SLOTH: SCALING LAWS FOR LLM SKILLS TO PREDICT MULTI-BENCHMARK PERFORMANCE ACROSS FAMILIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling laws for large language models (LLMs) predict model performance based on parameters like size and training data. However, differences in training configurations and data processing across model families lead to significant variations in benchmark performance, making it difficult for a single scaling law to generalize across all LLMs. On the other hand, training family-specific scaling laws requires training models of varying sizes for every family. In this work, we propose Skills Scaling Laws (SSLaws, pronounced as `Sloth`), a novel scaling law that leverages publicly available benchmark data and assumes LLM performance is driven by low-dimensional latent skills, such as reasoning and instruction following. These latent skills are influenced by computational resources like model size and training tokens but with varying efficiencies across model families. `Sloth` exploits correlations across benchmarks to provide more accurate and interpretable predictions while alleviating the need to train multiple LLMs per family. We present both theoretical results on parameter identification and empirical evaluations on 12 prominent benchmarks, from Open LLM Leaderboard v1/v2, demonstrating that `Sloth` predicts LLM performance efficiently and offers insights into scaling behaviors for downstream tasks such as coding and emotional intelligence applications.

1 INTRODUCTION

Large Language Model (LLM) scaling laws for benchmarks and downstream tasks efficiently predict the performance of an LLM based on its parameter count and training set size. However, variations in training configurations and data processing across different model families often lead to significant differences in benchmark performance, even for models with comparable compute budgets (Ruan et al., 2024). Consequently, a single scaling law typically fails to predict performance across all LLMs accurately (Choshen et al., 2024). In contrast, creating family-specific scaling laws requires training multiple models of increasing size, which is resource-intensive.

In this work, we propose a new class of scaling laws called `Sloth` to solve this dilemma. These scaling laws are fitted using publicly available data (*e.g.*, from LLM leaderboards) across multiple benchmarks, leveraging information shared among benchmarks and model families to improve prediction power and interpretability through parameter efficiency, *i.e.*, fewer parameters without hurting performance. Specifically, we utilize the correlations in benchmark scores to make the scaling law simpler in terms of parameter count without harming prediction power by assuming that LLM performance is driven by a set of low-dimensional latent skills, such as reasoning and instruction following, which can be easily interpreted. Furthermore, we hypothesize that these latent skills are similarly influenced by computational resources, such as model size and training tokens, across different LLM families, with the key distinction being each family's efficiency in converting compute into skill levels–something that can be estimated with one or more models per family during testing.

In summary, our main contributions are

- Introducing a new class of scaling laws, `Sloth`, that borrows strength across the available benchmarks and LLM families to make more accurate and interpretable performance predictions of (hypothetical) LLMs in given benchmarks of interest. Specifically, we assume that benchmark performances directly depend on low-dimensional LLM skills, which are influenced by factors such as the number of training tokens and the number of parameters.

- Providing a theoretical result regarding the identification of `Sloth`'s parameters and empirically demonstrating that our scaling laws can (i) accurately predict the performance of large models in 12 prominent LLM benchmarks and (ii) provide interpretable insights into LLM scaling behavior.
- Demonstrating how predicted latent skills can be used to predict model performance in complex downstream tasks such as coding and emotional intelligence applications.

1.1 Related work

Scaling laws for deep neural networks In recent years, researchers have studied scaling laws from different angles. Rosenfeld et al. (2019) provides experimental scaling laws that predict model loss as a function of training set size, model width, and model depth. Likewise, Kaplan et al. (2020) establishes scaling laws that primarily measure loss (perplexity) and not accuracy on downstream tasks or benchmarks. Motivated by the presence of hard limits on the size of trainable data sets but a hypothetical unlimited ability to scale models, the authors of Muennighoff et al. (2023) establish scaling laws in constrained data settings. They find that perhaps unsurprisingly, increasing computing provides diminishing returns if data does not scale. Gadre et al. (2024) addresses the gap between the assumptions in scaling laws and how training is performed in practice; in particular, they construct scaling laws that both perform well in the over-training regime and predict performance on downstream tasks. In a similar but distinct direction, some works try not only to estimate scaling laws but also respond to the following strategic question: "Given a fixed FLOPs budget, how should one trade-off model size and the number of training tokens?" For example, Hoffmann et al. (2022) provides a partial answer, introducing the celebrated family of Chinchilla scaling laws and finding that training tokens and parameter size should roughly scale together. This contrasts with the older work of Kaplan et al. (2020) that provides a series of power laws that imply that simply increasing parameter count will provide good returns. Each of these referenced works trains models with a particular pretraining setting (e.g., architecture) at various sizes and ultimately seeks to predict test loss. Our focus is distinct, we fit scaling laws on existing benchmark data of multiple model families and predict LLM benchmark performance with minimal amount of data on the new family being predicted. The closest related works are Owen (2024); Ruan et al. (2024); Gadre et al. (2024); we will provide a detailed comparison with their work throughout the paper.

LLMs latent skills Given that the performance of large language models (LLMs) in different and diverse benchmarks is correlated, it makes sense to think that those models have some low-dimensional latent skills that are reflected in downstream tasks. In this direction, Ilić (2023) extracts a general intelligence factor ("g-factor") for LLMs using the Open LLM Leaderboard (Beeching et al., 2023) and GLUE (Wang et al., 2018) using factor analysis. They also verify that this "g-factor" positively correlates with model size. In a similar direction Burnell et al. (2023) uses HELM (Liang et al., 2022) data to reveal that LLM intelligence may be constituted by three distinct, yet correlated factors. They also verify a positive correlation between model size and these latent skills, yet they do not propose a formal scaling law. In their study, the authors do not account for the training set size or model family information, leading to a poor fit of the regression model; this leaves good extrapolation as an open problem we address. In Kipnis et al. (2024), a unidimensional item response theory model is applied to each one of the 6 (filtered) benchmarks of the Open LLM Leaderboard. A factor analysis on the skill parameters shows that the main factor (carrying 80% of the data variability) is highly correlated with the "grand" (average) score of LLMs. In a related but different direction, Maia Polo et al. (2024a;b) show that inferring low-dimensional latent skills of LLMs can make model evaluation much more efficient, saving up to 140x in computing power. In this work, we explicitly model LLM skills as a function of computing resources, which enables the creation of accurate and interpretable scaling laws for benchmark performances.

2 Scaling laws for benchmark data

2.1 Problem Statement

In this section, we describe the setup we work on and what our objectives are. Within a family of LLMs i (*e.g.*, LLaMA 3), our objective is to estimate the performance of a big LLM, *e.g.*, with 70 billion parameters, in a benchmark j, *e.g.*, MMLU, given evaluation data from smaller models in the same family. Let s represent the size of the LLM, defined as the number of parameters, and let t denote the number of training tokens. We define $Y_{ij}(s,t) \in [0,1]$ as the score of an LLM from family i, with size s and trained on t tokens, on benchmark j. Our goal is to approximate the following

quantity:

$$\mu_{ij}(s,t) = \mathbf{E}[Y_{ij}(s,t)]. \tag{2.1}$$

Here, $\mathbf{E}[\cdot]$ should be interpreted as a central tendency summary measure of a random variable, such as the mean or median. Ideally, the model for μ_{ij} will be simple and some of its parameters will be shared among model families and benchmarks; in this case, the model becomes more interpretable and more data can be used in the fitting process, making the model better estimated. From now on, we denote the set of model families as $\mathcal{I} = \{1, \cdots, I\}$ and the set of benchmarks as $\mathcal{J} = \{1, \cdots, J\}$.

2.2 Previous approaches to scaling laws for benchmarks

The closest works to ours that propose models for $\mu_{ij}(s,t)$ (2.1) are Owen (2024); Ruan et al. (2024), and Gadre et al. (2024). While Gadre et al. (2024) indirectly model the quantity of interest via the LLMs perplexity in specific datasets, which might not be readily available, Owen (2024) and Ruan et al. (2024) model $\mu_{ij}(s,t)$ directly through a regression model connecting compute and benchmark performance. One assumption they made is that the performance on benchmarks only depends on s and t through the total amount of training FLOPs, which can be approximated by $c(s,t) = 6st$. That is, if $\sigma : \mathbf{R} \to [0,1]$ denotes a fixed activation function, *e.g.*, the standard logistic (sigmoid) function, and $\gamma_j \in [0,1]$, then it is assumed that

$$\mu_{ij}(s,t) = \gamma_j + (1-\gamma_j)\sigma(\eta_{ij}(s,t)), \tag{2.2}$$

where $\eta_{ij} : \mathbf{R}^2 \to \mathbf{R}$ denotes a linear predictor such that $\eta_{ij}(s,t) = \alpha_{ij} + \beta_{ij} \log c(s,t)$, which can be easily interpreted. Here, γ_j adjusts the lower asymptote of μ_{ij} and accounts for the probability of LLMs scoring correctly by chance. Owen (2024), in their best performing models, considers the case in which $\gamma_j = 0$ (or adds a similar, but slightly different, offset parameter to the model) and the parameters α_{ij} and β_{ij} are independent of the model family i. On the other hand, Ruan et al. (2024) consider both α_{ij} and β_{ij} to be family-dependent and, in their most general model, γ_j can assume values in $[0,1]$.

The biggest issue with previous approaches when modeling μ_{ij} is that they are either too restrictive or too flexible. From the restrictive side, they assume that (i) μ_{ij} depends on s and t only through FLOPs, (ii) there are no family-dependent parameters, or (iii) the activation function σ is fixed and well-specified. From the flexibility side, Ruan et al. (2024) assume both α_{ij} and β_{ij} to be family dependent making estimation hard (or impossible) depending on the number of models we see for each family. From Ruan et al. (2024): "(...) fitting such a scaling law can be tricky, as each model family f and downstream benchmark has its own scaling coefficients β_f and α_f. This means that scaling experiments, especially for post-training analysis, are often fitted on very few (3-5) models sharing the same model family (...)." Thus, in their experiments, they consider a different problem setting, where a large LLM has been trained and evaluated on some benchmarks and use their method to predict its performance on other benchmarks. In this work we wish to instead predict the performance of a larger LLM *without having to train it*, thus allowing practitioners to make decisions regarding investing resources into training large LLMs. However, we note that it is possible to extend their method to achieve the same goal and we consider such extension among our baselines.

3 Scaling laws for LLMs skills with Sloth

3.1 Model architecture

We present a novel scaling law called `Sloth`, which introduces several modifications to (2.2). The key innovation of `Sloth` lies in its explicit modeling of the correlation structure between benchmarks, resulting in improved predictive accuracy and interpretability. Moreover, `Sloth` proposes that (i) LLM capabilities should scale with computing resources similarly across families up to an efficiency factor, (ii) benchmark performance can depend on s and t not only through the total number of FLOPs, and (iii) that the function σ can also be learned in cases in which predictive performance is important. We detail each one of these points.

Inspired by the latent skills (*e.g.*, reasoning, language modeling, instruction following) inferred from benchmark data in Burnell et al. (2023); Ilić (2023); Ruan et al. (2024); Gor et al.; Kipnis et al. (2024); Maia Polo et al. (2024a;b), we propose creating a scaling law for LLMs skills by leveraging the correlation structure of the benchmarks; for example, we model how the construct "reasoning" scales with compute instead of modeling benchmarks scores directly. The two major advantages of this approach are better performance prediction since we have fewer parameters to fit (reducing

overfitting) and extra interpretability/insights. Concretely, we model $\eta_{ij}(s,t)$'s simultaneously for benchmarks $j \in \mathcal{J}$ as each being a linear combination of the same low-dimensional latent skills $\theta_i(s,t) \in \mathbf{R}^d$ plus a bias term $b \in \mathbf{R}^J$, where $d \ll J = |\mathcal{J}|$. Denote $\eta_i(s,t) \in \mathbf{R}^J$ as the vector of $\{\eta_{ij}(s,t)\}_{j\in\mathcal{J}}$. Mathematically, we have

$$\eta_i(s,t) = \Lambda\theta_i(s,t) + b. \tag{3.1}$$

One can see that $\Lambda \in \mathbf{R}^{J\times d}$ encodes the correlation structure between the benchmarks; in particular, it tells us which benchmark measures overlapping (or distinct) skills. Interestingly, our model has a strong connection with factor analysis (FA) models, which we elaborate on in detail in Appendix C. In FA, the matrix Λ is known as factor loadings while $\theta_i(s,t)$ are known as factors.

Next, we propose a model for $\theta_i(s,t)$. Inspired by models used in Economics, we propose using the family of translog production functions from stochastic frontier analysis (Kumbhakar & Lovell, 2003):

$$\begin{aligned} \theta_{ik}(s,t) &= \alpha_{ik} + \beta_k^\top x(s,t); \quad 1 \le k \le d, \\ x(s,t) &= (\log(s), \log(t), \log(s)\log(t)). \end{aligned} \tag{3.2}$$

Note that (i) the intercept parameter α_{ik} is indeed family-dependent while each skill slope is shared across families and (ii) θ_i can depend on s and t not only through $c(s,t)$. In economic terms, the intercept term α_{ik} can be interpreted as an efficiency measure of the family i in converting compute to performance for skill k and, in practice, will absorb all hidden factors specific to family i such as data quality, post-training factors, *etc.*. We note that the interaction term in $(\log(s)\log(t))$ accounts for the fact that the impact of $\log(s)$ and $\log(t)$ on skills might depend on each other; in Appendix D, we show some evidence that this is indeed the case. Additionally to the changes in η_{ij}, we propose making the activation function σ trainable and specific to each benchmark j if needed. To that end, we adopt a semi-parametric single-index model using neural networks (Bietti et al., 2022). To make the results more behaved if (out-of-support) generalization is needed, we assume $\sigma_j : \mathbf{R} \to [0,1]$ is given by a monotonic (increasing) neural network, which can be achieved by constraining its weights to be non-negative (Sill, 1997) and its last activation function to be sigmoid. We note, however, that one can always forgo training of the link function and instead assume a sigmoid structure as this simplifies model fitting and may make the model more interpretable.

