Initial Findings on a Cross Term for Scaling Laws

Oam Patel, Arjun Puri, Kevin Luo

May 4, 2023

Abstract

We study the toy problem of *multitask sparse parity*, introduced in Michaud et al. [2023]. The original authors showed both theoretically and empirically that in the regime of near-infinite data, the loss demonstrated a power law with respect to the parameters, and likewise, in the regime of near-infinite parameters, the loss followed a power law with respect to the amount of data seen. Existing scaling laws assume a decomposition into three terms, one being the irreducible noise, another being a power of the parameter count, and the last being a power of the dataset size – crucially, none consider any interaction between these two factors. We seek to establish in the multitask sparse parity setting whether, in the regime where both model size and dataset size are assumed finite, there exists such a cross-term.

1 Introduction and Related Work

Scaling laws have recently emerged as an exciting area of inquiry for science of deep learning. The study of scaling extends to well before the age of LLMs; however, exciting work recently has shown that a power law holds for scaling both data and model size Kaplan et al. [2020]. Several possible models of power law scaling have been proposed in the past. Borrowing heavily from the literature review of Michaud et al. [2023], we note that a model of neural networks performing piecewise linear approximations on the input data manifold was proposed by Sharma and Kaplan [2020]. This was extended by Bahri et al. [2021] to training data scaling; however, this relied heavily on the power law spectrum of certain kernels. Additionally, Maloney et al. [2022] propose a model of scaling where the power law coefficient arises from the power law spectra of the data feature-feature-covariance matrix and Hutter [2021] propose a toy model where a power law over features in the data distribution imposes a power law over the learned loss landscape. None of these settings consider the cross term between parameter count and training examples.

Scaling laws to date have mostly followed a sum of power laws model. For example, compute-optimal scaling laws for the Chinchilla series of LLMs where an N parameter model sees D training examples have the following functional form Hoffmann et al. [2022].

$$L(N, D) = \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + E \tag{1}$$

Note that there is no interaction between number of parameters and number of training examples. This strikes us as a lossy abstraction, and while it may be a good approximation in large N, large D settings, we expect that this functional form doesn't hold up in other settings. The first half of Michaud et al. [2023] considers the more toy setting of multitask sparse parity, based on the initial single-task sparse parity results from Bahri et al. [2021]. In the setting where task frequency follows a power law distribution, there exists power law scaling with respect to both N and D separately, when the other parameter is assumed effectively infinite. We seek to establish whether in settings where both parameters are finite, there exists an interaction term between N and D in the form of the loss.

2 Problem Setting

We consider the multitask sparse parity setting from Michaud et al. [2023] which extends the sparse parity setting from Bahri et al. [2021] to multiple tasks. The sparse parity prediction problem is as follows: given

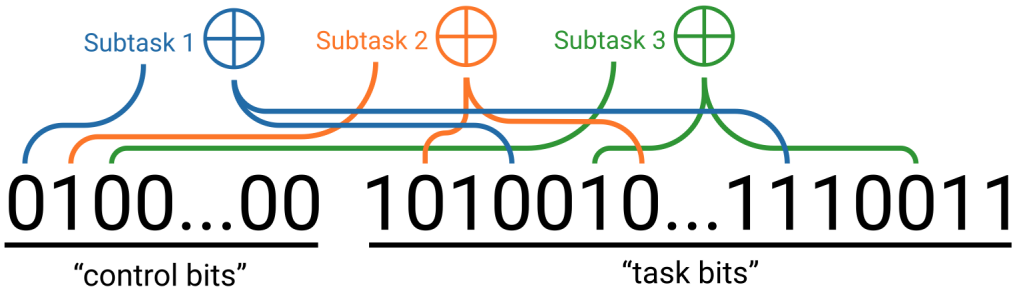


Figure 1: Example of a multitask sparse parity problem. Here the 2nd task is active, and the correct prediction would be 0. Credit: Michaud et al. [2023]

a bit string of length n, compute the XOR of a fixed subset of k bits. In the multitask setting, we add n_{task} control bits that one-hot encode whether a particular task is active or not. For constructing the dataset, we take in a given n, k, n_{task}, construct n_{task} subsets of n each of size k. We use this same dataset for all of training (in an online fashion). Given $n > 100$, seeing the same training example twice becomes extremely unlikely due to the combinatorial explosion of subsets. In any given training sample, only one task is active at a time.

