BROKEN NEURAL SCALING LAWS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a smoothly broken power law functional form that accurately models 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, or training dataset size varies) for 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, arithmetic, and reinforcement learning. This functional form yields extrapolations of scaling behavior that often is an order of magnitude more accurate than the ones obtained by other functional forms for neural scaling behavior. Moreover, this functional form accurately models 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. Lastly, we use this functional form to glean insights about the limit of the predictability of scaling behavior.

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 accurate (often by an order of magnitude) 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

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

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

where y represents the performance evaluation metric (e.g. prediction error, cross entropy, BLEU score percentage, F1 score percentage, reward, or Elo rating) and x represents a quantity that is being scaled (e.g. number of model parameters, amount of compute used for training, or training dataset size). The remaining parameters $a, b, c, d_1...d_n, f_1...f_n, g_1...g_n$ 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) as 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 represents the slope of the first 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 linear region (on a log-log plot) occurs. Constant f_i represents the slope of the $(i + 1)$th linear region on a log-log plot. Constant g_i represents the sharpness of break between the (i)th and the $(i + 1)$th linear region 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.

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 of the form $y = ax^b + c$ in which x refers to training dataset size and y refers to test error. Hestness et al. (2017) showed that this functional form holds over many orders of magnitude. Rosenfeld et al. (2019) demonstrated that the same functional form applies when x refers to model size (number of parameters). Kaplan et al. (2020) brought "neural" scaling laws to the mainstream and showed that there is an additional scaling law with respect to compute. Zhai et al. (2021) introduced the functional form $y = a(x + d)^b + c$, 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. Abnar et al. (2021) proposed a functional form that relates downstream performance to upstream performance. Alabdulmohsin et al. (2022) proposed functional form $(y - \epsilon_\infty)/((\epsilon_0 - y)^a) = bx$, 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 that form is mathematically equivalent to our BNSL with a single break, it is important to note that (i) Hernandez et al. (2021) uses 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 (as explored in our work); (ii) Hernandez et al. (2021) only mentioned this equation in passing, without trying to fit this functional form to 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 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.

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, on a linear-linear plot, the extra expressiveness of broken neural scaling laws appears to be necessary (and sufficient). Figure 1 and Figure 3, provide examples of BNSLs (with a single break-point, $n = 1$) producing double descent and inflection points, respectively, establishing the capacity of this functional form to fit these phenomena.

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	$a(x^{-1}+d)^{-b} + c$	$\frac{ab}{x(1+dx)(d+1/x)^b}$	$abx^{(b-2)}(1+dx)^{(-2-b)}(b-1-2dx)$
$f(x) = g^{-1}(x)$	$g(y)$	$g'(y)$	$g''(y)$
M4	$\frac{1}{b}\frac{y-\epsilon_\infty}{(\epsilon_0-y)^a}$,		

Table 1: Previously proposed functional forms and their (first and second order) derivatives. For $a, b, c, d, x > 0$, $b < 1$, the expressions for each of the derivatives is a product of non-zero multiplicands, and thus none of these functional forms can express functions that are non-monotonic or have inflection points.

Why M1-M4 functional forms cannot model nonmonotonic behavior and/or inflection points? We now explain why standard functional forms are incapable of exhibiting either behavior. 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 $a, b, c, d, x > 0$, $b < 1$, since $x > 0$ - namely the scaling variable (model, data or compute amount) is always non-negative. This implies that they are monotonic (as the first derivative never changes sign), and that they lack inflection points (since an inflection point must have $f''(x) = 0$).

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. $x = g(y) = 1/b(y - \epsilon_\infty)/((\epsilon_0 - y)^a)$. 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 to the possibility of modeling an inflection point, 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. However, it is easy to show via simple algebraic computation that second derivative $g''(y)$ never equals zero, thus ruling out the possibility of inflection points with this functional form as well.

5 Empirical Results: BNSL Accurately Fits and Predicts All Scaling Performance

We now show the fits and/or extrapolations of various functional forms. In all plots here and in the appendix A.4, black points are the points used for fitting a functional form, green points are the held-out points used for evaluating extrapolation of a functional form fit to the black points, and a red line is the BNSL that has been fit to the black points. Please refer to Appendix Section A.3 for further experimental details on fitting BNSL.

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 $y = a(x^{-1} + d)^{-b} + c$
- M4 refers to functional form $(y - \epsilon_\infty)/((\epsilon_0 - y)^a) = bx$

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 A.1 for definition of root standard log error.

Domain	M1 ↑	M2 ↑	M3 ↑	M4 ↑	BNSL ↑
Downstream Image Classification	0%	1.39%	5.56%	4.17%	**88.89%**
Language	5%	10%	15%	25%	**45%**

Table 2: Percentage of tasks by domain where each functional form is the best for extrapolation. See Section 5.1 for more details. Numbers for M1, M2, M3, and M4 were obtained via correspondence with authors of Alabdulmohsin et al. (2022).

5.1 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 88.89% of tasks of any of the functional forms, while the next best functional form performs the best on only 4.17% of the tasks.

To view all plots of the BNSL on each of these tasks, see figures 4, 5, 6, 7 in Appendix A.4. To view all plots of M1, M2, M3, and M4 on each of these tasks, see Appendix A.4 of Alabdulmohsin et al. (2022).