3.2 Identifiability of model parameters

To interpret `Sloth` parameters, we need to guarantee they are identifiable. Given that our scaling law models the function $\mu_{ij}(s,t) = \mathbf{E}[Y_{ij}(s,t)]$, that condition is equivalent to the following statement: if two sets of parameters are responsible for characterizing $\mu_{ij}(s,t)$, then those set of parameters should be the same up to predictable variations such as translations or rotations. To prove identifiability, we work with a fixed and invertible σ, as usually done in the literature, and assume γ_j's are fixed. The last condition is reasonable since these constants are usually known beforehand, *e.g.*, it is well accepted that the lower asymptote γ_j for MMLU (Hendrycks et al., 2020) performance is 25% which is given by 100% divided by the number of multiple-choice alternatives. Denote our fixed design matrix as $X \in \mathbf{R}^{n\times p}$, where each row is given by an LLM and p equals 3 plus the number of families, and define

$$B = \begin{pmatrix} \beta_1 & \cdots & \beta_d \\ \alpha_{11} & \cdots & \alpha_{1d} \\ \vdots & \ddots & \vdots \\ \alpha_{m1} & \cdots & \alpha_{md} \end{pmatrix} \in \mathbf{R}^{p\times d}$$

such that the rows of $XB \in \mathbf{R}^{n\times d}$ give the skills vectors $\theta^{(i)} \triangleq (XB)^{(i)}$'s of all models in our dataset. Here n denotes the total number of models in the dataset and m is the total number of model families. To prove identifiability, we adopt standard assumptions from the factor analysis literature (Chen et al., 2019) or regression literature, which assumes that the skills vectors $\theta^{(i)} \in \mathbf{R}^{1\times d}$'s are standardized, *i.e.*, their average is null while their covariance matrix is fixed, $\text{rank}(\Lambda) = d$, and $\text{rank}(X) = p$.

Assumption 3.1 (Identifiability constraints)**.** *Assume that*

$$\frac{1}{n}\sum_{i=1}^n \theta^{(i)} = 0, \quad \frac{1}{n}\sum_{i=1}^n {\theta^{(i)}}^\top \theta^{(i)} = \Psi, \quad \textit{rank}(\Lambda) = d, \quad \textit{and} \quad \textit{rank}(X) = p,$$

where $\theta^{(i)}$ denotes the i-th row of XB and Ψ is a positive definite matrix.

One possible choice for the covariance matrix is $\Psi = I_d$ (Chen et al., 2019), which assumes uncorrelated skills. One implicit implication of Assumption 3.1 is that $n \geq p \geq d$ must be satisfied, otherwise the covariance matrix cannot be full rank. This condition is satisfied in our experiments. Under Assumption 3.1, we show the identifiability of the model parameters up to a transformation of Λ tied to a transformation of B, which leaves the outputs of the model unchanged. This means that we can potentially approximate the true values for Λ and B up to a transformation, which is usually the norm within the class of exploratory factor analysis models.

Theorem 3.2. *Given that the true set of model parameters is (Λ, b, B), if there is another set of parameters $(\tilde{\Lambda}, \tilde{b}, \tilde{B})$ that satisfy*

$$\sigma\left(\Lambda(XB)^{(i)\top} + b\right) = \sigma\left(\tilde{\Lambda}(XB)^{(i)\top} + \tilde{b}\right) \text{ for all } i \in [n],$$

then, under the Assumption 3.1, we have $\tilde{b} = b$, $\tilde{\Lambda} = \Lambda M$, and $\tilde{B} = B(M^\top)^{-1}$ for an invertible matrix $M \in \mathbf{R}^{d\times d}$. In particular, M is orthogonal if $\Psi = I_d$, i.e., $M^\top M = MM^\top = I_d$.

We place the proof of Theorem 3.2 in Appendix B. From our proof, we can see that the matrix M is dependent on the specification of Ψ.

3.3 Model fitting

Assume that for each model family i we observe a set of tuples (s,t)'s denominated by $\mathcal{E}_i$. Then, we fit the model by solving the following minimization problem

$$(\hat{\gamma}, \hat{\sigma}, \hat{b}, \hat{\Lambda}, \hat{\alpha}, \hat{\beta}) = \underset{\substack{\gamma_j \in [0,1], \text{ for } j\in\mathcal{J} \\ \sigma_j : \mathbf{R}\to[0,1] \text{ increasing , for } j\in\mathcal{J} \\ b_j\in\mathbf{R}, \text{ for } j\in\mathcal{J}; \Lambda\in\mathbf{R}^{J\times d} \\ \alpha_{ik}\in\mathbf{R}, \text{ for } i\in\mathcal{I} \text{ and } 1\leq k\leq d \\ \beta_k\in\mathbf{R}^3, \text{ for } 1\leq k\leq d}}{\arg\min} \sum_{i\in\mathcal{I}} \sum_{(s,t)\in\mathcal{E}_i} \sum_{j\in\mathcal{J}} \ell_\delta(\mu_{ij}(s,t), Y_{ij})$$

where ℓ_δ is given by the Huber loss with hyperparameter $\delta = .01$ and $\mu_{ij}(s,t)$ denotes the most general version of our model. We minimize the loss function via gradient descent using the Adam optimizer (Kingma & Ba, 2017) with a decaying learning rate. We parameterize γ_j using the sigmoid transformation to guarantee the constraints are satisfied. Similarly, we truncate the weights of the two-hidden-layer neural network σ_j to ensure the trainable function is increasing. If one desires, σ_j's can be set to fixed functions, *e.g.*, sigmoid, and γ_j's can be fixed beforehand. Unfortunately, the minimization problem is not convex as expected when fitting factor-analysis-like models; multiple initializations of the optimizer can be applied to guarantee a better fit.

3.4 Interpretability and practical considerations post model fitting

In practical situations, it is hard to fix the covariance matrix of skills to something meaningful before fitting the model, as suggested in Section 3.1. To make the model interpretable, we mirror a standard approach used in factor analysis, *e.g.*, in Chen et al. (2019)'s applications. First, we fit `Sloth` without any constraints on the covariance of skills obtaining the estimates $(\hat{\Lambda}, \hat{b}, \hat{B})$. Second, we find the matrix $A \in \mathbf{R}^{d\times d}$ such that the skills $X\hat{B}A$ have covariance identity, update $\hat{B} \leftarrow \hat{B}A$, and update $\hat{\Lambda} \leftarrow \hat{\Lambda}(A^\top)^{-1}$ so the model outputs remains unchanged, because $\hat{\Lambda}(X\hat{B})^\top = \hat{\Lambda}(A^\top)^{-1}(X\hat{B}A)^\top$. Third, we find a matrix $M \in \mathbf{R}^{d\times d}$ such that $\hat{\Lambda}M$ is easily interpretable (*e.g.*, it is a sparse matrix); there are different methods to find M and, in this paper, we use the *Geomin* (Yates, 1987; Chen et al., 2019) oblique rotation method to find a suitable M using the Python package FactorAnalyzer (Biggs, 2019). We then update $\hat{\Lambda} \leftarrow \hat{\Lambda}M$ and, to make the model invariant, we also update $\hat{B} \leftarrow \hat{B}(M^\top)^{-1}$; the resulting skills are still guaranteed to have unitary standard deviations, so their covariance equals their correlation. Finally, we standardize the columns of the skills $X\hat{B}$ to have zero mean, while keeping the correlation structure unchanged. This last step implies that $\hat{b}$ must be translated to make the model invariant.

4 Sloth in practice

In this section, we present some experimental results that provide evidence of the usefulness of `Sloth`. We perform experiments on a set of twelve benchmarks and state-of-the-art LLM families, including LLaMa 3 (Dubey et al., 2024), Qwen 2 (Yang et al., 2024), and Yi 1.5 (Young et al., 2024).

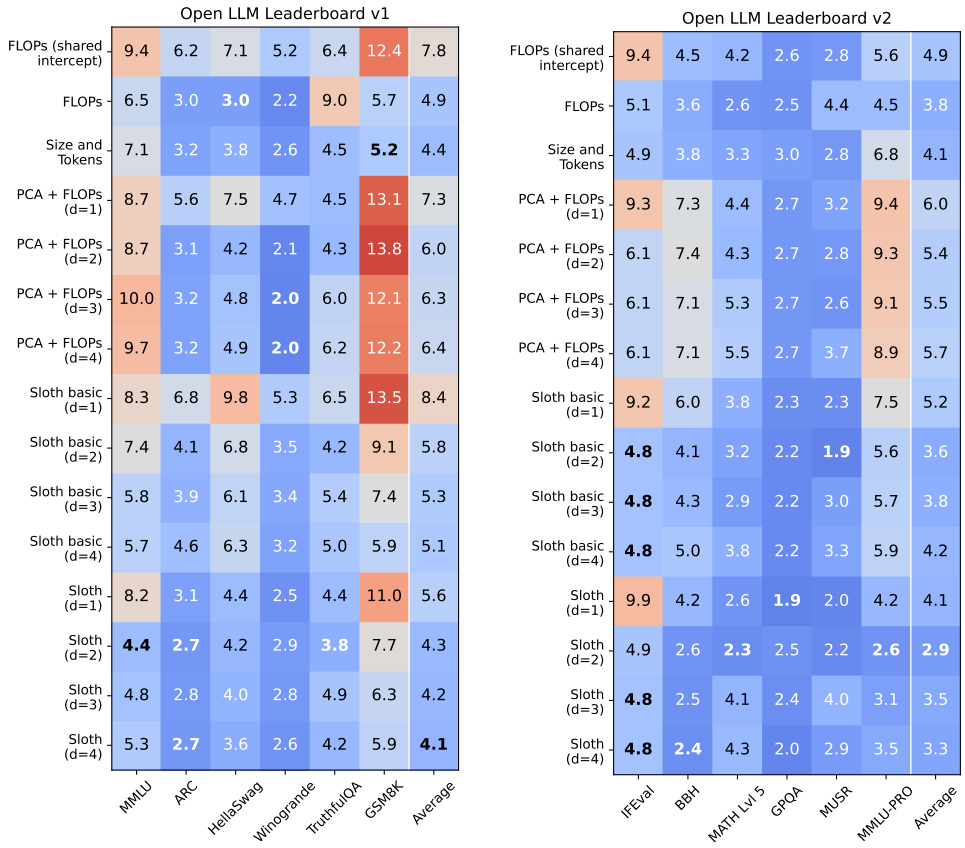


Figure 1: The figure shows the average (across LLM families) mean-absolute-error (MAE) (within a family) for different methods. `Sloth` performs competitively, with errors similar to or lower than the "Size and Tokens" variant, indicating its effective inductive bias.

We explore the following applications: (i) benchmark performance prediction for larger models from a specific LLM family, (ii) interpretability of the scaling of skills (can help practitioners allocate resources based on the skills of interest), and (iii) complex downstream tasks performance prediction.

4.1 DATA

We expand the dataset made available by Ruan et al. (2024), including more models from the HuggingFace Open LLM leaderboard v1 (Beeching et al., 2023) and v2 (Fourrier et al., 2024). In our extended dataset, we have a total of 30 families[1], which 28 are on v1 of the Open LLM Leaderboard and 17 families measured on v2 of the Open LLM Leaderboard. Furthermore, there are 15 families at the intersection of the two versions. Furthermore, we collect data and present results on the performance of a variety of instruction-tuned versions of the base models we consider. As far as we are aware, our dataset is the most comprehensive among prior works on benchmark data scaling laws. Please check Appendix F for details on the included models.

4.2 COMPARING SCALING LAWS IN TERMS OF PREDICTION ERRORS

In this section, we compare the predictive power of different scaling laws in predicting LLM performance in all the considered benchmarks; we focus on the two versions of the Open LLM Leaderboard, which include 12 benchmarks: GSM8k (Cobbe et al., 2021), MATH Lvl 5 (Hendrycks et al., 2021), MMLU (Hendrycks et al., 2020), MMLU-PRO (Wang et al., 2024), BBH (Suzgun et al., 2022), GPQA (Rein et al., 2023), MUSR (Sprague et al., 2023), TruthfulQA (Lin et al., 2021), HellaSwag (Zellers et al., 2019), Winogrande (Sakaguchi et al., 2019), ARC (Clark et al., 2018),

[1]If we consider that instruct and base models are from different families, we end up with 53 families.

and IFEval (Zhou et al., 2023). We apply a leave-one-out cross-validation algorithm to obtain test errors for each family of models. We consider base models and instruct models to belong to distinct families (they will not share the same intercept in our model, for example), but we do not include the instruct (resp. base) family in the training set when the corresponding base (resp. instruct) family is in the test set. Moreover, we do not test older versions of recent families if they are available in the training set, *e.g.*, we do not include LLaMa 2 in the set of test families if LLaMa 3 is present in the training set. In this section, we present results for the two leaderboards separately; in Figures 11 and 16 of the Appendix, we also present results for the intersection of the two leaderboards.

In the first set of experiments, we consider the case in which only the smallest model of the test family is observed at training time. Because of that reason, we cannot fit the general scaling law in (2.2) in which both the intercept and slope are family dependent. In this scenario, we consider our main baselines to be (i) the model in (2.2) with shared intercept and slope (Owen, 2024) ("FLOPs (shared intercept)"), (ii) the same model but only with shared slope ("FLOPs"), (iii) a version of the PCA idea[2] proposed by Ruan et al. (2024) in which we predict the principal components using the FLOPs model with shared slope that are then mapped to the benchmark scores ("PCA + FLOPs"), (iv) and our model with trainable activation function but assuming Λ is identity ("Size and Tokens"; implies $d = J$). Moreover, we include two versions of `Sloth`. In the "basic" one, we fix σ to be sigmoid, and γ_j's are given by the 100% over the number of alternatives in the case of multiple-choice benchmarks[3] and 0 otherwise, except for TruthfulQA, which we empirically compute the first percentile of the scores coming from the full Open LLM Leaderboard and fix the lower asymptote to that value.