As in Michaud et al. [2023], we use a uniform distribution over the task bits and a Zipfian distribution over the control bits. The probability that a given sample has control bit i active is $\frac{1}{Z} i^{-(\alpha+)}$ where Z is a normalizing constant. This imposes a power law distribution over the the subtasks. Multitask sparse parity is a clean setting to consider scaling laws because the n_{task} parameter provides an easy handhold for smaller models (they can at least learn the most frequent task or two) and continues to be difficult for large models. Accordingly, we can construct a dataset on which even large models with lots of data won't easily achieve near-zero loss.

2.1 Theoretical Framework in Michaud et al.

We give a brief overview of the theoretical arguments in Michaud et al. [2023] that show that a power-law distribution on the tasks (quanta) yields a power-law for the loss with respect to the parameter count and dataset size. With respect to dataset size, we focus only on the online training regime, and defer the fixed-data multi-epoch setting to future work. Let p_k denote the frequency of the kth task. Since they are drawn according to a power law, we have $p_k = \frac{1}{\zeta(\alpha+1)} k^{-(\alpha+1)} \propto k^{-(\alpha+1)}$.

Assume that each a sample from any given task contributes a loss of a. Therefore the initial expected loss is $L_n = \sum_{k=1}^{n} a p_k$. Assume that learning a given task drops the loss on samples from that task from a to 0. Assume that the model is able to learn some n number of the tasks. Then the expected loss becomes

$$\begin{aligned} L &= \sum_{k=1}^{n} 0 \cdot p_k + \sum_{k=n+1}^{\infty} a \cdot p_k \\ &\approx \frac{1}{\zeta(\alpha+1)} \int_{n}^{\infty} k^{-(\alpha+1)} \, \mathrm{d}k \\ &= \frac{a}{\alpha \zeta(\alpha+1)} n^{-\alpha} \end{aligned}$$

Hence the loss is indeed a power law with respect to the number of tasks learned. The authors then propose that if the parameter count is N, each task requires possibly C of those parameters to learn, and hence the total number of tasks learned is N/C once the model is trained to capacity, yielding a power law with respect to the parameters. With respect to data, they believe that, given sufficient width, the model must see a given task T times to learn it, and hence having seen S total samples, only $Sk^{-(\alpha+1)}$ of them are of task k. We therefore expect the last task n that is learned to satisfy $Sn^{-(\alpha+1)} \approx T$, so $n \approx (S/T)^{1/(\alpha+1)}$. Substituting this then also yields a scaling law for S.

These proposed scaling laws agree with the form of Equation 1. The predictions given here are in the regime where one of the quantities is very large, and observe the scaling with respect to the other (the resolution limited regime, Bahri et al. [2021]). When we think of both quantities being finite, it does not seem intuitive that the parameter C would depend on the amount of data seen; however one could imagine that the number of samples needed to learn a task, T, could depend on the model size. This would then yield a potential cross term in the total scaling law.

3 Methodology

We train a series of 1 hidden layer RELU MLPs on an infinite dataset that is the same for every training run, varying the parameter size from a hidden layer of size 10 up to 500. For the strength of the power law over the subtasks, we try three values of α, namely $0.2, 0.4$, and 0.8. For each model, we save out the loss curve over all iterations, and save out the per-subtask loss every 5 gradient update steps. The training parameters are identical to those given in Michaud et al. [2023]. We use a batch size of $20,000$, a learning rate of $1e-3$, and the Adam optimizer. We observed that when training the largest models, the loss flattened by $100,000$ iterations, and thus trained every model for this many steps rather than $200,000$.

Training was performed on a series of AWS instances equipped with GPUs, as well as local hardware. The full set of runs is located in the pickled tar file in our repository. This contains only per-batch losses, and does not include losses specific to each subtask. These were too large for GitHub.

4 Observed Results

In this section attempt to verify empirically examine assumptions made in Michaud et al. [2023] in their theoretical derivations of scaling laws, and attempt to see whether any of them would lead to cross terms.

4.1 A Sanity Check

A view of our terminal losses reproduces Figure 2 of Michaud et al. [2023], indicating that our training settings at least agree with theirs. We now move to investigating the existence of a cross term.