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

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	2.49e-2 ± 1.2e-3	**3.79e-3 ± 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	2.48e-2 ± 5.1e-4	**4.96e-3 ± 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	4.14e-2 ± 7.8e-4	**9.98e-3 ± 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	4.59e-2 ± 1.6e-3	**8.41e-3 ± 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	1.87e-2 ± 7.2e-4	**9.81e-3 ± 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	**9.81e-3 ± 5.4e-4**	1.51e-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	5.09e-2 ± 1.8e-3	**7.24e-3 ± 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	1.63e-2 ± 5.1e-4	**8.00e-3 ± 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	1.85e-2 ± 6.3e-4	**5.02e-3 ± 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	4.88e-2 ± 1.8e-3	**1.05e-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	1.52e-2 ± 5.8e-4	**5.87e-3 ± 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	3.28e-2 ± 1.1e-3	**1.21e-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	3.00e-2 ± 1.2e-3	**6.08e-3 ± 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	2.63e-2 ± 5.4e-4	**5.79e-3 ± 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	1.86e-2 ± 5.0e-4	**5.74e-3 ± 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	2.65e-2 ± 9.0e-4	**5.06e-3 ± 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	1.26e-2 ± 5.3e-4	**6.32e-3 ± 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	1.56e-2 ± 5.9e-4	**6.93e-3 ± 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	9.28e-2 ± 3.9e-3	**1.53e-2 ± 2.9e-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	1.23e-2 ± 5.7e-4	**9.74e-3 ± 7.1e-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	2.60e-2 ± 1.2e-3	**8.61e-3 ± 9.5e-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	4.76e-2 ± 1.7e-3	**1.55e-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	5.60e-2 ± 1.8e-3	**1.73e-2 ± 1.8e-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	2.47e-2 ± 7.4e-4	**1.15e-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	4.29e-2 ± 3.4e-3	**9.32e-3 ± 3.0e-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	2.36e-2 ± 1.5e-3	**2.00e-2 ± 1.8e-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	2.08e-2 ± 9.3e-4	**8.15e-3 ± 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	2.79e-2 ± 1.3e-3	**8.47e-3 ± 1.6e-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	**4.81e-2 ± 1.4e-3**	1.20e-1 ± 1.4e-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	2.19e-2 ± 7.0e-4	**1.34e-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	4.57e-2 ± 2.8e-3	**1.58e-2 ± 2.6e-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**	4.91e-2 ± 1.3e-3	1.95e-2 ± 1.0e-3
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	2.05e-2 ± 7.4e-4	**6.77e-3 ± 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	3.14e-2 ± 1.1e-3	**1.18e-2 ± 1.3e-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	5.11e-2 ± 1.4e-3	**1.19e-2 ± 1.2e-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	3.09e-2 ± 7.5e-4	6.40e-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	3.54e-2 ± 6.3e-3	**6.42e-3 ± 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	6.31e-2 ± 4.4e-3	**5.37e-3 ± 2.2e-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	2.11e-1 ± 1.7e-2	**3.72e-2 ± 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	1.92e-1 ± 1.6e-2	**3.73e-2 ± 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	5.54e-2 ± 4.3e-3	**1.67e-2 ± 5.4e-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**	3.95e-2 ± 3.1e-3	3.31e-2 ± 5.3e-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	5.23e-2 ± 2.7e-3	**1.13e-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	5.95e-2 ± 4.1e-3	**2.13e-3 ± 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	1.74e-1 ± 1.7e-2	**2.83e-2 ± 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	1.03e-1 ± 9.5e-3	**1.98e-2 ± 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	4.24e-2 ± 3.6e-3	**6.37e-3 ± 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	**2.35e-2 ± 2.3e-3**	2.38e-2 ± 4.7e-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	8.82e-2 ± 5.0e-3	**3.17e-3 ± 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	4.49e-2 ± 2.1e-3	**1.33e-2 ± 2.2e-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	1.23e-1 ± 6.0e-3	**3.59e-3 ± 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	3.19e-2 ± 2.6e-3	**3.19e-2 ± 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	6.59e-2 ± 4.7e-3	**2.33e-2 ± 6.0e-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	4.06e-2 ± 2.7e-3	**2.47e-2 ± 3.8e-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	2.13e-2 ± 5.8e-4	**5.95e-3 ± 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**	1.77e-2 ± 2.7e-4	8.93e-3 ± 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	1.80e-2 ± 2.7e-4	**5.88e-3 ± 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	9.48e-3 ± 2.4e-4	**3.81e-3 ± 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	3.43e-2 ± 4.4e-4	**2.85e-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	1.14e-2 ± 2.4e-4	**1.97e-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	2.18e-2 ± 7.0e-4	**4.86e-3 ± 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**	1.70e-2 ± 2.5e-4	8.14e-3 ± 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	1.47e-2 ± 2.7e-4	**2.86e-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	6.09e-3 ± 2.5e-4	**1.94e-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	3.26e-2 ± 5.4e-4	**6.91e-3 ± 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	1.16e-2 ± 2.2e-4	**3.09e-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	1.01e-2 ± 5.3e-4	**2.55e-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	1.03e-2 ± 2.3e-4	**4.00e-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	1.45e-2 ± 2.5e-4	**3.57e-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	5.66e-3 ± 3.1e-4	**1.63e-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	2.88e-2 ± 3.6e-4	**4.97e-3 ± 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	1.44e-2 ± 2.2e-4	**3.00e-3 ± 1.7e-4**

Table 3: Extrapolation Results for Vision Tasks. See Section 5.1 for more details. Numbers for M1, M2, M3, and M4 obtained via correspondence with authors of Alabdulmohsin et al. (2022).

5.2 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 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 for the largest number of tasks of any of the functional forms. BNSL outperforms the next best functional form by 25%.

To view all plots of the BNSL on each of these tasks, see Figures 8, 9, 10 in Appendix A.4.