Figure 2: Average prediction error across LLM families and benchmarks. `Sloth` performs well while the scaling laws in which intercept and slope are family-dependent underperform.

Figure 1 gives the results for the first set of experiments. It depicts the average mean absolute error of all methods when predicting LLM benchmark performance, which is measured in percentage points. It shows the competitiveness of `Sloth` in terms of prediction quality. One important thing to notice is that `Sloth` errors are similar or lower than the "Size and Tokens" variant, suggesting that the assumed low-dimensional structure between benchmarks results is a good inductive bias. We highlight that the analysis includes recent families like LLaMa 3, Qwen 2, and Yi 1.5. For more details on the tested models and extra related results, including model-specific results, please check Appendix G.1. The extra results are qualitatively similar to the ones in Figure 1, in which `Sloth` is competitive and often beats the baselines.

In the second set of experiments, we consider the case in which the two smallest models of the test family are observed at training time. In this way, we can fit (2.2) making both parameters family dependent. For this analysis, we modify all methods that depend on FLOPs, by letting the slope (in addition to the intercept) depend on the family. All variations of our method are kept unchanged. Figure 2 shows the average mean-absolute-error (MAE) across families and benchmarks for different methods. We see that the different variants of our approach are still the most successful ones in performance prediction. We also replicate the observation made by Ruan et al. (2024) that fitting both intercept and slope per family performs poorly. For more detailed results, including family-specific results see Appendix G.2.

[2]We include more details about the PCA approach in Appendix E.

[3]When the benchmark has subsections with a different number of alternatives, we compute the asymptote parameters per subsection and then compute an overall asymptote using a weighted average in which the weights are proportional to the number of examples in each subsection.

In an extra set of experiments, we show that family-specific intercept models are not always needed; we can still get good prediction results for some benchmarks even if we consider a shared intercept between families. The advantage of this approach is that we can claim for a *general* scaling law that holds for all families. Figure 3 shows us a subset[4] of Figure 11 in the appendix and it is built under the same conditions as Figure 1. It is possible to see that, for a subset of benchmarks `Sloth` with shared intercept is a strong alternative to the FLOPs model used by Owen (2024). In some cases, it gets similar prediction errors relative to more complete versions of `Sloth`.

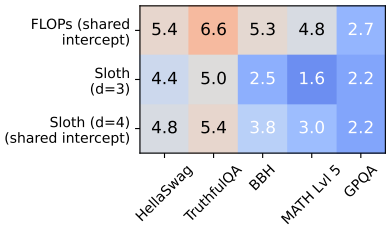


Figure 3: Running `Sloth` with shared intercept can offer a great way to model scaling laws that are family-independent.

4.3 Interpreting the latent skills

In this section, we use the intersection between the two leaderboards, aiming to get more insights from the combined data. Since we have an identifiability result for the "basic" version of `Sloth`, in which we fix the lower asymptotes γ_j's and the link function to be sigmoid (see Section 3.2), we opt for interpreting that version of the model. We set $d = 3$ as that model version achieved the best prediction results in Figure 11. Figure 4 illustrates the model loadings, Λ, from which we assign names to the three dimensions based on our subjective interpretation. We include the loadings for $d = 2$ and $d = 4$ in Appendix H. To complement our exploration, we include Figure 5, which gives us the level curves of different skills (disregarding the family-specific intercept term), and Figure 6 that compare the skills of base and instruction-tuned models; in this figure, we include LLM families with more number of models. In both figures, the numbers are given in terms of standard deviations as the skills are standardized to have zero mean and unitary standard deviation.

Reasoning skill The first dimension, with strong loadings from benchmarks such as GSM8K, MATH, GPQA, MMLU(-PRO), BBH, and MUSR, is labeled "Reasoning." The benchmarks GSM8k and MATH Lvl 5 consist entirely of mathematical word problems while MMLU/MMLU-PRO and GPQA also contain mathematical and advanced science questions. On the other hand, BBH includes logic deduction and linguistic reasoning puzzles. The strong dependence of BBH on the "Reasoning" skill suggests that in language models, there is an association between logical reasoning, general linguistic ability, and mathematical ability. Finally, MuSR is a benchmark that evaluates "multistep soft reasoning tasks specified in a natural language narrative" (Sprague et al., 2023). Figure 5 shows that Reasoning is primarily a function of model size, with a small dependence on the number of training tokens used. Moreover, the first plot of Figure 6 compares base models versus their instruction-tuned versions in terms of Reasoning and we found that there is no clear rule: instruction tuning can either increase or decrease the ability of an LLM to reason. These findings are robust for different values of d as we can see in the figures of Appendix H.

Knowledge skill The second dimension is positively loaded on ARC, HellaSwag, and Winogrande. These three benchmarks measure the ability of LLMs to remember common sense and basic knowledge; we denominate this skill as "Knowledge". More specifically, ARC consists of grade school-level science questions, HellaSwag is meant for sentence completion for common scenarios, and Winogrande common sense pronoun resolution problems. Contrasting with Reasoning, Figure 5 shows that Knowledge is highly influenced by both model size and number of training tokens. Moreover, we can see that the range of standard deviations in the middle plot is much greater than in the other two plots, giving us evidence that this skill might be more sensitive to increases in compute resources and less dependent on the LLM families themselves. On the other hand, Figure 6 does not show any strong evidence of the effect of instruction tuning on this skill. These findings are similar to the ones reported in Appendix H for different values of d, even though there is no clear one-to-one correspondence of Knowledge in those results.

Instruction following skill: IFEval, which is positively and heavily loaded in this skill, measures how well language models produce answers that follow a verifiable set of instructions; for example, including a keyword x number of times in responses. Therefore, we call it "Instruction Following". An interesting fact is that instruction tuning has a strong positive effect on this skill for all depicted

[4]We selected the best d for both versions of `Sloth`.

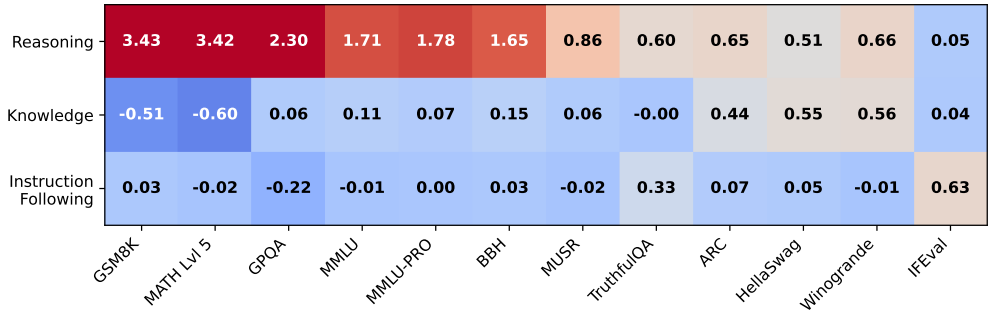


Figure 4: Needed skills for each benchmark. In this figure, we report the estimated loadings Λ and, based on their values, we give them appropriate names.

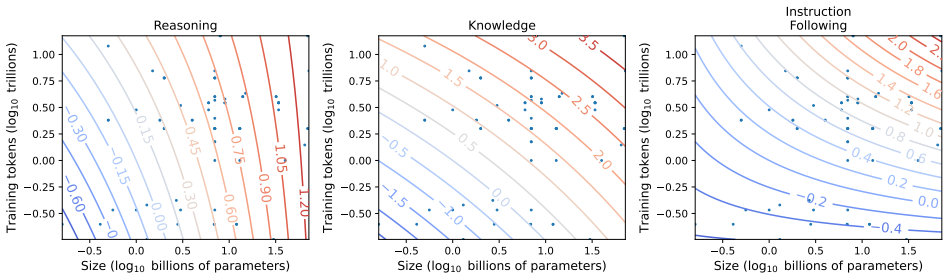


Figure 5: Level curves in producing different latent skills from parameter count and training tokens.

families we can see in Figure 6. The effect can also be observed in Figure 27 of the appendix. When $d = 2$, instruction following gets mixed with other skills and we are not able to see this effect. Regarding Figure 5, we see that Instruction Following depends on both model size and tokens. Unfortunately, this interpretation does not hold when $d = 4$ as seen in Appendix H; in that case, instruction following abilities are almost constant with respect to size and number of tokens for models trained on bigger datasets and size is negatively correlated to it when the models are trained on smaller datasets.

4.4 Predicting LLM performance on downstream tasks

Another useful application of `Sloth`, which is inspired by Ruan et al. (2024), is to predict the performance in a downstream task for a large model from a relatively small number of prior performance observations from that task. We use Sloth to estimate the latent skills of hypothetical LLMs and then use them to predict the performance of those LLMs in downstream tasks. With this approach, we expand on the experiments by Ruan et al. (2024), which do not consider performance prediction of hypothetical LLMs; as we have seen in Section 4.2 (from the "PCA + FLOPs" baselines in Figures 1 and 2), their method could be adapted to this task but it has, in general, poor predictive performance.

The basic prediction pipeline is as follows. First, use standard LLM leaderboards to fit a scaling law for skills using `Sloth`. Second, use existing LLM performance on the downstream task to model how performance can be predicted from latent skills. Third, use `Sloth` to predict the skills of a (hypothetical) LLM of interest, *e.g.*, a larger version of an existing LLM. Finally, use the model fitted in the second step to predict the performance of the hypothetical model in the downstream task.

We evaluate this pipeline on two downstream tasks, predicting the performance of `meta-llama-3-70B` and `meta-llama-3-70B-instruct` on code completion and `meta-llama-3-70B-instruct` on emotional intelligence tasks. We fit the same model shown in Section 4.3, but *do not include* `meta-llama-3-70B` *or* `meta-llama-3-70B-instruct` *in the training set* (see Figure 30 for the loadings of the latent skills, which is similar to Figure 4). Next, using either HumanEval (Chen et al., 2021) or EQ bench data (Paech, 2024), we fit a regression model with logistic link using latent skills as features and performance on the downstream task as target. Together, this provides us with sufficient information to predict the performance of the held-out models on

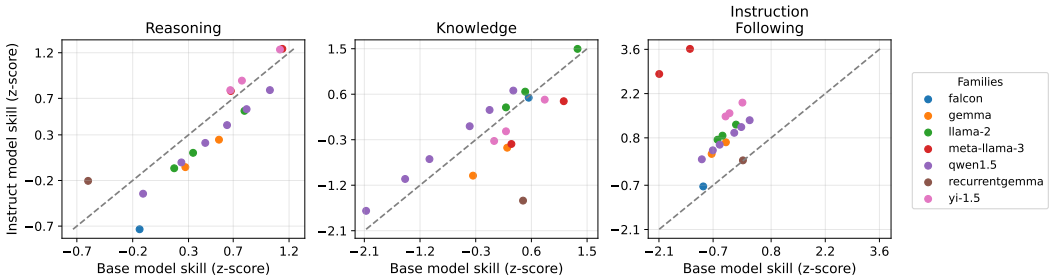
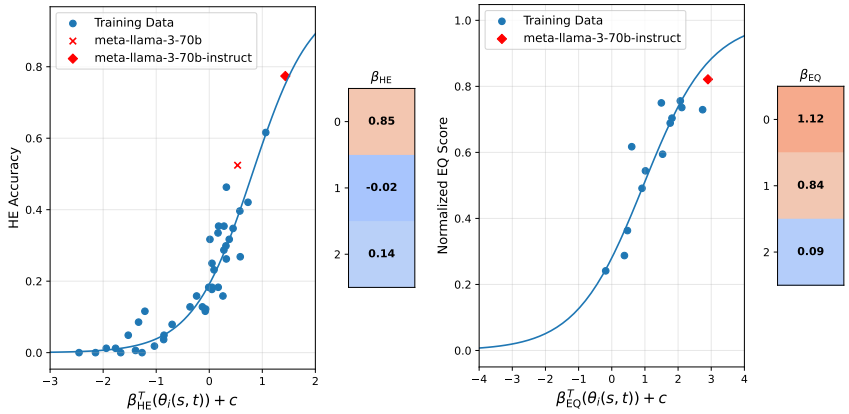


Figure 6: Gains from instruction tuning for different families on three latent skills. Findings include a large and positive impact on instruction following.

both tasks with decent accuracy. Figure 7 depicts this logistic curve and the actual values. Moreover, we can see that "Reasoning" is by far the most important skill in predicting coding ability while a mixture of "Reasoning" and "Knowledge" is needed for emotional intelligence (see Figure 30 for a more accurate interpretation of the loadings). In Appendix I, a similar test is provided for agentic



(a) Code Completion

(b) Emotional Intelligence

Figure 7: Predicting code completion and EQ of LLaMa 3 70B (base/instruct).

capability measured by AgentBench (Liu et al., 2023), although to avoid overfitting, in this case, we must fit `Sloth` with no family-specific intercept.

5 Conclusion

In conclusion, we have introduced the `Sloth` scaling laws as a novel approach to predicting the performance of large language models across benchmarks and model families. By leveraging the correlations between benchmark scores and assuming that LLM performance is governed by a set of interpretable, low-dimensional latent skills, our approach offers a more efficient and flexible framework for understanding and predicting LLM behavior. The ability to estimate model performance across a variety of benchmarks and tasks, even with minimal data from individual model families, highlights the practical utility of `Sloth` scaling laws.