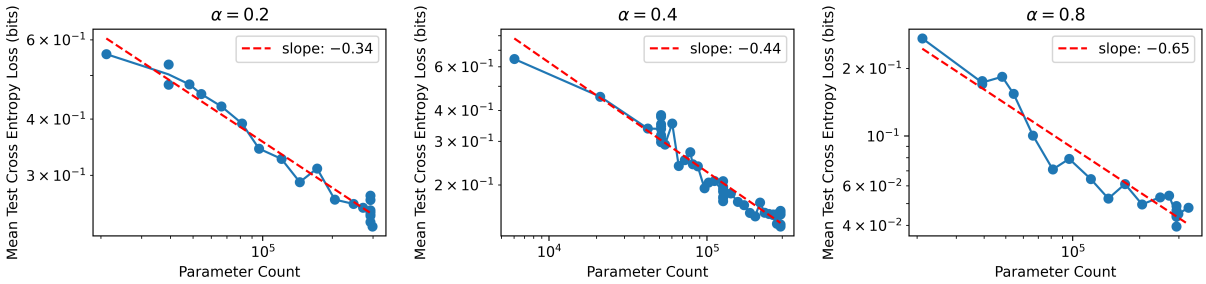


Figure 2: Losses after 1e5 gradient update steps.

4.2 Verifying the Theoretical Framework for the Loss

The framework used in Michaud et al. [2023], discussed in Section 2.1, suggests that there will be no cross term if the time taken to learn each subtask, T, is independent of the model size. For each subtask, they observed that there exists sharp phase transition in which the loss for datapoints depending on that quanta rapidly decreases. We first empirically establish the concentration of this size S, for sufficiently large models.

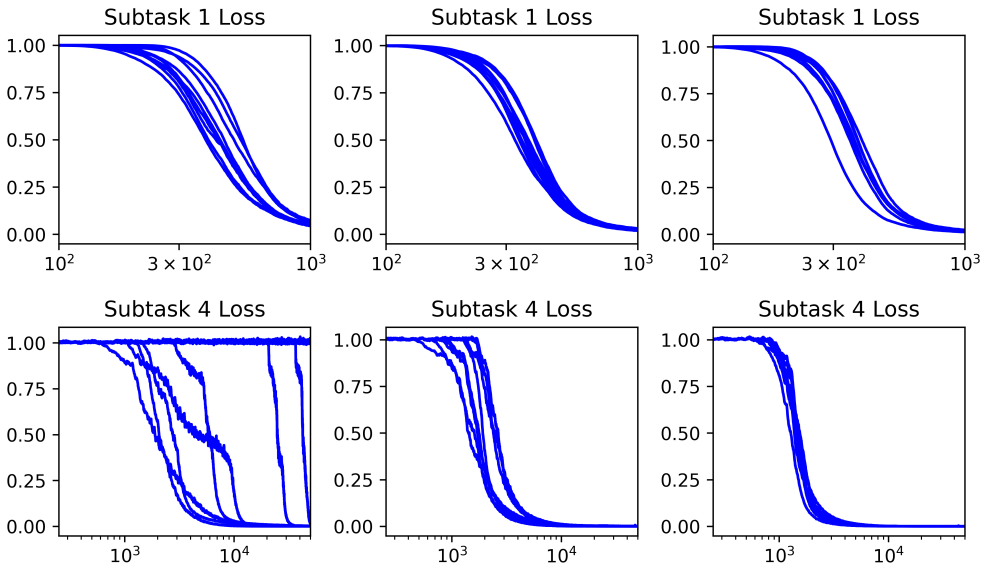


Figure 3: The point of phase transition for subtasks 1 and 4, for models of size 85, 210, and 485, respectively. All are under $\alpha = 0.4$.

Figure shows the loss on subtask 1 (the most common subtask) for models of three sizes, run under the same data settings. It is clear that for the first subtask, the phase transition point for all three sizes of models occurs at a consistent point in time, one that is remains constant across all model initializations, and is also somewhat consistent across model sizes (investigated more in 4.2.4.)

On the other hand, we see that for subtask 4, this begins to longer be the case. Instead, we see large variance in the point of phase transition in the smallest model, while the other two remain relatively consistent. As we see now, this is because the smallest model is beginning to reach capacity.