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	3.09e-2 ± 9.2e-4	**4.67e-3 ± 1.4e-4**	1.49e-2 ± 5.9e-4	4.96e-3 ± 4.8e-5
BB	date understanding, 2-shot	2.62e+8 Param	2.86e-2 ± 6.2e-4	2.44e-2 ± 5.6e-4	4.83e-3 ± 4.1e-4	1.58e-2 ± 5.0e-4	**2.55e-3 ± 6.2e-4**
BB	linguistic mappings, 1-shot	2.62e+8 Param	1.66e-2 ± 5.5e-4	**1.41e-2 ± 4.7e-4**	1.66e-2 ± 5.5e-4	1.57e-2 ± 4.8e-4	2.22e-2 ± 8.2e-4
BB	linguistic mappings, 2-shot	2.62e+8 Param	1.70e-2 ± 6.5e-4	1.30e-2 ± 5.4e-4	1.70e-2 ± 6.5e-4	9.25e-3 ± 5.1e-4	**8.09e-3 ± 5.1e-4**
BB	mult data wrangling, 1-shot	2.62e+8 Param	1.07e-2 ± 1.0e-3	6.71e-3 ± 7.6e-4	1.07e-2 ± 1.0e-3	8.27e-3 ± 8.4e-4	**6.17e-3 ± 8.4e-4**
BB	mult data wrangling, 2-shot	2.62e+8 Param	1.57e-2 ± 1.5e-3	7.22e-3 ± 8.9e-4	1.57e-2 ± 1.5e-3	**5.01e-3 ± 5.0e-4**	5.45e-3 ± 9.2e-4
BB	qa wikidata, 1-shot	2.62e+8 Param	**4.27e-3 ± 8.9e-4**	5.14e-3 ± 5.3e-4	4.27e-3 ± 8.9e-4	5.05e-3 ± 5.6e-4	4.34e-3 ± 6.1e-4
BB	qa wikidata, 2-shot	2.62e+8 Param	4.39e-3 ± 7.0e-4	7.35e-3 ± 6.1e-4	4.39e-3 ± 7.0e-4	7.12e-3 ± 6.0e-4	**4.20e-3 ± 6.0e-4**
BB	unit conversion, 1-shot	2.62e+8 Param	8.30e-3 ± 4.4e-4	6.77e-3 ± 4.0e-4	**1.48e-3 ± 2.7e-4**	2.32e-3 ± 2.3e-4	6.32e-3 ± 2.5e-4
BB	unit conversion, 2-shot	2.62e+8 Param	1.07e-2 ± 4.4e-4	9.15e-3 ± 4.7e-4	7.50e-3 ± 5.5e-4	**2.83e-3 ± 4.8e-4**	5.21e-3 ± 5.2e-4
LM	upstream test cross-entropy	1.07e+9 Param	1.71e-2 ± 6.0e-4	1.02e-3 ± 3.5e-5	4.50e-3 ± 5.9e-5	**3.79e-4 ± 4.0e-5**	8.26e-4 ± 3.2e-5
LM	upstream test cross-entropy	4.53e+8 Param	1.65e-2 ± 6.6e-4	6.47e-4 ± 8.5e-5	6.58e-4 ± 6.6e-5	8.54e-4 ± 4.7e-5	**5.10e-4 ± 7.7e-5**
LM	upstream test cross-entropy	2.62e+8 Param	1.55e-2 ± 7.2e-4	1.01e-3 ± 9.4e-5	3.97e-3 ± 1.3e-4	**9.37e-4 ± 9.7e-5**	1.80e-3 ± 5.6e-5
LM	upstream test cross-entropy	1.34e+8 Param	1.43e-2 ± 4.8e-4	1.33e-3 ± 6.5e-5	**6.46e-4 ± 5.1e-5**	8.14e-4 ± 5.3e-5	8.09e-4 ± 5.5e-5
LM	upstream test cross-entropy	1.68e+7 Param	6.37e-3 ± 9.4e-5	**3.08e-4 ± 1.3e-5**	1.24e-3 ± 3.2e-5	3.27e-4 ± 1.5e-5	4.05e-4 ± 1.8e-5
NMT	upstream test cross-entropy	28 Enc, 6 Dec	1.71e-1 ± 0	5.63e-2 ± 0	3.37e-2 ± 0	1.50e-2 ± 0	**5.13e-3 ± 0**
NMT	upstream test cross-entropy	6 Enc, 28 Dec	2.34e-1 ± 0	5.27e-2 ± 0	1.65e-2 ± 0	3.21e-2 ± 0	**9.81e-3 ± 0**
NMT	upstream test cross-entropy	6 Enc, 6 Dec	2.62e-1 ± 0	3.84e-2 ± 0	8.92e-2 ± 0	**1.05e-2 ± 0**	1.41e-2 ± 0
NMT	upstream test cross-entropy	Dec-only, LM	2.52e-1 ± 0	1.03e-2 ± 0	3.28e-2 ± 0	8.88e-3 ± 0	**2.79e-3 ± 0**
NMT	upstream test cross-entropy	Transformer-Enc, LSTM-Dec	1.90e-1 ± 0	1.26e-2 ± 0	6.32e-2 ± 0	1.31e-2 ± 0	**3.08e-3 ± 0**

Table 4: Extrapolation Results for Language Tasks. See Section 5.2 for more details. Numbers for M1, M2, M3, and M4 were obtained via correspondence with authors of Alabdulmohsin et al. (2022).

5.3 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 1. Other functional forms are mathematically incapable of expressing non-monotonic behaviors (as shown in Section 3).