Our empirical results demonstrate that `Sloth` can accurately predict the performance of LLMs across multiple benchmarks while providing insights into the relationship between computational resources and model capabilities. By reducing the reliance on resource-intensive model training and offering parameter-efficient scaling laws, this work paves the way for more informed decisions about LLM development and deployment. Future work could explore extending this framework to additional model families and downstream tasks, further enhancing the understanding of LLM scaling behaviors in diverse real-world applications.

We include a paragraph about limitations in Appendix A.

REFERENCES

Edward Beeching, Clémentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard., 2023. URL https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard,2023.

Alberto Bietti, Joan Bruna, Clayton Sanford, and Min Jae Song. Learning single-index models with shallow neural networks. *Advances in Neural Information Processing Systems*, 35:9768–9783, 2022.

Jeremy Biggs. Factor_analyzer documentation. *Release 0.3*, 1, 2019.

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

Ryan Burnell, Han Hao, Andrew RA Conway, and Jose Hernandez Orallo. Revealing the structure of language model capabilities. *arXiv preprint arXiv:2306.10062*, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. 2021.

Yunxiao Chen, Xiaoou Li, and Siliang Zhang. Joint maximum likelihood estimation for high-dimensional exploratory item factor analysis. *Psychometrika*, 84:124–146, 2019.

Leshem Choshen, Yang Zhang, and Jacob Andreas. A hitchhiker's guide to scaling law estimation. *arXiv preprint arXiv:2410.11840*, 2024.

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. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. Open llm leaderboard v2. https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard, 2024.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, Rui Xin, Marianna Nezhurina, Igor Vasiljevic, Jenia Jitsev, Luca Soldaini, Alexandros G. Dimakis, Gabriel Ilharco, Pang Wei Koh, Shuran Song, Thomas Kollar, Yair Carmon, Achal Dave, Reinhard Heckel, Niklas Muennighoff, and Ludwig Schmidt. Language models scale reliably with over-training and on downstream tasks, 2024. URL https://arxiv.org/abs/2403.08540.

Maharshi Gor, Tianyi Zhou, III Hal Daumé, and Jordan Boyd-Graber. Do great minds think alike? investigating human-ai complementarity for question answering.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, October 2020.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874.

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.

David Ilić. Unveiling the general intelligence factor in language models: A psychometric approach. *arXiv preprint arXiv:2310.11616*, 2023.

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. URL https://arxiv.org/abs/2001.08361.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, January 2017.

Alex Kipnis, Konstantinos Voudouris, Luca M Schulze Buschoff, and Eric Schulz. metabench - a sparse benchmark to measure general ability in large language models. *arXiv preprint arXiv:2407.12844*, 2024.

Subal C Kumbhakar and CA Knox Lovell. *Stochastic frontier analysis*. Cambridge university press, 2003.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.

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

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents, October 2023.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. *arXiv preprint arXiv:2402.14992*, 2024a.

Felipe Maia Polo, Ronald Xu, Lucas Weber, Mírian Silva, Onkar Bhardwaj, Leshem Choshen, Allysson Flavio Melo de Oliveira, Yuekai Sun, and Mikhail Yurochkin. Efficient multi-prompt evaluation of llms. *arXiv preprint arXiv:2405.17202*, 2024b.

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, 2023. URL https://arxiv.org/abs/2305.16264.

David Owen. How predictable is language model benchmark performance? *arXiv preprint arXiv:2401.04757*, 2024.

Samuel J. Paech. Eq-bench: An emotional intelligence benchmark for large language models, 2024. URL https://arxiv.org/abs/2312.06281.

Mark D Reckase. 18 multidimensional item response theory. *Handbook of statistics*, 26:607–642, 2006.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

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

Yangjun Ruan, Chris J. Maddison, and Tatsunori Hashimoto. Observational Scaling Laws and the Predictability of Language Model Performance, July 2024.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Joseph Sill. Monotonic networks. *Advances in neural information processing systems*, 10, 1997.

Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. *arXiv preprint arXiv:2310.16049*, 2023.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022. URL https://arxiv.org/abs/2210.09261.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

A Yates. *Multivariate exploratory data analysis: A perspective on exploratory factor analysis*. State University of New York Press, 1987.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A LIMITATIONS

From the predictive side of `Sloth`, we believe that the main limitation is that the model is still dependent, most of the time, on seeing data from at least one LLM from the LLM family of interest. Moreover, we train the link function in the best version of `Sloth` using flexible neural networks, which can interpolate data very well, but have no guarantee of extrapolation when the (hypothetical) LLM of interest is very different from others in the training set. From the interpretability side, we only understand the identifiability problems, such as transformations in the latent space, that can arise in the most simple case of `Sloth`. This fact limits our understanding and interpretability of the most advanced versions of the model.

B IDENTIFIABILITY THEOREM PROOF

Theorem 3.2. We start proving that $b = \tilde{b}$. Because σ is invertible, we get

$$\Lambda(XB)^{(i)\top} + b = \tilde{\Lambda}(X\tilde{B})^{(i)\top} + \tilde{b} \text{ for all } i \in [n],$$

and consequently by the standardization of the latent skills

$$\Lambda\left[\frac{1}{n}\sum_{i=1}^{n}(XB)^{(i)\top}\right] + b = \tilde{\Lambda}\left[\frac{1}{n}\sum_{i=1}^{n}(X\tilde{B})^{(i)\top}\right] + \tilde{b} \Rightarrow b = \tilde{b}.$$

Now, we prove that $\tilde{\Lambda} = \Lambda M$. Given that $b = \tilde{b}$, we have

$$\Lambda(XB)^{(i)\top} = \tilde{\Lambda}(X\tilde{B})^{(i)\top} \text{ for all } i \in [n],$$

and consequently by the standardization of the latent skills

$$\Lambda\left[\frac{1}{n}\sum_{i=1}^{n}(XB)^{(i)\top}(XB)^{(i)}\right]\Lambda^\top = \tilde{\Lambda}\left[\frac{1}{n}\sum_{i=1}^{n}(X\tilde{B})^{(i)\top}(X\tilde{B})^{(i)}\right]\tilde{\Lambda}^\top \Rightarrow \Lambda\Psi\Lambda^\top = \tilde{\Lambda}\Psi\tilde{\Lambda}^\top.$$

By Cholesky's decomposition, we can write $\Psi = LL^\top$, for a lower triangular matrix L. If we define $\Lambda' \triangleq \Lambda L$ and $\tilde{\Lambda}' \triangleq \tilde{\Lambda} L$, then

$$\Lambda'\Lambda'^\top = \tilde{\Lambda}'\tilde{\Lambda}'^\top.$$

Because $\text{rank}(\Lambda) = d$, we have that $\text{rank}(\Lambda') = d$ and we claim that $\tilde{\Lambda}' = \Lambda' U$ for an orthogonal matrix $U \in \mathbf{R}^{d\times d}$. To see that, first, realize that

- $\text{rank}(\Lambda') = \text{rank}(\Lambda'\Lambda'^\top) = \text{rank}(\tilde{\Lambda}'\tilde{\Lambda}'^\top) = \text{rank}(\tilde{\Lambda}')$. We see this by realizing that the null spaces of $\Lambda'^\top$ and $\Lambda'\Lambda'^\top$ are the same: for an arbitrary vector z, $\Lambda'^\top z = 0 \Rightarrow \Lambda'\Lambda'^\top z = 0$ and $\Lambda'\Lambda'^\top z = 0 \Rightarrow \Lambda'^\top\Lambda'\Lambda'^\top z = 0 \Rightarrow \Lambda'^\top z = 0$, where the last implication follows from the assumption that $\Lambda'^\top\Lambda'$ is full rank ($\text{rank}(\Lambda') = d$). Because the null spaces of $\Lambda'^\top$ and $\Lambda'\Lambda'^\top$ are the same, their ranks should be the same as well. The same reasoning applies to $\tilde{\Lambda}'\tilde{\Lambda}'^\top$ and $\tilde{\Lambda}'$, proving this intermediate result.
- Because Λ' and $\Lambda'\Lambda'^\top$ have the same rank, the column space of these two matrices must be the same as the columns of $\Lambda'\Lambda'^\top$ are given by linear combinations of columns of Λ'. Same for $\tilde{\Lambda}'$ and $\tilde{\Lambda}'\tilde{\Lambda}'^\top$. Consequently, the column spaces of Λ' and $\tilde{\Lambda}'$ are the same.

Because the column spaces of Λ' and $\tilde{\Lambda}'$ are the same, there must be an invertible matrix U such that $\tilde{\Lambda}' = \Lambda' U$. But then

$$\Lambda'\Lambda'^\top = \tilde{\Lambda}'\tilde{\Lambda}'^\top = \Lambda' UU^\top\Lambda'^\top \Rightarrow \Lambda'^\top\Lambda'\Lambda'^\top\Lambda' = \Lambda'^\top\Lambda' UU^\top\Lambda'^\top\Lambda' \Rightarrow UU^\top = I$$

and

$$UU^\top = I \Rightarrow U^\top UU^\top U = U^\top U \Rightarrow U^\top U = I$$

Because $\tilde{\Lambda}' = \Lambda' U$, we have that

$$\tilde{\Lambda} L = \Lambda L U \Rightarrow \tilde{\Lambda} = \Lambda L U L^{-1} = \Lambda M.$$

If $\Psi = I_d$, then $L = I_d$ and $M = U$.

Finally, we prove that $\tilde{B} = B(M^\top)^{-1}$. From previous considerations, we can write

$$\begin{aligned}
\Lambda B^\top X^\top = \Lambda M \tilde{B}^\top X^\top &\Rightarrow \Lambda^\top \Lambda B^\top X^\top = \Lambda^\top \Lambda M \tilde{B}^\top X^\top \\
&\overset{\text{rank}(\Lambda)=d}{\Rightarrow} XB = X\tilde{B}M^\top \\
&\Rightarrow X^\top XB = X^\top X\tilde{B}M^\top \\
&\overset{\text{rank}(X)=p}{\Rightarrow} B = \tilde{B}M^\top \\
&\Rightarrow \tilde{B} = B(M^\top)^{-1}
\end{aligned}$$

If $\Psi = I_d$, then $L = I_d$ and $(M^\top)^{-1} = U$. □

C Connections with factor analysis

`Sloth` is heavily inspired by (exploratory) factor analysis models. Factor analysis is a statistical technique used to identify underlying relationships between observed variables by reducing the data's dimensionality (Bishop & Nasrabadi, 2006; Chen et al., 2019). It assumes that multiple observed variables are influenced by a smaller number of unobserved/latent variables called factors (skills $\theta^{(i)}$, in our case). These factors help explain the correlations among the observed variables. The method aims to model the observed variability and reveal the structure behind the data by estimating the factor loadings (Λ, in our case). The classical model assumes

$$Y_i = \Lambda\theta_i + b + \varepsilon_i,$$

where Y_i is a vector of variables of interest and ε_i is an error term. There are versions for the factor model in which a nonlinear model for Y_i is assumed, *e.g.*, in item response theory (IRT) (Reckase, 2006; Chen et al., 2019). It is usually the case that θ_i is estimated using a random effects model, *i.e.*, practitioners place a prior distribution on θ_i. In our work, we assume θ_i is given by a function of observable covariates and a family-specific intercept, which is fitted using data.

D Motivating the interaction term in Sloth

As shown in Section 3, we include an interaction term between $\log(s)$ and $\log(t)$. In the first place, we consider this as a natural extension of the model that depends on s and t only through FLOPs, since we recover that formulation if $\beta_{k1} = \beta_{k2}$ and $\beta_{k3} = 0$. In the second place, we believe that the dependence of benchmark performances on $\log(s)$ depends on $\log(t)$ (and possibly vice-versa). To motivate this idea we show some plots for two benchmarks we use: MMLU-PRO and BBH. For these plots, we only keep families with a higher number of models. First, realize that in both Figures 8 and 9, the performance within families in the middle plot can be well approximated by a line. Also, the slope of this line has a strong relation with the number of tokens in the last plots. For example, Pythia was trained in a small dataset and its (hypothetical) slopes on the second plot are almost zero in both cases. On the other hand, Qwen2 was trained on more data and its (hypothetical) slope on the middle plots is high. Certainly, this relationship does not always exist, but adding an interaction term in our model helps us to leverage this pattern when it exists.

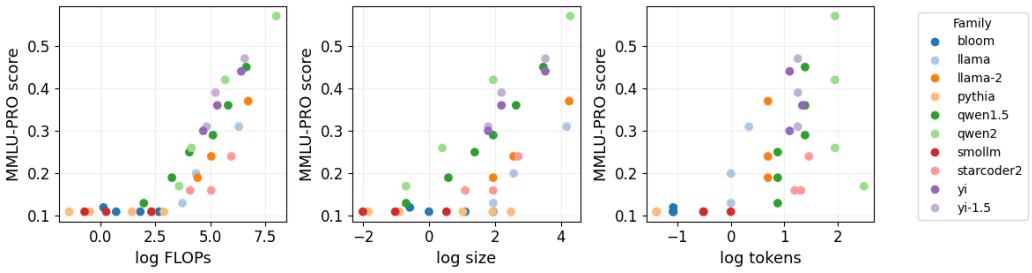


Figure 8: Inputs vs MMLU-PRO scores.