4.2.1 Model Capacity

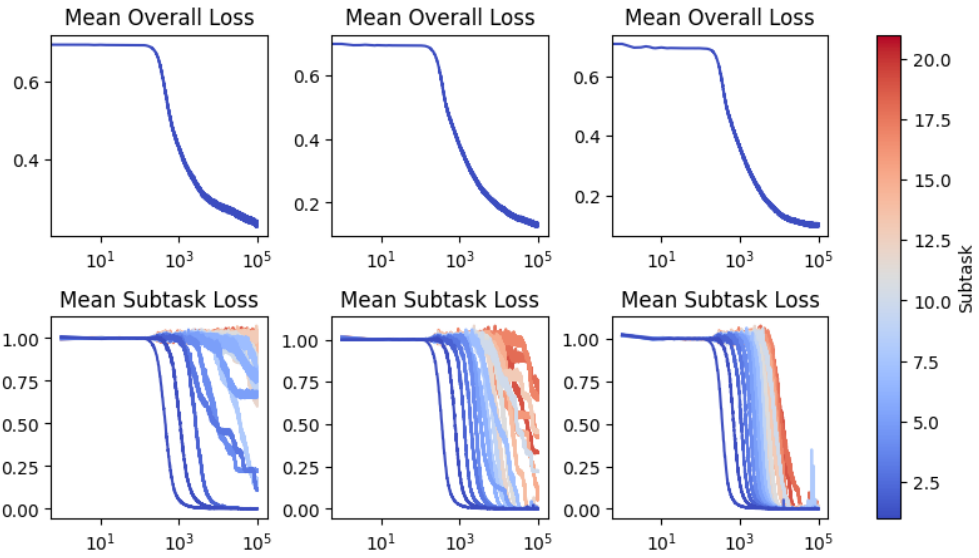


Figure 4: Per subtask losses for each model size (85, 210, 485). All under $\alpha = 0.4$.

As described in 2.1, Michaud et al. [2023] establish a scaling law with respect to parameters in the data limit by supposing that each learning each subtask takes a portion of the model's total capacity, and that this capacity scales linearly in the number of parameters.

Figure 4 illustrates what occurs for the first 20 subtasks as these models of different sizes are trained. The stair-like subtask loss curves seen in the model of size 85 indicate that some of the models managed to learn the subtask sooner than others, which was shown in Figure 4.2, while the fact that this does not emerge for the model of size 485 indicates that all of the models still learn the task at the same time. These data validate the idea that models have some capacity that depends on their parameter count, but also shows that the number of tasks able to be learned is still sensitive to the initialization and likely the training process, since it remains possible for some of the smaller models to learn more subtasks, but not others. This effect likely explains why the plot of parameter count against test loss in Figure 2 of Michaud et al. [2023] is much noisier than the others. Overall, it appears that having more scale in fact makes training easier.

4.2.2 The number of parameters per task

We look at the estimated number of parameters needed to learn each task, denoted C in Michaud et al. [2023], to determine whether this is indeed consistent across models.

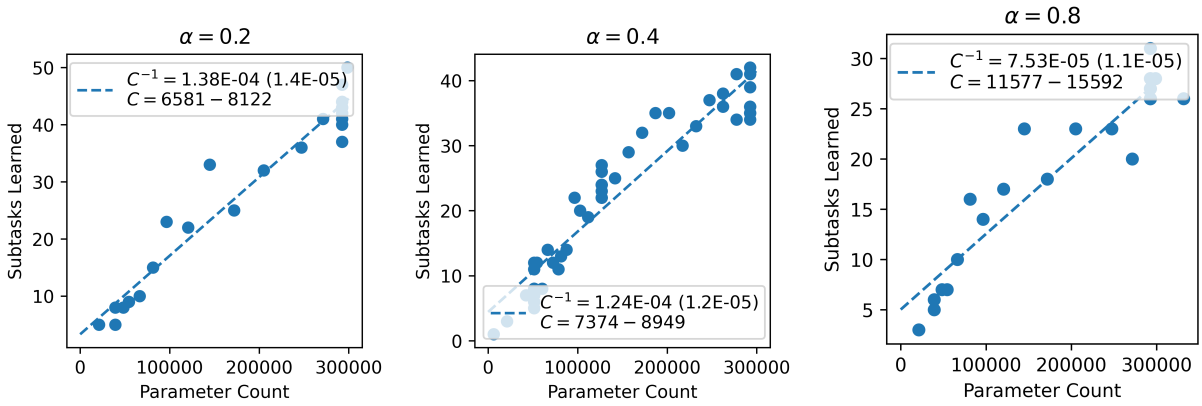


Figure 5: Estimated number of parameters needed to store a given task. Standard error in parenthesis, range given is inverted 95% confidence interval under OLS assumptions.