5.4 REINFORCEMENT LEARNING

We show that BNSL accurately models the scaling behaviors of various reinforcement learning algorithms trained in various environments. In 2 left, BNSL accurately models the scaling behavior of the Reanalyse algorithm trained to play the Atari game Ms. Pacman from Figure 1 of Schrittwieser et al. (2021). In 2 right, BNSL accurately models the scaling behavior of the Proximal Policy Optimization (PPO) algorithm Schulman et al. (2017) trained to play the Procgen game called Heist.

5.5 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 3) and as a

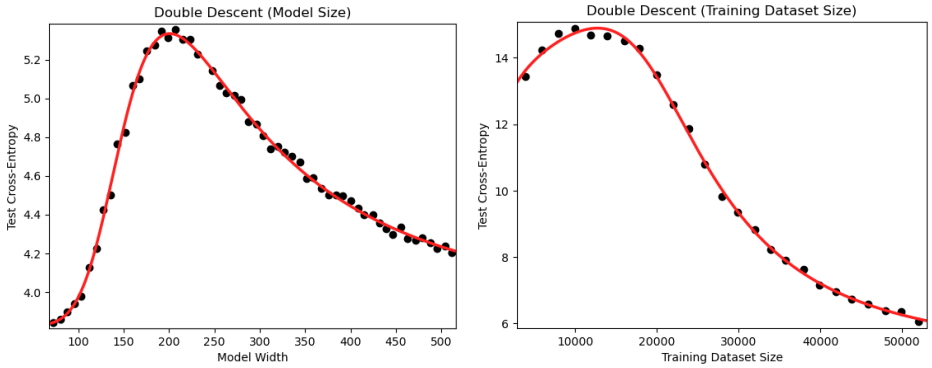

Figure 1: Fit of BNSL to Double Descent. Both plots are of transformers trained to do neural machine translation via minimizing cross-entropy. Experimental data of left figure is obtained from Figure 8 top of Nakkiran et al. (2021); "Model Width" on the x-axis refers to embedding dimension d_{model} of the transformer. Experimental data of the right figure is obtained from Figure 11b of Nakkiran et al. (2021).

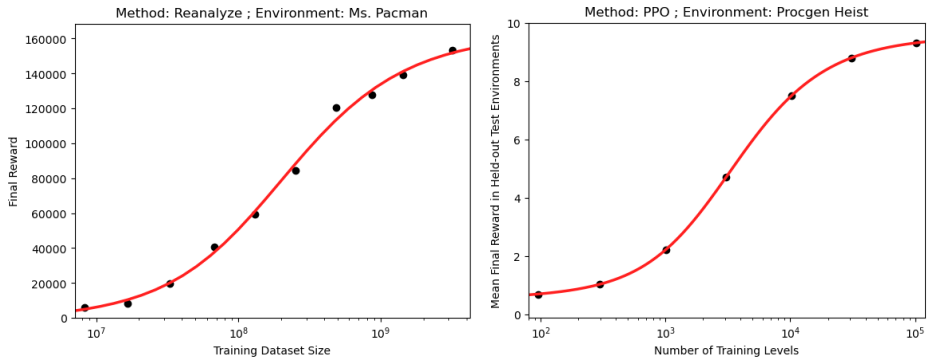

Figure 2: Fit of BNSL to Reinforcement Learning Scaling Data. Experimental Data of the left plot is from Figure 1 of Schrittwieser et al. (2021). Experimental Data of the right plot is from Figure 2 of Cobbe et al. (2020).

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 3 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 A.2.

5.6 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., 2022a) 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 3 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 3 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

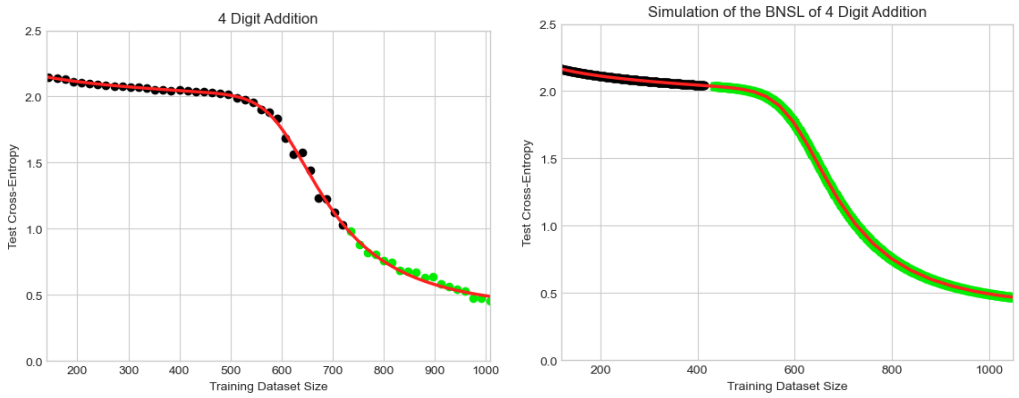

Figure 3: 4 Digit Addition. Note that these plots are linear-linear. Each point in the left plot is the mean of greater than 100 seeds at that dataset size. In the left plot, each point is gathered from a model trained to do the task of 4-digit addition. In the right plot, each point is gathered from a noiseless simulation of the BNSL of the task of 4-digit addition.

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.

6 Conclusions

Summary. 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 often are an order of magnitude 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 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. Finally, we used this functional form to obtain insights about the limit of the predictability of scaling behavior.

Limitations. A limitation of the current approach is the potential need to collect enough samples of system's performance (i.e. the (x,y) points required for estimating the scaling laws parameters). A small number of samples may not be sufficient to accurately fit and extrapolate the BNSL functional form, but obtaining a large number of such samples can sometimes be costly.