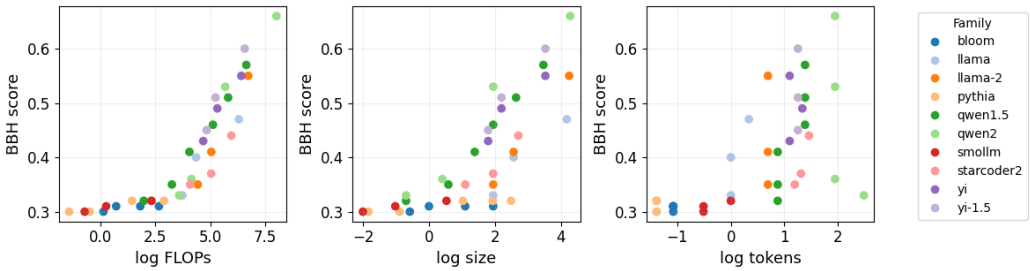


Figure 9: Inputs vs BBH scores.

E PCA APPROACH FORMULATION

We follow the ideas of Ruan et al. (2024) as closely as possible to create a prediction method. Moreover, we follow their code[5] and apply PCA with the same set of hyperparameters. Assume we have a matrix of scores $Y \in [0,1]^{n \times J}$ in which columns represent benchmarks and each row represents a language model. We compute the covariance matrix of benchmark scores using Y and then compute its eigenvector matrix U, where the columns give the ordered eigenvectors (from the highest eigenvalue to the lowest one). To reduce the dimensions of Y, we keep only the first d columns of YU, resulting in $\tilde{Y} \in \mathbf{R}^{n \times d}$. For each column of $\tilde{Y}$ (principal components; PCs), we train a linear regression model using logFLOPs as the covariate; in this case, either the intercept or both the intercept and slope can be family-dependent. At test time, we predict the PCs of a held-out model and then go back to the original coordinate axis to obtain the final predictions by computing $\sum_{j=1}^{d} \hat{\mathrm{PC}}_j U_{\cdot,j} \in \mathbf{R}^J$.

[5]See https://github.com/ryoungj/ObsScaling.

F MODELS IN OUR DATASET

Table 1 gives a detailed view of our dataset. The column "Family" considers that base and instruct models are from different families, while "OriginalFamily" does not. The column "TestFamily" tells if that specific family is considered to be part of the test set in our experiment while the remaining three columns tell if the data is available for these specific benchmarks. For the EQ data, only the following models are available 'gemma-7b-it', 'llama-2-13b-chat', 'llama-2-70b-chat', 'llama-2-7b-chat', 'meta-llama-3-70b-instruct', 'meta-llama-3-8b-instruct', 'qwen1.5-1.8b-chat', 'qwen1.5-14b-chat', 'qwen1.5-32b-chat', 'qwen1.5-4b-chat', 'qwen1.5-7b-chat', 'yi-1.5-34b-chat', 'yi-1.5-6b-chat', 'yi-1.5-9b-chat', 'yi-34b-chat'.

	Model	Family	OriginalFamily	TestFamily	Leader-board1	Leader-board2	HumanEval
0	bloom	bloom	bloom	True	True	False	True
1	bloom-1b1	bloom	bloom	True	True	True	True
2	bloom-3b	bloom	bloom	True	True	True	True
3	bloom-560m	bloom	bloom	True	True	True	True
4	bloom-7b1	bloom	bloom	True	True	True	True
5	blossom-v5.1-34b	blossom-v5.1	yi-1.5	False	True	True	False
6	blossom-v5.1-9b	blossom-v5.1	yi-1.5	False	False	True	False
7	codegen-16b-nl	codegen-nl	codegen	True	True	False	True
8	codegen-6b-nl	codegen-nl	codegen	True	True	False	True
9	codellama-13b	codellama	codellama	True	True	False	True
10	codellama-34b	codellama	codellama	True	True	False	True
11	codellama-70b	codellama	codellama	True	True	False	True
12	codellama-7b	codellama	codellama	True	True	False	True
13	deepseek-coder-1.3b-base	deepseek-coder-base	deepseek-coder	True	True	False	True
14	deepseek-coder-33b-base	deepseek-coder-base	deepseek-coder	True	True	False	True
15	deepseek-coder-6.7b-base	deepseek-coder-base	deepseek-coder	True	True	False	True
16	dolly-v2-12b	dolly-v2	pythia	True	True	True	True
17	dolly-v2-3b	dolly-v2	pythia	True	False	True	False
18	dolly-v2-7b	dolly-v2	pythia	True	True	True	False
19	dolphin-2.9.1-yi-1.5-34b	dolphin-2.9.1-yi-1.5	yi-1.5	True	True	True	False
20	dolphin-2.9.1-yi-1.5-9b	dolphin-2.9.1-yi-1.5	yi-1.5	True	True	True	False
21	dolphin-2.9.2-qwen2-72b	dolphin-2.9.2-qwen2	qwen2	True	False	True	False
22	dolphin-2.9.2-qwen2-7b	dolphin-2.9.2-qwen2	qwen2	True	False	True	False
23	falcon-180b	falcon	falcon	True	True	False	False
24	falcon-40b	falcon	falcon	True	True	True	False
25	falcon-40b-instruct	falcon-instruct	falcon	True	False	True	False
26	falcon-7b	falcon	falcon	True	True	True	False
27	falcon-7b-instruct	falcon-instruct	falcon	True	True	True	False
28	gemma-2-2b	gemma-2	gemma-2	True	False	True	False
29	gemma-2-2b-it	gemma-2-it	gemma-2	True	False	True	False
30	gemma-2-9b	gemma-2	gemma-2	True	False	True	False
31	gemma-2-9b-it	gemma-2-it	gemma-2	True	False	True	False
32	gemma-2b	gemma	gemma	True	True	True	True
33	gemma-2b-it	gemma-it	gemma	True	True	True	True
34	gemma-7b	gemma	gemma	True	True	True	True
35	gemma-7b-it	gemma-it	gemma	True	True	True	True
36	gpt-j-6b	gpt-j-neo-neox	gpt-neo/j	True	True	False	True
37	gpt-neo-1.3b	gpt-j-neo-neox	gpt-neo/j	True	True	True	True
38	gpt-neo-125m	gpt-j-neo-neox	gpt-neo/j	True	True	False	True
39	gpt-neo-2.7b	gpt-j-neo-neox	gpt-neo/j	True	True	True	True
40	gpt-neox-20b	gpt-j-neo-neox	gpt-neo/j	True	True	False	True
41	internlm2-20b	internlm2	internlm2	True	True	False	False
42	internlm2-7b	internlm2	internlm2	True	True	False	False
43	llama-13b	llama	llama	False	True	True	True
44	llama-2-13b	llama-2	llama-2	False	True	True	True
45	llama-2-13b-chat	llama-2-chat	llama-2	False	True	True	True
46	llama-2-70b	llama-2	llama-2	False	True	True	True
47	llama-2-70b-chat	llama-2-chat	llama-2	False	True	True	True
48	llama-2-7b	llama-2	llama-2	False	True	True	True
49	llama-2-7b-chat	llama-2-chat	llama-2	False	True	True	True
50	llama-3-sauerkrautlm-70b-instruct	llama-3-sauerkrautlm-instruct	meta-llama-3	True	False	True	False
51	llama-3-sauerkrautlm-8b-instruct	llama-3-sauerkrautlm-instruct	meta-llama-3	True	True	True	False

52	llama-30b	llama	llama	False	True	False	True
53	llama-65b	llama	llama	False	True	True	True
54	llama-7b	llama	llama	False	True	True	True
55	meta-llama-3-70b	meta-llama-3	meta-llama-3	True	True	True	True
56	meta-llama-3-70b-instruct	meta-llama-3-instruct	meta-llama-3	True	True	True	True
57	meta-llama-3-8b	meta-llama-3	meta-llama-3	True	True	True	True
58	meta-llama-3-8b-instruct	meta-llama-3-instruct	meta-llama-3	True	True	True	True
59	mpt-30b	mpt	mpt	True	True	False	True
60	mpt-30b-chat	mpt-chat	mpt	True	True	False	False
61	mpt-30b-instruct	mpt-instruct	mpt	True	True	False	False
62	mpt-7b	mpt	mpt	True	True	False	True
63	mpt-7b-chat	mpt-chat	mpt	True	True	False	False
64	mpt-7b-instruct	mpt-instruct	mpt	True	True	False	False
65	olmo-1b	olmo	olmo	True	True	True	False
66	olmo-7b	olmo	olmo	True	True	True	False
67	open_llama_13b	open_llama_	openllama	False	True	False	False
68	open_llama_3b	open_llama_	openllama	False	True	False	False
69	open_llama_3b_v2	open_llama__v2	openllamav2	False	True	False	False
70	open_llama_7b	open_llama_	openllama	False	True	False	False
71	open_llama_7b_v2	open_llama__v2	openllamav2	False	True	False	False
72	openhermes-13b	openhermes	llama-2	False	True	True	False
73	openhermes-7b	openhermes	llama-2	False	True	True	False
74	opt-1.3b	opt	opt	True	True	True	True
75	opt-125m	opt	opt	True	True	False	True
76	opt-13b	opt	opt	True	True	False	True
77	opt-2.7b	opt	opt	True	True	False	True
78	opt-30b	opt	opt	True	True	True	True
79	opt-350m	opt	opt	True	True	False	True
80	opt-6.7b	opt	opt	True	True	False	True
81	opt-66b	opt	opt	True	True	False	True
82	orca-2-13b	orca-2	llama-2	False	True	True	False
83	orca-2-7b	orca-2	llama-2	False	True	True	False
84	orca_mini_v3_13b	orca_mini_v3_	llama-2	False	True	True	False
85	orca_mini_v3_70b	orca_mini_v3_	llama-2	False	False	True	False
86	orca_mini_v3_7b	orca_mini_v3_	llama-2	False	True	True	False
87	orca_mini_v7_72b	orca_mini_v7_	qwen2	False	False	True	False
88	orca_mini_v7_7b	orca_mini_v7_	qwen2	False	False	True	False
89	pythia-1.4b	pythia	pythia	True	True	False	True
90	pythia-12b	pythia	pythia	True	True	True	True
91	pythia-160m	pythia	pythia	True	True	True	True
92	pythia-1b	pythia	pythia	True	True	False	True
93	pythia-2.8b	pythia	pythia	True	True	True	True
94	pythia-410m	pythia	pythia	True	True	True	True
95	pythia-6.9b	pythia	pythia	True	True	True	True
96	pythia-70m	pythia	pythia	True	True	False	True
97	qwen-14b	qwen	qwen	False	True	False	True
98	qwen-72b	qwen	qwen	False	True	False	True
99	qwen-7b	qwen	qwen	False	True	False	True
100	qwen1.5-0.5b	qwen1.5	qwen1.5	False	True	True	True
101	qwen1.5-0.5b-chat	qwen1.5-chat	qwen1.5	False	True	True	False
102	qwen1.5-1.8b	qwen1.5	qwen1.5	False	True	True	True
103	qwen1.5-1.8b-chat	qwen1.5-chat	qwen1.5	False	True	True	False
104	qwen1.5-14b	qwen1.5	qwen1.5	False	True	True	True
105	qwen1.5-14b-chat	qwen1.5-chat	qwen1.5	False	True	True	False
106	qwen1.5-32b	qwen1.5	qwen1.5	False	True	True	True
107	qwen1.5-32b-chat	qwen1.5-chat	qwen1.5	False	True	True	False
108	qwen1.5-4b	qwen1.5	qwen1.5	False	True	True	True
109	qwen1.5-4b-chat	qwen1.5-chat	qwen1.5	False	True	True	False
110	qwen1.5-72b	qwen1.5	qwen1.5	False	True	False	True
111	qwen1.5-72b-chat	qwen1.5-chat	qwen1.5	False	True	False	True
112	qwen1.5-7b	qwen1.5	qwen1.5	False	True	True	True
113	qwen1.5-7b-chat	qwen1.5-chat	qwen1.5	False	True	True	False
114	qwen2-0.5b	qwen2	qwen2	True	True	True	False
115	qwen2-0.5b-instruct	qwen2-instruct	qwen2	True	False	True	False
116	qwen2-1.5b	qwen2	qwen2	True	True	True	False
117	qwen2-1.5b-instruct	qwen2-instruct	qwen2	True	False	True	False
118	qwen2-72b	qwen2	qwen2	True	True	True	False
119	qwen2-72b-instruct	qwen2-instruct	qwen2	True	False	True	False
120	qwen2-7b	qwen2	qwen2	True	True	True	False
121	qwen2-7b-instruct	qwen2-instruct	qwen2	True	False	True	False
122	rwkv-4-14b-pile	rwkv-4-pile	rwkv	True	True	False	False
123	rwkv-4-169m-pile	rwkv-4-pile	rwkv	True	True	False	False
124	rwkv-4-1b5-pile	rwkv-4-pile	rwkv	True	True	False	False
125	rwkv-4-3b-pile	rwkv-4-pile	rwkv	True	True	False	False
126	rwkv-4-430m-pile	rwkv-4-pile	rwkv	True	True	False	False
127	rwkv-4-7b-pile	rwkv-4-pile	rwkv	True	True	False	False
128	sauerkrautlm-gemma-2b	sauerkrautlm-gemma	gemma	True	True	True	False

129	sauerkrautlm-gemma-7b	sauerkrautlm-gemma	gemma	True	True	True	False
130	smollm-1.7b	smollm	smollm	True	False	True	False
131	smollm-1.7b-instruct	smollm-instruct	smollm	True	False	True	False
132	smollm-135m	smollm	smollm	True	False	True	False
133	smollm-135m-instruct	smollm-instruct	smollm	True	False	True	False
134	smollm-360m	smollm	smollm	True	False	True	False
135	smollm-360m-instruct	smollm-instruct	smollm	True	False	True	False
136	stablelm-base-alpha-3b	stablelm-base-alpha	stablelm	True	True	False	False
137	stablelm-base-alpha-7b	stablelm-base-alpha	stablelm	True	True	False	False
138	starcoder2-15b	starcoder2	starcoder2	True	True	True	True
139	starcoder2-3b	starcoder2	starcoder2	True	True	True	True
140	starcoder2-7b	starcoder2	starcoder2	True	True	True	True
141	starcoderbase	starcoderbase	starcoder	False	True	False	True
142	starcoderbase-1b	starcoderbase	starcoder	False	True	False	True
143	starcoderbase-3b	starcoderbase	starcoder	False	True	False	True
144	starcoderbase-7b	starcoderbase	starcoder	False	True	False	True
145	wizardlm-13b-v1.0	wizardlm-v1.0	llama-2	False	False	True	False
146	wizardlm-70b-v1.0	wizardlm-v1.0	llama-2	False	False	True	False
147	xglm-1.7b	xglm	xglm	True	True	False	True
148	xglm-4.5b	xglm	xglm	True	True	False	True
149	xglm-564m	xglm	xglm	True	True	False	True
150	xglm-7.5b	xglm	xglm	True	True	False	True
151	yi-1.5-34b	yi-1.5	yi-1.5	True	True	True	False
152	yi-1.5-34b-chat	yi-1.5-chat	yi-1.5	True	True	True	False
153	yi-1.5-6b	yi-1.5	yi-1.5	True	True	True	False
154	yi-1.5-6b-chat	yi-1.5-chat	yi-1.5	True	True	True	False
155	yi-1.5-9b	yi-1.5	yi-1.5	True	True	True	False
156	yi-1.5-9b-chat	yi-1.5-chat	yi-1.5	True	True	True	False
157	yi-34b	yi	yi	False	True	True	True
158	yi-34b-200k	yi-200k	yi-200k	False	True	False	False
159	yi-34b-chat	yi-chat	yi	False	True	False	False
160	yi-6b	yi	yi	False	True	True	True
161	yi-6b-200k	yi-200k	yi-200k	False	True	False	False
162	yi-6b-chat	yi-chat	yi	False	False	True	False
163	yi-9b	yi	yi	False	True	True	False

G EXTRA PERFORMANCE PREDICTION RESULTS

In this section, we present the full versions of the figures presented in the main text.

