UNIFIED NEURAL SCALING LAWS

Anonymous authors
Paper under double-blind review

ABSTRACT

We present a functional form (that we refer to as a *Unified Neural Scaling Law (UNSL)*) that accurately models and extrapolates the scaling behaviors of deep neural networks as multiple dimensions all vary simultaneously (i.e. how the evaluation metric of interest varies as one simultaneously varies the number of model parameters, training dataset size, number of training steps, number of inference steps, and various hyperparameters) for various architectures and for each of various tasks within a varied set of upstream and downstream tasks. When compared to other functional forms for neural scaling, this functional form yields **extrapolations** of scaling behavior that are considerably more accurate on this set.

1 INTRODUCTION

Training today's state-of-the-art neural networks requires significant amounts of computational resources and training data. Given a wide range of available methods and architectures to choose from, accurate forecasting of their performance is essential for selecting those that are likely to perform best at scale, especially since the top-performing methods at smaller scales often fail to maintain their performance at larger scales (Sutton, 2019; Tolstikhin et al., 2021). Moreover, accurate forecasting of neural network behaviors at scale is critical not only for identifying the top-performing approaches but also for ensuring AI safety, as predicting the emergence of novel capabilities at scale is essential for responsible development and deployment of advanced AI systems. This realization motivated the study of *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; Brown et al., 2020; Bahri et al., 2021; Alabdulmohsin et al., 2022; Caballero et al., 2023) which aim to predict the behavior of large-scale models as the amount of compute, data, and model parameters increases.

In principle, the accuracy, as well as the confidence, of predictions made by neural scaling laws can only increase (or remain the same) as a larger number of relevant predictors are included, due to the standard conditional entropy inequality, $H(Y|\mathbf{X}) \leq H(Y)$, where $\mathbf{X}$ is the vector of predictive variables and Y is the performance evaluation metric. Namely, as the number of predictive variables $X_i, i = 1, ..., m$ increases, the conditional entropy $H(Y|(X_1, ..., X_m))$ can only decrease (or remain the same). Ultimately, to obtain the maximal achievable reduction in the entropy of Y, one would need to identify the set of all possible X_i that are causally related to Y, and develop a complete model $P(Y|\mathbf{X})$ that can serve as a "unified functional form" of neural network behavior(s) at scale.

To address this need for a (more) unified functional form, we present *Unified Neural Scaling Laws (UNSL)*, a functional form that accurately models and extrapolates the scaling behaviors of deep neural networks as multiple dimensions all vary simultaneously. When compared to other functional forms for neural scaling, this functional form yields **extrapolations** of scaling behavior that are considerably more accurate on this set. Additionally, this functional form accurately models and extrapolates multivariate scaling behavior that other functional forms are incapable of expressing such as the nonmonotonic transitions present in the scaling behavior of overfitting and hyperparameters (such as learning rate and standard deviation of weights at initialization) that have a nonmonotonic relationship with the performance evaluation metric.

2 THE FUNCTIONAL FORM OF UNIFIED NEURAL SCALING LAWS

Let y denote a performance evaluation metric of interest (e.g. prediction error or cross-entropy), either "upstream" (i.e. measured on the validation dataset from the pretraining data distribution)

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

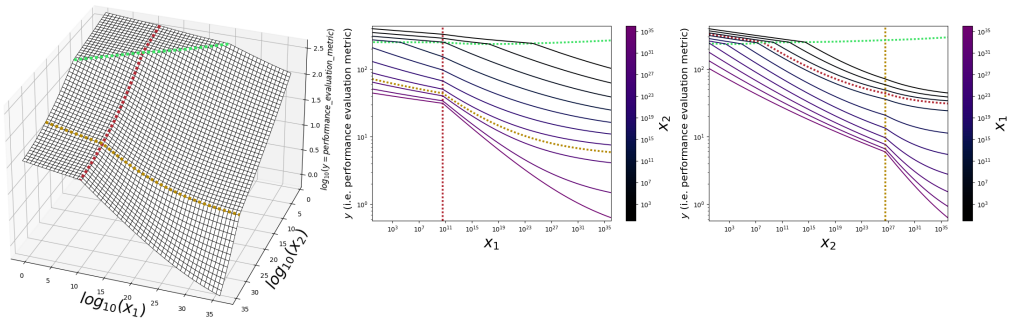

Figure 1: An illustration of a Unified Neural Scaling Law (UNSL) (dark solid lines) with two input dimensions, x_1 and x_2, in multi-log space (Section 2.1; left) and sliced along each input dimension (center, right; line color = value of the other input). In this particular example, the UNSL contains 3 hyperbreaks (smooth transitions between approximately planar pieces), highlighted by dotted lines. The red (dark-yellow) hyperbreak is created by an x_1 (x_2) bottleneck component, which depends on x_1 (x_2) alone, so it lies at a fixed value of x_1 (x_2). The green hyperbreak is created by a non-bottleneck component, so its position along x_1 shifts with x_2 and vice versa. See Sections 2.1 and 2.2 for a detailed explanation of hyperbreaks and components.

or "downstream" (i.e. measured on new data and/or tasks that the model does not encounter during pretraining), and let $(x_i)_{i=1}^m \in \overline{\mathbb{R}}_{>0}^m$ denote a tuple of m quantities ($\overline{\mathbb{R}}$ denotes the extended reals, so an x_i may be ∞) that can be viewed as predictors of y, e.g. number of model parameters, training dataset size, number of training steps, number of inference steps, and values of various hyperparameters.

Overview. We construct UNSL in five steps (Section 2.2). Each step states desiderata that neural scaling behavior satisfies (from Desiderata D1, D2, D3, ..., D7, D8; Appendix 17 gives the empirical evidence that each desideratum holds) and adds the part of UNSL that enforces them (Appendix 16 explains how each part enforces its desiderata); Table 1 (end of Section 2.2) summarizes them. Each part is built from the multivariate broken neural scaling law K (Section 2.1) and constants a via sums and reciprocals. For $A, B > 0$, $\max(A, B) \leq A + B \leq 2\max(A, B)$ and $\frac{1}{2}\min(A, B) \leq (A^{-1} + B^{-1})^{-1} \leq \min(A, B)$; as a result, in log space (up to a factor of 2) a *sum* is approximately its largest term and a *reciprocal of a sum of reciprocals* is approximately its smallest term. Steps 1, 2, 4, and 5 yield the ablations A1, A2, A3 (Section 4.0.1; the misperformance limits of Step 3 are part of A3) and UNSL (Eq. 6), which all have the same supremal expressivity (i.e. with enough hyperbreaks, each can approximate any positive continuous function arbitrarily well on any compact domain; Appendix 15); as a result, we attribute their differences in Section 4 to the desiderata that each of them enforces.

2.1 The Building Block and the Additive Symmetry

Equation 1 (below) and Equations 4–6 (Section 2.2), which are built from it, are interpreted as follows. We use the term ***multi-log space*** to refer to the $(m+1)$-dimensional space obtained by applying the logarithmic transformation to each dimension $(x_1, \ldots, x_m, y)$; in it, a power law is a hyperplane. The building block of UNSL is a *Multivariate Broken Neural Scaling Law (MBNSL)*:

$$K(M, n, k) = \underbrace{b_k \cdot \left(\prod_{i \in M} x_i^{-c_{i0_k}}\right)}_{\text{first hyperplane}} \prod_{j=1}^{n} \underbrace{\left(1 + \left(\frac{\prod_{i \in M} x_i^{c_{ij_k}}}{d_{j_k}}\right)^{\left|\frac{1}{f_{j_k}}\right|}\right)^{-f_{j_k}}}_{j\text{-th hyperbreak}}, \tag{1}$$

where $M \subseteq \{1, \ldots, m\}$ is the set of input dimensions that K depends on, n is its number of hyperbreaks, and k only indexes its constants (e.g. c_{ij_k} is the exponent of x_i in its j-th hyperbreak). Equation 1 is an extension of the univariate *broken neural scaling law (BNSL)* of Caballero et al. (2023) to multivariate settings; when $|M| = 1$, adding a constant (the performance limit) to it gives exactly the expressivity of the univariate BNSL. When $|M| > 1$, it defines smoothly connected, approximately planar pieces (hyperplanes) in multi-log space (each of dimensionality $|M|$) joined by

n smooth "hyperbreaks" (i.e. transitions, each of dimensionality $|M| - 1$), which form a sequence of $n + 1$ hyperplanes when the hyperbreaks are parallel; when $n = 0$, it becomes $b_k \prod_{i \in M} x_i^{-c_{i_{0_k}}}$. In multi-log space, b_k corresponds to the offset and $-c_{i_{0_k}}$ to the slope along $\log x_i$ of the first hyperplane (where every $\prod_{i \in M} x_i^{c_{i_{j_k}}} \ll d_{j_k}$). The j-th hyperbreak lies where $\prod_{i \in M} x_i^{c_{i_{j_k}}} = d_{j_k}$; crossing it (toward larger values of this product) changes the slope along $\log x_i$ by $-\operatorname{sign}(f_{j_k}) \cdot c_{i_{j_k}}$ (e.g. for $|M| = n = 1$: $y \propto x^{-c_0}$ before and $y \propto x^{-(c_0 + \operatorname{sign}(f) \cdot c_1)}$ after the break). f_{j_k} is its sharpness: smaller values of $|f_{j_k}|$ yield a sharper hyperbreak and regions (before and after it) that have less curvature in multi-log space; larger values yield a smoother (wider) hyperbreak and regions that have more curvature.

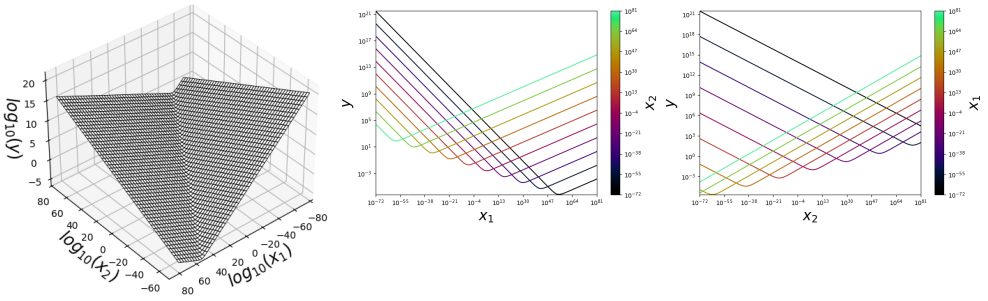

Figure 2: An example of Equation 2 (a decreasing plus an increasing power law) with two input dimensions, x_1 and x_2, in multi-log space (left) and sliced along each input dimension (center, right; line color = value of the other input). All 3 plots are of the same scaling behavior: one smooth nonmonotonic transition.

The additive symmetry: adding two power laws creates exactly one smooth hyperbreak, but one whose sharpness is not free. The following expression[1] implicitly shows up in several places (when an addition takes place) in Equations 4, 5, and 6:

$$y = b \cdot \left(\prod_{i=1}^{m} x_i^{-c_{i_0}} \right) + g \cdot \left(\prod_{i=1}^{m} x_i^{h_i} \right), \tag{2}$$

and is equivalent to a ($n = 1, M = \{1, \ldots, m\}$) version of Equation 1:

$$y = b \cdot \left(\prod_{i=1}^{m} x_i^{-c_{i_0}} \right) \left(1 + \left(\frac{\prod_{i=1}^{m} x_i^{c_{i_1}}}{d} \right)^{\left|\frac{1}{f}\right|} \right)^{-f}, \tag{3}$$

when all these equalities are simultaneously true:

$$f = -1 \ , \qquad c_{i_1} = c_{i_0} + h_i \ , \qquad d = b/g \ .$$

Equation 2 is different from Equation 3 in that (assuming b, g, and d are positive numbers):

1. For Equation 2, as any x_i increases, the slope in multi-log space can only increase from the 1st to the 2nd hyperplane, i.e. a sum of power laws only bends upward (Figure 2); for Equation 3, the slope can change by any amount of either sign.
2. For Equation 2, the sharpness of this hyperbreak is dependent solely on the amount of change in slope between the 1st and the 2nd hyperplane; for Equation 3, this sharpness is dependent on the value of f (and as a result is decoupled from the amount of change in slope).

As a result, transitions whose sharpness must be free (D1 below) get it from f inside K, while each addition in UNSL creates a transition of the form of Equation 2, i.e. Equation 3 with $f = -1$. Empirically, we observe that nonmonotonic transitions always seem to be characterized by Equation 2 rather than by Equation 3 with a free f. As a result, UNSL uses additions to model phenomena (e.g. overfitting and hyperparameters such as learning rate and standard deviation of weights at

[1] In Equation 2, b, c_{i_0}, g, and h_i are constants estimated by fitting Equation 2 to $(x_1, \ldots, x_m, y)$ data points.

initialization) that are capable of exhibiting a nonmonotonic relationship with the performance evaluation metric (for hyperparameters in Step 4, Equation 5, and for overfitting in Step 5, Equation 6, of Section 2.2). Equation 2 sums two $n = 0$ versions of MBNSL of Equation 1; to extend these relations to a summation of two MBNSLs that each have an arbitrary number of hyperbreaks n, see Appendix 7.

2.2 Constructing UNSL from Its Desiderata

The UNSL functional form satisfies all of the following desiderata (**Appendix 16 explains how UNSL satisfies each; Appendix 17 gives the empirical evidence that each holds**). "Optimal values" are the values of $(x_i)_{i=1}^{m} \in \overline{\mathbb{R}}_{>0}^{m}$ that yield the global optimum of y; we phrase everything for metrics such as error or loss, for which lower is better. A *performance limit* is a limit as to how far the value of y can be reduced (e.g. the irreducible entropy or Bayes error), and a *misperformance limit* is a limit as to how far the value of y can be increased (e.g. the loss or error of random guessing). An input x_i is a *bottleneck* when all other inputs are at their optimal values, so that the performance limit depends on x_i alone (e.g. model size when the training dataset is effectively unlimited); similarly, x_i is a bottleneck of a misperformance limit when all other inputs are at the values that yield the globally worst value of y. An *oppositional force* is a nonnegative term that is added to another part of the functional form and counteracts it (e.g. the effect of overfitting or of hyperparameters such as a too-large learning rate or a too-large standard deviation of weights at initialization, each of which counteracts learning).

Step 1: curve shape (D1, D2) and the building block K.

- **D1.** Each univariate scaling behavior (y versus one x_i, the others fixed) is a univariate *broken neural scaling law (BNSL)* of Caballero et al. (2023), i.e. approximately straight segments in log-log space joined by smooth breaks. This means that (as shown by item 2 of Section 2.1) for a significant subset of transitions between consecutive hyperplanes (in multi-log space), the sharpness needs to be decoupled from the amount of change in slope, i.e. these transitions need the free sharpness f of Equation 3 (and Equation 1).
- **D2.** Within hyperbreaks created by non-bottleneck components (each a K of several inputs; see Step 2), the position of each break of a univariate scaling behavior is shifted multiplicatively (i.e. by an offset in log space) by an amount that depends on the other input dimensions.

In Figure 7 (Appendices 17.1, 17.2), a form in which a break's position shifts multiplicatively with the other inputs, as D2 states, fits and extrapolates accurately if f is free but unsatisfactorily if $f = 1$ (i.e. if the sharpness is fixed and hence tied to the amount of change in slope), as D1 states. K enforces both: each univariate slice of K is a BNSL whose sharpness f_{j_k} is a separate constant (Appendix 16.1), and its j-th hyperbreak lies where a product of powers of the inputs equals d_{j_k}, so the position of a break along one x_i shifts multiplicatively with the other inputs (Appendix 16.2), e.g. the green hyperbreak in Figure 1. K with $M = \{1, \ldots, m\}$ alone is the ablation baseline A1 (Section 4.0.1).

Step 2: best values of y (D3, D4), bottleneck components, and a_0.

- **D3.** Whenever all but one x_i simultaneously go to their optimal values, the resulting performance limit depends on that single x_i (the one bottlenecking performance) and on no other input dimension. Sufficiently close to the global optimum of y, the transition to that performance limit follows $y = a + \sum_{t \in T} b_t \cdot x_t^{-c_t}$, where T indexes the bottlenecking x_t (when all but one x_t are at their optimal values, only that x_t's term remains).
- **D4.** The performance limit as all x_i simultaneously go to their optimal values is a constant (e.g. the irreducible entropy or Bayes error); the transition to this performance limit follows an entire functional form plus a constant (in UNSL, the constant a_0 below).

Both are observed empirically in prior work such as Hoffmann et al. (2022) (Appendices 17.3, 17.4). More generally, we empirically observe that transitions to or from regions in which the slope with respect to some x_i is zero (e.g. reaching a bottleneck or a best value, as in D3–D4, or a worst value, as in D5–D6 below; Appendices 17.3–17.6) always seem to follow Equation 2 with $h_i = 0$ for each such x_i (i.e. the added term is flat along x_i); UNSL uses that version for the additions in Equation 4 below and for the additions of the constants $a_0, a_1, \ldots$ in Steps 2–5. To enforce D3, UNSL sums a non-bottleneck component of $(x_i)_{i \in U_r}$ and "bottleneck" components, each of which depends on a single x_t, $t \in T_r$, and sets the performance limit when x_t is the bottleneck:

$$R(r) = \underbrace{K(\,U_r,\ n_{r_0},\ r{\cdot}(m{+}1))}_{\text{non-bottleneck component}} + \underbrace{\sum_{t \in T_r} K(\{t\},\ n_{r_t},\ r{\cdot}(m{+}1){+}t)}_{\text{bottleneck components}}. \tag{4}$$

In Equation 4, $U_r, T_r \subseteq \{1, \ldots, m\}$; the index r and the third argument of each K only serve to give each K its own constants. Once the non-bottleneck component is negligible, $R(r)$ is approximately the sum of the bottleneck components, so near the global optimum $a_0 + R(r)$ reduces to $a_0 + \sum_{t \in T_r} b_t \cdot x_t^{-c_t}$, i.e. D3's form with $a = a_0$ (Appendix 16.3), e.g. the form used by Hoffmann et al. (2022) (called "CF" in Section 3). To enforce D4, the constant a_0, which corresponds to the limit as to how far the value of y can be reduced even if all of $x_1, \ldots, x_m$ go to the values of $(x_i)_{i=1}^m \in \overline{\mathbb{R}}_{>0}^m$ that yield the global optimum of y, is added to the entire functional form (Appendix 16.4); $a_0 + R(r)$ is the ablation baseline A2.

Step 3: global worst value of y (D5) and misperformance limits.

D5. The misperformance limit (e.g. upper limits when using metrics such as error or loss) that is not bottlenecked by any single x_i is a constant, e.g. the loss or error of a random guessing (maximum entropy) model or, in other scenarios (e.g. very large learning rates with very few training steps; Figure 9), a value much larger than that; the transition to this misperformance limit follows the reciprocal of the sum of a functional form and a constant.

Figures 8 and 9 show such limits (Appendix 17.5). UNSL enforces D5 via terms $(A^{-1} + a^{-1})^{-1}$, which are approximately A when $A \ll a$ and approach the misperformance limit a otherwise (Appendix 16.5): $R(q)$ capped by a misperformance limit a_q (e.g. the cross-entropy or test error rate of random guessing) is the first term of Equation 5 below, and everything except a_0 is capped by a_2 (so $y \leq a_0 + a_2$), the misperformance limit due to the evaluation metric's own upper bound. For metrics unbounded above such as cross-entropy, $a_2 = \infty$ (i.e. $a_2^{-1} = 0$); for metrics bounded above such as error rate, $a_2 < \infty$ (Equation 6 below).

Step 4: bottlenecked worst values, nonmonotonic transitions (D6, D7), and oppositional forces.

D6. Whenever all but one x_i simultaneously go to the values that yield the globally worst value of y, the resulting misperformance limit depends on that single x_i (the one bottlenecking misperformance) and on no other input dimension, e.g. the high loss or error obtained when training dataset size is too small (i.e. such that overfitting occurs). Sufficiently far from the global optimum of y, the transition to that misperformance limit follows $y = \left(a + \sum_{t \in T} b_t \cdot x_t^{-c_t}\right)^{-1}$ (where c_t may be negative), i.e. the reciprocal of the form in D3.

D7. Nonmonotonic transitions (e.g. due to overfitting and hyperparameters such as learning rate and standard deviation of weights at initialization) are characterized by the additive symmetry discussed in Section 2.1, i.e. by a sum of power laws (Equation 2).

Figure 10 shows D6 (Appendix 17.6); for D7, at an equal number of learned parameters, removing the additive symmetries of Equation 5 below (i.e. fitting $a_0 + R(r)$ instead) raises extrapolation RMSLE (root mean squared log error, defined in Appendix 8) from 5.11e-2 to 8.09e-2 (Table 3; Appendix 17.7). UNSL enforces both with "oppositional forces", i.e. nonnegative terms $(R(q+s) + a_{q+s}^{-1})^{-1}$ that oppose the capped term of Step 3. Each force is approximately the reciprocal of R (whereas the capped term of Step 3 is approximately R), which turns the bottleneck form of D3 into the reciprocal form of D6 (Appendix 16.6); it increases as $R(q+s)$ decreases, up to its own misperformance limit a_{q+s}. Each force is *added* to the capped term of Step 3, which creates a transition of the form of Equation 2 (here a nonmonotonic one, as in D7; Appendix 16.7):

$$Q(q) = \underbrace{\left((R(q))^{-1} + {a_q}^{-1}\right)^{-1}}_{R(q) \text{ capped at misperformance limit } a_q} + \underbrace{\sum_{s=1}^{S} \left(R(q+s) + {a_{q+s}}^{-1}\right)^{-1}}_{\text{oppositional force of hyperparameters}}. \tag{5}$$

These forces model hyperparameters (such as learning rate and standard deviation of weights at initialization) that can worsen the performance evaluation metric (an "oppositional" relationship); for example, when learning rate and/or standard deviation of weights at initialization are too large, they exert an "oppositional force" on the value of $Q(q)$. S is the number of misperformance limits of the "oppositional force" of hyperparameters; S **does not represent the number of hyperparameters** (it only counts these misperformance limits): one term can model several hyperparameters, since $R(q+s)$ can depend on several inputs. **In practice,** $S \leq 1$ **except in relatively contrived scenarios** (e.g. fewer than 5 training steps together with a learning rate greater than 1) such as Figure 9 of

Appendix 17.5 (in which $S = 2$). The form $a_0 + ((Q(q))^{-1} + a_2^{-1})^{-1}$, in which Q's forces must model every oppositional effect (overfitting included), is the ablation baseline A3 (Section 4.0.1 uses other index names).