Ethical considerations. An ethical concern one might have about our work is that revealing BNSL might differentially (Hendrycks & Mazeika, 2022) improve AI capabilities progress relative to AI safety/alignment progress. A counter-argument is that BNSL will also allow AI safety/alignment field to extrapolate the scaling behaviors of its methods for aligning AI system and as a result will also accelerate alignment/safety progress. Existing scaling laws besides BNSL struggle especially to model downstream performance, e.g. on safety-relevant evaluations; we believe our work could differentially help in forecasting emergence of novel capabilities (such as reasoning (Wei et al., 2022b)) or behaviors (such as deception or dishonesty (Evans et al., 2021; Lin et al., 2021)), and thus help avoid unpleasant surprises.

REFERENCES

Direct detection of a break in the teraelectronvolt cosmic-ray spectrum of electrons and positrons. *Nature*, 552(7683):63–66, nov 2017. doi: 10.1038/nature24475. URL https://doi.org/10.1038%2Fnature24475.

Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training, 2021.

Ibrahim Mansour I Alabdulmohsin, Behnam Neyshabur, and Xiaohua Zhai. Revisiting neural scaling laws in language and vision. In *NeurIPS 2022*, 2022. URL https://arxiv.org/abs/2209.06640.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Corinna Cortes, Lawrence D Jackel, Sara A Solla, Vladimir Vapnik, and John S Denker. Learning curves: Asymptotic values and rate of convergence. In *Advances in Neural Information Processing Systems*, pp. 327–334, 1994.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Owain Evans, Owen Cotton-Barratt, Lukas Finnveden, Adam Bales, Avital Balwit, Peter Wills, Luca Righetti, and William Saunders. Truthful ai: Developing and governing ai that does not lie. *arXiv preprint arXiv:2110.06674*, 2021.

Li Fei-Fei, Rob Fergus, and Pietro Perona. 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*, pp. 178–178. IEEE, 2004.

Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, et al. Predictability and surprise in large generative models. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1747–1764, 2022.

Dan Hendrycks and Mantas Mazeika. X-risk analysis for ai research. *arXiv preprint arXiv:2206.05862*, 2022.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling Laws for Transfer. *arXiv e-prints*, art. arXiv:2102.01293, February 2021.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep Learning Scaling is Predictable, Empirically. *arXiv e-prints*, art. arXiv:1712.00409, December 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. *arXiv e-prints*, art. arXiv:2001.08361, January 2020.

Andrej Karpathy. mingpt. https://github.com/karpathy/minGPT, 2020.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pp. 491–507. Springer, 2020.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Jonathan S. Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *CoRR*, abs/1909.12673, 2019. URL `http://arxiv.org/abs/1909.12673`.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in Neural Information Processing Systems*, 34:24261–24272, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022a.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022b.

Peter Welinder, Steve Branson, Takeshi Mita, Catherine Wah, Florian Schroff, Serge Belongie, and Pietro Perona. Caltech-ucsd birds 200. 2010.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *CoRR*, abs/2106.04560, 2021. URL `https://arxiv.org/abs/2106.04560`.

A APPENDIX

You may include other additional sections here.

A.1 DEFINITION OF ROOT STANDARD LOG ERROR

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

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

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

$$Root_Standard_Log_Error = \sqrt{\mu_{error} + \frac{\sigma_{error}}{\sqrt{len(\hat{y})}}} - \sqrt{\mu_{error}}$$

A.2 EXPERIMENTAL DETAILS OF 5.5

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

A.3 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 $(a, b, c, d_1...d_n, f_1...f_n, g_1...g_n)$ 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:

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

A.4 PLOTS OF BNSL EXTRAPOLATIONS ON SCALING LAWS BENCHMARK

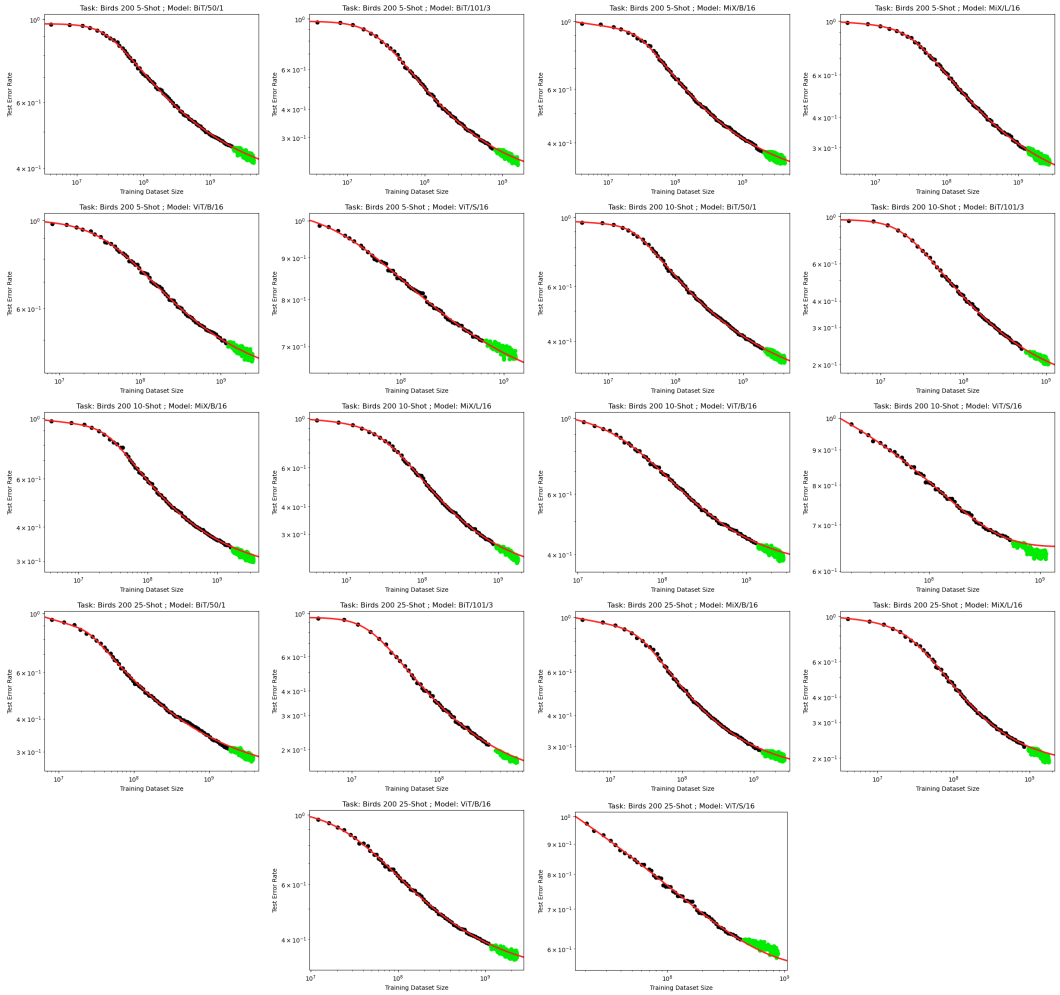

Figure 4: Birds 200