G.1 TEST FAMILIES HAVE EXACTLY ONE MODEL IN THE TRAINING SET

G.1.1 AVERAGE PREDICTION LOSS ACROSS MODELS

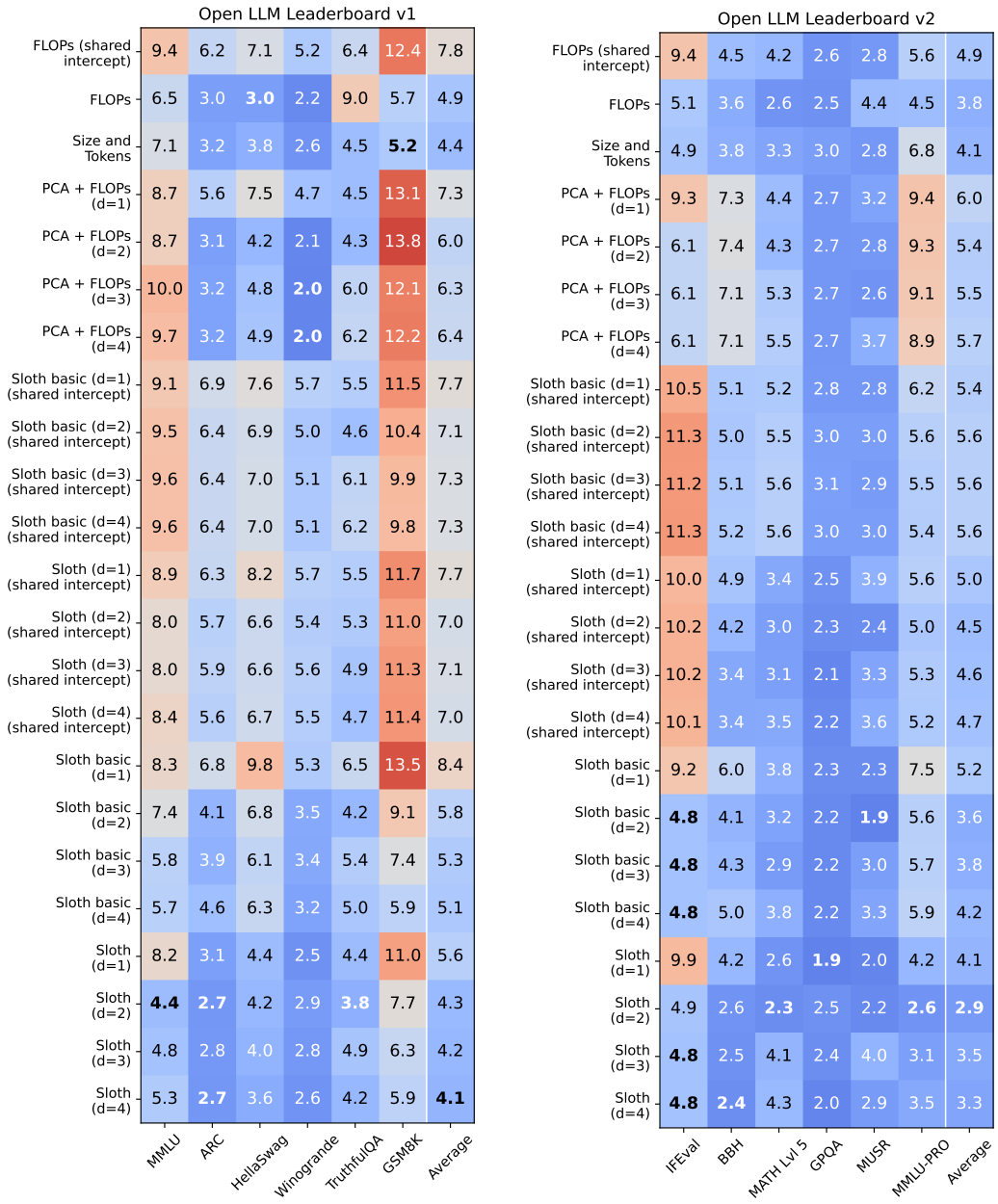


Figure 10: The figure shows the average (across LLM families) mean-absolute-error (MAE) (within a family) for different methods. This is a complete version of Figure 1, in which we include `Sloth` versions with shared intercept.

Open LLM Leaderboard v1/v2

	MMLU	ARC	HellaSwag	Winogrande	TruthfulQA	GSM8K	IFEval	BBH	MATH Lvl 5	GPQA	MUSR	MMLU-PRO	Average
FLOPs (shared intercept)	9.7	5.1	5.4	4.3	6.6	11.4	7.9	5.3	4.8	2.7	2.6	7.5	6.1
FLOPs	5.6	3.9	**2.6**	2.7	7.1	**5.3**	4.2	3.5	2.2	2.2	4.7	5.4	4.1
Size and Tokens	**4.8**	3.7	3.1	2.5	4.8	6.4	4.1	2.2	6.6	2.2	2.6	6.3	4.1
PCA + FLOPs (d=1)	11.5	6.1	7.9	4.8	4.9	14.7	7.2	4.5	3.3	2.1	2.7	5.9	6.3
PCA + FLOPs (d=2)	11.2	5.1	5.2	3.4	5.2	15.3	7.2	4.4	2.9	2.2	2.7	5.9	5.9
PCA + FLOPs (d=3)	11.0	4.5	4.1	2.5	5.3	15.0	5.7	4.4	3.0	2.1	2.8	6.0	5.6
PCA + FLOPs (d=4)	12.0	4.5	4.4	**2.3**	6.3	13.3	**3.9**	5.9	3.1	2.4	2.9	7.5	5.7
Sloth basic (d=1) (shared intercept)	9.9	5.7	5.5	4.7	6.4	11.8	7.9	5.5	5.3	2.6	3.0	8.0	6.4
Sloth basic (d=2) (shared intercept)	9.2	6.1	5.7	3.8	6.8	9.8	7.9	5.7	5.6	2.8	3.1	7.6	6.2
Sloth basic (d=3) (shared intercept)	9.1	6.4	5.8	3.8	6.7	8.2	7.6	6.3	6.3	3.0	3.4	7.8	6.2
Sloth basic (d=4) (shared intercept)	9.1	6.4	5.8	3.8	6.8	8.3	7.6	6.3	6.4	3.3	3.7	7.8	6.3
Sloth (d=1) (shared intercept)	8.0	5.2	6.7	4.7	5.9	10.7	7.8	4.5	3.2	2.3	3.1	6.7	5.7
Sloth (d=2) (shared intercept)	8.2	4.7	4.9	3.8	5.2	9.9	8.1	4.3	10.5	2.6	2.4	7.5	6.0
Sloth (d=3) (shared intercept)	7.4	4.7	4.9	4.1	5.6	9.8	8.6	4.0	3.2	2.4	2.2	6.9	5.3
Sloth (d=4) (shared intercept)	8.0	5.0	4.8	4.2	5.4	9.4	8.1	3.8	3.0	2.2	2.2	6.0	5.2
Sloth basic (d=1)	11.3	7.0	7.7	5.6	6.6	15.9	8.6	5.0	3.0	2.3	2.4	6.7	6.8
Sloth basic (d=2)	7.9	5.8	6.0	3.9	4.4	8.9	8.4	3.3	2.0	2.2	1.9	4.7	4.9
Sloth basic (d=3)	7.5	5.8	6.1	3.8	4.6	8.6	4.0	3.4	2.3	1.9	2.2	4.3	4.5
Sloth basic (d=4)	6.7	6.0	6.2	3.8	8.4	7.1	**3.9**	4.0	2.3	**1.8**	4.1	4.3	4.9
Sloth (d=1)	8.6	5.1	7.2	5.3	**4.3**	12.3	6.7	2.8	2.9	2.0	1.7	4.4	5.3
Sloth (d=2)	5.8	3.8	5.0	3.3	5.1	7.5	7.2	2.6	2.2	2.2	1.9	3.3	4.2
Sloth (d=3)	5.4	**3.0**	4.4	3.1	5.0	6.4	4.7	2.5	**1.6**	2.2	**1.5**	**2.5**	**3.5**
Sloth (d=4)	5.4	4.0	5.0	2.9	6.5	6.8	4.1	**1.8**	2.8	2.1	3.8	3.8	4.1

Figure 11: The figure shows the average (across LLM families) mean-absolute-error (MAE) (within a family) for different methods when fitting only one scaling law for both leaderboards.

G.1.2 FAMILY-SPECIFIC PREDICTION LOSSES

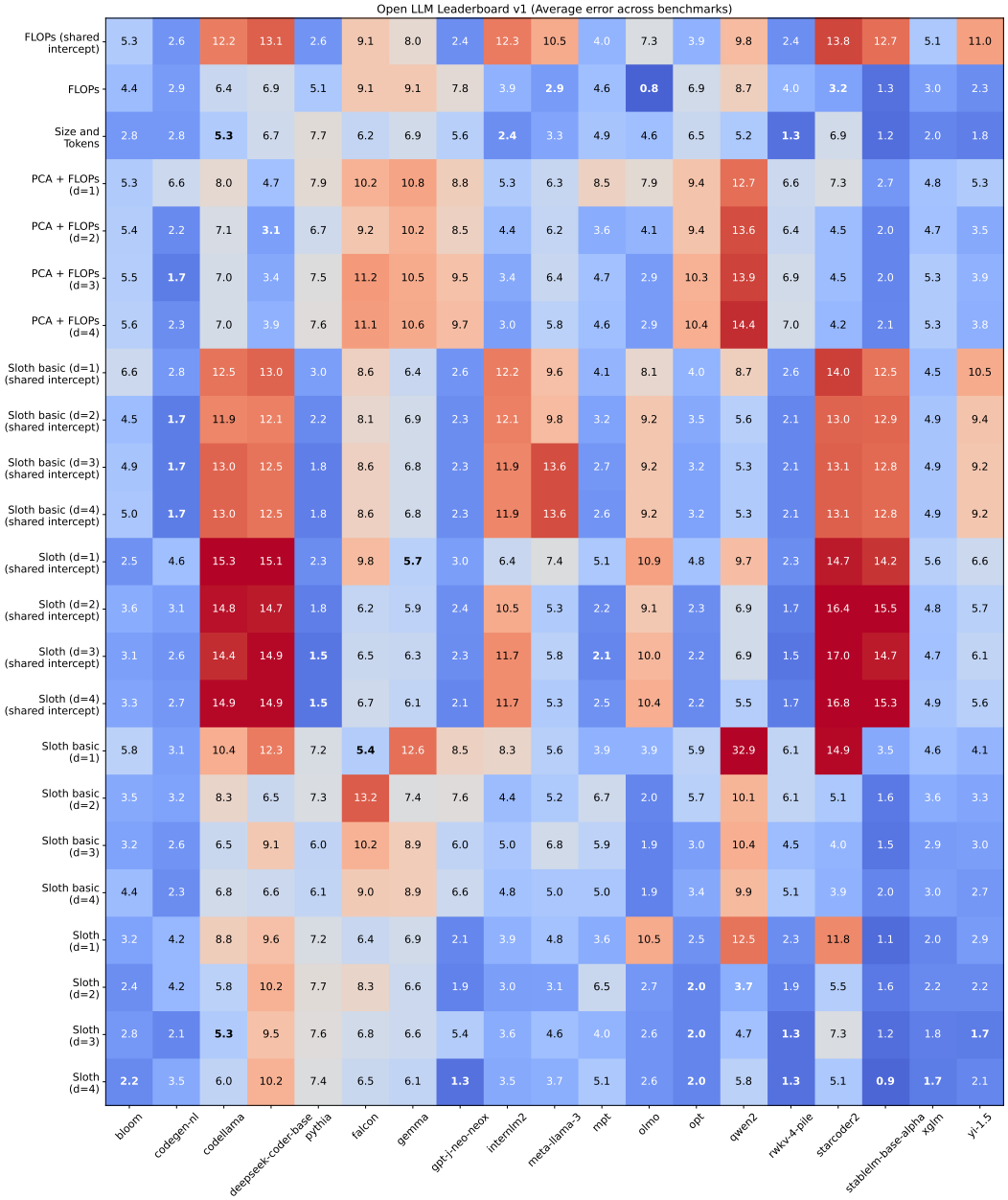


Figure 12: The figure shows the average (across benchmarks) mean-absolute-error (MAE) for each family considering only Open LLM Leaderboard v1.