Step 5: overfitting (D8) and the full UNSL.

D8. The "oppositional forces" of hyperparameters oppose both "good learning" (i.e. learning that improves the evaluation metric) and "bad learning" (i.e. overfitting, which worsens it).

In A3, all forces are added together, so no hyperparameter force can reduce the force of overfitting; UNSL, which differs from A3 only in this step, extrapolates better than A3 on 14 of the 21 bivariate vision tasks of Table 5 (in Appendix 18; the argument is in Appendix 17.8). UNSL therefore gives overfitting its own oppositional force $(Q(S+4)+a_1^{-1})^{-1}$,[2] which cannot exceed the misperformance limit a_1 (as in D5; often significantly worse than random guessing). Because this force is a reciprocal, the hyperparameter forces inside $Q(S+4)$ reduce it, i.e. oppose "bad learning" (Appendix 16.8), while those in $Q(3)$ oppose "good learning". Adding the force to $Q(3)$ yields the **Unified Neural Scaling Law (UNSL)**:

$$y = \underbrace{a_0}_{\text{performance limit}} + \left(\left(Q(3) + \underbrace{\left(Q(S+4) + {a_1}^{-1}\right)^{-1}}_{\text{oppositional force of overfitting}} \right)^{-1} + \underbrace{{a_2}^{-1}}_{\text{metric's misperformance limit}} \right)^{-1} \tag{6}$$

with Q, R, and K as in Equations 5, 4, and 1; the unknown constants, estimated by fitting it to the $(x_1, \ldots, x_m, y)$ data points, are all parameters named a (limits of y), b (offsets), c (exponents), d (hyperbreak locations), or f (hyperbreak sharpnesses), with any subscripts, while n and S are selected by validation (Appendix 10) and the remaining symbols are indices or index sets (footnote 2). When one trains a model for more than one epoch, the oppositional force of overfitting becomes non-negligible and, added to $Q(3)$, yields the nonmonotonic behavior of overfitting (D7).

Step		Desideratum (Section 2.2)	Part of UNSL that enforces it (Appendix 16)	Empirical evidence (Appendix 17)
1	D1	univariate slices are BNSLs	free sharpness f_{j_k} of each hyperbreak of K (Eq. 1)	Fig. 7: accurate if f is free, not if $f = 1$
	D2	breaks shift multiplicatively	hyperbreak where $\prod_{i \in M} {x_i}^{c_{i j_k}} = d_{j_k}$ in K (Eq. 1)	Fig. 7 (left): accurate extrapolation/fit
2	D3	bottlenecked best values	bottleneck components of R (Eq. 4)	prior work, e.g. Hoffmann et al. (2022)
	D4	global best value	constant a_0 added to all else (Eq. 6)	prior work, e.g. Hoffmann et al. (2022)
3	D5	global worst values	$(A^{-1} + a^{-1})^{-1}, a \in \{a_1, a_2, \ldots\}$ (Eqs. 5, 6)	Figs. 8, 9
4	D6	bottlenecked worst values	inverted R: $(R + a^{-1})^{-1}$ (Eqs. 5, 6)	Fig. 10
	D7	nonmonotonic transitions	adding oppositional forces (Eqs. 5, 6)	Tab. 3: RMSLE 5.11e-2 vs. 8.09e-2 without them
5	D8	hyperparameters oppose both good & bad learning	hyperparameter forces also inside $Q(S+4)$ (Eq. 6)	Tab. 5: UNSL beats A3 on 14 of 21 tasks

Table 1: Summary of Section 2.2, grouped by construction step.

3 Related Work

To the best of our knowledge, Rosenfeld et al. (2019) was the first to describe a functional form with multivariate input; this functional form is $y = a + b_1 {x_1}^{-c_1} + b_2 {x_2}^{-c_2}$ in which x_1 is number of model parameters and x_2 is training dataset size. Hoffmann et al. (2022) used this same functional form (Kaplan et al. (2020) used related forms), but had x_2 be number of training steps multiplied by training batch size; we refer to this functional form as "CF".

Muennighoff et al. (2023) introduced this functional form (which we call "DC") with trivariate input:

[2]The purpose of the variables i, j, k, q, r, s, t is indexation. n is a bound of a product operator; as a result, each of n_{r_0} and n_{r_t} implicitly is a bound of a product operator. S is a bound of a summation operator. $M \subseteq \{1, \ldots, m\}$. M is a product index set; as a result, U_r implicitly is a product index set. T_r is a summation index set. K, Q, R are functions and the contents of the parentheses in $K(\cdot), Q(\cdot), R(\cdot)$ are arguments of those functions. Whenever an argument of K, Q, or R is obtained via addition(s) and/or multiplication(s), the sole reason that those additions and multiplications occur is to cause each instantiation of K to have a unique value for k, i.e. its own constants (e.g. in Equations 5 and 6, $Q(3)$ uses $R(3), \ldots, R(3{+}S)$ and $a_3, \ldots, a_{3+S}$, so $S{+}4$ is the next unused index).

$y = a + b_1 \cdot (U_N + U_N \cdot d_1 \cdot (1 - e^{(-1 \cdot R_N/(d_1))}))^{-c_1} + b_2 \cdot (x_3 + x_3 \cdot d_2 \cdot (1 - e^{(-1 \cdot R_D/(d_2))}))^{-c_2}$; in that functional form: $R_D = \max(0, (x_2/x_3) - 1)$, $U_N = \min(x_1, (x_3 \cdot ((c_1 \cdot b_1)/(c_2 \cdot b_2))^{(1/(c_1+c_2))})^{(c_2/c_1)} \cdot ((c_1 \cdot b_1)/(c_2 \cdot b_2))^{(1/(c_1+c_2))})$, $R_N = \max(0, (x_1/U_N) - 1)$, x_1 is number of model parameters, x_2 is number of training steps multiplied by training batch size, and x_3 is training dataset size. When training dataset size is so large that one only trains for one epoch, "DC" is mathematically identical to "CF". See Appendix 20 for additional related work.

4 Empirical Results: Fits & Extrapolations of Functional Forms

We now show the fits and **extrapolations** of various functional forms. **In all plots from here onward and in the appendix, triangle-shaped points are points used for fitting a functional form, circle-shaped points are held-out points used for evaluating extrapolation of functional form fit to the triangle-shaped points, and lines are the functional form that has been fit to triangle-shaped points. The color of each line and (the inside of) each point represents its value along the color bar dimension. Lines of the functional form are intentionally only rendered at the values of the color bar dimension for which there exists at least one (triangle-shaped or circle-shaped) point; this means that the vertical distance of each point from the line (that is the same color as that point) represents the error of the functional form when fitting (or extrapolating to) that point. 100% of the plots from here onward and in the appendix contain circle-shaped point(s) for evaluating extrapolation.**

See Appendix 10 for details on fitting UNSL, Appendix 13 for how the number of observed points used for fitting affects extrapolation accuracy, Appendix 14 for UNSL accurately extrapolating to scales an order of magnitude larger in multiple dimensions simultaneously, and Appendix 12 for how to obtain the compute-optimal values of the input dimensions from a fitted UNSL.

All the extrapolation evaluations reported in the tables (that have ↓ symbol in the top row) are reported in terms of root mean squared log error (RMSLE) ± root standard log error. See Appendix 8 for definition of RMSLE and Appendix 9 for definition of root standard log error.

4.0.1 Ablation functional forms

A1 functional form refers to the baseline ablation functional form in which all the additive symmetries discussed in Section 2.1 have been removed such that this A1 baseline functional form is Equation 1 with $M = \{1, \ldots, m\}$ (i.e. a single MBNSL of all input dimensions; the subscript k is dropped).

A2 functional form refers to the baseline ablation functional form that consists solely of Equation 4 (which consists of Equation 1) plus the constant a_0 (which corresponds to the limit as to how far the value of y can be reduced even if all of $x_1, \ldots, x_m$ go to the values of $(x_i)_{i=1}^m \in \overline{\mathbb{R}}_{>0}^m$ that yield the global optimum of y):

$$y = a_0 + K(\, U_0,\ n_{0_0},\ 0) + \sum_{t \in T_0} K(\{t\},\ n_{0_t},\ t), \quad \text{where } U_0, T_0 \subseteq \{1, \ldots, m\}.$$

A3 functional form refers to the baseline ablation functional form that consists solely of Equation 5 (which consists of Equation 4 (which consists of Equation 1)) plus the constant a_0 (which corresponds to the limit as to how far the value of y can be reduced even if all of $x_1, \ldots, x_m$ go to the values of $(x_i)_{i=1}^m \in \overline{\mathbb{R}}_{>0}^m$ that yield the global optimum of y):

$$y = a_0 + \left(\left(\left((R(0))^{-1} + {a_1}^{-1} \right)^{-1} + \sum_{s=1}^{S} \left(R(s) + {a_{s+2}}^{-1} \right)^{-1} \right)^{-1} + a_2^{-1} \right)^{-1}.$$

A2 functional form incorporates more of the additive symmetries discussed in Section 2.1 than A1 functional form does, A3 more than A2, and UNSL all of them (i.e. more than A3); correspondingly, A1, A2, A3, and UNSL enforce Desiderata D1–D2, D1–D4, D1–D7, and D1–D8, respectively (Table 1).

Domain	CF ↑	DC ↑	A1 ↑	A2 ↑	A3 ↑	UNSL ↑
Downstream Image Classification	0.00%	0.00%	8.70%	8.70%	21.74%	**60.87%**
Language (Downstream & Upstream)	0.00%	0.00%	0.00%	11.11%	0.00%	**88.89%**

Table 2: Percentage of tasks by domain where each functional form is the best for **extrapolation** of scaling behavior. See Sections 4.1 and 4.2 for more details.

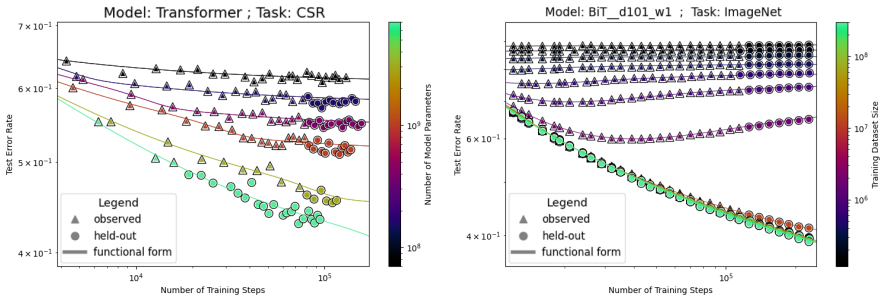

Figure 3: UNSL accurately extrapolating downstream performance; there are many additional accurate extrapolation results in Appendix 19. Experimental data of scaling behavior in left plot is downstream zero-shot performance on CSR (Common Sense Reasoning); see Section 4.2 for more details. Experimental data of scaling behavior in right plot is few-shot downstream performance on ImageNet; see Section 4.1 for more details.

4.0.2 Summary of Results

As shown in the Appendix, A1, A2, A3, and UNSL all have the exact same supremal expressivity. As a result, we attribute the fact that UNSL is better for extrapolation than A1, A2, and A3 in Table 2 to the fact that UNSL enforces more of the desiderata (of Section 2.2) (e.g. via incorporating all of the symmetries discussed in Section 2.1) than A1, A2, and A3 do.

4.1 Vision

We evaluate how well various functional forms extrapolate performance on downstream vision tasks as multiple dimensions vary simultaneously. The tasks that are evaluated are test error rate on each of various few-shot downstream image classification tasks; the downstream tasks are: Birds 200 (Welinder et al., 2010), Cars 196 (Krause et al., 2013), and ImageNet (Deng et al., 2009). The following architectures of various sizes are pre-trained on subsets of JFT-300M (Sun et al., 2017): vision transformers (ViT) (Dosovitskiy et al., 2020), MLP mixers (MiX) (Tolstikhin et al., 2021), and big-transfer residual neural networks (BiT) (Kolesnikov et al., 2020). The bivariate subset of this scaling behavior data is obtained via correspondence with authors of Alabdulmohsin et al. (2022); the simultaneously varying dimensions of the bivariate scaling behavior are training dataset size and number of training steps. The trivariate subset of this scaling behavior data is obtained from the ViT/16 results of Zhai et al. (2022); the simultaneously varying dimensions of the trivariate scaling behavior are training dataset size, number of training steps, and number of model parameters. As can be seen in Table 2 (and Tables 4 and 5 of Appendix 18), UNSL yields extrapolations with the lowest RMSLE for 60.87% of tasks of any of the functional forms, while the next best functional form performs the best on only 21.74% of the tasks. Plots of UNSL, DC, A1, A2, and A3 on these bivariate (trivariate) scaling behaviors are, respectively, Figures 17–21 (22–26) in Appendix 19.5.

We additionally show that UNSL accurately extrapolates the multivariate scaling behavior of reinforcement learning (Appendix 19.1), multivariate scaling behavior as width and depth vary simultaneously (Appendix 19.3), multivariate scaling behavior when batch size is an input dimension to UNSL (Appendix 19.4), and the trivariate scaling behavior as learning rate, standard deviation of weights at initialization, and number of training steps all vary simultaneously (Figure 11 of Appendix 17.7).

4.2 Language

We evaluate how well various functional forms extrapolate performance on downstream (and upstream) language tasks as multiple dimensions vary simultaneously. As can be seen in Table 2 (and Tables 6 and 7 of Appendix 18), UNSL yields extrapolations with the lowest RMSLE for 88.89%

of tasks of any of the functional forms, while the next best functional form performs the best on only 11.11% of the tasks. Plots of UNSL, DC, A1, A2, and A3 on trivariate scaling behavior are, respectively, Figures 27–31 in Appendix 19.6.1; this trivariate scaling behavior data is from scaling behavior data released by Muennighoff et al. (2023), and the simultaneously varying dimensions of these trivariate scaling behaviors are number of model parameters, number of tokens processed, and number of tokens in training dataset. Plots of UNSL, CF, A1, and A2 on the bivariate scaling behaviors of Table 7 are, respectively, Figures 32–35 in Appendix 19.6.2; the simultaneously varying dimensions of these bivariate scaling behaviors are number of model parameters and number of training steps (or number of tokens processed). There is no A3 in Table 7 because UNSL becomes A3 in the scenario of Table 7, i.e. the scenario in which training dataset size is effectively infinite such that one only trains for one epoch. These bivariate scaling behaviors are obtained from Shen et al. (2024) and via correspondence with authors of Hoffmann et al. (2022). CSR (Common Sense Reasoning) is zero-shot mean test error rate across HellaSwag (Zellers et al., 2019), ARC (easy and challenge) (Clark et al., 2018), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), OpenBookQA (Mihaylov et al., 2018), SIQA (Sap et al., 2019), and BoolQ (Clark et al., 2019).

In Appendix 19.2, we additionally show that UNSL accurately extrapolates the multivariate scaling behavior of inference (i.e. test-time) scaling.

5 The Limit of the Predictability of Scaling Behavior

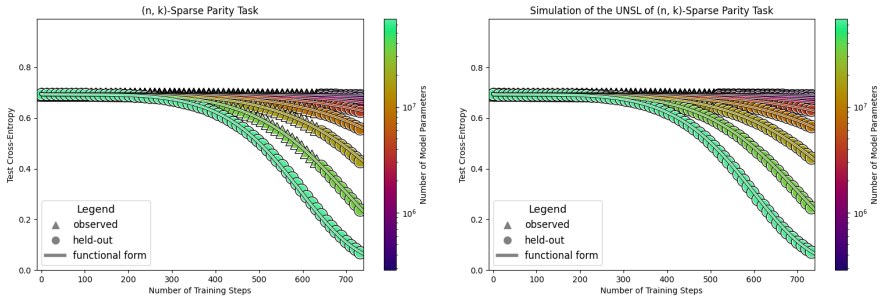

Figure 4: Extrapolation of UNSL on scaling behavior of an MLP trained for a single epoch on the (n, k)-sparse parity task (the parity of $k = 4$ of $n = 40$ input bits) of Barak et al. (2022). Each point in the left plot is the mean of greater than 100 seeds of that MLP. In the right plot, each point is gathered from a noiseless simulation of the UNSL of the scaling behavior of that (n, k)-sparse parity task. See Section 5 and Appendix 11 for more details.

We use UNSL to glean insights about the limit of the predictability of scaling behavior. In Figure 4 left, UNSL accurately extrapolates the scaling behavior of the sparse parity task of Barak et al. (2022), despite the fact that this task famously does not exhibit any observable progress in loss (or error) for the first few hundred training steps. In Figure 4 right, we use a noiseless simulation of the UNSL of the scaling behavior of the sparse parity task 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 curve-fitting) the learned constants of the UNSL as well as possible. We observe the following:

- To accurately extrapolate past each hyperbreak, the shortest distance to each hyperbreak from (the convex hull of) the points used for fitting must be sufficiently small.

6 Discussion

We have presented the Unified Neural Scaling Law (UNSL) functional form that accurately models and extrapolates the scaling behaviors of deep neural networks as multiple dimensions all vary simultaneously (i.e. how the evaluation metric of interest varies as one simultaneously varies the number of model parameters, training dataset size, number of training steps, number of inference steps, and various hyperparameters) for various architectures and for each of various tasks within a varied set of upstream and downstream tasks. When compared to other functional forms for neural scaling, this functional form yields **extrapolations** of scaling behavior that are considerably more accurate on this set.

AI USE STATEMENT

We used generative AI tools to proofread and review this paper, to help with its LaTeX formatting, to help run the experiments, and to help us understand the related literature. The authors take full responsibility for the content of this paper.

REPRODUCIBILITY STATEMENT

Appendix 10 specifies how UNSL is fit. Appendices 8 and 9 define the reported error metrics, and Appendix 11 gives experimental details for Section 5 and Appendices 13, 14, and 17. Sections 4.1 and 4.2 give the sources of the scaling behavior data summarized in Table 2, Appendix 18 reports the per-task results, and Appendix 19 plots them.

REFERENCES

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

Ben Adlam and Jeffrey Pennington. Understanding double descent requires a fine-grained bias-variance decomposition. In *Advances in Neural Information Processing Systems*, volume 33, pp. 11022–11032, 2020.

Ibrahim M. Alabdulmohsin, Xiaohua Zhai, Alexander Kolesnikov, and Lucas Beyer. Getting ViT in shape: Scaling laws for compute-optimal model design. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

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.

Fadhel Ayed and Soufiane Hayou. Data pruning and neural scaling laws: fundamental limitations of score-based algorithms. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.

Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Aleksandar Botev and James Martens. KFAC-JAX, 2022. URL https://github.com/google-deepmind/kfac-jax.

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.

Ethan Caballero, Kshitij Gupta, Irina Rish, and David Krueger. Broken neural scaling laws. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2210.14891.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018.

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.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989. doi: 10.1007/BF02551274.

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.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3460–3468, 2015.

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.

Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: Robust and efficient hyperparameter optimization at scale. In *International Conference on Machine Learning*, pp. 1437–1446. PMLR, 2018.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: Training ImageNet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Sam Greydanus and Dmitry Kobak. Scaling down deep learning with MNIST-1D. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16404–16415. PMLR, 2024.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of Statistics*, 50(2):949–986, 2022.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B. Brown, Prafulla Dhariwal, Scott Gray, Chris Hallacy, Benjamin Mann, Alec Radford, Aditya Ramesh, Nick Ryder, Daniel M. Ziegler, John Schulman, Dario Amodei, and Sam McCandlish. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Danny Hernandez, Jared Kaplan, Tom Henighan, and Sam McCandlish. Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*, 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.

Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning. *arXiv preprint arXiv:2301.13442*, 2023.

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

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2): 251–257, 1991. doi: 10.1016/0893-6080(91)90009-T.

Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization (LION)*, pp. 507–523. Springer, 2011.

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.

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.

Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pp. 554–561, 2013.

Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural Networks*, 6(6):861–867, 1993. doi: 10.1016/S0893-6080(05)80131-5.

Hao Li, Yang Zou, Ying Wang, Orchid Majumder, Yusheng Xie, R. Manmatha, Ashwin Swaminathan, Zhuowen Tu, Stefano Ermon, and Stefano Soatto. On the scalability of diffusion-based text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9400–9409, 2024.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52, 2018.

Zhengyang Liang, Hao He, Ceyuan Yang, and Bo Dai. Scaling laws for diffusion transformers. *arXiv preprint arXiv:2410.08184*, 2024.

Ziming Liu, Ouail Kitouni, Niklas S. Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *International Conference on Learning Representations*, 2023.

