101 10-shot	MiX/B/16	**1.35e-1 ± 1.4e-2**	1.35e-1 ± 1.4e-2	1.35e-1 ± 1.4e-2	1.92e-1 ± 1.6e-2	2.04e-1 ± 9.7e-3
Caltech101 10-shot	MiX/L/16	1.25e-1 ± 1.3e-2	1.25e-1 ± 1.3e-2	**1.25e-1 ± 1.3e-2**	1.30e-1 ± 1.2e-2	2.13e-1 ± 1.5e-2
Caltech101 10-shot	ViT/B/16	7.76e-2 ± 4.3e-3	7.76e-2 ± 4.3e-3	**3.11e-2 ± 3.0e-3**	5.75e-2 ± 4.4e-3	4.02e-2 ± 3.9e-3
Caltech101 10-shot	ViT/S/16	1.95e-1 ± 6.0e-3	3.41e-2 ± 2.9e-3	**2.40e-2 ± 2.0e-3**	3.41e-2 ± 2.9e-3	2.40e-2 ± 2.0e-3
Caltech101 25-shot	BiT/101/3	1.15e-1 ± 6.5e-3	1.15e-1 ± 6.5e-3	1.15e-1 ± 6.5e-3	1.15e-1 ± 6.5e-3	**9.86e-2 ± 8.0e-3**
Caltech101 25-shot	BiT/50/1	3.60e-1 ± 1.9e-2	8.80e-2 ± 5.5e-3	1.43e-1 ± 7.6e-3	4.76e-2 ± 3.6e-3	**1.55e-2 ± 1.6e-3**
Caltech101 25-shot	MiX/B/16	**8.28e-2 ± 1.2e-2**	8.28e-2 ± 1.2e-2	8.28e-2 ± 1.2e-2	1.65e-1 ± 1.7e-2	1.93e-1 ± 1.3e-2
Caltech101 25-shot	MiX/L/16	9.66e-2 ± 1.0e-2	9.66e-2 ± 1.0e-2	9.66e-2 ± 1.0e-2	**9.66e-2 ± 1.0e-2**	1.49e-1 ± 1.3e-2
Caltech101 25-shot	ViT/B/16	1.03e-1 ± 5.6e-3	**3.33e-2 ± 2.5e-3**	4.46e-2 ± 3.6e-3	3.33e-2 ± 2.5e-3	3.95e-2 ± 5.4e-3
Caltech101 25-shot	ViT/S/16	1.77e-1 ± 5.4e-3	3.79e-2 ± 3.1e-3	**2.80e-2 ± 1.8e-3**	3.79e-2 ± 3.1e-3	3.29e-2 ± 2.1e-3
Caltech101 5-shot	BiT/101/3	2.12e-1 ± 1.2e-2	2.12e-1 ± 1.2e-2	2.12e-1 ± 1.2e-2	1.65e-1 ± 9.4e-3	**1.87e-2 ± 4.3e-3**
Caltech101 5-shot	BiT/50/1	2.34e-1 ± 6.1e-3	4.13e-2 ± 2.1e-3	**1.61e-2 ± 1.3e-3**	4.69e-2 ± 2.1e-3	4.10e-2 ± 2.1e-3
Caltech101 5-shot	MiX/B/16	2.43e-1 ± 1.2e-2	2.43e-1 ± 1.2e-2	2.35e-1 ± 1.1e-2	7.28e-2 ± 4.3e-3	**1.92e-2 ± 1.9e-3**
Caltech101 5-shot	MiX/L/16	1.38e-1 ± 9.7e-3	1.38e-1 ± 9.7e-3	1.38e-1 ± 9.7e-3	**1.37e-1 ± 9.9e-3**	1.63e-1 ± 1.1e-2
Caltech101 5-shot	ViT/B/16	1.10e-1 ± 6.3e-3	1.10e-1 ± 6.3e-3	6.02e-2 ± 4.7e-3	6.81e-2 ± 4.8e-3	**3.87e-2 ± 3.4e-3**
Caltech101 5-shot	ViT/S/16	1.90e-1 ± 4.7e-3	3.82e-2 ± 2.6e-3	5.04e-2 ± 2.9e-3	3.82e-2 ± 2.6e-3	**2.78e-2 ± 1.8e-3**
ImageNet 10-shot	BiT/101/3	1.27e-1 ± 2.0e-3	1.27e-1 ± 2.0e-3	7.36e-2 ± 1.1e-3	3.06e-2 ± 7.0e-4	**2.08e-2 ± 5.5e-4**