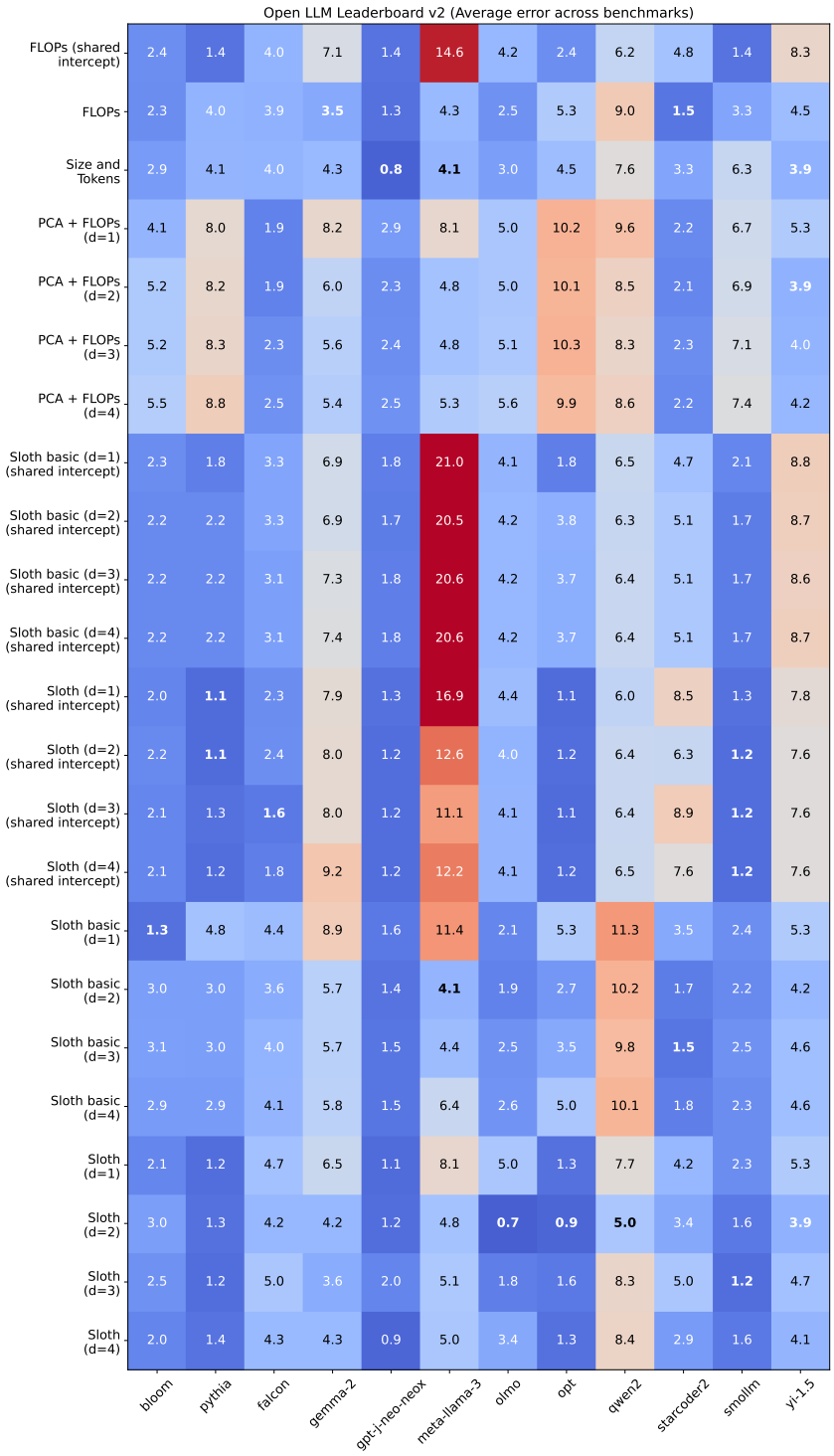


Figure 13: The figure shows the average (across benchmarks) mean-absolute-error (MAE) for each family considering only Open LLM Leaderboard v2.

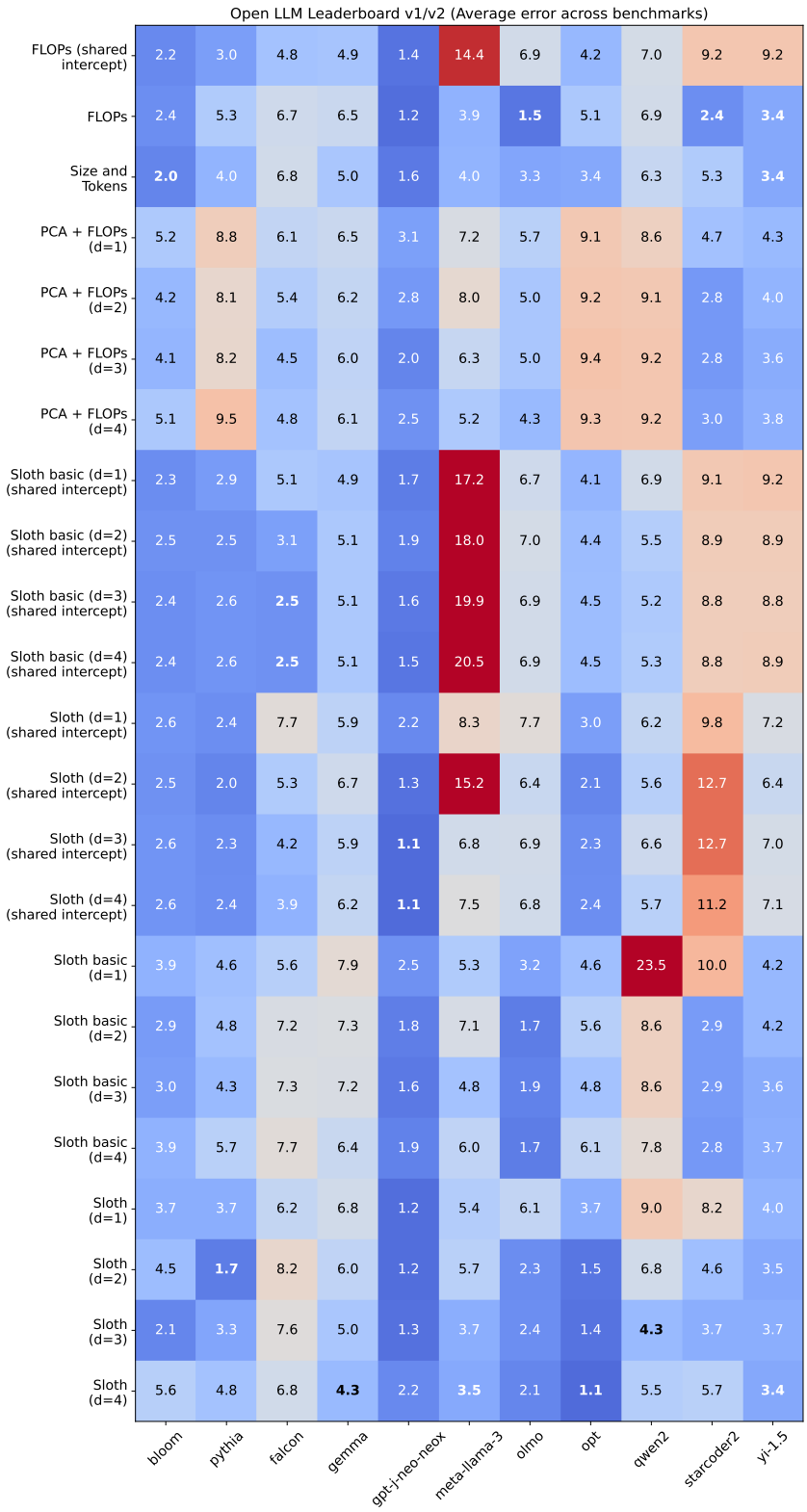


Figure 14: The figure shows the average (across benchmarks) mean-absolute-error (MAE) for each family considering Open LLM Leaderboard v1/v2.

G.2 TEST FAMILIES HAVE EXACTLY TWO MODELS IN THE TRAINING SET

G.2.1 AVERAGE PREDICTION LOSS ACROSS MODELS

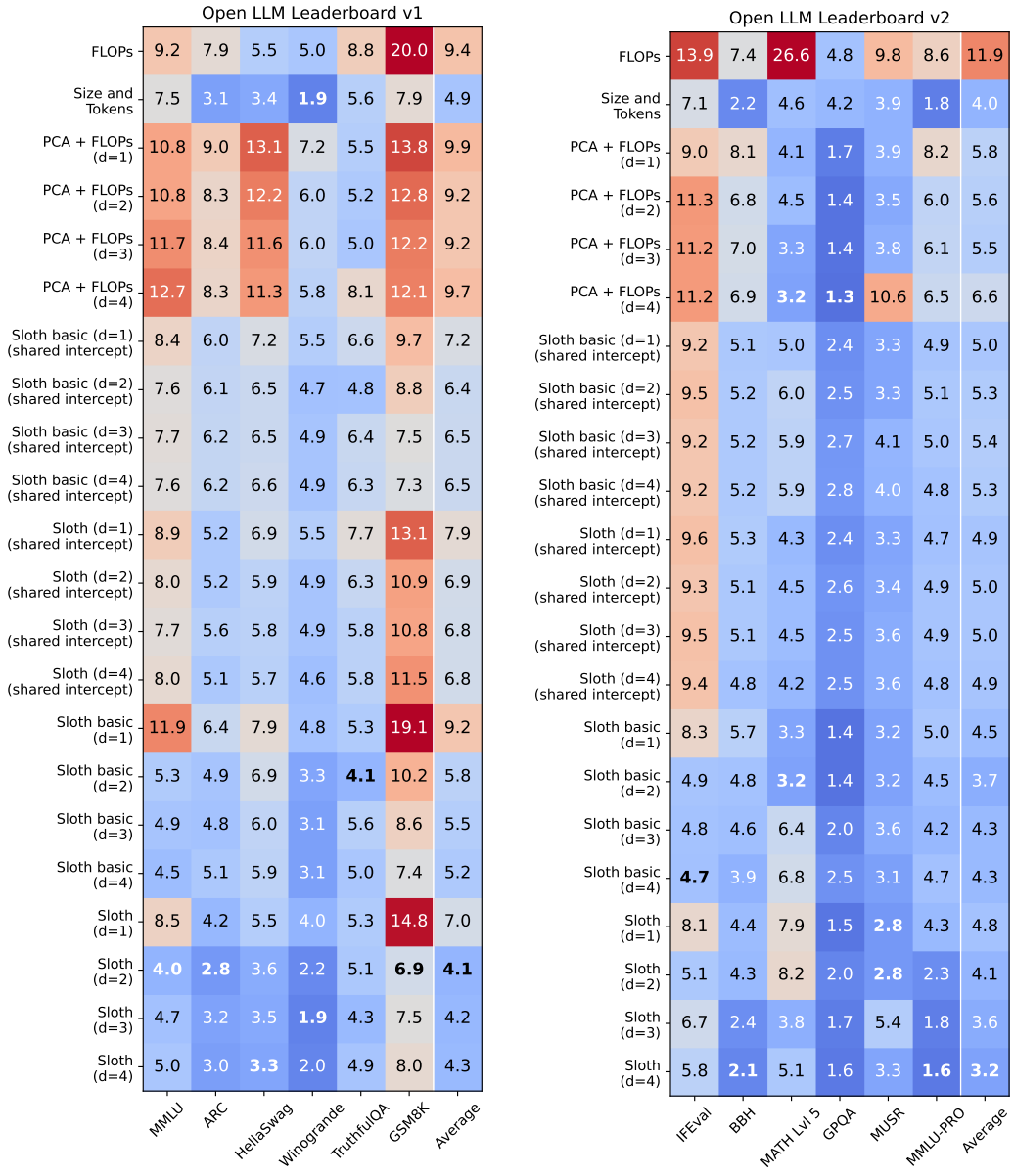


Figure 15: The figure shows the average (across LLM families) mean-absolute-error (MAE) (within a family) for different methods. This is a complete version of Figure 2.