Mathematical Association of America. American invitational mathematics examination 2024. `https://artofproblemsolving.com/wiki/index.php/American_Invitational_Mathematics_Examination`, 2024. Accessed: 2025-10-05.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Niklas Muennighoff, Alexander M Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. *arXiv preprint arXiv:2305.16264*, 2023.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

Oren Neumann and Claudius Gros. Scaling laws for a multi-agent reinforcement learning model. *arXiv preprint arXiv:2210.00849*, 2023.

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.

Ranajoy Sadhukhan, Zhuoming Chen, Haizhong Zheng, Yang Zhou, Emma Strubell, and Beidi Chen. Kinetics: Rethinking test-time scaling laws. *arXiv preprint arXiv:2506.05333*, 2025.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4463–4473, 2019.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? In *Advances in Neural Information Processing Systems*, volume 36, pp. 55565–55581, 2023.

Xuyang Shen, Dong Li, Ruitao Leng, Zhen Qin, Weigao Sun, and Yiran Zhong. Scaling laws for linear complexity language models. *arXiv preprint arXiv:2406.16690*, 2024.

Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

Ben Sorscher, Robert Geirhos, Shashank Shekhar, Surya Ganguli, and Ari S. Morcos. Beyond neural scaling laws: beating power law scaling via data pruning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 19523–19536, 2022.

Stefano Spigler, Mario Geiger, Stéphane d'Ascoli, Levent Sagun, Giulio Biroli, and Matthieu Wyart. A jamming transition from under- to over-parametrization affects generalization in deep learning. *Journal of Physics A: Mathematical and Theoretical*, 52(47):474001, 2019.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Richard Sutton. The bitter lesson. *Incomplete Ideas (blog)*, 2019. URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Mingxing Tan and Quoc V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114. PMLR, 2019.

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.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

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

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs V: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

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

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12104–12113, 2022.

APPENDIX

7 EXTENSION OF THE ADDITIVE SYMMETRY RELATIONS DISCUSSED IN SECTION 2.1 TO A SUMMATION OF TWO MBNSLS THAT EACH HAVE AN ARBITRARY NUMBER OF HYPERBREAKS n

Note that Equation 2 sums two $n = 0$ versions of MBNSL of Equation 1. To extend the relations discussed in Section 2.1 to a summation of two MBNSLs that each have an arbitrary number of hyperbreaks n, for each of those two MBNSLs one needs to obtain the $n = 0$ version $(w_b \prod_{i=1}^{m} x_i^{w_{c_i}})$ of MBNSL of Equation 1 that is the tangent hyperplane in multi-log space. The values of w_b and w_{c_i} that yield the tangent hyperplane in multi-log space are:

$$w_{c_i} = -c_{i_0} - \sum_{j=1}^{n} \text{sign}(f_j) \cdot c_{i_j} \cdot \left(1 + \left(\frac{\prod_{l=1}^{m} x_l^{c_{l_j}}}{d_j} \right)^{-\left|\frac{1}{f_j}\right|} \right)^{-1},$$

$$w_b = b \cdot \left(\prod_{l=1}^{m} x_l^{-c_{l_0}} \right) \left(\prod_{j=1}^{n} \left(1 + \left(\frac{\prod_{l=1}^{m} x_l^{c_{l_j}}}{d_j} \right)^{\left|\frac{1}{f_j}\right|} \right)^{-f_j} \right) \prod_{i=1}^{m} x_i^{-w_{c_i}}.$$

8 DEFINITION OF ROOT MEAN SQUARED LOG ERROR

$$Root_Mean_Squared_Log_Error = RMSLE = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (\log(y_i) - \log(\hat{y}_i))^2}$$

9 DEFINITION OF ROOT STANDARD LOG ERROR

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

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

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

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

10 EXPERIMENTAL DETAILS OF FITTING UNSL

We fit the UNSL by implementing it in KFAC-JAX (Botev & Martens, 2022) and minimizing mean squared log error (MSLE):

$$MSLE = \frac{1}{N} \sum_{i=1}^{N} (\log(y_i + \epsilon) - \log(\hat{y_i} + \epsilon))^2, \tag{7}$$

with $\epsilon = 10^{-16}$. We also employ L2 regularization on the exponents of the UNSL with a weighting of λ relative to the MSLE loss term.

The values of n (from Equation 1) (and S from Equation 5) and λ that yield the lowest extrapolation error can be obtained as follows. Split the set of observed points (i.e. the triangle shaped points) used for fitting into two sets, a validation set and a training set; for each of every point in the validation set, the training set should not contain a point that is simultaneously larger than each of every x dimension ($x_1 ... x_m$) of that validation set point. The values of n, S, and λ with the lowest validation error when fitting on the remaining training points are then used. Note that once the values of n, S, and λ are identified, the validation set is added back to the training set; (and the hold-out points (i.e. the circle shaped points) are still held out to evaluate extrapolation RMSLE). In practice, $S \leq 1$ unless the scaling behavior of interest is an extravagant scaling behavior that is similar to the scaling behavior shown in Figure 9 of Appendix 17.5.

It takes 20000 training steps and 20 seeds of random initialization for KFAC-JAX to converge when fitting a UNSL. We use the JAX default "LeCun Normal" initialization as the distribution from which each random initialization (for each seed) is drawn from for parameters of UNSL. Unlike the values of n (from Equation 1) (and S from Equation 5) and λ, the optimal seed that is selected is that which yields the lowest training error (not the lowest validation error).

11 Experimental Details of Sections 5, 13, 17 (besides Figure 10), and 14

For all figures in Sections 5, 13, 17 (besides Figure 10), and 14:

- The batch size is 80000. No regularization is used because training dataset size is $\sim$infinite such that model is only trained for a single epoch. Adam is used. Adam hyperparameters are $\beta_1 = 0$ and $\beta_2 = 0$ (except for Figures 11 and 12 (and Table 3) in Section 17.7 in which $\beta_1 = 0.9$ and $\beta_2 = 0.999$). Except when learning rate and/or standard deviation of weights at initialization are explicitly varied in the plots of figures, learning rate and standard deviation of weights at initialization are held constant.

In Figure 4, number of model parameters is varied by varying width when depth is held constant.

12 Obtaining the Compute-Optimal Values of the Input Dimensions

Let $\mathcal{D}$ be the index set that contains the indexes of dimensions of $(x_1, ..., x_m)$ that directly contribute to amount of training compute used (e.g. number of model parameters, number of training steps, etc.). Let $\mathcal{H}$ be the index set that contains the indexes of dimensions of $(x_1, ..., x_m)$ that do not directly contribute to amount of training compute used (e.g. learning rate, standard deviation of weights at initialization, etc.). C is amount of training compute used. C_0 is a constant (e.g. equal to 6 in Hoffmann et al. (2022)) such that $C_0 = C/(\prod_{i \in \mathcal{D}} x_i)$. λ is a Lagrange multiplier.

To obtain the values of $(x_1, ..., x_m)$ that yield the lowest value of y for a given value of C, one solves following system of equations:

$$\frac{\partial y}{\partial x_\ell} + \lambda \frac{C}{x_\ell} = 0, \quad \ell \in \mathcal{D},$$
$$\frac{\partial y}{\partial x_v} = 0, \quad v \in \mathcal{H},$$
$$C - C_0 \prod_{\ell \in \mathcal{D}} x_\ell = 0.$$

13 Effect of varying the number of observed points used for fitting UNSL functional form

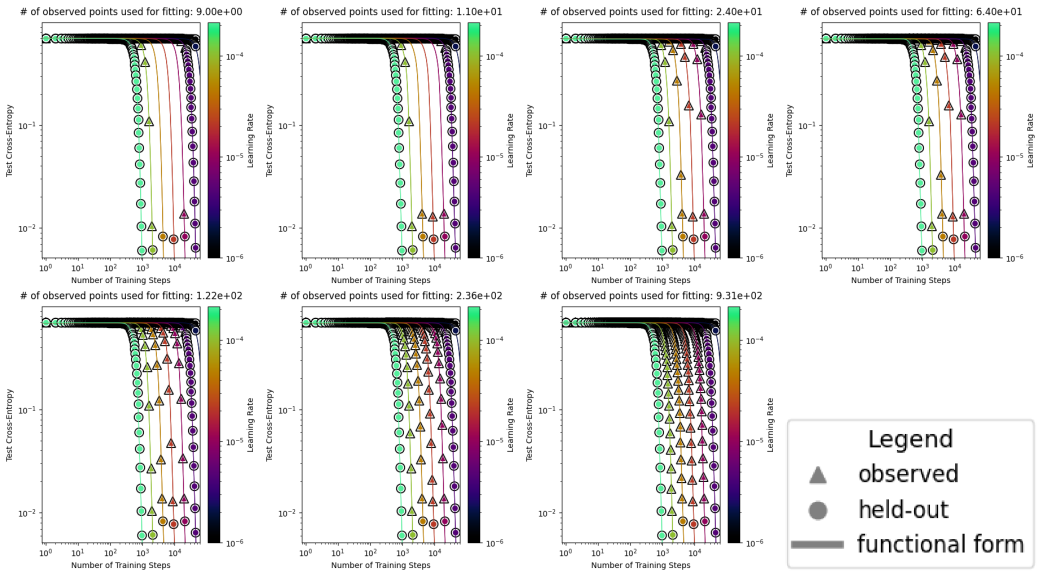

Figure 5: Varying the number of observed points used for fitting UNSL functional form from 9e0 (in top left plot) to 9e2 (in bottom right plot). Scaling behavior is that of an MLP trained for a single epoch on the (n, k)-sparse parity task (with $n = 40$ and $k = 4$) of Barak et al. (2022). See Appendix 13 for more details.

In Figure 5, we observe that UNSL accurately extrapolates scaling behavior when only a small number of observed points are used for fitting UNSL functional form.

14 UNSL ACCURATELY EXTRAPOLATING TO SCALES AN ORDER OF MAGNITUDE LARGER IN MULTIPLE DIMENSIONS SIMULTANEOUSLY

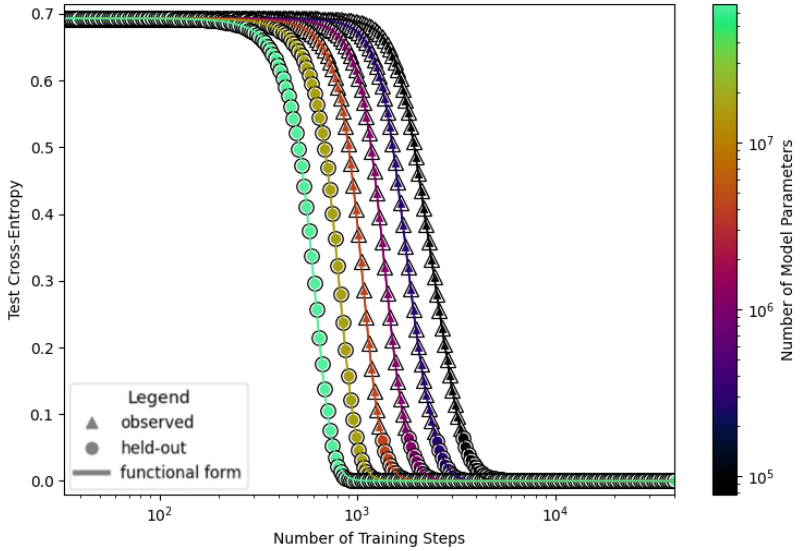

Figure 6: Extrapolation of UNSL on scaling behavior of an MLP trained for a single epoch on the (n, k)-sparse parity task (with $n = 40$ and $k = 4$) of Barak et al. (2022). Each point in the plot is the mean of greater than 100 seeds. See Section 14 and Appendix 11 for more details.

In Figure 6, UNSL accurately extrapolates to scales an order of magnitude larger in multiple dimensions simultaneously.

15 SUPREMAL EXPRESSIVITY EQUIVALENCE OF A1, A2, A3, AND UNSL

In multi-log space, MBNSL (i.e. Equation 1 and A1) with $|M| = m$ and n hyperbreaks is

$$\log y \;=\; \log b \;-\; \left(\sum_{i=1}^{m} c_{i_0}\, \log x_i\right) \;-\; \sum_{j=1}^{n} f_j \cdot \mathrm{softplus}\left(\left|\frac{1}{f_j}\right|\left(\sum_{i=1}^{m} c_{i_j}\, \log x_i \;-\; \log d_j\right)\right).$$

which is a single-hidden-layer feedforward network with softplus activation, linear skip connection, and n hidden units. Since the softplus function is continuous and non-polynomial, the universal approximation theorem for non-polynomial activations (Leshno et al., 1993; Cybenko, 1989; Hornik, 1991) ensures that $\{\text{A1} : n \in \mathbb{N}\}$ is dense in $C(\Omega, \mathbb{R}_{>0})$ for any compact $\Omega \subseteq \mathbb{R}^m_{>0}$.

A2, A3, and UNSL generate positive continuous functions of $(x_1, \ldots, x_m)$, which can therefore be arbitrarily well approximated by A1. Conversely, A1 is derived from each by specifying the corresponding parameters: $a_i^{-1} = 0$ for all i, ignoring various K components using their parameters, and (for UNSL) $S = 0$. Hence A1, A2, A3, and UNSL have identical supremal expressivity.

16 EXPLANATION OF HOW UNSL FUNCTIONAL FORM SATISFIES ALL OF THE DESIDERATA OF SECTION 2.2

16.1 EXPLANATION OF HOW UNSL FUNCTIONAL FORM SATISFIES DESIDERATUM 1

Desideratum 1 says that for each single input dimension x_i, the scaling behavior follows a univariate broken neural scaling law of Caballero et al. (2023), i.e.:

$$y = b \cdot x_i^{-c_{i0}} \prod_{j=1}^{n} \left(1 + \left(\frac{x_i^{c_{ij}}}{d_j}\right)^{\left|\frac{1}{f_j}\right|}\right)^{-f_j},$$

where b, c_{i0}, $c_{i1}...c_{ij}$, $d_1...d_j$, and $f_1...f_j$ are learned parameters. (Note that "performance limit" term a from Caballero et al. (2023) is intentionally removed here because it is addressed by other desiderata.)

This is implemented in Equation 1, where each univariate scaling behavior is modeled as a Broken Neural Scaling Law (BNSL):

$$K(M, n, k) = b_k \cdot \left(\prod_{i \in M} x_i^{-c_{i_0 k}}\right) \prod_{j=1}^{n} \left(1 + \left(\frac{\prod_{i \in M} x_i^{c_{i_{j_k}}}}{d_{j_k}}\right)^{\left|\frac{1}{f_{j_k}}\right|}\right)^{-f_{j_k}}.$$

For the pedagogical purposes of this Section 16.1, by setting $M = \{1, \ldots, m\}$ and removing subscript k one can simplify Equation 1 to:

$$y = b \cdot \left(\prod_{i=1}^{m} x_i^{-c_{i_0}}\right) \prod_{j=1}^{n} \left(1 + \left(\frac{\prod_{i=1}^{m} x_i^{c_{i_j}}}{d_j}\right)^{\left|\frac{1}{f_j}\right|}\right)^{-f_j}.$$

In that equation, when one varies only a single input dimension x_i, all x with a subscript in "$\{1, \ldots, m\} \setminus \{i\}$" become constants that can be folded into b or d_j, hence recovering the univariate broken neural scaling law of Caballero et al. (2023), i.e.:

$$y = b \cdot x_i^{-c_{i0}} \prod_{j=1}^{n} \left(1 + \left(\frac{x_i^{c_{ij}}}{d_j}\right)^{\left|\frac{1}{f_j}\right|}\right)^{-f_j}.$$

16.2 EXPLANATION OF HOW UNSL FUNCTIONAL FORM SATISFIES DESIDERATUM 2

In Equation 1, the j-th hyperbreak (i.e. smooth transition from the j-th hyperplane to the (j+1)-th hyperplane) occurs at the values of $(x_i)_{i \in M}$ for which this equality is true:

$$d_{j_k} = \prod_{i \in M} x_i^{c_{i_{j_k}}}.$$

As can be seen from this equality, the location at which each hyperbreak occurs is shifted via multiplicative interaction between (exponentiations of) input dimensions $(x_i)_{i \in M}$.

16.3 EXPLANATION OF HOW UNSL FUNCTIONAL FORM SATISFIES DESIDERATUM 3

For the pedagogical purposes of this Section 16.3, by removing subscript k one can simplify Equation 1 to:

$$y = b \cdot \left(\prod_{i \in M} x_i^{-c_{i_0}} \right) \prod_{j=1}^{n} \left(1 + \left(\frac{\prod_{i \in M} x_i^{c_{i_j}}}{d_j} \right)^{\left| \frac{1}{f_j} \right|} \right)^{-f_j}.$$

When c_{i_j} and f_j are constrained to force that functional form to always be monotonic (and assuming $x_i > 0,\ y > 0,\ b > 0,\ d_j > 0$), that functional form effectively becomes the following monomial when maximally close to the global minima with respect to y:

$$y = \left(b \cdot \prod_{\substack{j=1 \\ f_j > 0}}^{n} d_j \right) \cdot \prod_{i \in M} x_i^{-\left(c_{i_0} + \sum_{\substack{j=1 \\ f_j > 0}}^{n} c_{i_j} \right)}.$$

When Equation 1 becomes a monomial, the expressivity of Equation 4 becomes equivalent to the expressivity of this functional form:

$$b \cdot \left(\prod_{i \in U_r} x_i^{-c_{i_{0_0}}} \right) + \sum_{t \in T_r} b_t \, x_t^{-c_{t_{0_t}}},$$

which effectively becomes

$$\sum_{t \in T_r} b_t \, x_t^{-c_{t_{0_t}}}$$

when

$$b \prod_{i \in U_r} x_i^{-c_{i_{0_0}}} \ll \sum_{t \in T_r} b_t \, x_t^{-c_{t_{0_t}}}.$$

As a result, it is also true that

$$a + b \cdot \left(\prod_{i \in U_r} x_i^{-c_{i_{0_0}}} \right) + \sum_{t \in T_r} b_t \, x_t^{-c_{t_{0_t}}}$$

effectively becomes

$$a + \sum_{t \in T_r} b_t \, x_t^{-c_{t_{0_t}}}$$

when

$$b \prod_{i \in U_r} {x_i}^{-c_{i_{0_0}}} \ll \sum_{t \in T_r} b_t \, {x_t}^{-c_{t_{0_t}}}.$$

Additionally, the following functional form

$$a \; + \; \sum_{t \in T_r} b_t \, {x_t}^{-c_{t_{0_t}}}$$

effectively becomes

$$a \; + \; b_v \, {x_v}^{-c_{v_{0_v}}}, \quad (\text{where } v \in T_r)$$

when

$$\left(a + \sum_{t \in T_r \setminus \{v\}} b_t \, {x_t}^{-c_{t_{0_t}}} \right) \ll \left(a + b_v \, {x_v}^{-c_{v_{0_v}}} \right).$$

16.4 Explanation of how UNSL functional form satisfies Desideratum 4

Recall that Equation 6 is:

$$y = a_0 + \left(\left(Q(3) + \left(Q(S+4) + a_1^{-1} \right)^{-1} \right)^{-1} + a_2^{-1} \right)^{-1}.$$

Desideratum 4 is captured by the addition between a_0 and $\left(\left(Q(3) + \left(Q(S+4) + a_1^{-1} \right)^{-1} \right)^{-1} + a_2^{-1} \right)^{-1}$ in this equation, where a_0 represents the ultimate limit of performance.

16.5 Explanation of how UNSL functional form satisfies Desideratum 5

This is captured in Equations 6 and 5 when (the reciprocal of) each of every variable in the set $\{a_i\}_{i>0}$ is summed with a functional form and each resultant sum is then raised to the -1 power. The set $\{a_i\}_{i>0}$ contains multiple variables rather than a single variable because misperformance caused by different phenomena often have different misperformance limits. For example, misperformance caused by overfitting often has a misperformance limit that is significantly worse than the performance of random guessing; meanwhile, misperformance caused by nonoptimal hyperparameters often has at least one misperformance limit that is equal to the performance of random guessing. The reason that in Equation 5 a value of S greater than 1 (rather than equal to 1) is sometimes used is that there sometimes are multiple misperformance limits a_{q+s} (e.g. as in Figure 9 of Appendix 17.5): a misperformance limit that is significantly larger than random guessing (that usually is noticeable when the number of training steps is small) and a misperformance limit that approximately is less than or equal to random guessing (that usually is noticeable when the number of training steps is large).

16.6 Explanation of how UNSL functional form satisfies Desideratum 6

Recall Appendix 16.3. As a result of Appendix 16.3, Desideratum 6 is captured by each of every instance in which $R(r)$ of Equation 4 is effectively raised to the -1 power; an instance in which $R(r)$ occurs is considered "effectively raised to the -1 power" if the count of reciprocal operations whose scope contains that instance is odd. Instances in which this occurs are $\left(R(q+s) + {a_{q+s}}^{-1} \right)^{-1}$ from Equation 5 and $\left(Q(S+4) + {a_1}^{-1} \right)^{-1}$ from Equation 6 (which contains $\left((R(q))^{-1} + {a_q}^{-1} \right)^{-1}$ from Equation 5).

16.7 Explanation of how UNSL functional form satisfies Desideratum 7

This desideratum is captured when $\left((R(q))^{-1} + {a_q}^{-1}\right)^{-1}$ is summed with the "oppositional force of hyperparameters" in Equation 5, and when $Q(3)$ is summed with the "oppositional force of overfitting" in Equation 6.