ImageNet 10-shot	BiT/50/1	9.54e-2 ± 7.2e-4	9.54e-2 ± 7.2e-4	**5.75e-3 ± 2.0e-4**	1.86e-2 ± 2.8e-4	1.97e-2 ± 2.7e-4
ImageNet 10-shot	MiX/B/16	9.34e-2 ± 7.9e-4	9.34e-2 ± 7.9e-4	3.37e-2 ± 2.9e-4	2.32e-2 ± 3.0e-4	**1.68e-2 ± 2.5e-4**
ImageNet 10-shot	MiX/L/16	9.83e-2 ± 1.3e-3	9.83e-2 ± 1.3e-3	9.83e-2 ± 1.3e-3	**4.01e-3 ± 1.9e-4**	1.44e-2 ± 2.9e-4
ImageNet 10-shot	ViT/B/16	4.62e-2 ± 7.1e-4	4.62e-2 ± 7.1e-4	4.62e-2 ± 7.1e-4	1.44e-2 ± 3.0e-4	**7.73e-3 ± 2.7e-4**
ImageNet 10-shot	ViT/S/16	4.74e-2 ± 5.6e-4	4.74e-2 ± 5.6e-4	1.66e-2 ± 2.5e-4	7.18e-3 ± 2.0e-4	**3.71e-3 ± 1.4e-4**
ImageNet 25-shot	BiT/101/3	1.42e-1 ± 2.3e-3	1.42e-1 ± 2.3e-3	6.67e-2 ± 9.1e-4	3.31e-2 ± 8.7e-4	**1.85e-2 ± 6.2e-4**
ImageNet 25-shot	BiT/50/1	1.17e-1 ± 9.2e-4	1.17e-1 ± 9.2e-4	**4.06e-3 ± 1.7e-4**	1.84e-2 ± 2.6e-4	1.96e-2 ± 2.4e-4
ImageNet 25-shot	MiX/B/16	9.59e-2 ± 9.3e-4	9.59e-2 ± 9.3e-4	5.39e-2 ± 4.9e-4	2.04e-2 ± 3.1e-4	**8.56e-3 ± 2.3e-4**
ImageNet 25-shot	MiX/L/16	1.03e-1 ± 1.3e-3	1.03e-1 ± 1.3e-3	1.03e-1 ± 1.3e-3	**6.33e-3 ± 2.2e-4**	7.60e-3 ± 2.6e-4
ImageNet 25-shot	ViT/B/16	5.17e-2 ± 8.8e-4	5.17e-2 ± 8.8e-4	5.17e-2 ± 8.8e-4	**1.52e-2 ± 3.8e-4**	1.98e-2 ± 4.3e-4
ImageNet 25-shot	ViT/S/16	5.52e-2 ± 4.4e-4	4.12e-2 ± 3.4e-4	9.65e-3 ± 2.3e-4	7.78e-3 ± 2.1e-4	**6.11e-3 ± 2.4e-4**
ImageNet 5-shot	BiT/101/3	9.24e-2 ± 1.4e-3	9.24e-2 ± 1.4e-3	9.24e-2 ± 1.4e-3	2.09e-2 ± 7.9e-4	**8.05e-3 ± 5.0e-4**
ImageNet 5-shot	BiT/50/1	8.95e-2 ± 6.7e-4	8.95e-2 ± 6.7e-4	1.53e-2 ± 2.2e-4	1.11e-2 ± 2.3e-4	**7.94e-3 ± 2.1e-4**
ImageNet 5-shot	MiX/B/16	9.09e-2 ± 7.2e-4	9.09e-2 ± 7.2e-4	3.01e-2 ± 2.8e-4	1.95e-2 ± 2.7e-4	**9.60e-3 ± 2.3e-4**
ImageNet 5-shot	MiX/L/16	7.99e-2 ± 9.7e-4	7.99e-2 ± 9.7e-4	7.99e-2 ± 9.7e-4	9.92e-3 ± 4.5e-4	**5.68e-3 ± 2.4e-4**
ImageNet 5-shot	ViT/B/16	4.11e-2 ± 6.3e-4	4.11e-2 ± 6.3e-4	4.11e-2 ± 6.3e-4	1.55e-2 ± 2.8e-4	**1.29e-2 ± 2.7e-4**
ImageNet 5-shot	ViT/S/16	4.20e-2 ± 4.1e-4	4.20e-2 ± 4.1e-4	2.40e-2 ± 2.6e-4	8.02e-3 ± 1.9e-4	**5.51e-3 ± 1.7e-4**