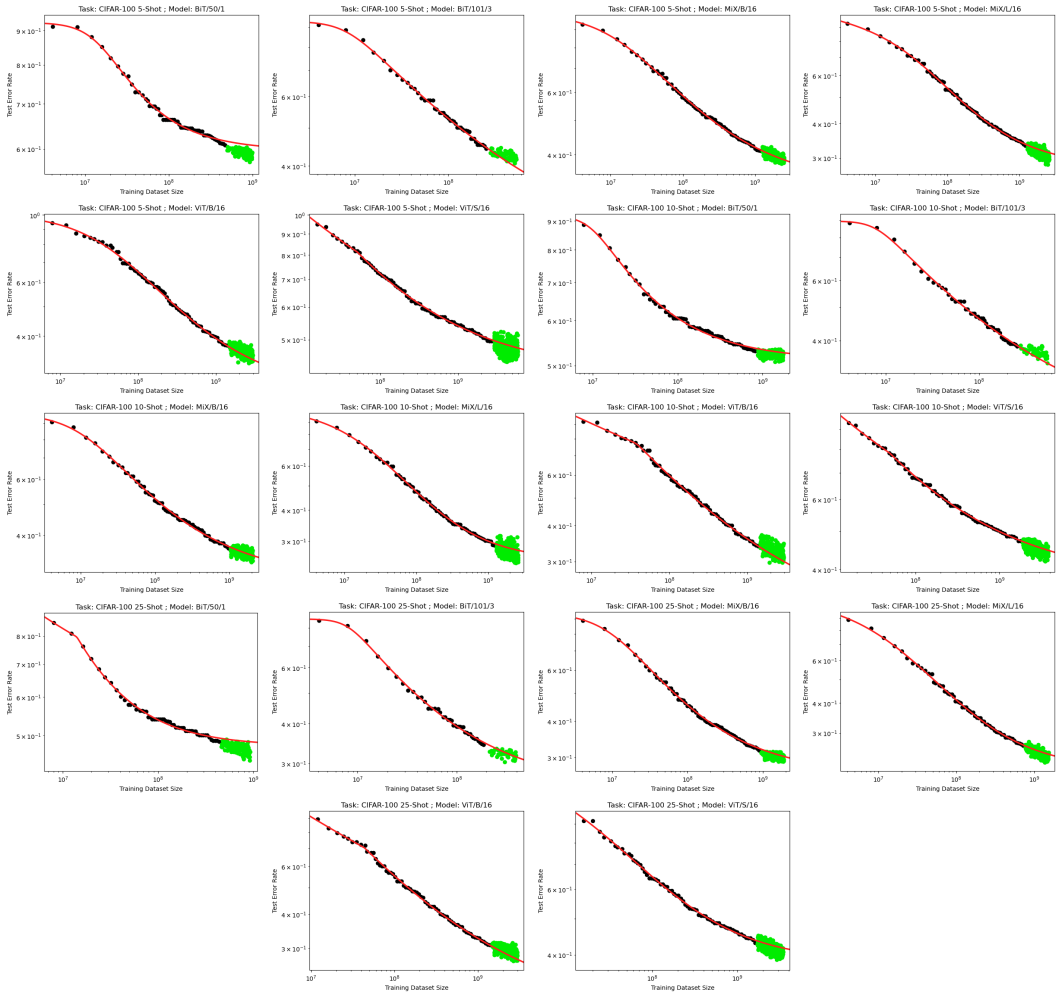

Figure 5: CIFAR-100

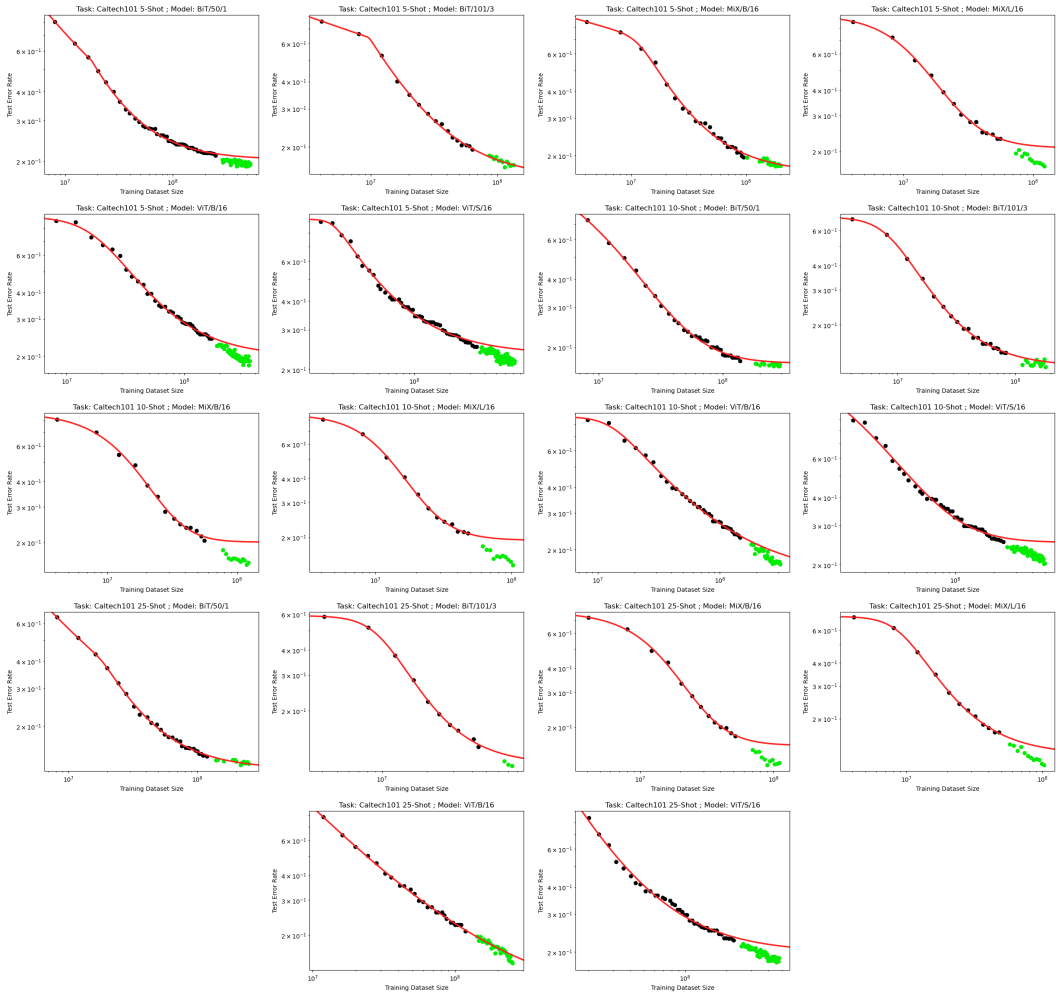

Figure 6: Caltech101

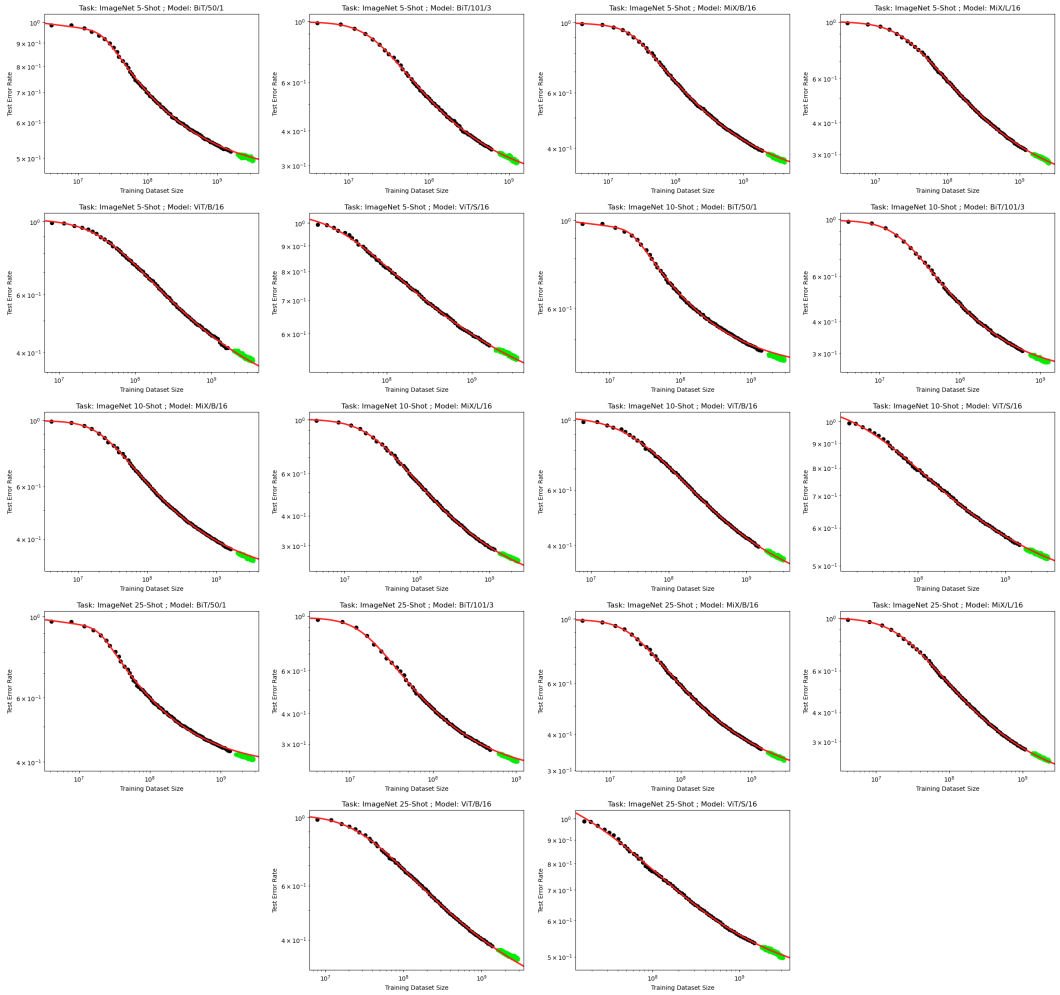

Figure 7: ImageNet