Figure 5 illustrates that the trends look relatively linear. However, for the two larger values of α, there appears to be a drop-off at higher parameter counts. While our set of models is relatively small, we believe that there may be reasons for this. The first is that the larger models may not have been trained long enough. We examined the loss curves and they seemed to have flattened, but there may have been more capacity to learn more tasks. The second is that, for higher values of α, the higher subtasks become less common more quickly. For example, for $n = 50$, $\alpha = 0.8$, we expect only around 9 (with a standard deviation of 3) examples of that task to appear in every batch of $20,000$, as opposed to 43 (with standard deviation 6). Hence, it is possible that the gradient information is washed out over the batch, and thus it becomes more difficult to learn these rarer tasks.

In any case, the value of C here is not very consistent, but we are not confident enough in our methods to assert that this refutes the existence of such a constant. The values of C^{-1}, the slope, in 0.4 and 0.8 look as though they could easily have been overestimated, thus giving more consistent values of C.

4.2.3 The number of examples needed per task

We now investigate whether the number examples needed to be seen, denoted T in Michaud et al. [2023], for the model to learn a specific task, remains consistent across different model sizes, and different α.

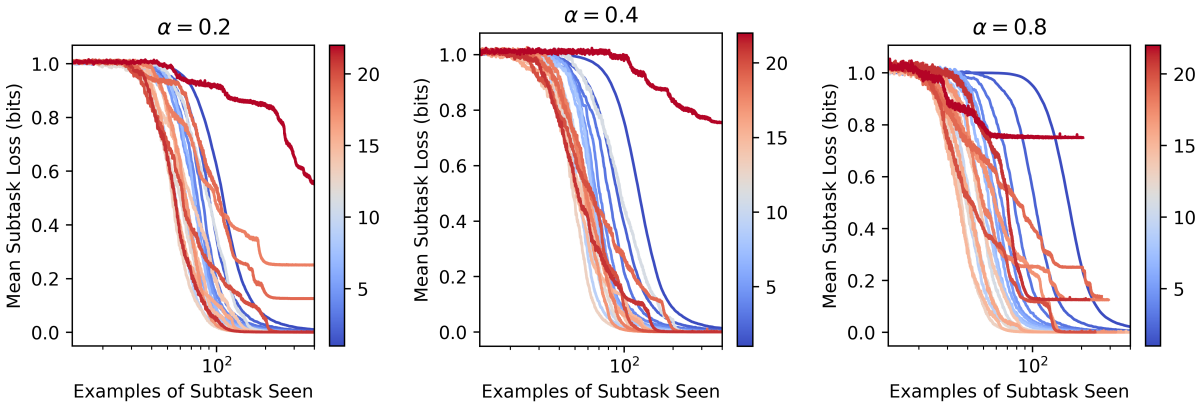


Figure 6: Subtask loss with respect to estimated number of examples of specific subtask seen. Model size is 485.

Figure 6 plots the subtask-level loss for the first 22 subtasks. The x-axis is not the exact number of samples of the subtask seen, but instead an estimated number based on the frequencies of each subtask. We expect this to be reasonably accurate given the large batch size of $20,000$ and large step count.

For the $\alpha = 0.4$ example, we see that the 22nd task is the first to fail to be learned by all models, likely because it began to reach its expected capacity. What is interesting is that this does not appear to be so consistent across different values of α. While the crossover point appears to be in a similar range, quite a few of the tasks entirely fail to be learned for $\alpha = 0.2$ and $\alpha = 0.8$. We remark that the predicted scalings given in Michaud et al. [2023] held best for $\alpha = 0.4$, and are curious to know whether this subtask-level behavior coinciding with what is assumed is the reason this particular regime matches predictions so well.