16.8 Explanation of how UNSL functional form satisfies Desideratum 8

UNSL (i.e. Equation 6) functional form (expanded out for pedagogical purposes) is:

$$y = a_0 + \Bigg(\Bigg(\left((R(3))^{-1} + {a_3}^{-1}\right)^{-1} + \underbrace{\sum_{s=1}^{S} \left(R(3+s) + {a_{3+s}}^{-1}\right)^{-1}}_{\text{oppositional force of hyperparameters}} + \underbrace{\left({a_1}^{-1} + \left((R(S+4))^{-1} + {a_{S+4}}^{-1}\right)^{-1} + \underbrace{\sum_{s=1}^{S} \left(R(S+4+s) + {a_{S+4+s}}^{-1}\right)^{-1}}_{\text{oppositional force of hyperparameters}} \right)^{-1}}_{\text{oppositional force of overfitting}} \Bigg)^{-1} + {a_2}^{-1} \Bigg)^{-1}.$$

As can be seen in that expansion of UNSL, oppositional force(s) of hyperparameters oppose the "oppositional force of overfitting" and the subset of the UNSL functional form that is not the "oppositional force of overfitting". Note that each of every "oppositional force" is nonnegative and that what each of every "oppositional force" opposes is nonnegative.

17 Empirical Evidence of Desiderata of Section 2.2

17.1 Empirical Evidence of Desideratum 1

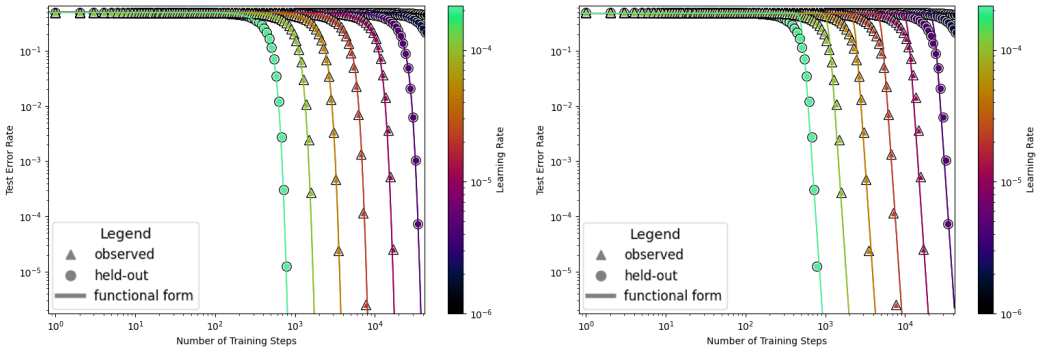

Figure 7: Extrapolation Results on scaling behavior of an MLP trained for a single epoch on the (n, k)-sparse parity task (with $n = 40$ and $k = 4$) of Barak et al. (2022). Left figure fits the functional form $y = \left(\left(\left(b\prod_{i=1}^{m} x_i^{-c_{i_0}}\right)\left(1+\left(\frac{\prod_{i=1}^{m} x_i^{c_{i_1}}}{d}\right)^{\left|\frac{1}{f}\right|}\right)^{-f}\right)^{-1} + a^{-1}\right)^{-1}$. Right figure fits the functional form of left figure when f is constrained to be 1 such that the functional form of right figure is $y = \left(\left(\left(b\prod_{i=1}^{m} x_i^{-c_{i_0}}\right)\left(1+\left(\frac{\prod_{i=1}^{m} x_i^{c_{i_1}}}{d}\right)^{\left|\frac{1}{1}\right|}\right)^{-1}\right)^{-1} + a^{-1}\right)^{-1}$. Observe that the fits and extrapolations in the top right quadrant of right figure are unsatisfactory. See Section 17.1 for more details.

In Figure 7, Desideratum 1 is true empirically. As can be seen in Figure 7, the sharpness needs to be decoupled from the amount of change in gradient (i.e. via the extra expressivity granted by f in Equation 3 (and Equation 1)) in order to accurately fit and accurately extrapolate the scaling behavior.

17.2 Empirical Evidence of Desideratum 2

Recall that in Equation 1 (with subscript k removed for pedagogical purposes) the j-th hyperbreak (i.e. smooth transition from the j-th hyperplane to the (j+1)-th hyperplane) occurs at the values of $(x_i)_{i\in M}$ for which this equality is true:

$$d_j = \prod_{i\in M} x_i^{c_{i_j}}.$$

As a result, desideratum 2 is true empirically because the functional form $y = \left(\left(\left(b\prod_{i=1}^{m} x_i^{-c_{i_0}}\right)\left(1+\left(\frac{\prod_{i=1}^{m} x_i^{c_{i_1}}}{d}\right)^{\left|\frac{1}{f}\right|}\right)^{-f}\right)^{-1} + a^{-1}\right)^{-1}$ accurately fits and accurately extrapolates the scaling behavior in Figure 7 left.

17.3 Empirical Evidence of Desideratum 3

Note that $(x_i)_{i=1}^{m} \in \overline{\mathbb{R}}_{>0}^{m}$ and that $(x_t)_{t\in T_r} \in \overline{\mathbb{R}}_{>0}^{T_r}$.

Desideratum 3 is observed empirically in several prior works such as Hoffmann et al. (2022) which empirically show that the scaling behavior follows $y = a + \sum_{t\in T_r} b_t \cdot x_t^{-c_t}$ when sufficiently close to the global optimum of y.

17.4 Empirical Evidence of Desideratum 4

Desideratum 4 is observed empirically in several prior works such as Hoffmann et al. (2022) in which the transition to the performance limit as all x_i dimensions in $x_1 ... x_m$ simultaneously go to the

values of $(x_i)_{i=1}^m \in \overline{\mathbb{R}}_{>0}^m$ that yield the global optimum of y is characterized by summing an entire functional form with a constant.

17.5 Empirical Evidence of Desideratum 5

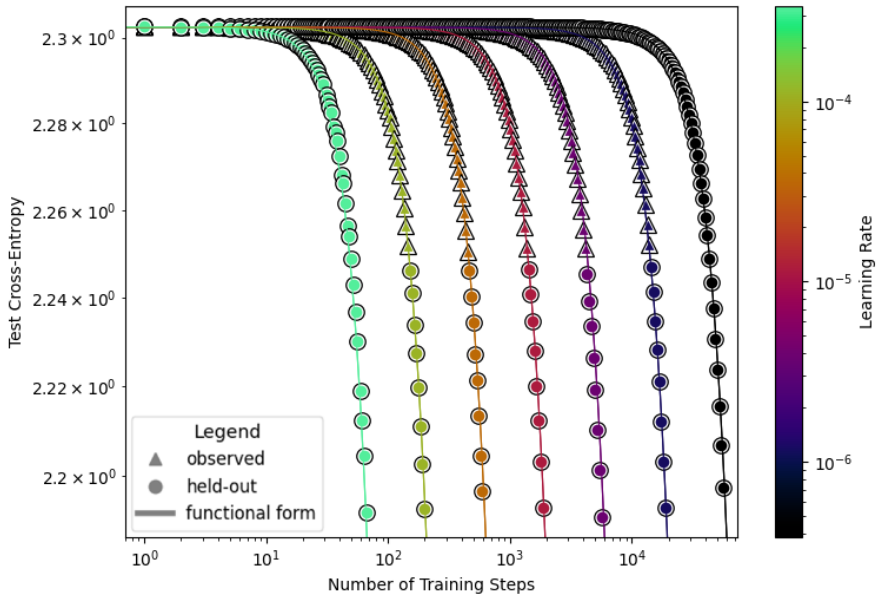

Figure 8: Extrapolation Results of functional form $y = \left(\left(b\prod_{i=1}^m x_i^{-c_i}\right)^{-1} + a^{-1}\right)^{-1}$. Scaling behavior is (top left region of) that of an MLP trained for a single epoch on dataset of Greydanus & Kobak (2024). See Section 17.5 for more details.

In Figure 8, Desideratum 5 is true empirically. As can be seen in Figure 8, the scaling behavior is characterized by raising to the -1 power the sum of a functional form and a constant.

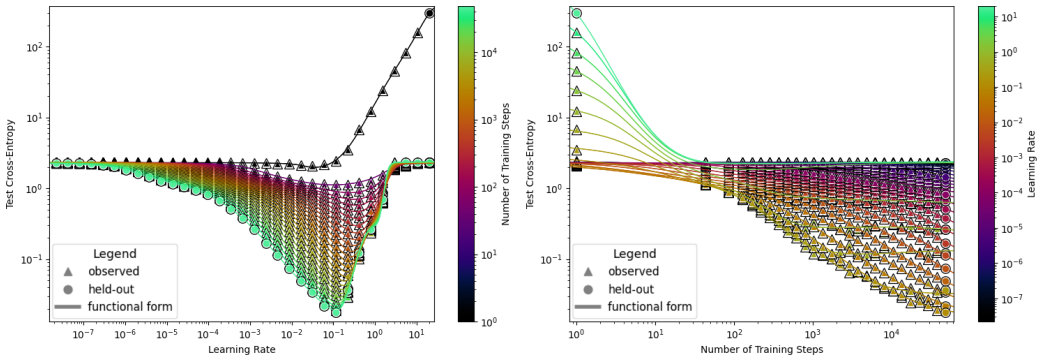

Figure 9: Extrapolation Results of UNSL functional form. Scaling behavior is that of an MLP (when standard deviation of weights at initialization is large) trained for a single epoch on dataset of Greydanus & Kobak (2024). See Section 17.5 for more details.

In Figure 9 with regards to Desideratum 5, there is a misperformance limit ~equal to random guessing performance (cross-entropy of 2.3) when it is simultaneously true that learning rate is large (i.e. greater than 3) and number of training steps is large (i.e. greater than 100); and an additional misperformance limit equal to a value significantly larger (i.e. larger than the largest y-axis value of Figure 9) than random guessing performance (cross-entropy of 2.3) occurs when it is simultaneously true that learning rate is large (i.e. significantly greater than 20) and number of training steps is small (i.e. less than 2). As a result, S (from Equation 5) is equal to 2 in Figure 9.

17.6 Empirical Evidence of Desideratum 6

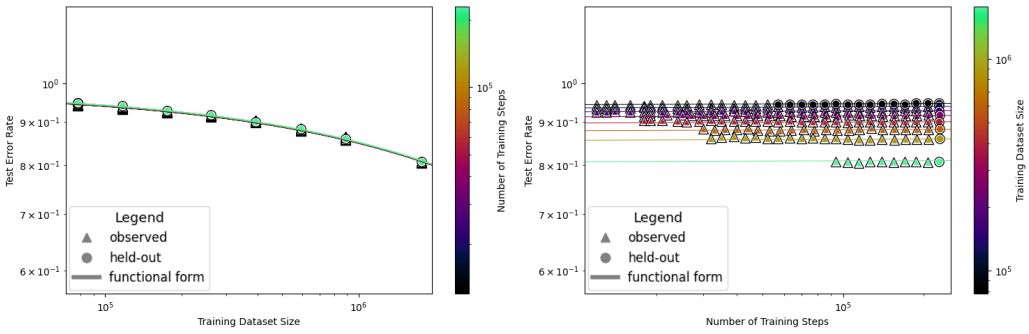

Figure 10: Extrapolation Results of functional form $y = \left(a + \sum_{t \in T_r} b_t \cdot x_t^{-c_t}\right)^{-1}$. Scaling behavior is (top region of the scaling behavior of) downstream ImageNet test error rate of ViT pre-trained on JFT. See Sections 17.6 and 4.1 for more details.

In Figure 10, Desideratum 6 is true empirically. As can be seen in Figure 10, the scaling behavior is characterized by the functional form $y = \left(a + \sum_{t \in T_r} b_t \cdot x_t^{-c_t}\right)^{-1}$.

17.7 Empirical Evidence of Desideratum 7

In Table 3 which summarizes Figures 11 and 12, Desideratum 7 is true empirically. We obtain the trivariate scaling behavior as learning rate, standard deviation of weights at initialization, and number of training steps vary when training an MLP for a single epoch on dataset of Greydanus & Kobak (2024). When holding the number of learned parameters of the functional forms constant, we compare the training and extrapolation RMSLE of UNSL to the following ablated functional form baseline in which the additive symmetries of Equation 5 are removed:

$$y = a_0 + K(U_0,\ n_{0_0},\ 0) + \sum_{t \in T_0} K(\{t\},\ n_{0_t},\ t), \quad \text{where } U_0, T_0 \subseteq \{1, \ldots, m\}. \tag{8}$$

As can be seen in Table 3 and Figures 11 and 12, when holding the number of learned parameters of the functional forms constant, UNSL yields fits and extrapolations with lower RMSLE than the ablated functional form baseline of Equation 8.

Set	Baseline ↓	UNSL ↓
Training	3.80e-2 ± 1.14e-3	**3.49e-2 ± 1.27e-3**
Extrapolation	8.09e-2 ± 5.90e-3	**5.11e-2 ± 4.30e-3**
All	4.14e-2 ± 1.26e-3	**3.60e-2 ± 1.23e-3**

Table 3: Results on trivariate scaling behavior in which Desideratum 7 is true empirically. See Section 17.7 for more details.

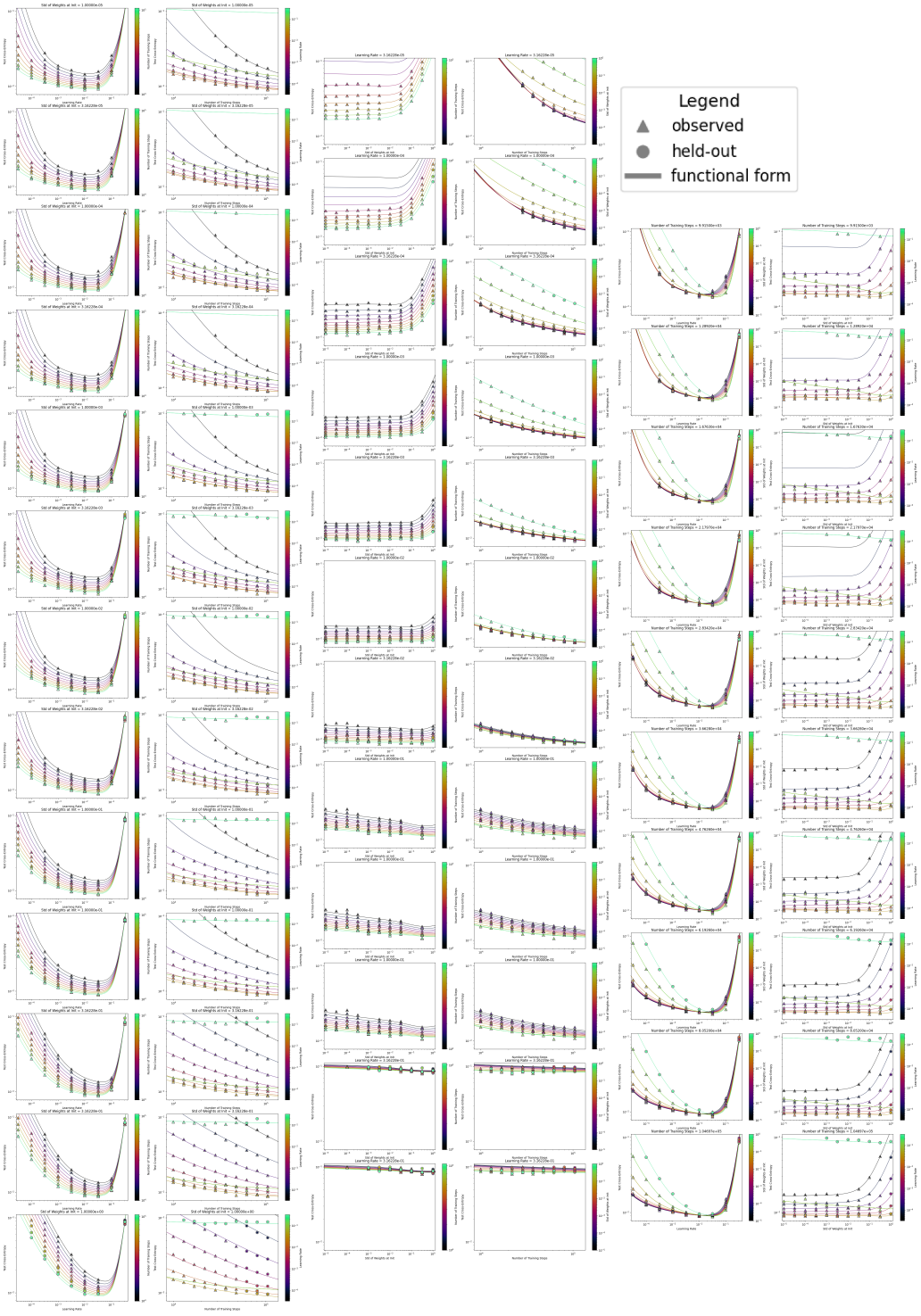

Figure 11: Extrapolation Results of UNSL. This trivariate scaling behavior is that of an MLP trained for a single epoch on dataset of Greydanus & Kobak (2024). See Section 17.7 for more details.

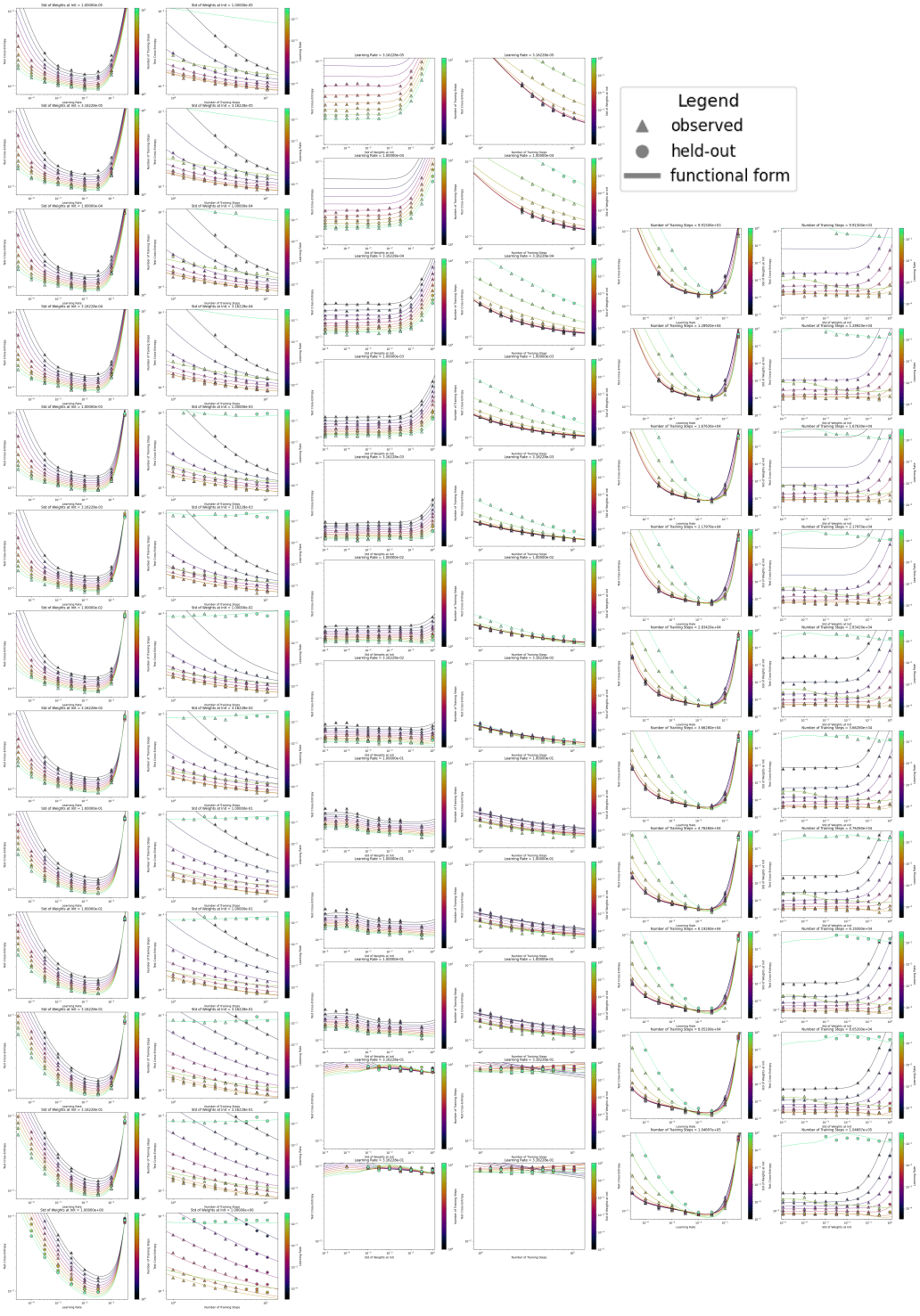

Figure 12: Extrapolation Results of ablation baseline of Equation 8. This trivariate scaling behavior is that of an MLP trained for a single epoch on dataset of Greydanus & Kobak (2024). See Section 17.7 for more details.

17.8 EMPIRICAL EVIDENCE OF DESIDERATUM 8

In Table 5, Desideratum 8 is true empirically because UNSL functional form outperforms A3 in the majority of instances.