Table 3: Extrapolation Results on scaling behavior of Downstream Vision Tasks. See Section C.3 for more details. Numbers for M1, M2, M3, and M4 obtained via correspondence with authors of Alabdulmohsin et al. (2022).

test cross-entropy of various models trained to do language modeling (LM) and neural machine translation (NMT). All LM and BB tasks use a decoder-only language model. As can be seen in Table 2 and 4, BNSL yields extrapolations with the lowest RMSLE (Root Mean Squared Logarithmic Error) for 55% of tasks of any of the functional forms, while the next best functional form performs the best on only 25% of the tasks.

To view all plots of the BNSL on each of these tasks, see Figures 13, 14, 15 in Appendix C.12.

To view plots of M1, M2, M3, and M4 on these tasks, see Figure 8 of Alabdulmohsin et al. (2022).

Domain	Task	Model	M1 ↓	M2 ↓	M3 ↓	M4 ↓	BNSL ↓
BB	date understanding, 1-shot	2.62e+8 Param	3.19e-2 ± 9.6e-4	3.19e-2 ± 9.6e-4	**4.67e-3 ± 1.4e-4**	3.19e-2 ± 9.6e-4	1.81e-2 ± 3.9e-4
BB	date understanding, 2-shot	2.62e+8 Param	2.86e-2 ± 6.2e-4	2.86e-2 ± 6.2e-4	**4.83e-3 ± 4.1e-4**	2.86e-2 ± 6.2e-4	5.41e-3 ± 1.0e-3
BB	linguistic mappings, 1-shot	2.62e+8 Param	1.66e-2 ± 5.5e-4	1.62e-2 ± 5.4e-4	1.66e-2 ± 5.5e-4	1.33e-2 ± 3.8e-4	**1.13e-2 ± 2.2e-4**
BB	linguistic mappings, 2-shot	2.62e+8 Param	1.70e-2 ± 6.5e-4	1.70e-2 ± 6.5e-4	1.70e-2 ± 6.5e-4	1.06e-2 ± 5.1e-4	**9.51e-3 ± 5.1e-4**
BB	mult data wrangling, 1-shot	2.62e+8 Param	1.07e-2 ± 1.0e-3	1.07e-2 ± 1.0e-3	1.07e-2 ± 1.0e-3	6.66e-3 ± 7.3e-4	**6.39e-3 ± 4.6e-4**
BB	mult data wrangling, 2-shot	2.62e+8 Param	1.57e-2 ± 1.5e-3	1.57e-2 ± 1.5e-3	1.57e-2 ± 1.5e-3	5.79e-3 ± 7.0e-4	**2.67e-3 ± 2.7e-4**
BB	qa wikidata, 1-shot	2.62e+8 Param	**4.27e-3 ± 8.9e-4**	4.32e-3 ± 8.2e-4	4.27e-3 ± 8.9e-4	4.32e-3 ± 8.2e-4	4.68e-3 ± 7.3e-4
BB	qa wikidata, 2-shot	2.62e+8 Param	**4.39e-3 ± 7.0e-4**	4.66e-3 ± 6.4e-4	4.39e-3 ± 7.0e-4	9.02e-3 ± 6.9e-4	8.05e-3 ± 7.3e-4
BB	unit conversion, 1-shot	2.62e+8 Param	8.30e-3 ± 4.4e-4	8.30e-3 ± 4.4e-4	**1.48e-3 ± 2.7e-4**	4.79e-3 ± 3.4e-4	5.73e-2 ± 4.6e-3
BB	unit conversion, 2-shot	2.62e+8 Param	1.07e-2 ± 4.4e-4	1.07e-2 ± 4.4e-4	**7.50e-3 ± 5.5e-4**	7.55e-3 ± 5.1e-4	7.74e-2 ± 5.7e-3
LM	upstream test cross-entropy	1.07e+9 Param	1.71e-2 ± 6.0e-4	1.66e-3 ± 5.1e-5	4.50e-3 ± 5.9e-5	1.28e-3 ± 3.9e-5	**9.71e-4 ± 3.2e-5**
LM	upstream test cross-entropy	1.34e+8 Param	1.43e-2 ± 4.8e-4	1.46e-3 ± 6.8e-5	**6.46e-4 ± 5.1e-5**	1.46e-3 ± 6.8e-5	9.01e-4 ± 5.5e-5
LM	upstream test cross-entropy	1.68e+7 Param	6.37e-3 ± 9.4e-5	**3.03e-4 ± 1.2e-5**	1.56e-3 ± 3.5e-5	3.03e-4 ± 1.2e-5	4.34e-4 ± 1.8e-5
LM	upstream test cross-entropy	2.62e+8 Param	1.55e-2 ± 7.2e-4	**9.20e-4 ± 9.7e-5**	3.97e-3 ± 1.3e-4	9.20e-4 ± 9.7e-5	2.05e-3 ± 5.6e-5
LM	upstream test cross-entropy	4.53e+8 Param	1.65e-2 ± 6.6e-4	7.41e-4 ± 9.8e-5	6.58e-4 ± 6.6e-5	7.41e-4 ± 9.8e-5	**5.86e-4 ± 7.7e-5**
NMT	upstream test cross-entropy	28 Enc, 6 Dec	1.71e-1 ± 0	5.64e-2 ± 0	3.37e-2 ± 0	1.81e-2 ± 0	**1.69e-2 ± 0**
NMT	upstream test cross-entropy	6 Enc, 28 Dec	2.34e-1 ± 0	5.27e-2 ± 0	1.65e-2 ± 0	4.44e-2 ± 0	**1.56e-2 ± 0**
NMT	upstream test cross-entropy	6 Enc, 6 Dec	2.62e-1 ± 0	3.84e-2 ± 0	8.92e-2 ± 0	2.05e-2 ± 0	**1.37e-3 ± 0**
NMT	upstream test cross-entropy	Dec-only, LM	2.52e-1 ± 0	1.03e-2 ± 0	3.28e-2 ± 0	8.43e-3 ± 0	**7.33e-3 ± 0**
NMT	upstream test cross-entropy	Transformer-Enc, LSTM-Dec	1.90e-1 ± 0	1.26e-2 ± 0	6.32e-2 ± 0	1.26e-2 ± 0	**8.30e-3 ± 0**

Table 4: Extrapolation Results on scaling behavior of Language Tasks. See Section C.4 for more details. Numbers for M1, M2, M3, and M4 were obtained via correspondence with authors of Alabdulmohsin et al. (2022). BB stands for BIG-Bench (Srivastava et al., 2022). NMT stands for Neural Machine Translation. LM stands for Language Modeling.

C.5 Non-Monotonic Scaling

We show that BNSL accurately models non-monotonic scaling behaviors that are exhibited by Transformers (Vaswani et al. (2017)) in double descent (Nakkiran et al., 2021) in Figure 5. Other functional forms are mathematically incapable of expressing non-monotonic behaviors (as shown in Section B).

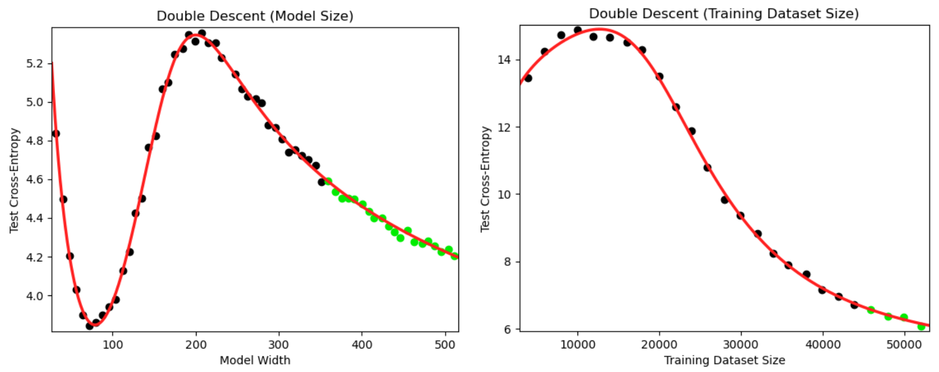


Figure 5: Fit of BNSL to Double Descent. Both plots are of transformers trained to do neural machine translation via minimizing cross-entropy. Experimental data of left figure is obtained from Figure 8 top of Nakkiran et al. (2021); "Model Width" on the x-axis refers to embedding dimension d_{model} of the transformer. Experimental data of the right figure is obtained from Figure 11b of Nakkiran et al. (2021).