4.2.4 An interesting trend: larger models learn faster

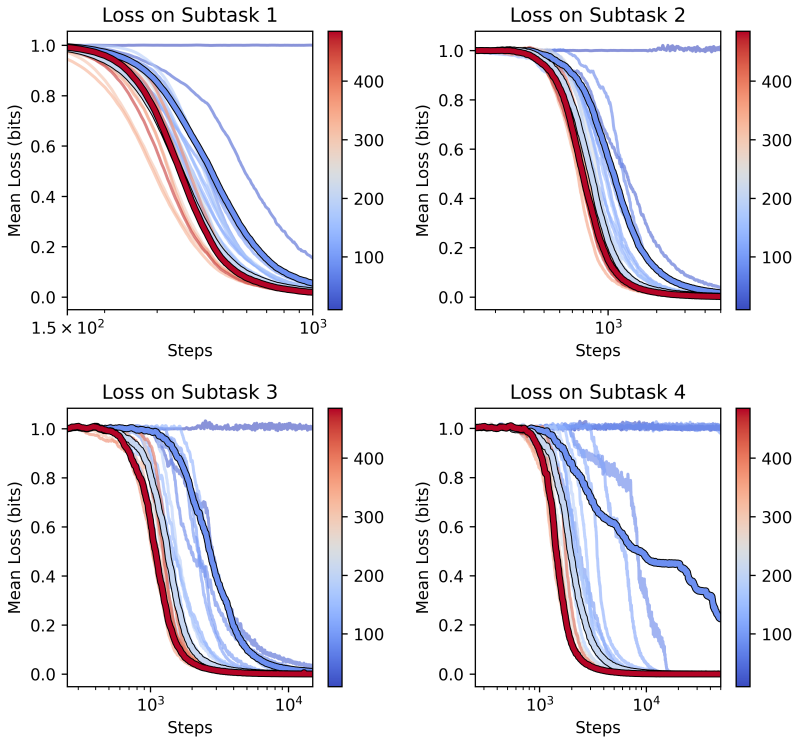
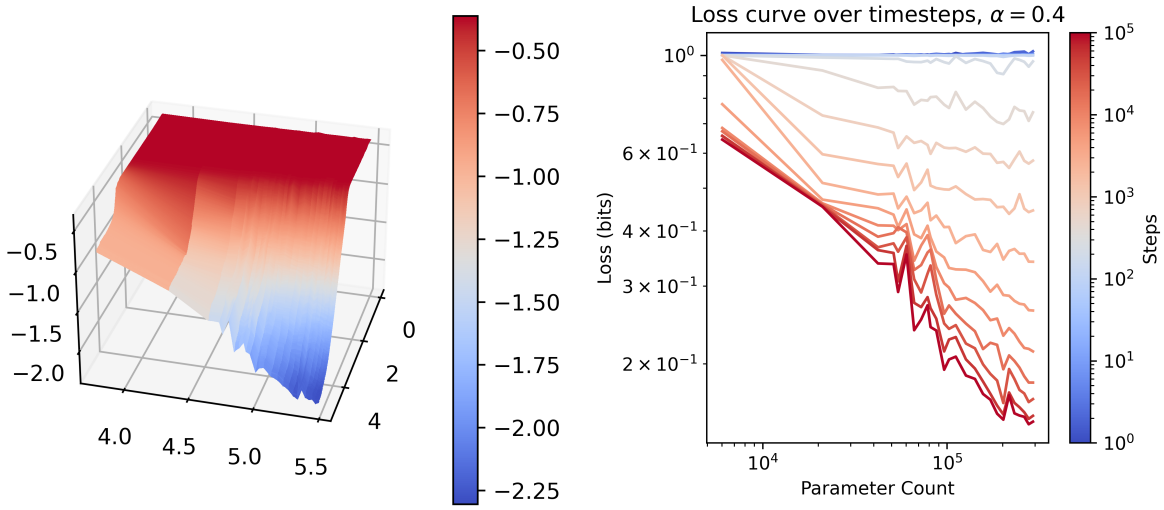


Figure 7: Per-subtask losses. The bolded lines highlight model sizes of 85 (dark blue), 210 (light blue), and 485 (dark red), where there were at least 8 models of the same configuration trained, and hence these lines have less variance.

While the time of the phase transitions remains relatively consistent across models, an interesting phenomenon we observed was that larger models did indeed learn faster. As shown in Figure 7, as the frequency of the subtasks diminishes, the advantages of large models begin to increase. Whether this is due to smaller models which begin to reach capacity having less degrees of freedom to fit the data or something else entirely is unknown. However, if this effect becomes pronounced in the aggregate loss, it would then necessitate a cross-term.

4.3 Loss surfaces

We plot the log-loss as a surface over the log-parameter and log-step count axes, for $\alpha = 0.4$.



(a) Loss surface, log scale on all axes.

(b) Loss curve evolving over time.

Figure 8: Surface plot of loss, as well as truncated views.

The loss surface in Figure 8 again illustrates the trends already remarked up on Michaud et al. [2023], with the loss staying constant before abruptly beginning to fall, and finally flattening out as the models begin to reach capacity.

Figure 8b illustrates the loss curves at a set number of gradient update steps for all models. It appears to show that, if the step count is held fixed, the relationship between loss and parameter count appears to behave as $L \propto N^{-k}$, with k evolving over time, as the curves appear linear on the log-scale, with changing slope. This is because if the functional form in 1 holds, and we view a sufficiently small slice of the parameter axis, the same kind of linear-like trends would appear.