Recall that A3 functional form is:

$$y = a_0 + \left(\left(\left((R(0))^{-1} + {a_1}^{-1} \right)^{-1} + \underbrace{\sum_{s=1}^{S} \left(R(s) + {a_{s+2}}^{-1} \right)^{-1}}_{\text{all oppositional forces in general}} \right)^{-1} + a_2^{-1} \right)^{-1}.$$

meanwhile UNSL functional form (expanded out for pedagogical purposes) is:

$$y = a_0 + \left(\left(\left((R(3))^{-1} + {a_3}^{-1} \right)^{-1} + \underbrace{\sum_{s=1}^{S} \left(R(3+s) + {a_{3+s}}^{-1} \right)^{-1}}_{\text{oppositional force of hyperparameters}} \right. \right.$$
$$\left. \left. + \underbrace{\left({a_1}^{-1} + \left((R(S+4))^{-1} + {a_{S+4}}^{-1} \right)^{-1} + \underbrace{\sum_{s=1}^{S} \left(R(S+4+s) + {a_{S+4+s}}^{-1} \right)^{-1}}_{\text{oppositional force of hyperparameters}} \right)^{-1}}_{\text{oppositional force of overfitting}} \right)^{-1} + {a_2}^{-1} \right)^{-1}.$$

As can be seen in that expansion of UNSL, oppositional force(s) of hyperparameters oppose the "oppositional force of overfitting" and the subset of the UNSL functional form that is not the "oppositional force of overfitting"; meanwhile, in A3 functional form, the "oppositional force(s) of hyperparameters" does not oppose the "oppositional force of overfitting". Note that each of every "oppositional force" is nonnegative and that what each of every "oppositional force" opposes is nonnegative.

18 PER-TASK EXTRAPOLATION RESULTS

Tables 4, 5, 6, and 7 report the per-task extrapolation results that are summarized in Table 2 (see Sections 4.1 and 4.2). In Table 7, the bivariate scaling behaviors that are referred to as "constant" are obtained from the LLaMA and HGRN2 portions of Figures 1 and 2 of Shen et al. (2024); they are referred to as "constant" because the learning rate is held constant and a learning rate schedule is not used. The bivariate scaling behaviors that are referred to as "chinchilla" are obtained via correspondence with authors of Hoffmann et al. (2022); they are called "chinchilla" because they use "chinchilla-scaling" (i.e. a learning rate schedule that is chosen to be training compute optimal as in Hoffmann et al. (2022)) and are the scaling behavior data from Hoffmann et al. (2022).

Task	DC ↓	A1 ↓	A2 ↓	A3 ↓	UNSL ↓
Birds	2.65e-1 ± 2.49e-2	7.38e-2 ± 1.41e-2	6.51e-2 ± 1.79e-2	4.77e-2 ± 7.51e-3	**4.03e-2 ± 5.51e-3**
Imagenet	2.54e-1 ± 2.02e-2	4.61e-2 ± 1.01e-2	3.39e-2 ± 1.06e-2	2.20e-2 ± 3.63e-3	**1.70e-2 ± 2.76e-3**

Table 4: Extrapolation Results for trivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

Task	Model	DC ↓	A1 ↓	A2 ↓	A3 ↓	UNSL ↓
Birds	BiT/d101/w3	3.97e-1 ± 9.84e-3	1.80e-2 ± 1.16e-3	2.12e-2 ± 1.43e-3	1.49e-2 ± 1.24e-3	**1.41e-2 ± 1.11e-3**
Birds	BiT/d50/w3	4.45e-1 ± 1.16e-2	4.29e-2 ± 2.94e-3	1.67e-2 ± 1.13e-3	**1.46e-2 ± 1.22e-3**	1.47e-2 ± 1.07e-3
Birds	MiX/L/16	4.92e-1 ± 1.15e-2	**1.39e-2 ± 8.46e-4**	1.72e-2 ± 1.05e-3	2.47e-2 ± 1.93e-3	2.62e-2 ± 1.56e-3
Birds	MiX/B/16	3.91e-1 ± 9.73e-3	2.12e-2 ± 1.88e-3	2.05e-2 ± 1.44e-3	1.90e-2 ± 1.36e-3	**1.89e-2 ± 1.23e-3**
Birds	BiT/d50/w1	3.50e-1 ± 9.39e-3	**1.16e-2 ± 7.67e-4**	1.36e-2 ± 1.03e-3	2.05e-2 ± 1.76e-3	1.59e-2 ± 1.38e-3
Birds	ViT/B/16	3.40e-1 ± 8.03e-3	5.86e-2 ± 6.60e-3	3.08e-2 ± 1.57e-3	2.98e-2 ± 1.72e-3	**2.27e-2 ± 1.38e-3**
Birds	BiT/d101/w1	3.97e-1 ± 9.84e-3	1.80e-2 ± 1.16e-3	1.29e-2 ± 9.28e-4	1.32e-2 ± 1.04e-3	**1.21e-2 ± 9.10e-4**
Cars	MiX/L/16	6.23e-1 ± 1.36e-2	5.83e-2 ± 5.45e-3	4.54e-2 ± 2.49e-3	2.15e-2 ± 2.23e-3	**2.13e-2 ± 2.13e-3**
Cars	MiX/B/16	7.05e-1 ± 1.42e-2	3.96e-2 ± 2.42e-3	2.46e-2 ± 2.15e-3	**2.30e-2 ± 1.83e-3**	3.31e-2 ± 3.26e-3
Cars	ViT/B/16	1.05e+0 ± 1.64e-2	1.36e-1 ± 9.15e-3	8.74e-2 ± 4.71e-3	4.77e-2 ± 3.14e-3	**2.74e-2 ± 1.73e-3**
Cars	BiT/d101/w3	3.03e-1 ± 7.80e-3	2.24e-2 ± 1.61e-3	2.12e-2 ± 1.43e-3	**1.75e-2 ± 1.44e-3**	1.91e-2 ± 1.35e-3
Cars	BiT/d101/w1	5.91e-1 ± 1.02e-2	3.89e-2 ± 1.97e-3	2.77e-2 ± 1.68e-3	2.46e-2 ± 1.66e-3	**2.33e-2 ± 1.69e-3**
Cars	BiT/d50/w3	3.87e-1 ± 1.29e-2	2.66e-2 ± 2.05e-3	**2.55e-2 ± 2.00e-3**	3.13e-2 ± 2.46e-3	2.87e-2 ± 2.29e-3
Cars	BiT/d50/w1	6.71e-1 ± 1.32e-2	1.99e-2 ± 1.45e-3	**1.93e-2 ± 1.28e-3**	2.51e-2 ± 1.90e-3	2.83e-2 ± 2.18e-3
Imagenet	MiX/L/16	4.30e-1 ± 9.59e-3	7.81e-3 ± 5.69e-4	1.13e-2 ± 8.20e-4	1.49e-2 ± 1.33e-3	**7.23e-3 ± 6.44e-4**
Imagenet	BiT/d101/w1	2.50e-1 ± 6.00e-3	9.52e-3 ± 8.50e-4	4.77e-3 ± 3.07e-4	5.97e-3 ± 5.66e-4	**3.74e-3 ± 2.45e-4**
Imagenet	BiT/d50/w1	2.17e-1 ± 5.36e-3	7.77e-3 ± 4.45e-4	4.40e-3 ± 2.47e-4	3.73e-3 ± 3.04e-4	**2.61e-3 ± 2.58e-4**
Imagenet	ViT/B/16	3.69e-1 ± 8.98e-3	1.41e-2 ± 1.05e-3	1.04e-2 ± 9.16e-4	1.43e-2 ± 1.02e-3	**8.57e-3 ± 7.78e-4**
Imagenet	MiX/B/16	3.21e-1 ± 8.18e-3	1.06e-2 ± 1.21e-3	7.86e-3 ± 4.83e-4	5.07e-3 ± 4.97e-4	**3.35e-3 ± 2.56e-4**
Imagenet	BiT/d101/w3	3.26e-1 ± 8.10e-3	5.44e-3 ± 4.89e-4	1.00e-2 ± 8.45e-4	**4.17e-3 ± 2.93e-4**	5.31e-3 ± 4.71e-4
Imagenet	BiT/d50/w3	3.09e-1 ± 8.07e-3	2.95e-2 ± 1.69e-3	6.19e-3 ± 3.91e-4	**3.79e-3 ± 2.50e-4**	4.20e-3 ± 2.40e-4

Table 5: Extrapolation Results for bivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

DC ↓	A1 ↓	A2 ↓	A3 ↓	UNSL ↓
6.24e-2 ± 6.00e-3	2.00e-2 ± 1.90e-3	1.96e-2 ± 3.62e-3	1.49e-2 ± 3.45e-3	**7.82e-3 ± 1.33e-3**

Table 6: Extrapolation Results for trivariate scaling behavior of language performance. See Section 4.2 for more details.

Task	Model	Scaling	CF ↓	A1 ↓	A2 ↓	UNSL ↓
Upstream	Transformer	Chinchilla	1.72e-2 ± 1.69e-3	9.85e-3 ± 1.30e-3	4.43e-3 ± 6.15e-4	**3.81e-3 ± 6.52e-4**
LAMBADA	Transformer	Chinchilla	2.08e-2 ± 2.48e-3	1.45e-2 ± 1.89e-3	1.30e-2 ± 1.80e-3	**1.13e-2 ± 1.60e-3**
CSR	Transformer	Constant	4.50e-2 ± 3.72e-3	1.43e-2 ± 1.24e-3	1.66e-2 ± 1.08e-3	**1.28e-2 ± 1.05e-3**
LAMBADA	Transformer	Constant	3.06e-2 ± 3.92e-3	4.17e-2 ± 3.52e-3	3.12e-2 ± 2.48e-3	**2.24e-2 ± 1.71e-3**
Upstream	Transformer	Constant	7.15e-2 ± 5.03e-3	3.98e-2 ± 3.27e-3	2.89e-2 ± 1.58e-3	**7.95e-3 ± 6.63e-4**
CSR	Recurrent	Constant	5.20e-2 ± 3.32e-3	2.65e-1 ± 2.87e-2	**1.15e-2 ± 9.39e-4**	1.22e-2 ± 9.15e-4
LAMBADA	Recurrent	Constant	3.02e-2 ± 2.63e-3	3.75e-2 ± 2.60e-3	4.31e-2 ± 3.62e-3	**1.66e-2 ± 1.38e-3**
Upstream	Recurrent	Constant	3.13e-2 ± 2.36e-3	3.07e-2 ± 1.99e-3	1.92e-2 ± 1.63e-3	**4.66e-3 ± 3.51e-4**

Table 7: Extrapolation Results for bivariate scaling behavior of downstream (and upstream) language performance. See Section 4.2 for more details.

19 PLOTS OF EXTRAPOLATION RESULTS

19.1 PLOTS OF REINFORCEMENT LEARNING EXTRAPOLATION RESULTS

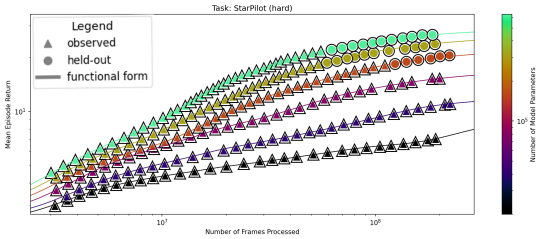

Figure 13: Extrapolation Results of UNSL on scaling behavior of reinforcement learning. Experimental data obtained from Figure 1a of Hilton et al. (2023). Scaling behavior is that of "StarPilot (hard)" task of Cobbe et al. (2020). X-axis is number of frames processed (i.e. "batch size" multiplied by "number of training steps") during training. See Section 19.1 for more details.

In Figure 13, UNSL accurately extrapolates the multivariate scaling behavior of reinforcement learning.

19.2 PLOTS OF INFERENCE SCALING EXTRAPOLATION RESULTS

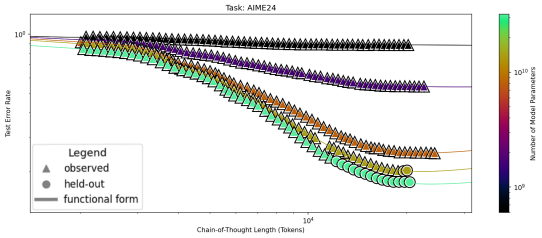

Figure 14: Extrapolation Results of UNSL on scaling behavior of inference scaling. Experimental data obtained from Figure 4a of Sadhukhan et al. (2025). Scaling behavior is that of test error rate on Mathematical Association of America (2024). X-axis is "Chain-of-Thought Length" (measured in number of tokens), i.e. how many "thinking" tokens a model outputs before outputting a final answer. See Section 19.2 for more details.

In Figure 14, UNSL accurately extrapolates the multivariate scaling behavior of inference (i.e. test-time) scaling.

19.3 PLOTS OF "WIDTH VS DEPTH" EXTRAPOLATION RESULTS

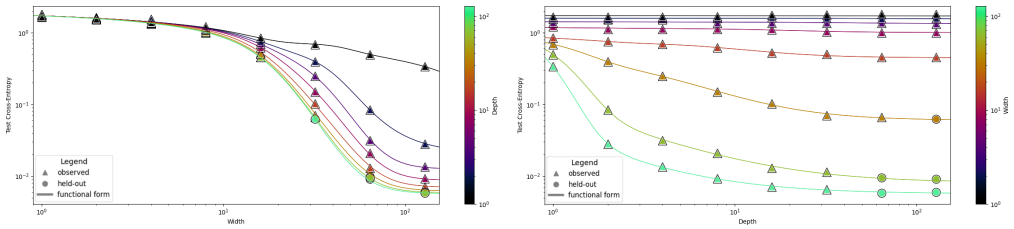

Figure 15: Extrapolation Results of UNSL on multivariate scaling behavior as width and depth simultaneously vary. Scaling behavior is that of an MLP trained for a single epoch on dataset of Greydanus & Kobak (2024). See Section 19.3 for more details.

In Figure 15, UNSL accurately extrapolates multivariate scaling behavior as width and depth vary simultaneously.

19.4 PLOTS OF MULTIVARIATE "BATCH SIZE" EXTRAPOLATION RESULTS

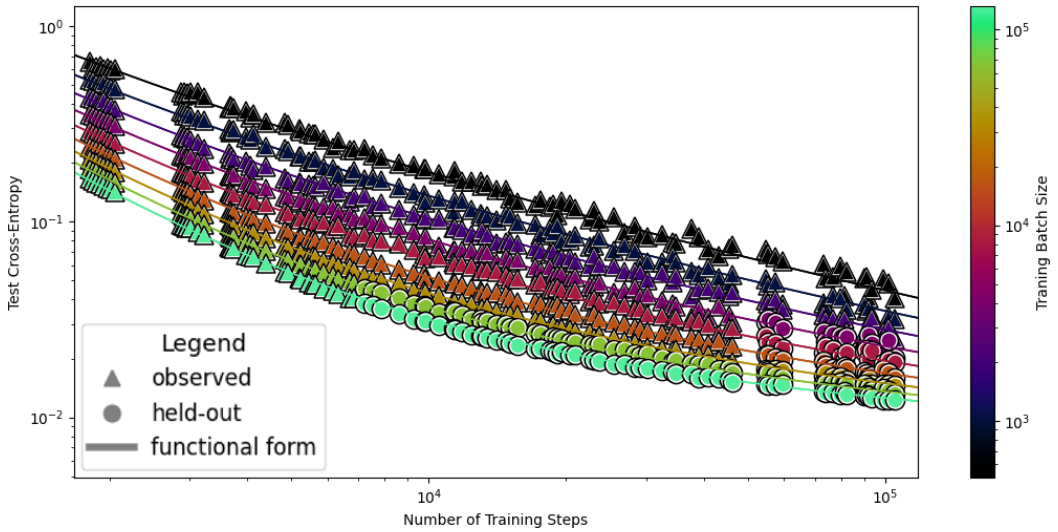

Figure 16: Extrapolation Results of UNSL on multivariate scaling behavior as batch size and number of training steps simultaneously vary. Scaling behavior is that of an MLP trained for a single epoch on dataset of Greydanus & Kobak (2024). See Section 19.4 for more details.

In Figure 16, UNSL accurately extrapolates multivariate scaling behavior when batch size is an input dimension to UNSL.

19.5 Plots of Vision Extrapolation Results

19.5.1 Bivariate

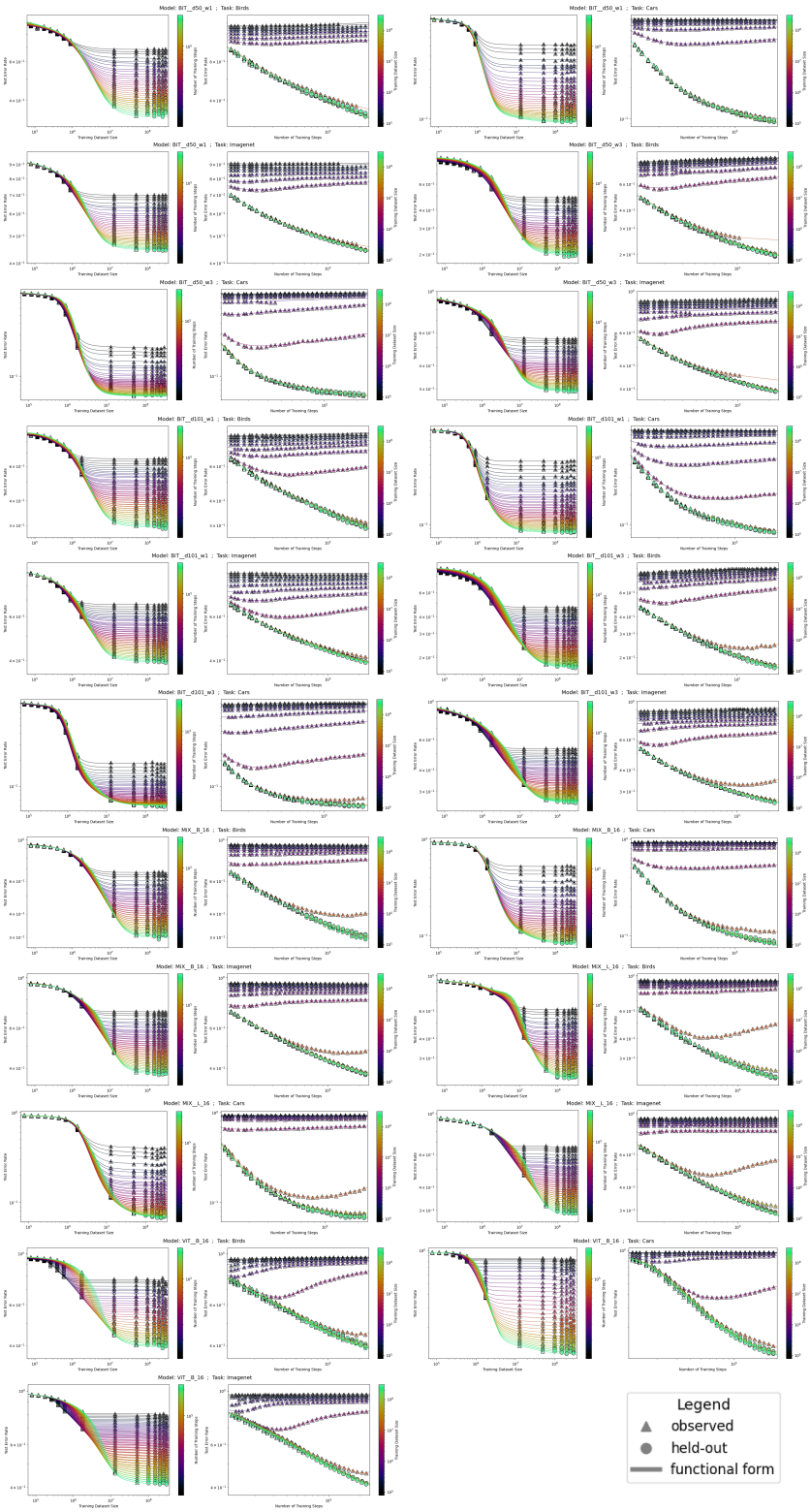

Figure 17: Extrapolation Results of UNSL on bivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