C.6 Reinforcement Learning

We show that BNSL accurately models and extrapolates the scaling behaviors of various multi-agent and single-agent reinforcement learning algorithms trained in various environments. In the top left plot and top right plot and bottom left plot of Figure 6, BNSL accurately models and extrapolates the scaling behavior of the AlphaZero algorithm trained to play the game Connect Four from Figure 4 and Figure 5 and Figure 3 respectively of Neumann and Gros (2022); the x-axes respectively are compute (FLOPs) used for training, training dataset size (states), and number of model parameters. In Figure 6 bottom right, BNSL accurately models and extrapolates the scaling behavior of the Proximal Policy Optimization (PPO) algorithm Schulman et al. (2017) trained to play the Procgen (Cobbe et al., 2020) game called Heist.

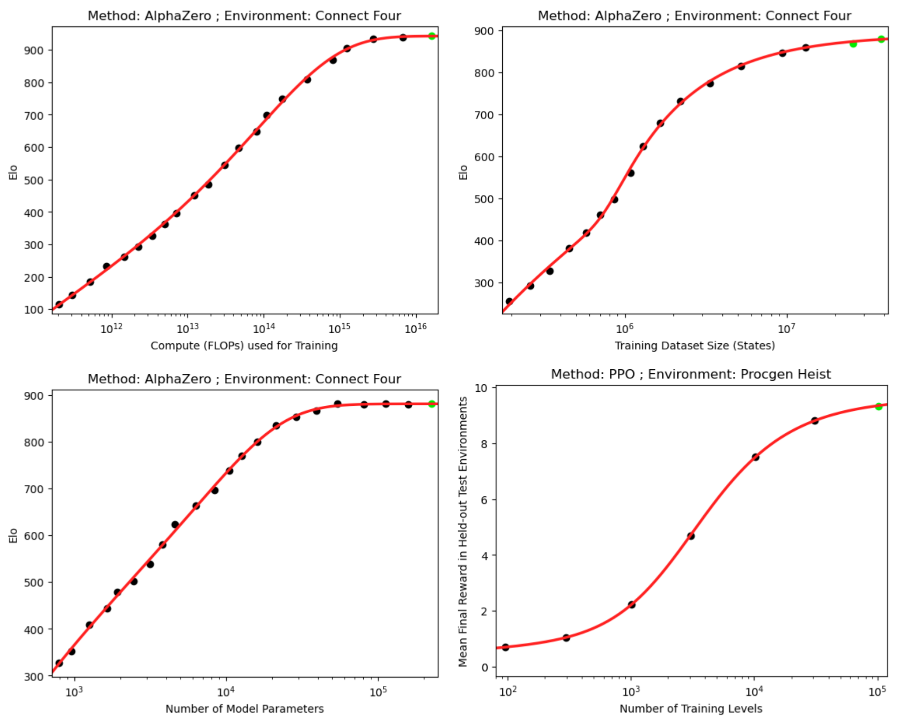


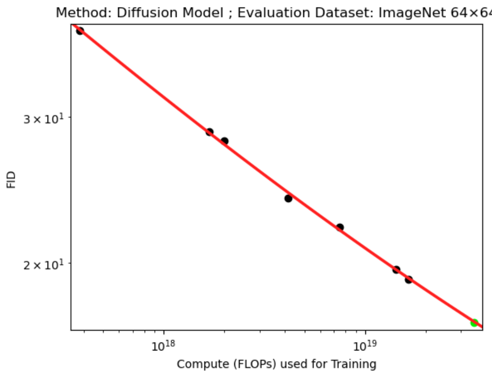
Figure 6: Extrapolation of BNSL on Reinforcement Learning Scaling Experimental Data. Experimental data of the top left plot and top right plot and bottom left plot is from Figure 4 and Figure 5 and Figure 3 respectively of Neumann and Gros (2022). Experimental Data of the bottom right plot is from Figure 2 of Cobbe et al. (2020). Top left plot is the compute-optimal Pareto frontier. See Section C.6 for more details.

C.7 Extrapolation Results for Diffusion Generative Models of Images

In Figure 7, we show that BNSL accurately extrapolates the scaling behavior of Diffusion Generative Models of Images from Figure 10 of Nichol and Dhariwal (2021) when Negative Log-likelihood (NLL) or Frechet Inception Distance (FID) score is on the y-axis and compute used for training is on the x-axis; compute is scaled in the manner that is Pareto optimal with respect to the performance evaluation metric on the y-axis.

C.8 Extrapolation Results when Upstream Performance is on the x-axis

In Figure 8, we show that BNSL accurately extrapolates the scaling behavior when upstream performance is on the x-axis and downstream performance is on the y-axis. The upstream task is supervised pretraining of ViT (Dosovitskiy et al., 2020) on subsets of JFT-300M (Sun et al., 2017). The downstream task is 20-shot ImageNet classification. The experimental data of this scaling behavior is obtained from Figure 5 of Abnar et al. (2021).