5 Hypothesis Testing the Cross Term

5.1 Set-up

We believe that the functional form of the loss is given by

$$L(N, D) = E + \frac{A}{N^{\alpha}} + \frac{B}{D^{\beta}} + \frac{F}{N^{\gamma} D^{\kappa}}, \tag{2}$$

where N is the number of parameters in the network and D is the amount of data used. We wish to test the null hypothesis $H_0 : F = 0$ against the alternative $H_a : F \neq 0$.

5.2 Heteroscedasticity

We begin by looking at the distribution of losses along our multiple test runs for neuron count 80, 210, and 485. To justify OLS, these should be approximately normal (with mean around the regression prediction) and homoscedastic. A plot of variances is shown below.

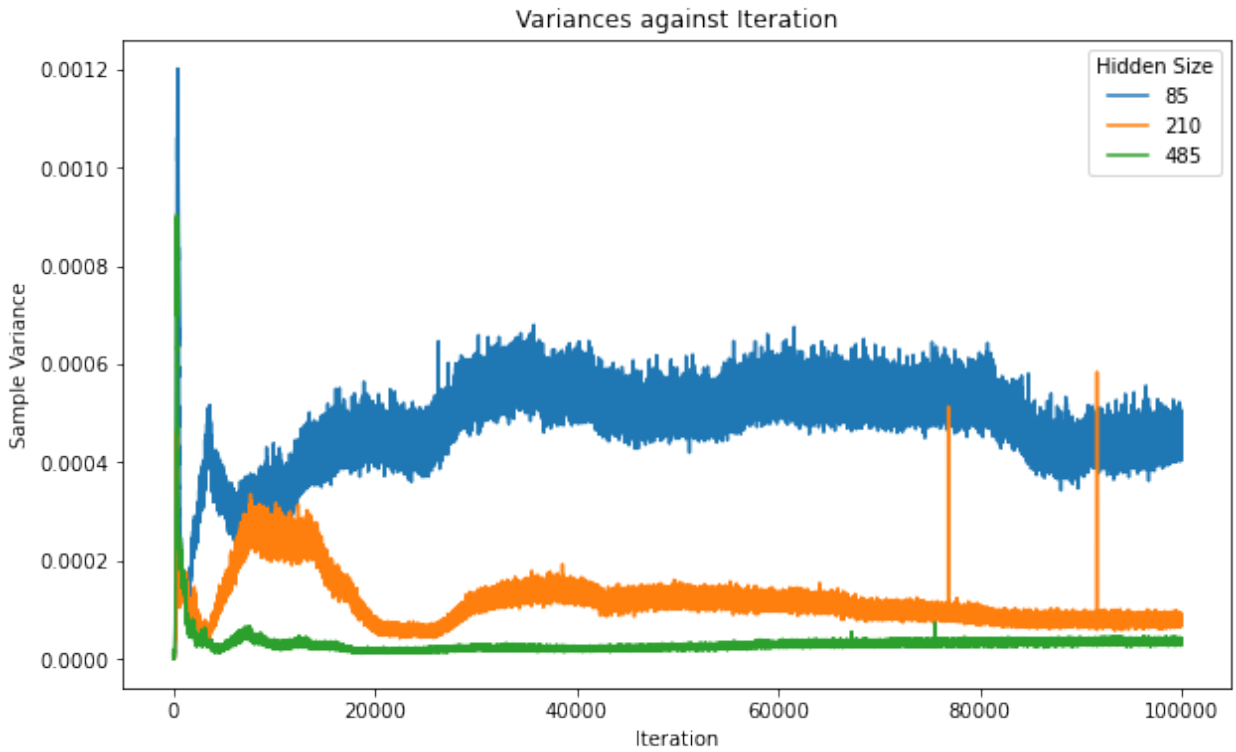


Figure 9: Variances against Model Size

We see here that the losses are emphatically not homoscedastic across parameter count, so hypothesis testing using standard OLS tests is not justified. We thus opt for a more nonparametric approach for the test, leveraging our wealth of training data. The method is a bit unique (as our form of data is itself unique), so we explain it in detail below.

Note that we have a wealth of samples along each data axis given that we have 100000 iterations run, and the residuals across these should be more or less iid for nearby points assuming that the residual distribution does not vary too much across nearby runs. Indeed, the above plot justifies that residuals nearby in data share variance. We thus opt to take the distribution of the k nearest neighbors along the same data run as a proxy for the true loss distribution at a point. We take k odd. The group of k chosen includes the data point itself and $\frac{k-1}{2}$ points to the left and right. We discuss soon optimization of choice of k.