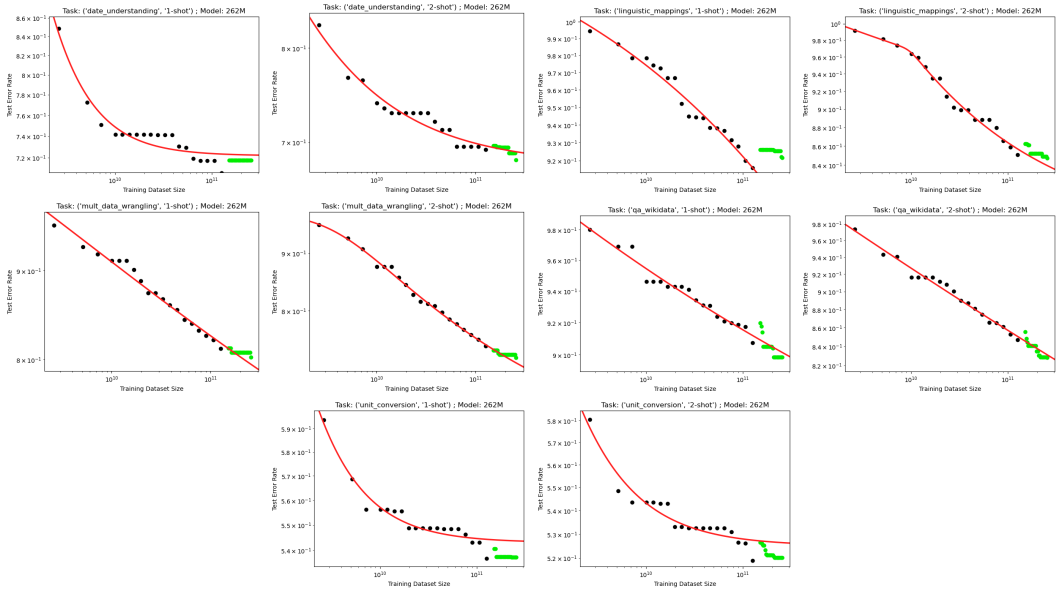

Figure 8: BIG Bench

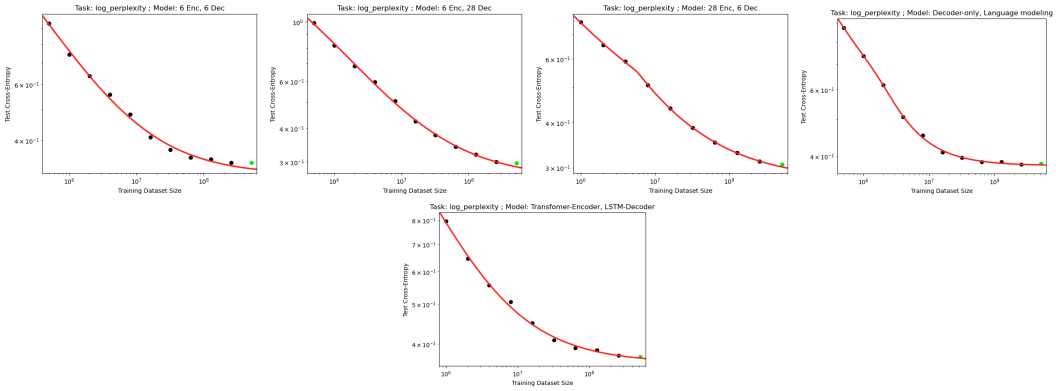

Figure 9: NMT

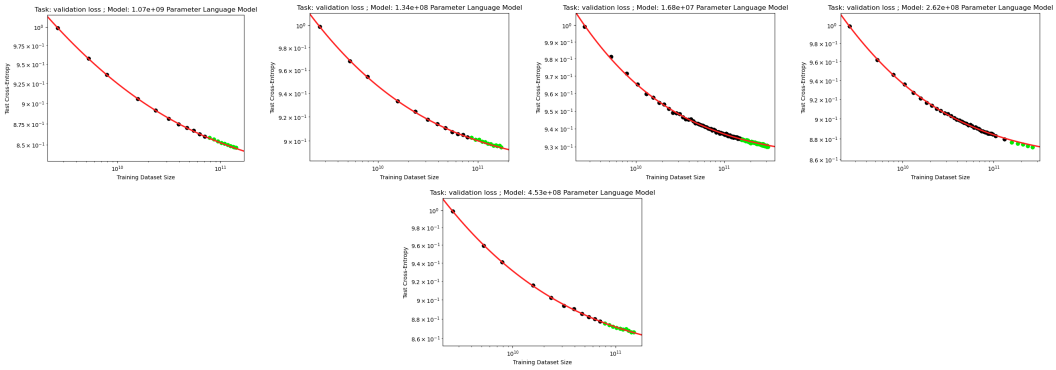

Figure 10: Language Modeling