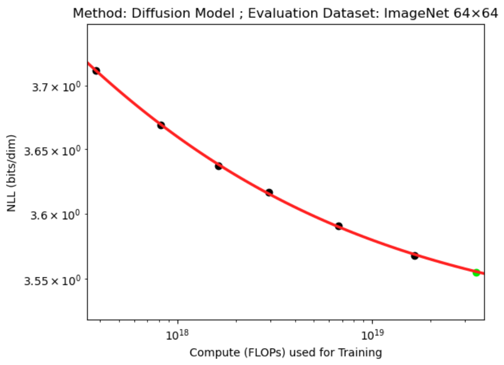


Figure 7: Extrapolation Results of BNSL for scaling behavior of Diffusion Generative Models of Images. Frechet Inception Distance (FID) score is on the y-axis in the left plot. Negative log-likelihood (NLL) is the y-axis in the right plot. For both plots, compute used for training is on the x-axis and Imagenet 64x64 is the evaluation dataset. Experimental data of scaling behavior obtained from Figure 10 of Nichol and Dhariwal (2021). See Section C.7 for more details.

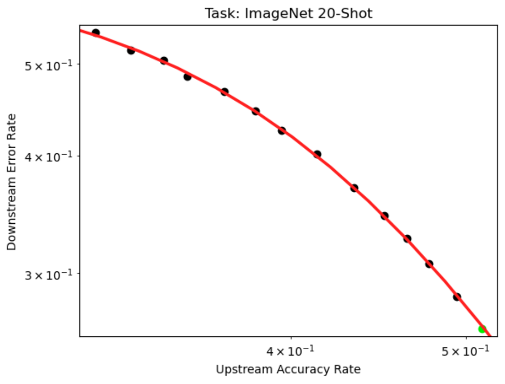


Figure 8: Extrapolation Results of BNSL for scaling behavior when Upstream Performance is on the x-axis and Downstream Performance is on the y-axis. Experimental data of scaling behavior obtained from Figure 5 of Abnar et al. (2021). The upstream task is supervised pretraining of ViT (Dosovitskiy et al., 2020) on subsets of JFT-300M (Sun et al., 2017). The Downstream Task is 20-shot ImageNet classification. See Section C.8 for more details.

C.9 Definition of Root Standard Log Error

$$error = (log(y_i) - log(\hat{y}_i))^2)$$

$$\mu_{error} = \frac{1}{N}\sum_{i=1}^{N} error$$

$$\sigma_{error} = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(error_i - \mu_{error})^2}$$

$$Root\_Standard\_Log\_Error = \sqrt{\mu_{error} + \frac{\sigma_{error}}{\sqrt{len(\hat{y})}}} - \sqrt{\mu_{error}}$$

C.10 Experimental details of Section 4.1

We perform an extensive set of experiments to model and extrapolate the scaling behavior for the 4-digit arithmetic addition task with respect to the training dataset size. Our code is based on the minGPT implementation Karpathy (2020). We set the batch size equal to the training dataset size.

We do not use dropout or a learning rate decay here. Each experiment was run on a single V100 GPU and each run took less than 2 hours. For our experiments we train the transformer model using the following set of hyperparameters:

D_{model}	128
D_{MLP}	512
Number of heads	2
Number of transformer blocks (i.e. layers)	1
Learning rate	0.0001
Weight Decay	0.1
Dropout Probability	0.0
Dataset sizes	144-1008
Vocab Size	10

Table 5: Hyperparameters for 4-digit addition task

C.11 Experimental details of fitting BNSL

We fit BNSL as follows: We first use scipy.optimize.brute to do a grid search of the values of the constants $(a, b, c_0, c_1...c_n, d_1...d_n, f_1...f_n)$ of BNSL that best minimize the mean squared log error (MSLE) between the real data and the output of BNSL. We then use the values obtained from the grid search as the initialization of the non-linear least squares algorithm of scipy.optimize.curve_fit. We then use the non-linear least squares algorithm of scipy.optimize.curve_fit to minimize the mean squared log error (MSLE) between the real data and the output of BNSL.

The version of MSLE we use for such optimization is the following numerically stable variant:

$$Numerically\_Stable\_MSLE = \sum_{i=1}^{n}((log(y_i + 1) - log(\hat{y}_i + 1))^2)/n$$

C.12 Plots of BNSL Extrapolations on Scaling Laws Benchmark of Alabdulmohsin et al. (2022)