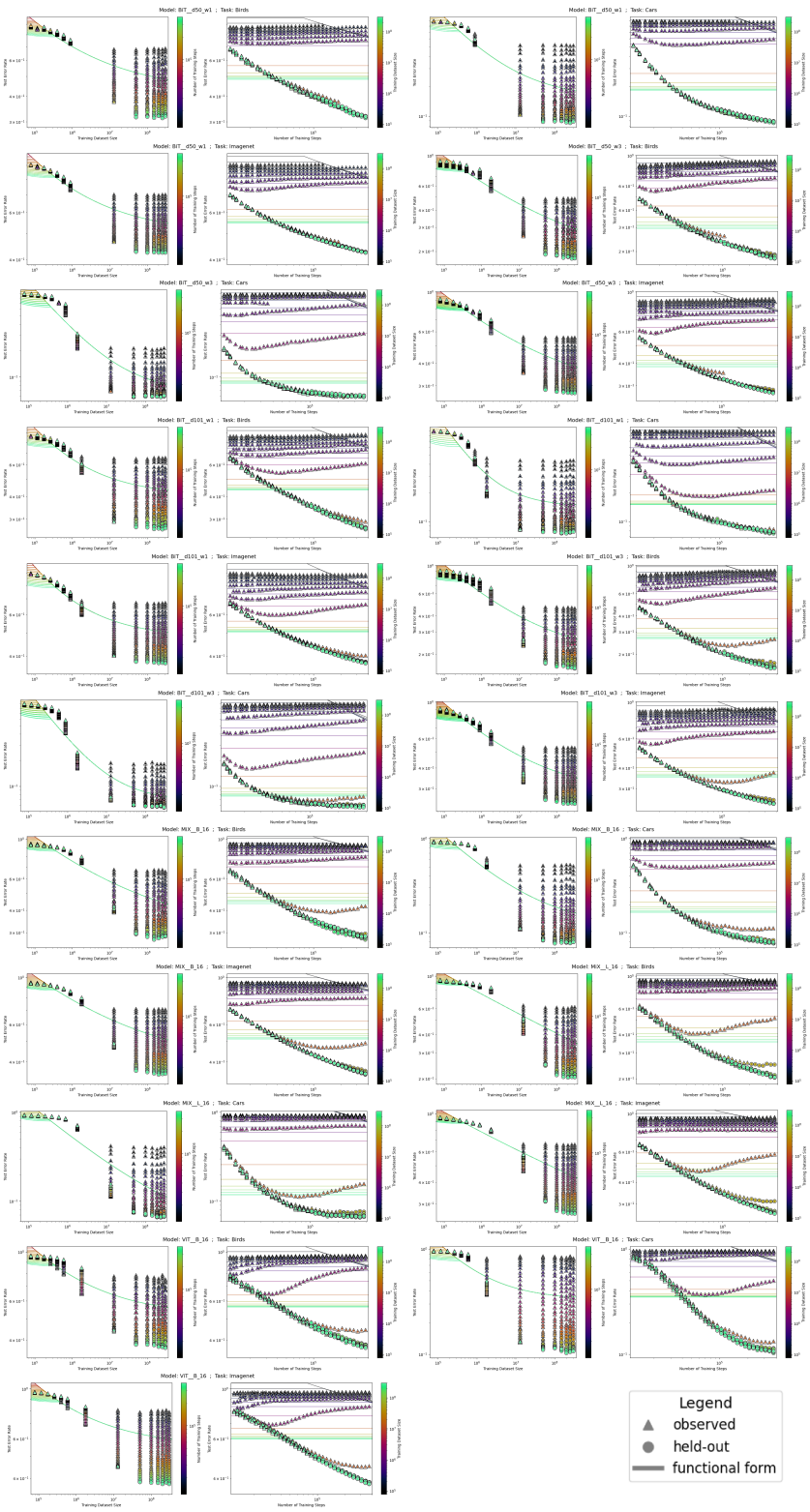

Figure 18: Extrapolation Results of "DC" functional form of Muennighoff et al. (2023) on bivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

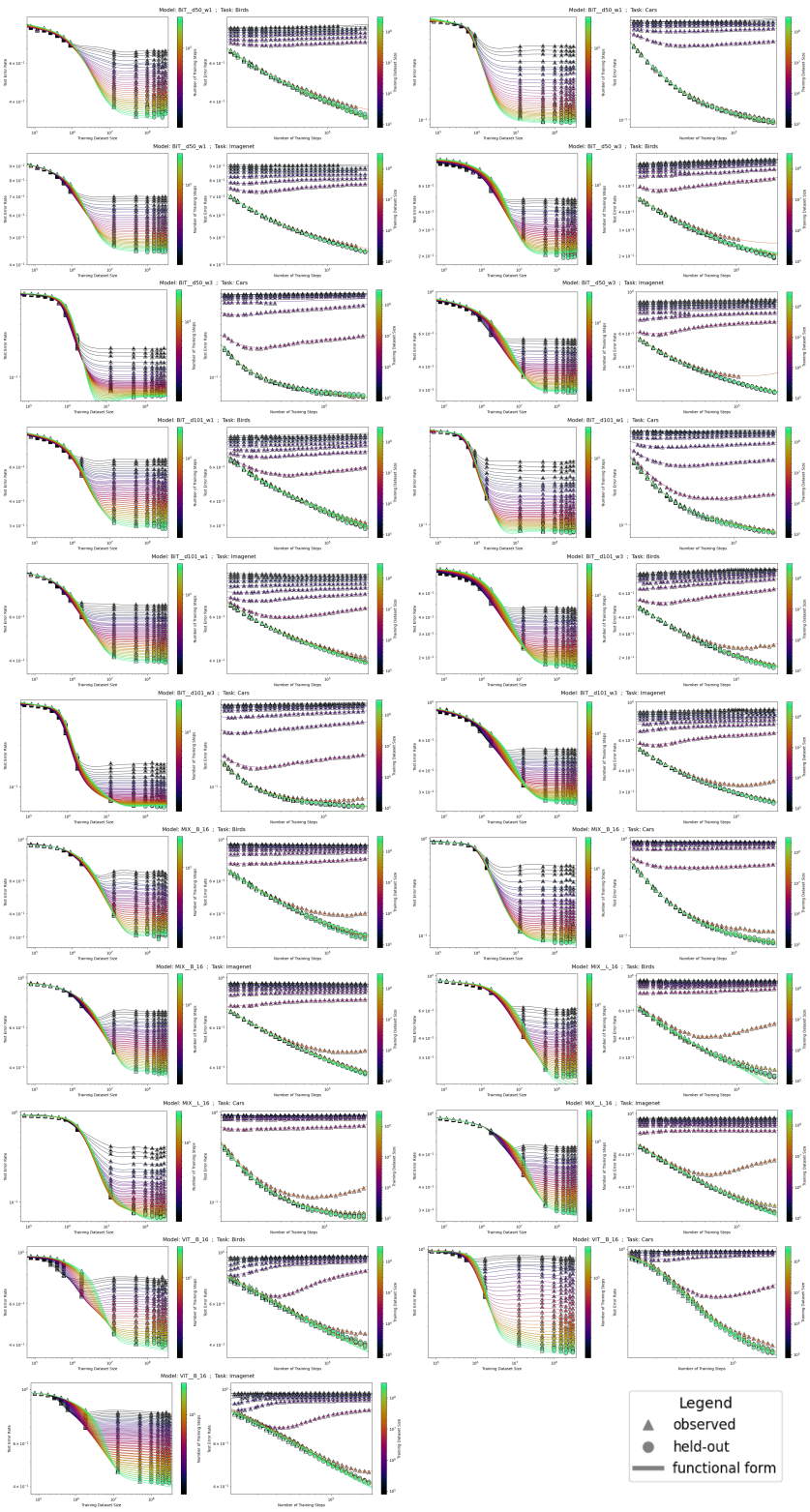

Figure 19: Extrapolation Results of A1 functional form on bivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

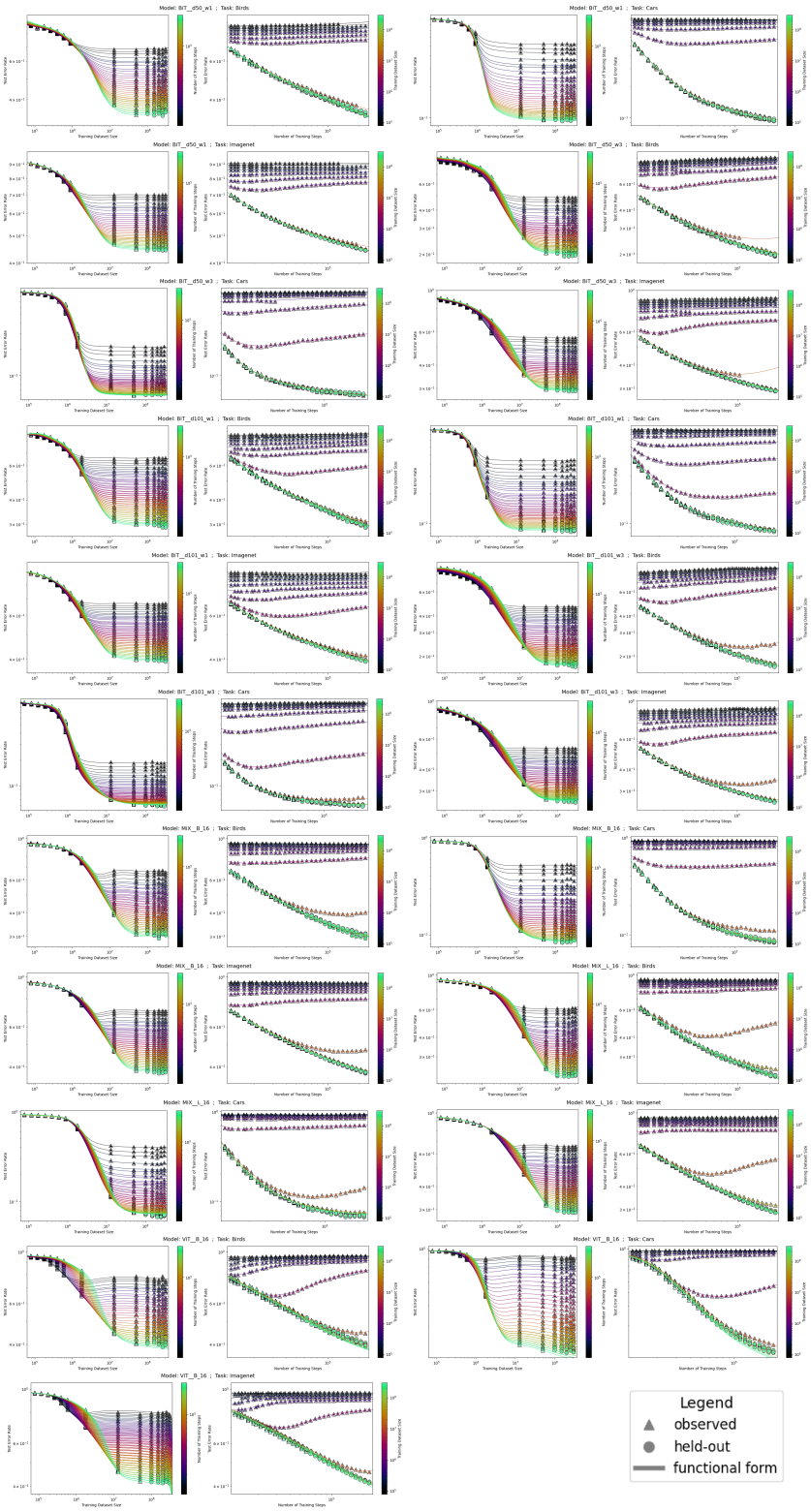

Figure 20: Extrapolation Results of A2 functional form on bivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

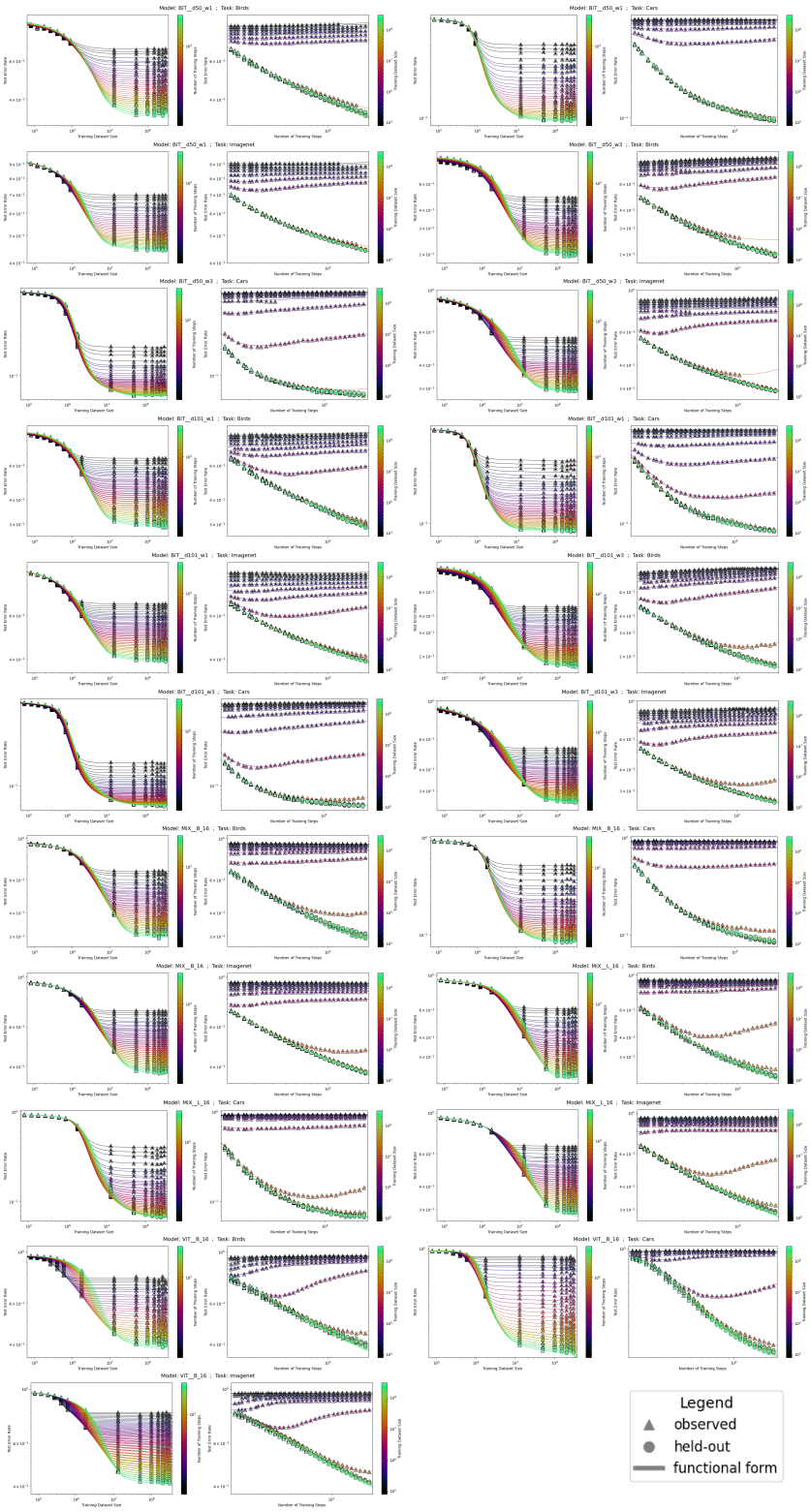

Figure 21: Extrapolation Results of A3 functional form on bivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

19.5.2 TRIVARIATE

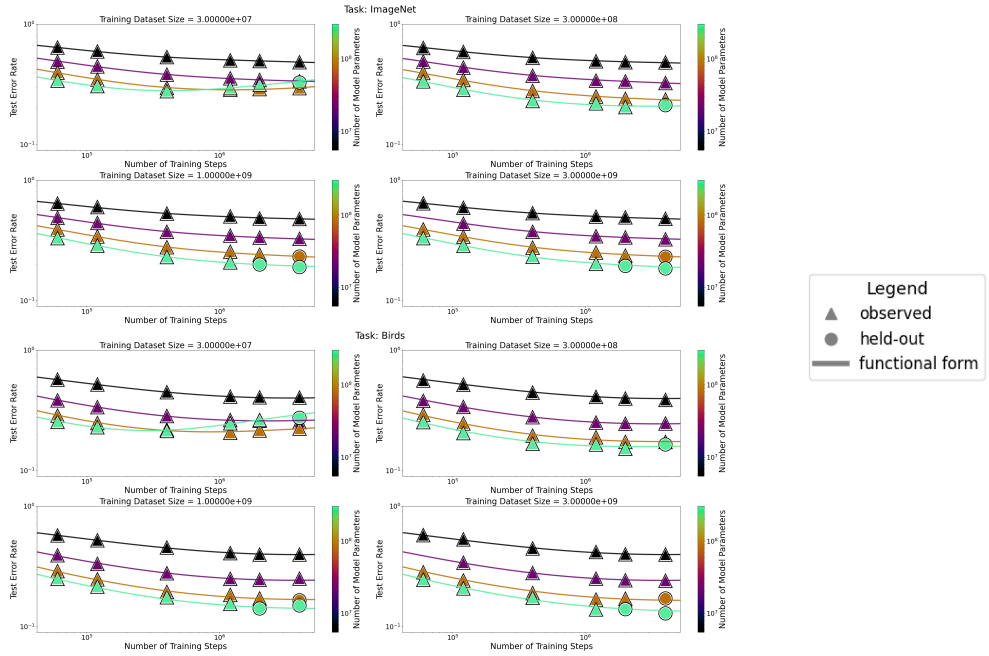

Figure 22: Extrapolation Results of UNSL functional form on trivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

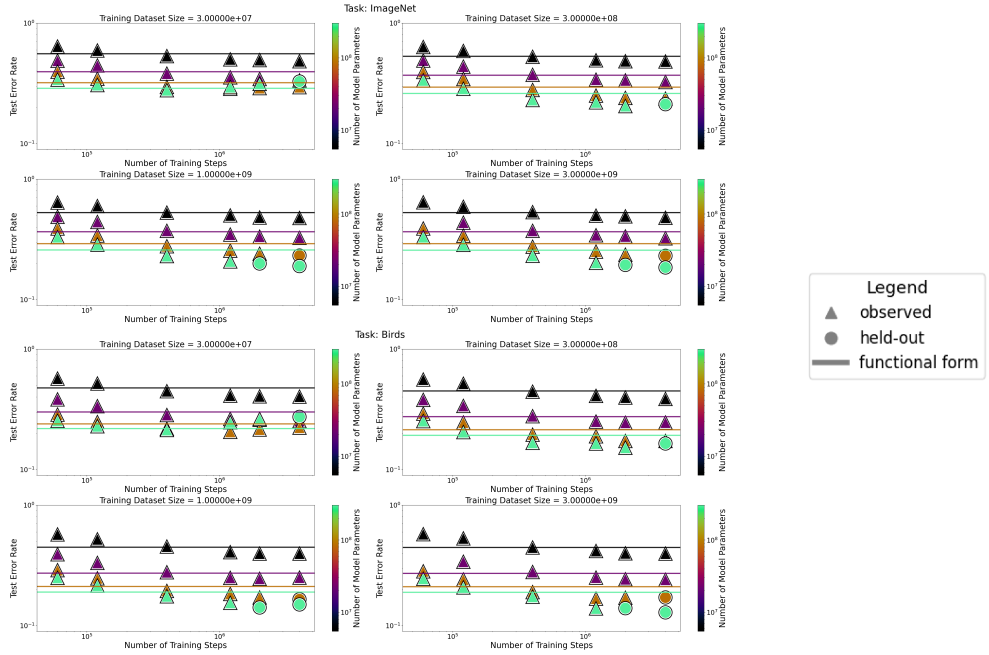

Figure 23: Extrapolation Results of "DC" functional form of Muennighoff et al. (2023) on trivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

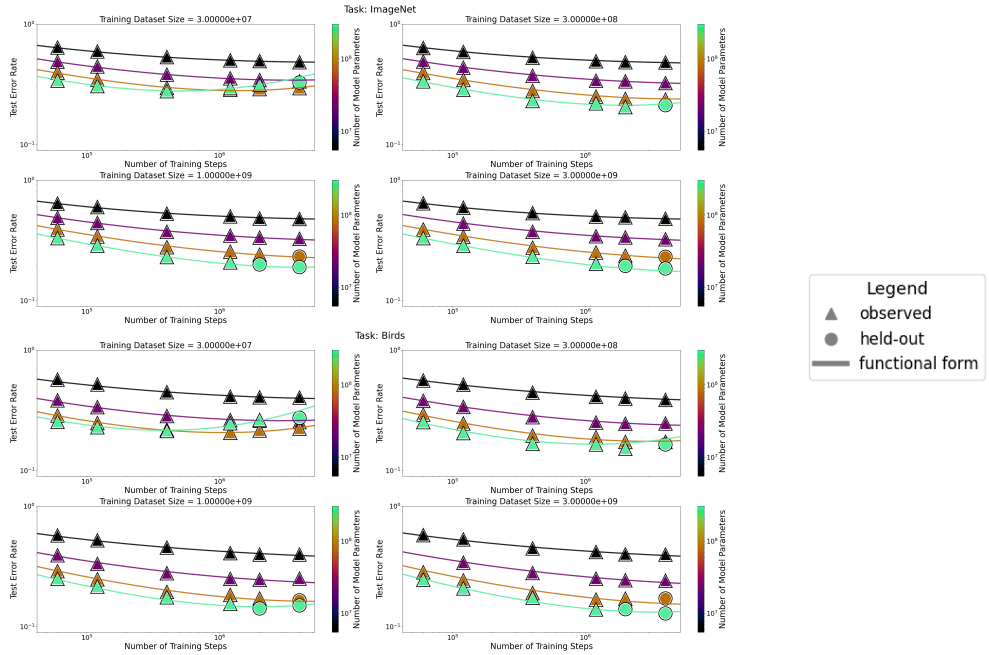

Figure 24: Extrapolation Results of A1 functional form on trivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