For the actual fitting, as long as the residuals have expectation 0 conditional on input, a least squares fit is consistent and unbiased, and is hence a good proxy for the MLE under the true distribution. We will thus fit using least squares, verify that we have expectation 0 residuals, and hypothesis test using our empirical distribution estimates. (We need not actually fit using least squares; it is feasible in theory to find an MLE model under the empirical distribution derived, but the compute did not end up being feasible.)

5.3 Choice of k

We want to find a heuristic for what k neighbors are iid with our given loss, and we will then use the residuals from these k neighbors to model the distribution of residuals at a given point by fitting them using a Gaussian kernel density estimate.

To do so, we choose one run of each of the 85, 210, and 485 models and choose a given value of k. We then compute a Gaussian kernel density estimate for the loss distribution implied at each point given the value of k. We compute the Jensen-Shannon divergence of this PDF with the empirical PDF implied by taking a Gaussian kernel density estimate of the multiple runs, and we choose the k that minimizes the total summed divergence across model sizes and data points. Using this method we find the optimal k to be $k = 19$.

5.4 Fitting the Models

We fit the models using OLS on our runs. The results are below. Use of OLS is justified as residuals consistently have approximately 0 mean across the data. The log-likelihood using the above sampling method is also calculated (we use a Gaussian KDE, with pdf that peaks at high values $\gg 1$, so the log likelihoods end up being positive). We note a few interesting things: all of the models found that $E = 0$, and the cross term convergence for $\alpha = 0.2$ and $\alpha = 0.8$ both converged to $F = 0$!

Given $E = 0$, it seems that this simple task is "fundamentally solvable" with infinite parameter size and data, which must be correct given the universal approximation theorem.

Table 1: Regression Results for $F = 0$

α	Model	Residual mean	MSE	Estimated Log-Likelihood
0.2	$L = \frac{11.2}{N^{0.37}} + \frac{619.7}{D^{0.47}}$	-0.00043	0.00034	7064598.452468417
0.4	$L = \frac{16.9}{N^{0.42}} + \frac{2026}{D^{0.54}}$	-0.00138	0.000527	12528845.110598337
0.8	$L = \frac{900}{N^{0.85}} + \frac{9783}{D^{0.65}}$	-0.00021	$9.95 \cdot 10^{-5}$	8293455.333301559

Table 2: Regression Results for $F \neq 0$

α	Model	Residual mean	MSE	Estimated Log-Likelihood
0.2	$L = \frac{11.2}{N^{0.37}} + \frac{619.7}{D^{0.47}}$	-0.00043	0.00034	7064598.452468417
0.4	$L = \frac{13.8}{N^{0.42}} + \frac{2384}{D^{0.55}} + \frac{13.0}{N^{0.37}D^{0.08}}$	-0.00133	0.000522	12528836.954350756
0.8	$L = \frac{900}{N^{0.85}} + \frac{9783}{D^{0.65}}$	-0.00021	$9.95 \cdot 10^{-5}$	8293455.333301559

5.5 Conclusion

The likelihoods above (and the fact that the unrestrained $\alpha = 0.2, 0.8$ models themselves converged to an $F = 0$ solution) make it clear that we fail to reject the null. It really does not look like there is a cross term...

We note that the fit coefficients scaling for N do approximately fit theory ($\alpha_N \approx \alpha$). The coefficient of D

6 Future Steps

Strengthening our results requires training more models in a larger variety of regimes. This includes attempting different values of α, adding in more runs for the ones we've tested to better quantify the variance of our results, and also changing parameters we held fix during this experiment, namely the batch size. The paper argues that for the theoretical results to hold, the batch size should be large, as to attain nearly perfect population gradient information. The issue with this is that there are many tasks that occur very rarely, and thus it is possible that $20,000$ is not sufficient. We are also curious as to what occurs if the batch size becomes small.

7 Contribution Statement

We contributed to global warming by training a ton of models. Kevin did most of the model running and the data visualization. Oam helped with debugging, torch tips, and literature summary. Arjun trained a few models and did the hypothesis testing.

References

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

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models, 2022.

Marcus Hutter. Learning curve theory, 2021.

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, 2020.

Alexander Maloney, Daniel A. Roberts, and James Sully. A solvable model of neural scaling laws, 2022.

Eric J. Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling, 2023.

Utkarsh Sharma and Jared Kaplan. A neural scaling law from the dimension of the data manifold, 2020.