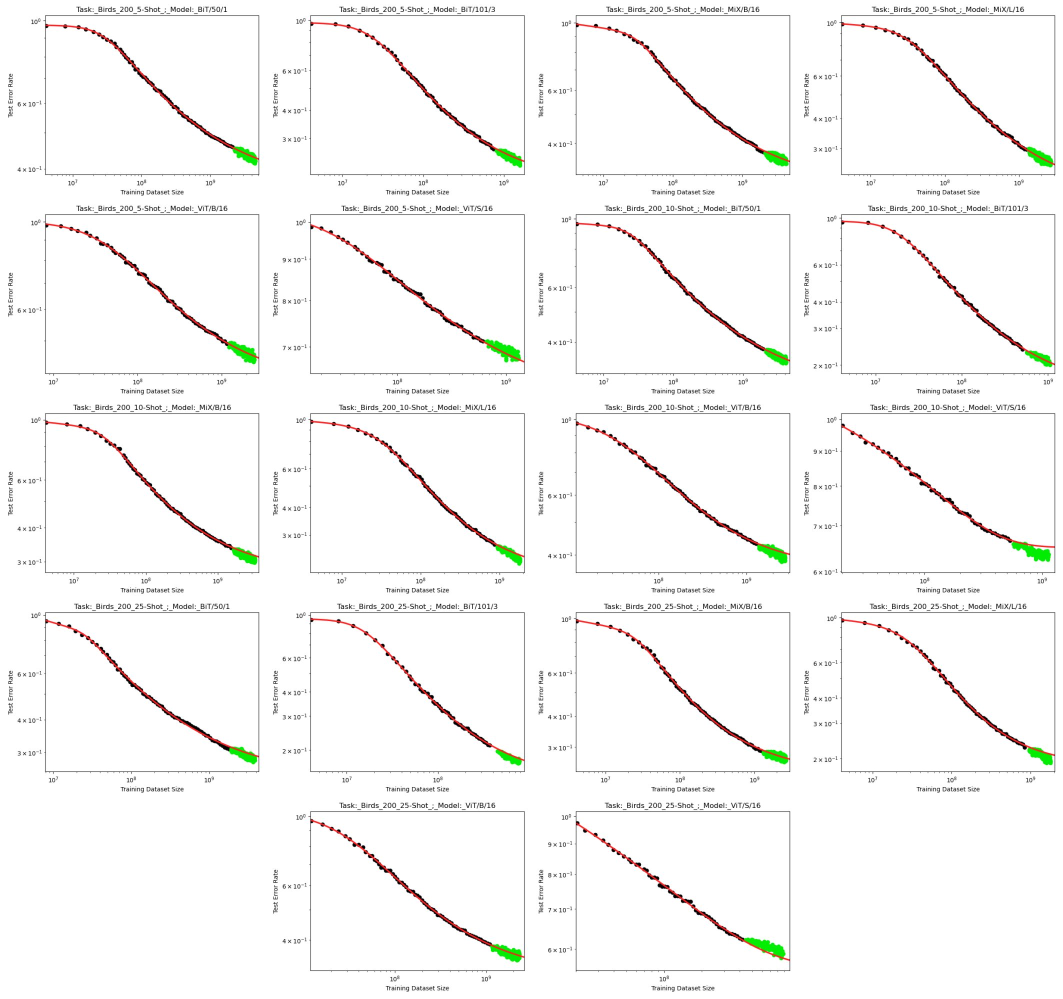


Figure 9: Birds 200

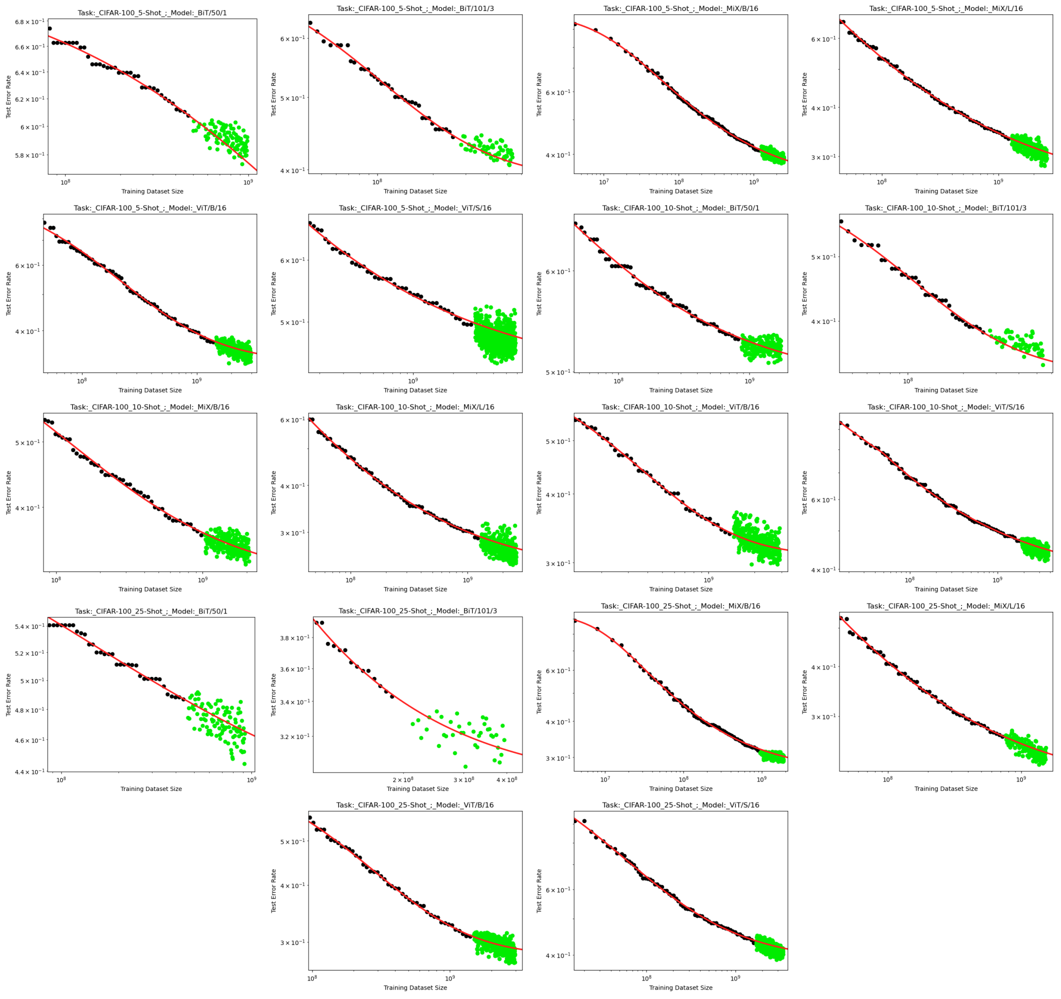


Figure 10: CIFAR-100

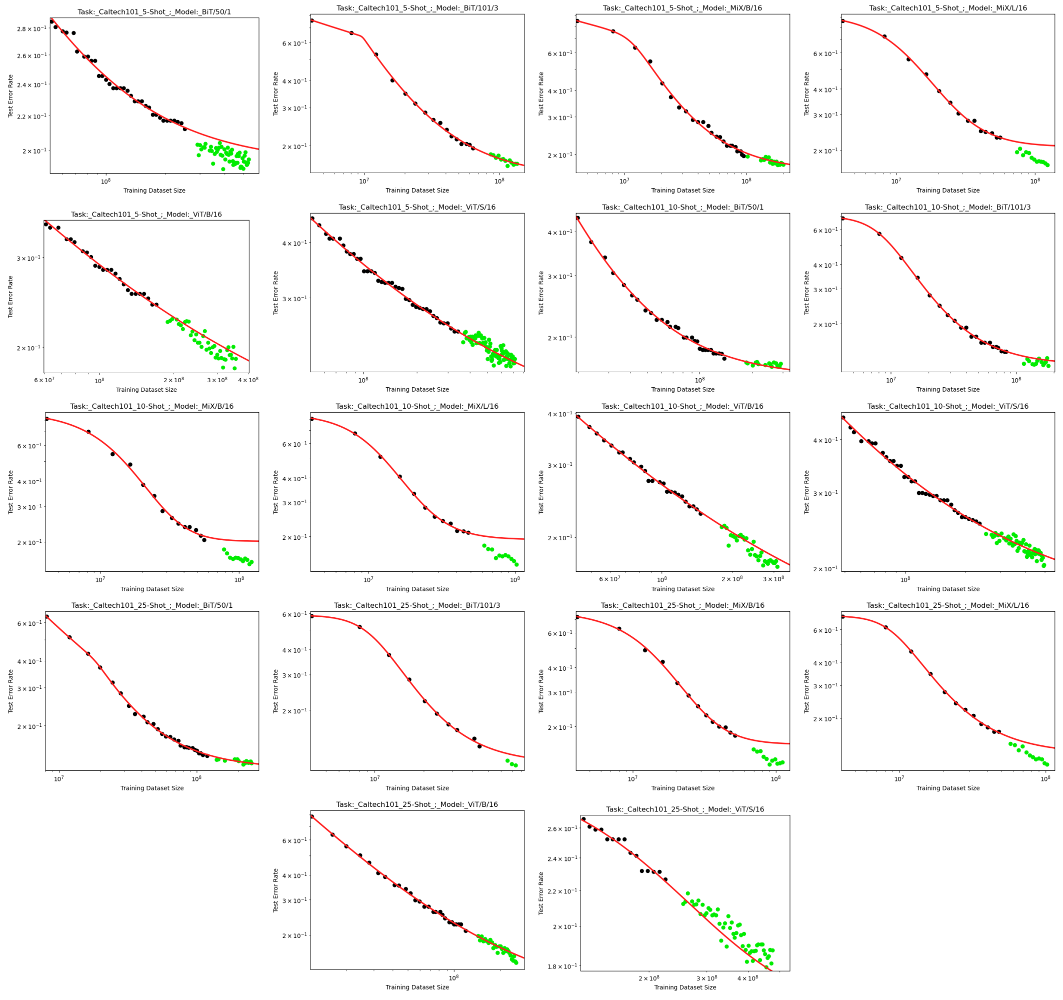


Figure 11: Caltech101. From eyeballing, we think the subset of Caltech101 with unsatisfactory extrapolations has unsatisfactory extrapolations due to the maximum (along the x-axis) of the black point used for fitting being near or before a break; this is accentuated by not having enough points for fitting for the SciPy fitter to be able to determine whether the break is an actual break or just noisy deviation.