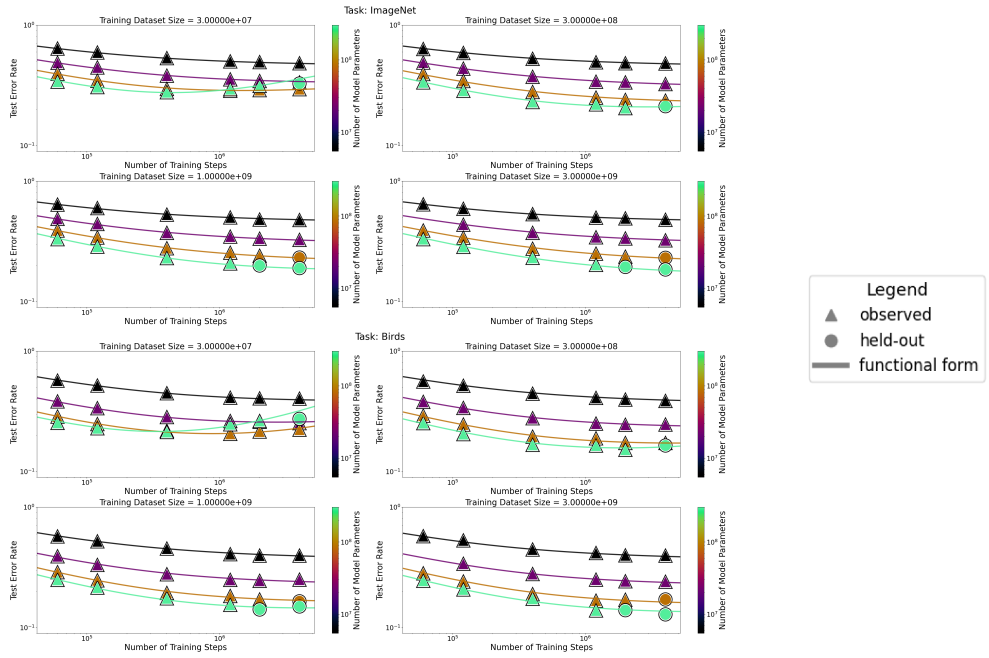

Figure 25: Extrapolation Results of A2 functional form on trivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

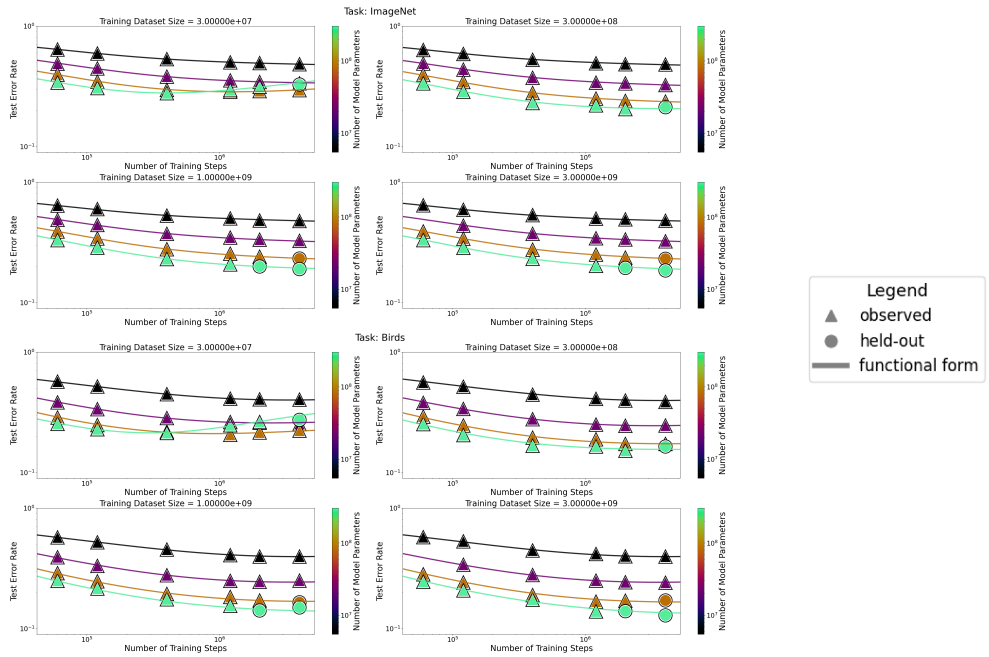

Figure 26: Extrapolation Results of A3 functional form on trivariate scaling behavior of downstream vision performance. See Section 4.1 for more details.

19.6 Plots of Language Extrapolation Results

19.6.1 Trivariate

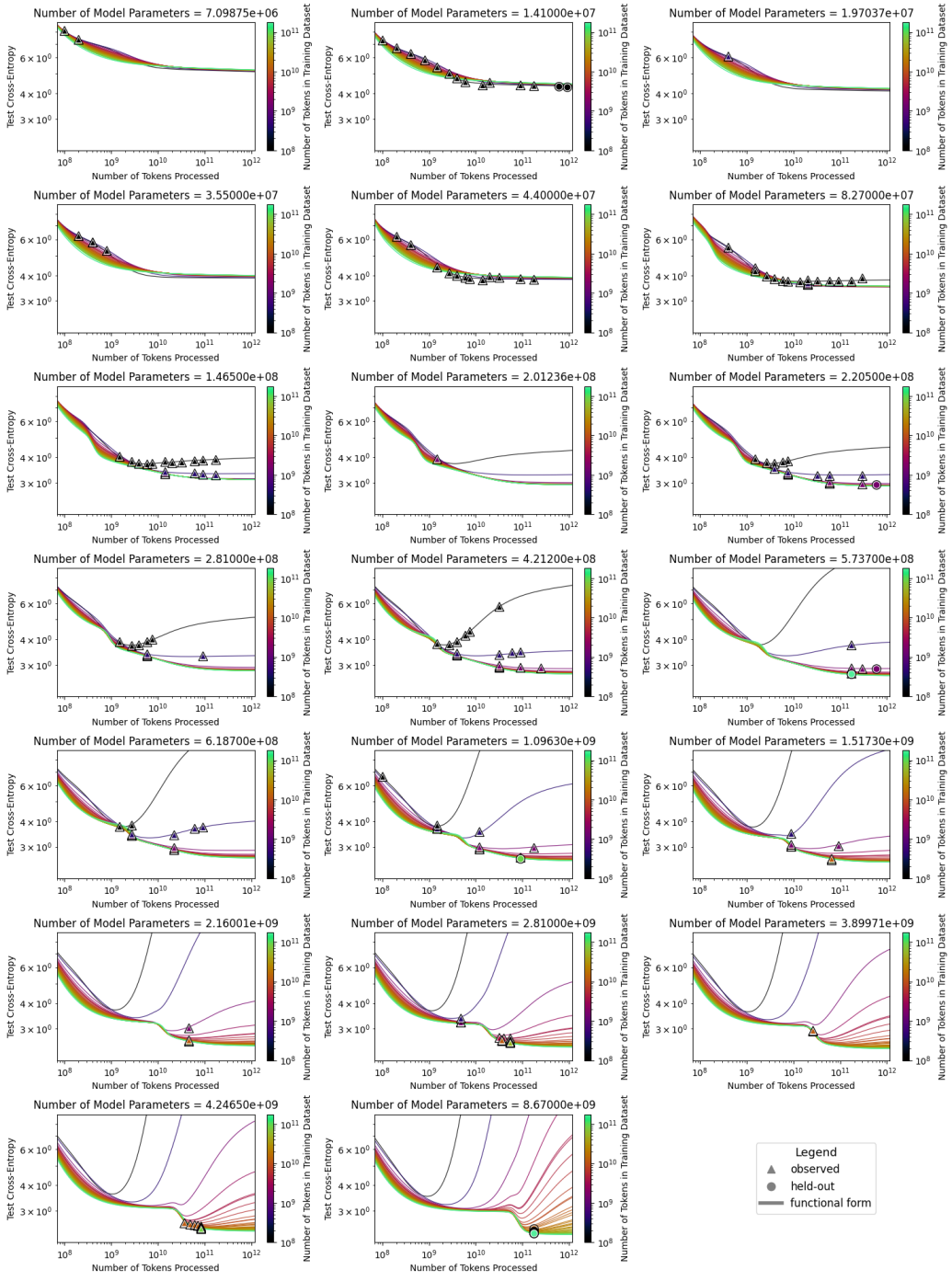

Figure 27: Extrapolation Results of UNSL on trivariate scaling behavior of language performance. All 20 plots are slices of single functional form fit to a single trivariate scaling behavior. The title of each plot represents the number of model parameters, the x-axis of each plot represents the number of training steps times the batch size, and the color bar of each plot represents the training dataset size. See Section 4.2 for more details.

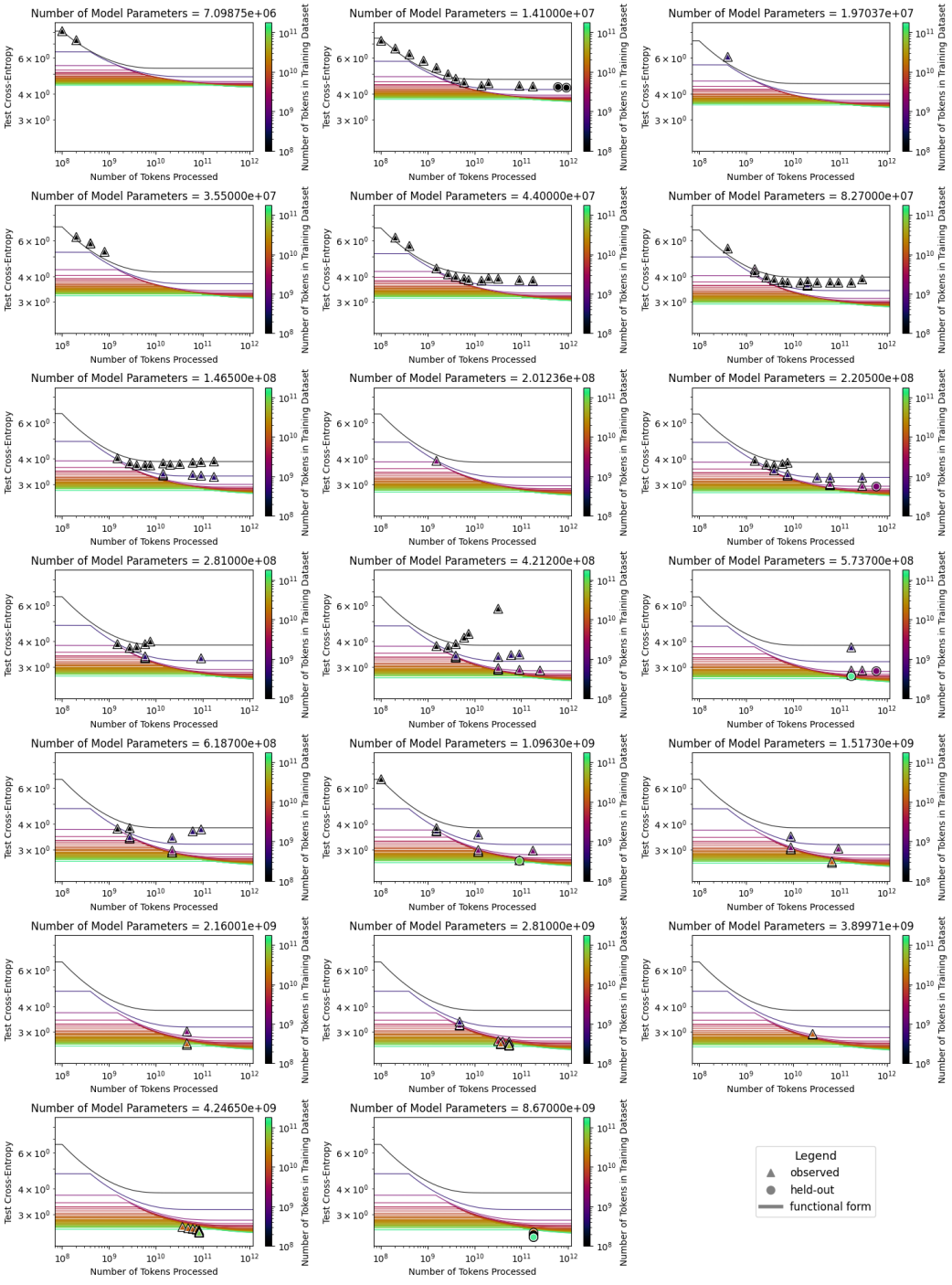

Figure 28: Extrapolation Results of "DC" functional form of Muennighoff et al. (2023) on trivariate scaling behavior of language performance. All 20 plots are slices of single functional form fit to a single trivariate scaling behavior. The title of each plot represents the number of model parameters, the x-axis of each plot represents the number of training steps times the batch size, and the color bar of each plot represents the training dataset size. See Section 4.2 for more details.

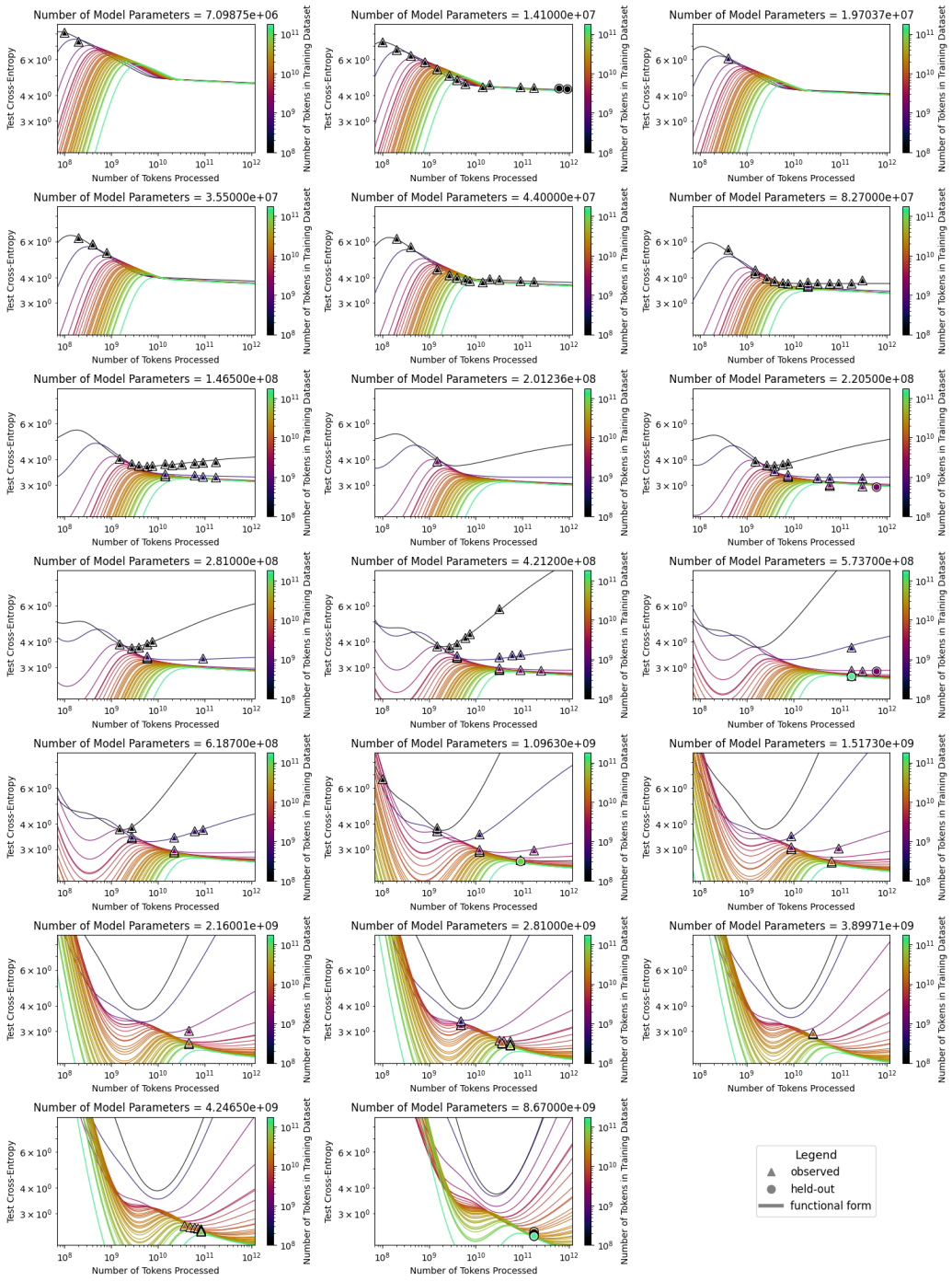

Figure 29: Extrapolation Results of A1 functional form on trivariate scaling behavior of language performance. All 20 plots are slices of single functional form fit to a single trivariate scaling behavior. The title of each plot represents the number of model parameters, the x-axis of each plot represents the number of training steps times the batch size, and the color bar of each plot represents the training dataset size. See Section 4.2 for more details.

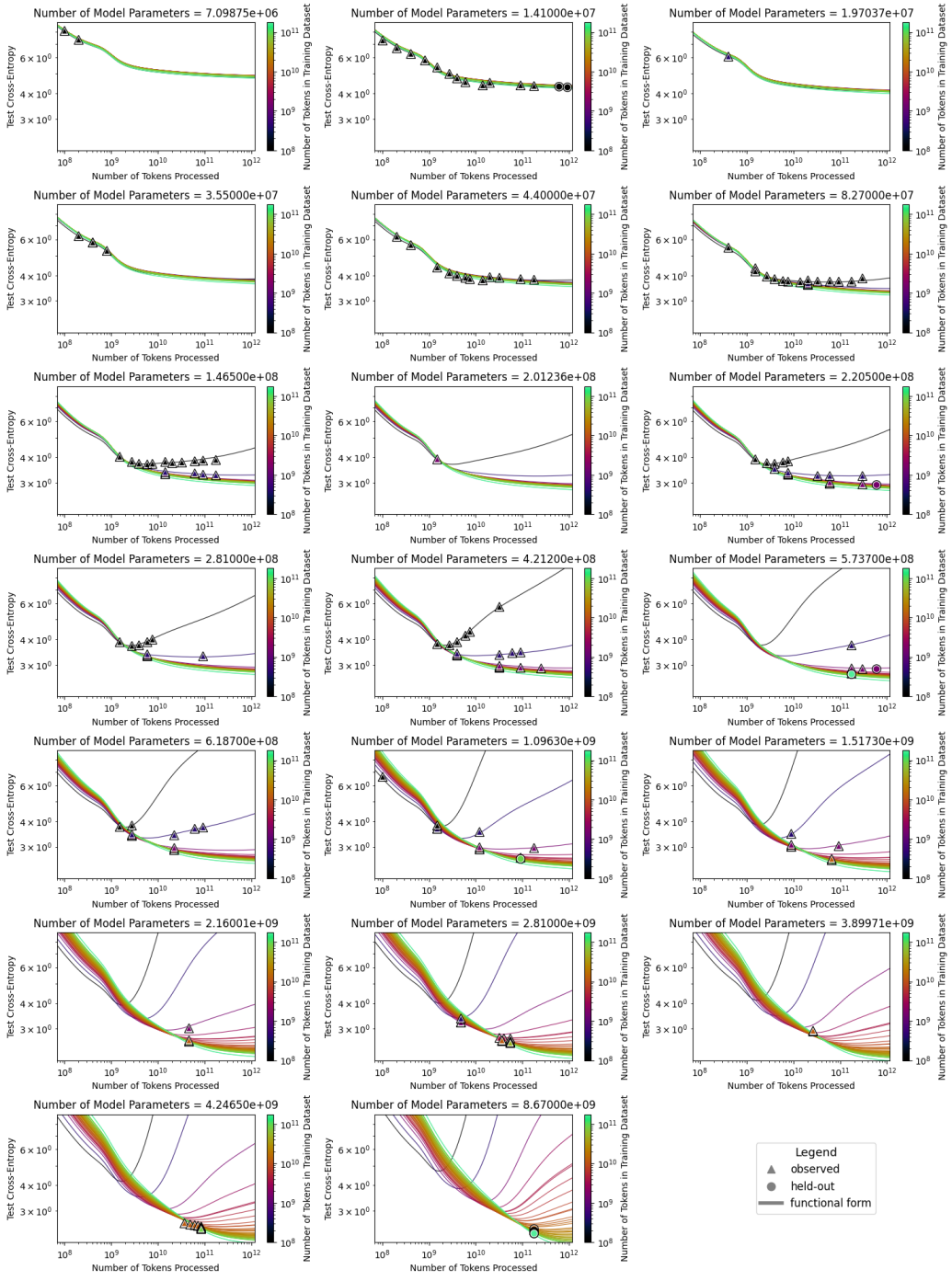

Figure 30: Extrapolation Results of A2 functional form on trivariate scaling behavior of language performance. All 20 plots are slices of single functional form fit to a single trivariate scaling behavior. The title of each plot represents the number of model parameters, the x-axis of each plot represents the number of training steps times the batch size, and the color bar of each plot represents the training dataset size. See Section 4.2 for more details.