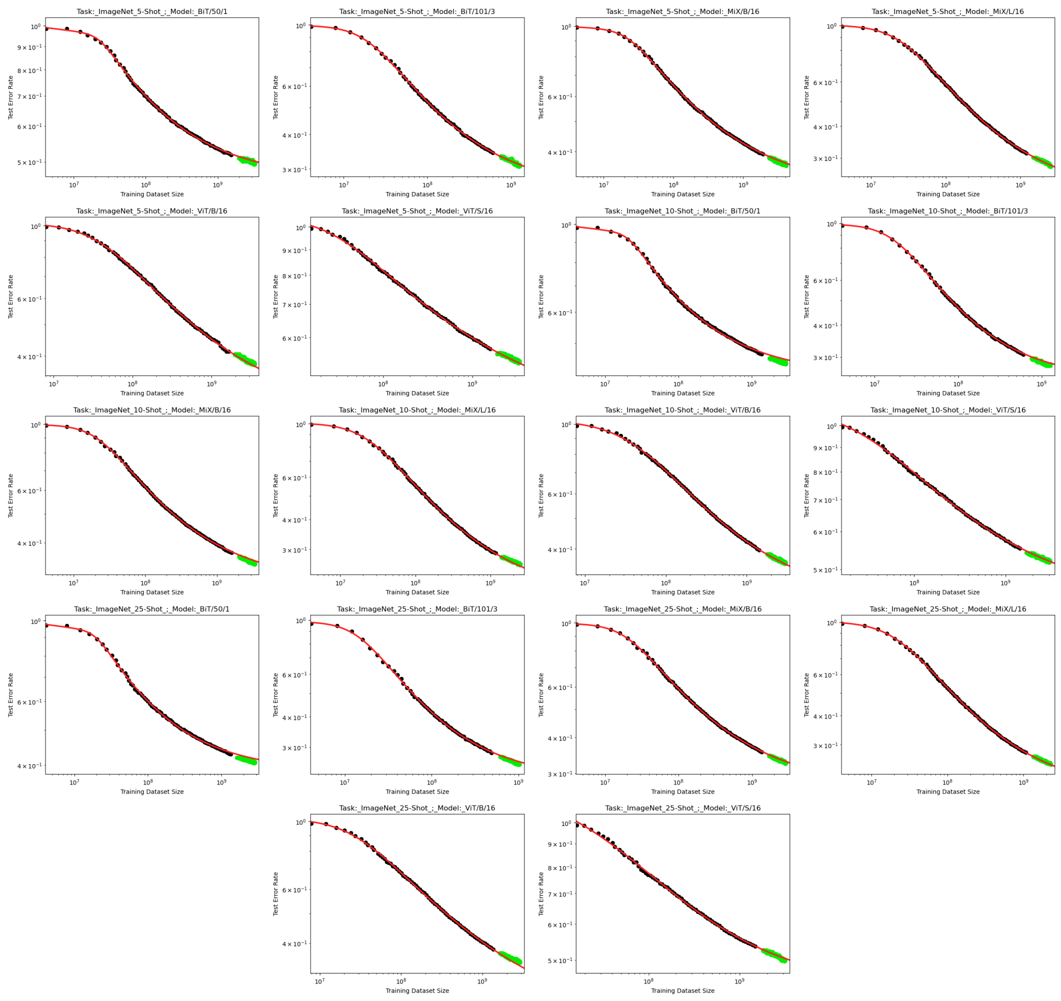


Figure 12: ImageNet

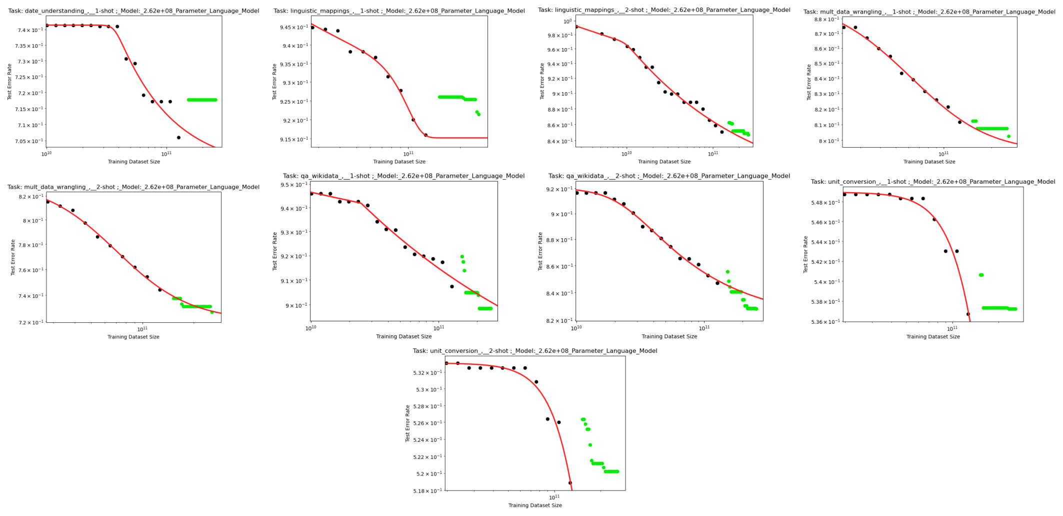


Figure 13: BIG-Bench (BB). From eyeballing, we think the subset of BIG-Bench with unsatisfactory extrapolations has unsatisfactory extrapolations due to the maximum (along the x-axis) of the black point used for fitting being near or before a break; this is accentuated by not having enough points for fitting for the SciPy fitter to be able to determine whether the break is an actual break or just noisy deviation.

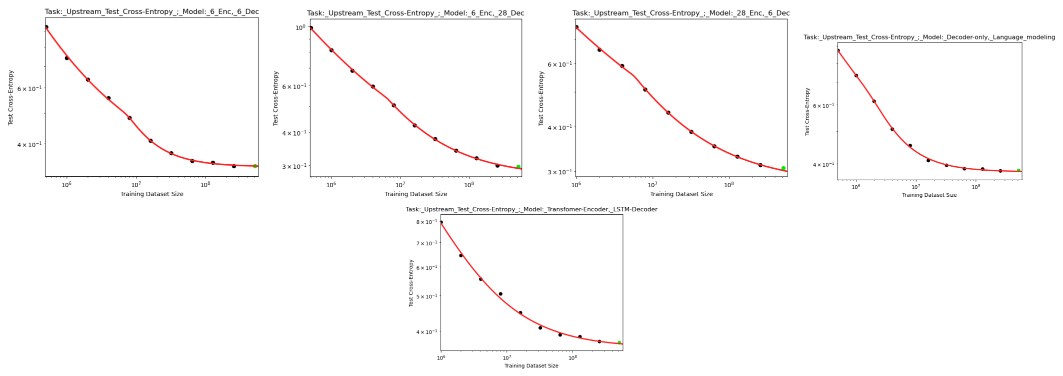


Figure 14: Neural Machine Translation (NMT)

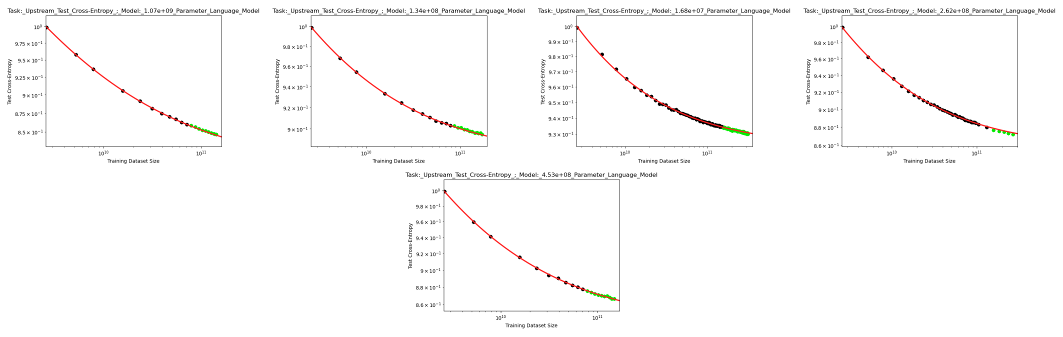


Figure 15: Language Modeling (LM)