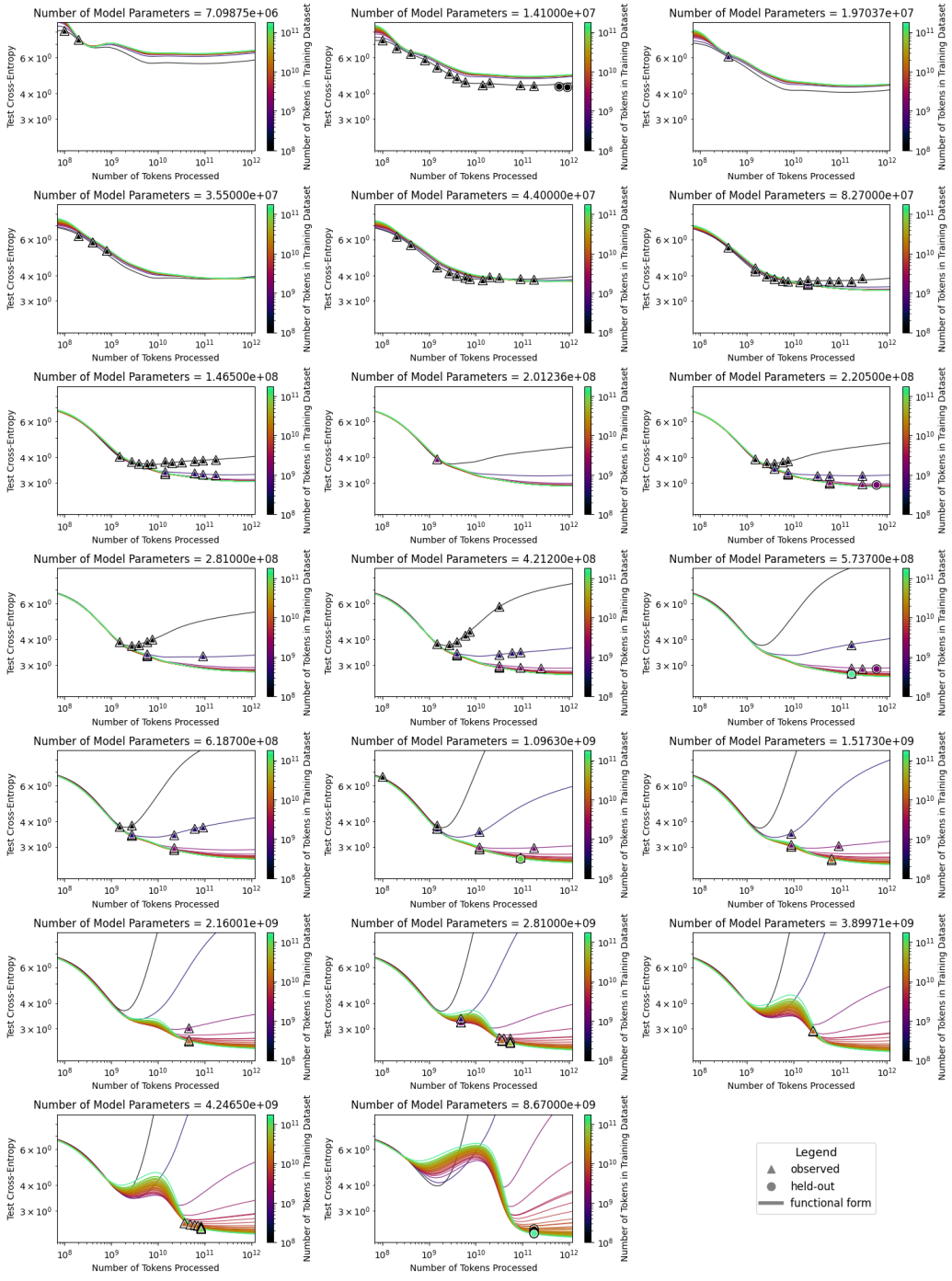

Figure 31: Extrapolation Results of A3 functional form on trivariate scaling behavior of language performance. All 20 plots are slices of single functional form fit to a single trivariate scaling behavior. The title of each plot represents the number of model parameters, the x-axis of each plot represents the number of training steps times the batch size, and the color bar of each plot represents the training dataset size. See Section 4.2 for more details.

19.6.2 BIVARIATE

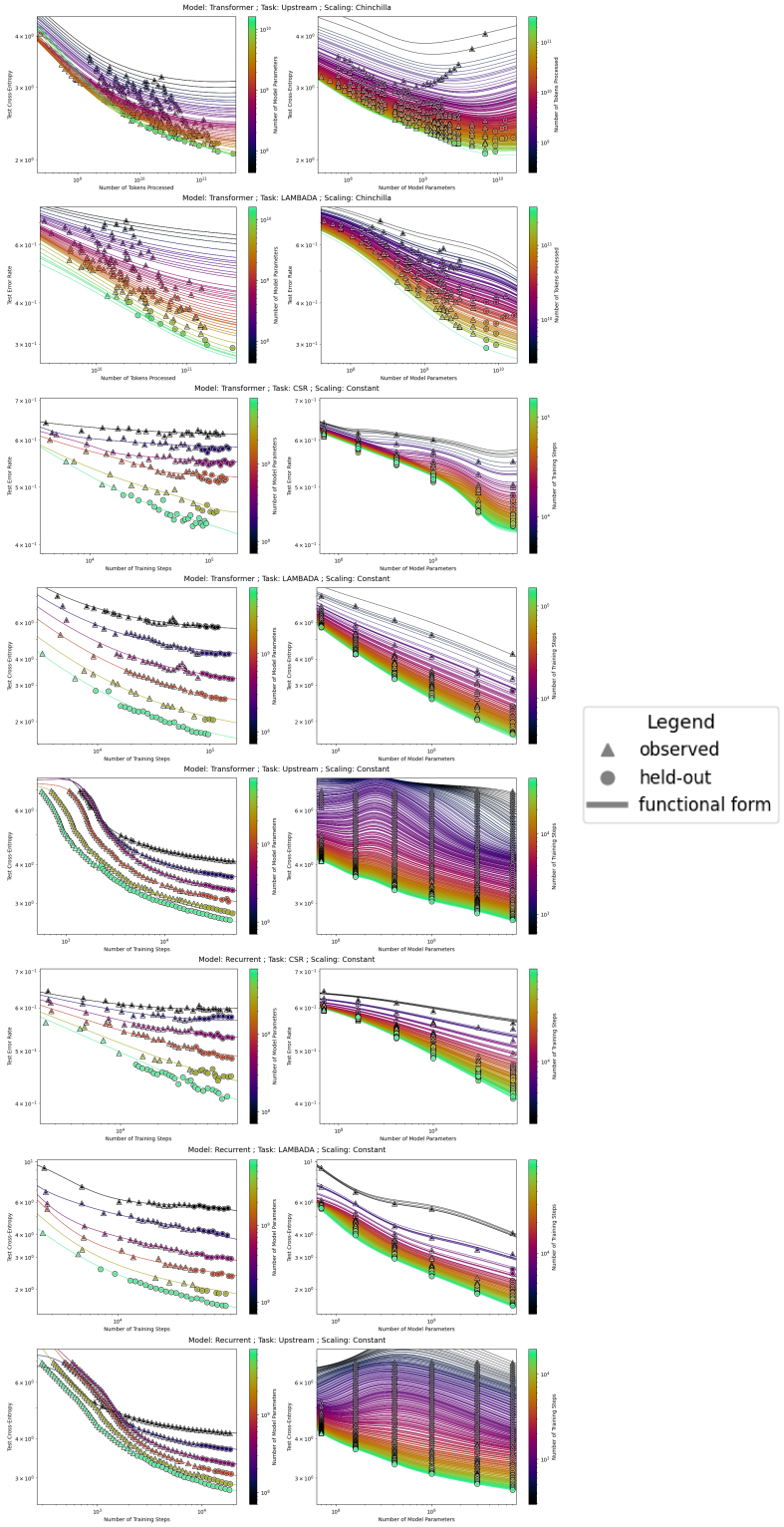

Figure 32: Extrapolation Results of UNSL on bivariate scaling behavior of downstream (and upstream) language performance. See Section 4.2 for more details.

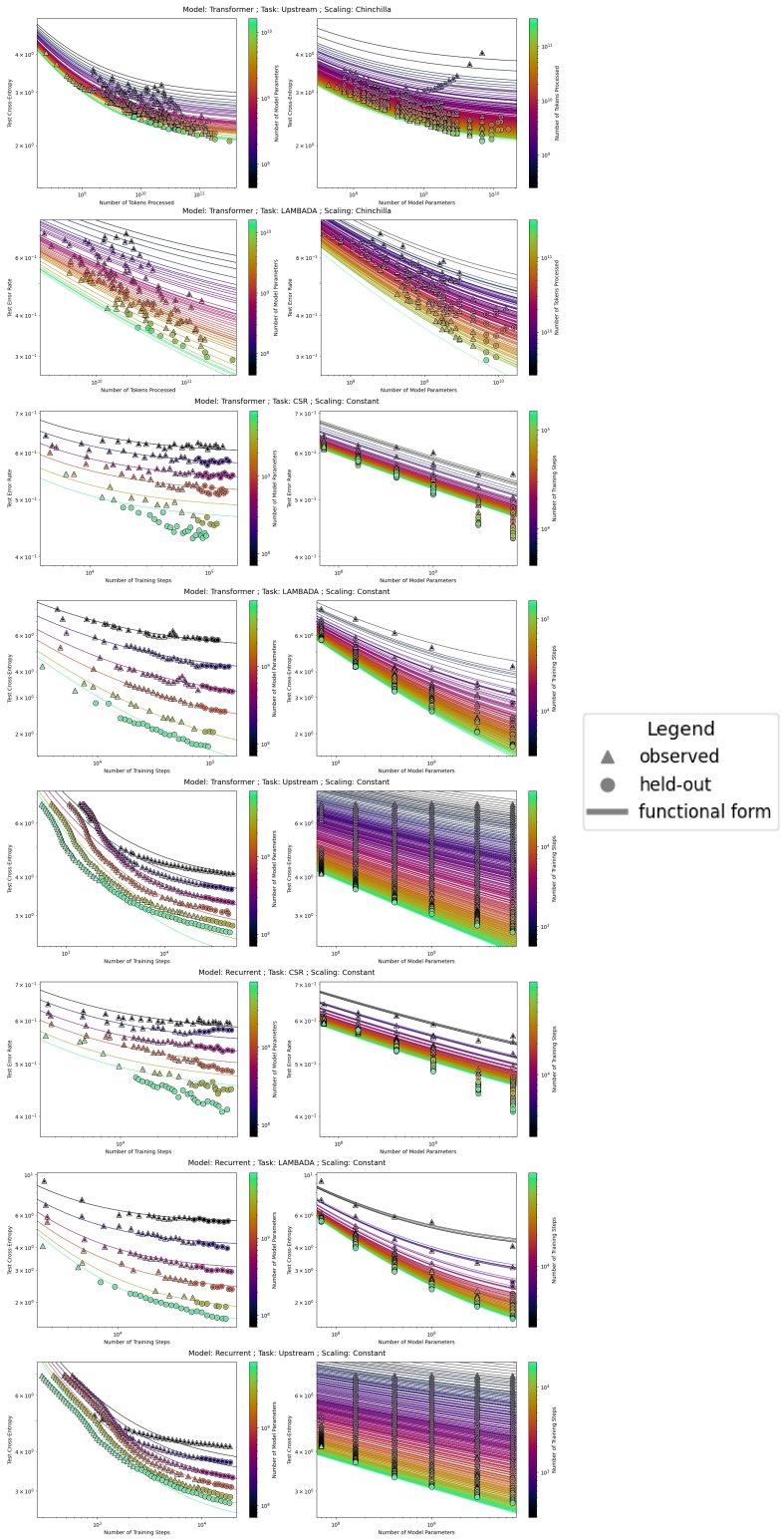

Figure 33: Extrapolation Results of "CF" functional form of Hoffmann et al. (2022) on bivariate scaling behavior of downstream (and upstream) language performance. See Section 4.2 for more details.

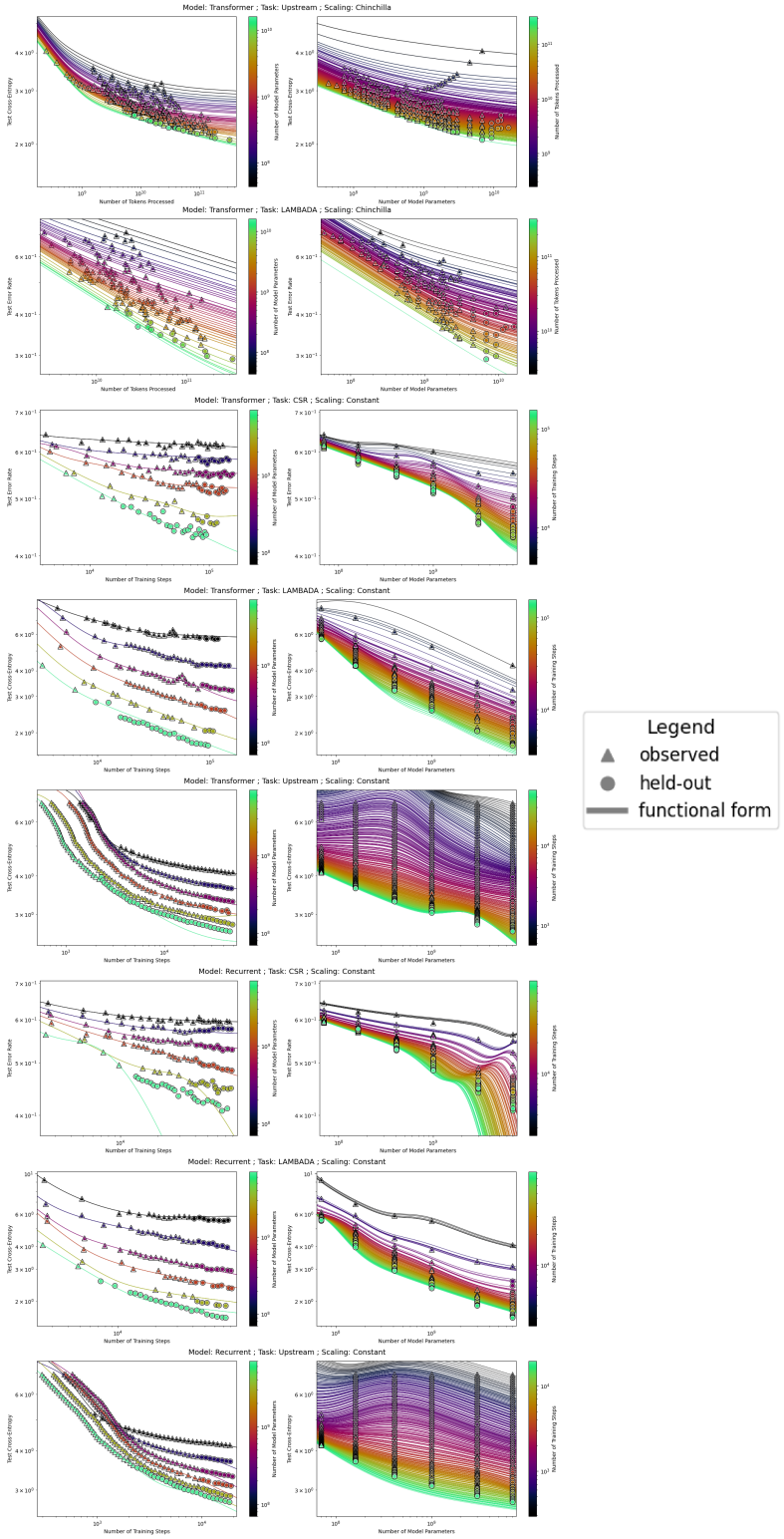

Figure 34: Extrapolation Results of A1 functional form on bivariate scaling behavior of downstream (and upstream) language performance. See Section 4.2 for more details.

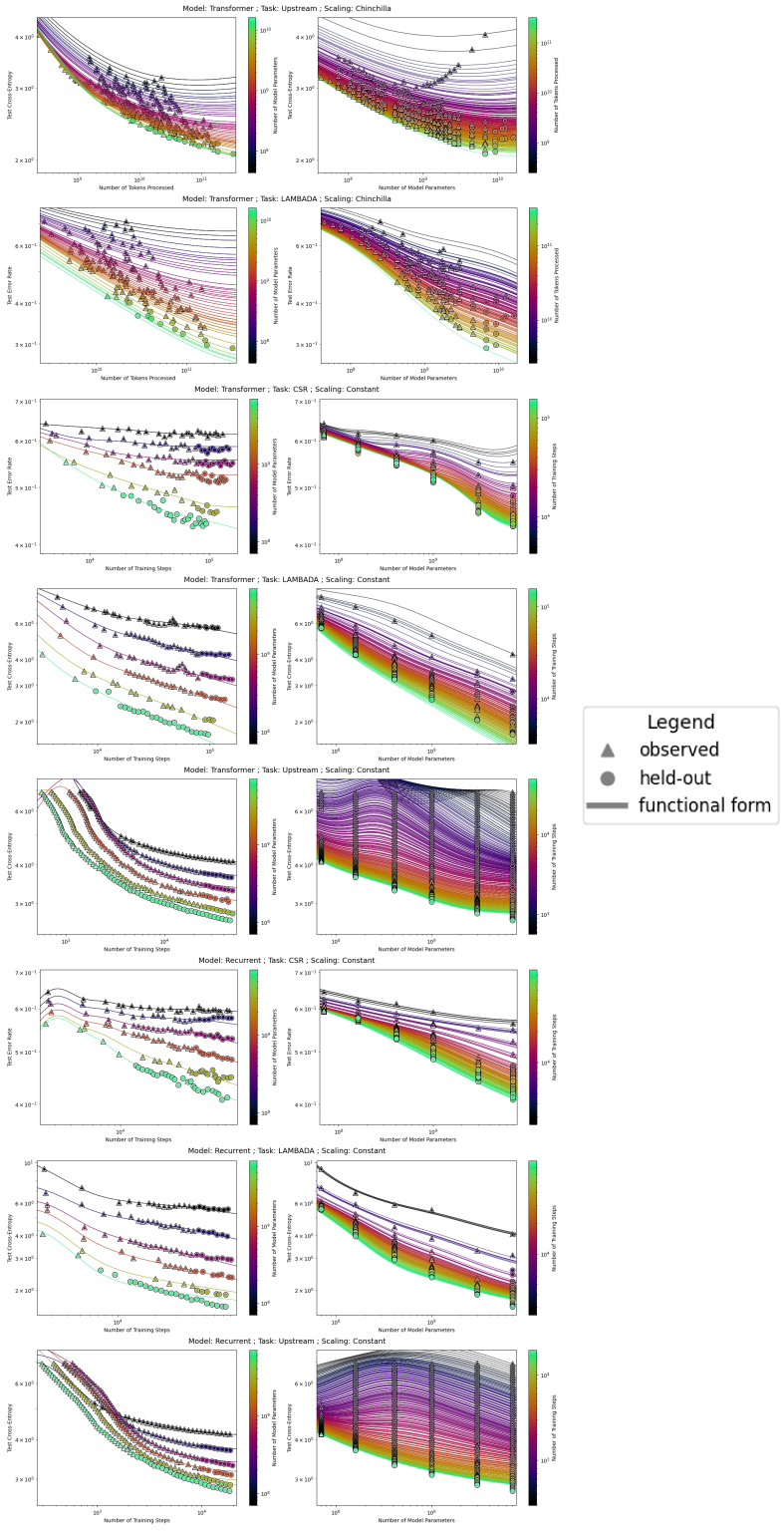

Figure 35: Extrapolation Results of A2 functional form on bivariate scaling behavior of downstream (and upstream) language performance. See Section 4.2 for more details.

20 Additional Related Work

There has been additional work on scaling law settings, interventions, and extensions besides those emphasized in this paper. Scaling laws have been applied in the context of autoregressive generative modeling along various axes of scale, e.g. compute/model/dataset (Henighan et al., 2020), as well as in the regime of transfer learning/fine-tuning, i.e. the joint effect of the scale of the pre-training task and the quantity of data available for downstream fine-tuning (Hernandez et al., 2021). In addition, the influence of dataset curation/selection on scaling laws is receiving increasing attention, e.g. via pruning or data selection with the goal of optimizing scaling exponents (Sorscher et al., 2022; Ayed & Hayou, 2023). Furthermore, the problem of compute-optimal scaling, i.e. the design of scaling laws beyond the number of parameters, e.g. with respect to depth, width, and shape, has been addressed (Alabdulmohsin et al., 2023). In parallel with the problem of predictive scaling-law fitting, a different line of work has focused on the design of explicit architectural scaling heuristics, e.g. compound scaling (Tan & Le, 2019). Other learning contexts and classes of models where scaling analyses have also been pursued include reinforcement learning (Neumann & Gros, 2023) and diffusion (-transformer) generative models (Liang et al., 2024; Li et al., 2024).

A related body of work has focused on the study of non-monotonic generalization properties and multi-regime phenomena with respect to scaling. Much of this work has focused on double descent phenomena that depend on model capacity, sample size, and/or training time, as seen in Belkin et al. (2019); Nakkiran et al. (2019); Hastie et al. (2022); Adlam & Pennington (2020). Other related statistical physics viewpoints on non-monotonicity and sharp transitions in terms of phase transitions/jamming are presented in Spigler et al. (2019). Another noteworthy example of non-monotonic behavior depending on optimization and regularization is "Grokking," or sudden late generalization far after the start of memorization, as seen in Power et al. (2022); Liu et al. (2022; 2023).

The application of scaling laws in forecasting is closely related to the problem of learning curve extrapolation, multi-fidelity hyperparameter optimization, and the scaling of hyperparameters. Because scaling laws are used operationally for forecasting on the basis of partial training runs, this problem is related to learning curve extrapolation and early termination of hyperparameter search, as discussed in (Domhan et al., 2015; Swersky et al., 2014). It is also related to multi-fidelity/bandit and BO-based methods for HPO, as discussed in (Li et al., 2018; Falkner et al., 2018; Snoek et al., 2012; Hutter et al., 2011). In the area of large batch training, there are studies on scaling rules for schedules and optimizers, which analyze how optimal hyperparameters (such as the learning rate, batch size, and momentum) change with scale and batch size, as discussed in (Goyal et al., 2017; Smith et al., 2017; You et al., 2017; McCandlish et al., 2018). Another line of work on theory-driven parameterization of schedules aims at enabling zero-shot transfer of hyperparameters across network widths (such as in μP) as discussed in (Yang et al., 2022).

Finally, the study of "emergent abilities" points out that the observation of clear transitions with respect to scale may actually indicate either real regime changes or the effects of nonlinear/discontinuous metrics and estimation at finite samples (Wei et al., 2022; Schaeffer et al., 2023).