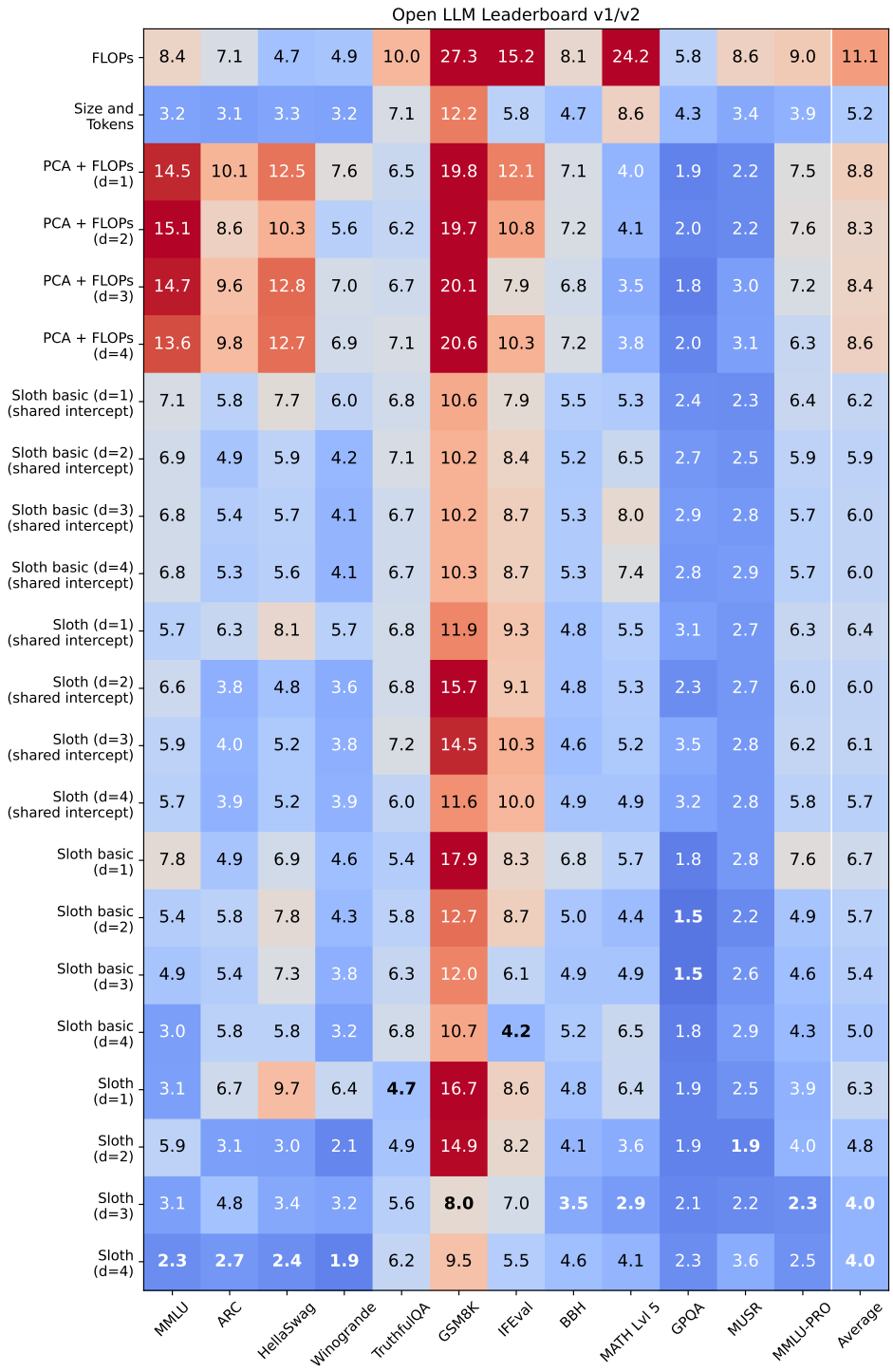


Figure 16: The figure shows the average (across LLM families) mean-absolute-error (MAE) (within a family) for different methods using the intersection of both leaderboards.

G.2.2 FAMILY-SPECIFIC PREDICTION LOSSES

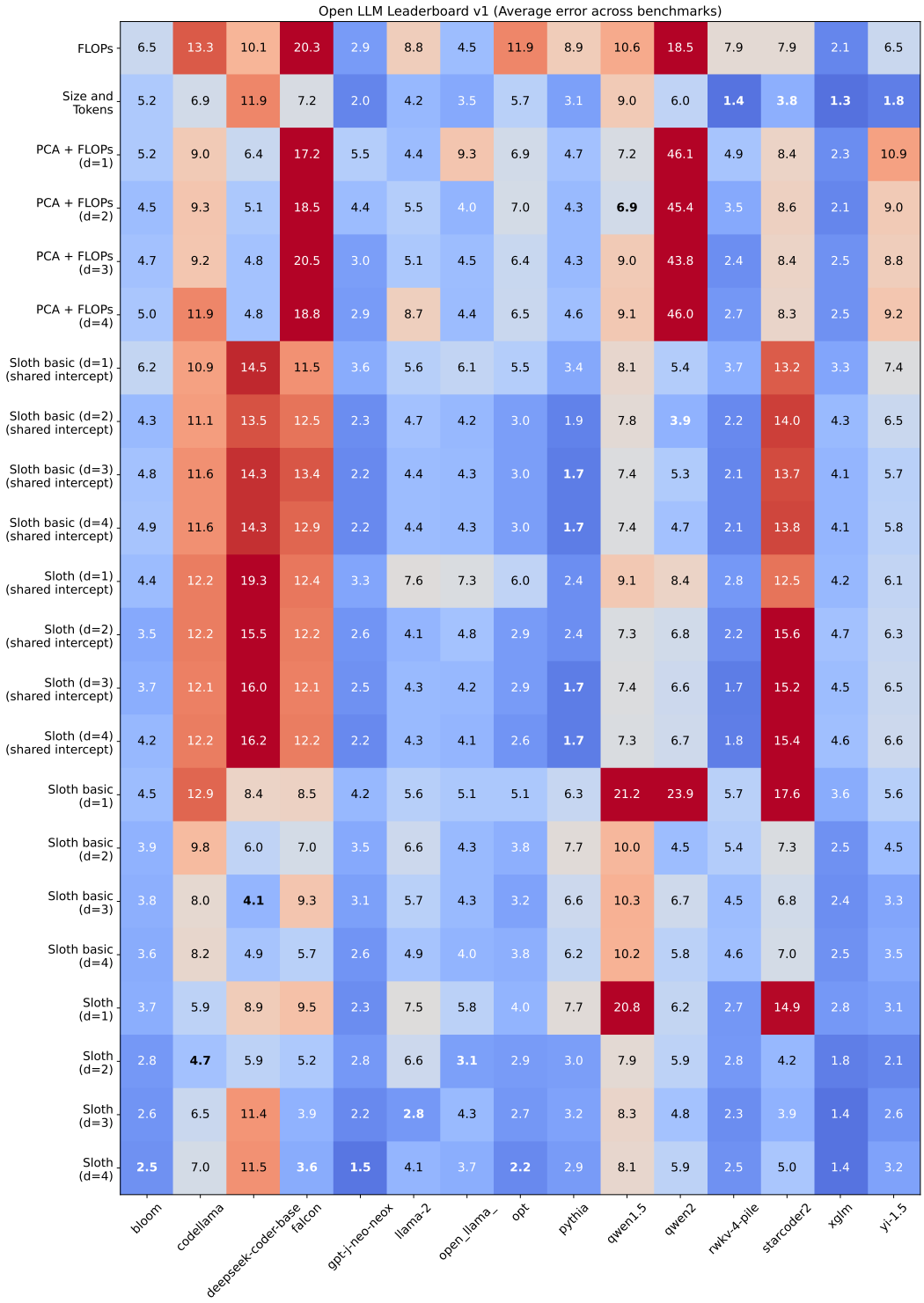


Figure 17: The figure shows the average (across benchmarks) mean-absolute-error (MAE) for each family considering only Open LLM Leaderboard v1.

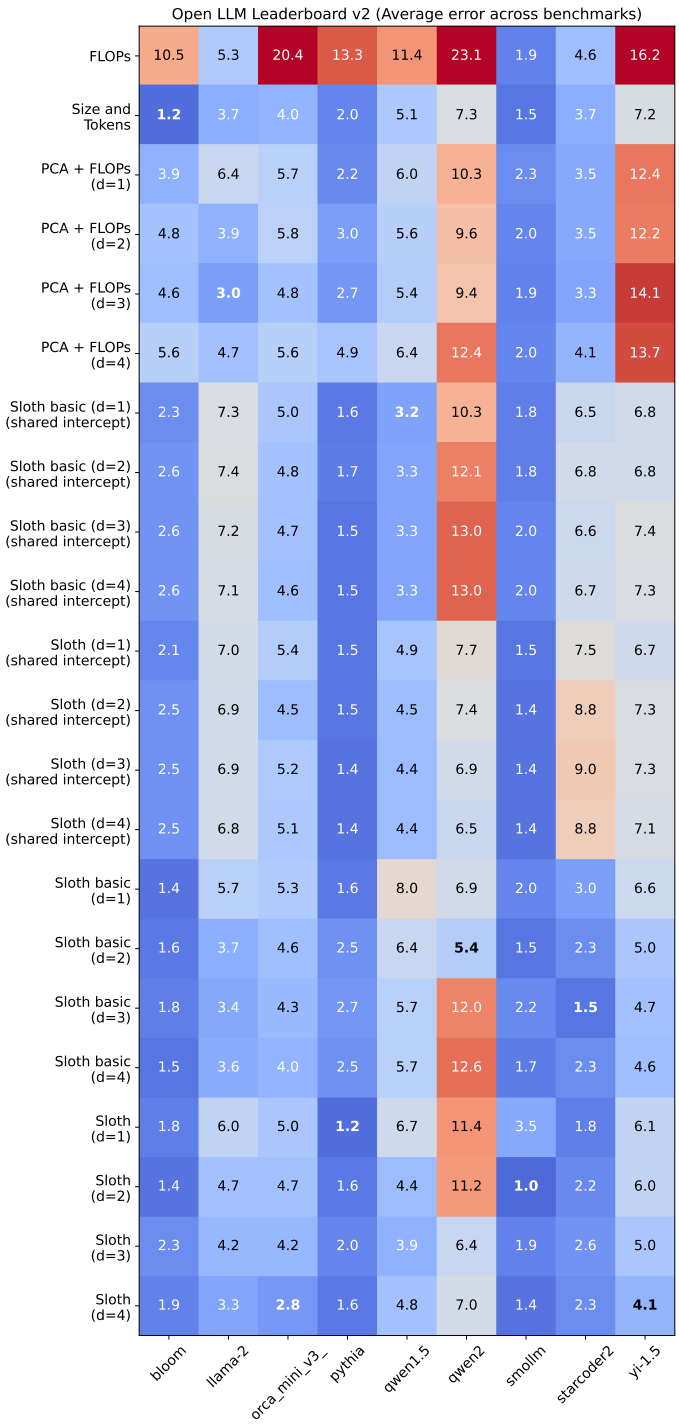


Figure 18: The figure shows the average (across benchmarks) mean-absolute-error (MAE) for each family considering only Open LLM Leaderboard v2.

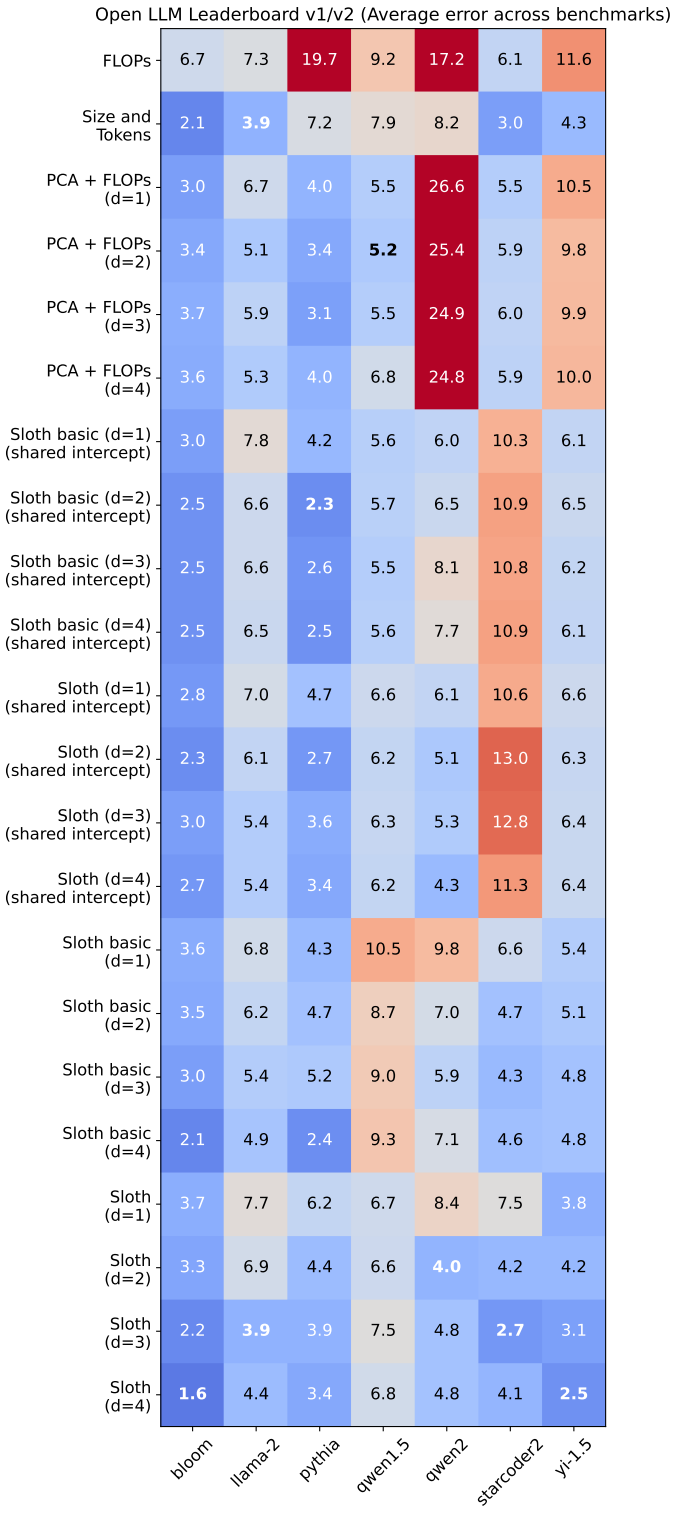


Figure 19: The figure shows the average (across benchmarks) mean-absolute-error (MAE) for each family considering only Open LLM Leaderboard v1/v2.

H EXTRA INTERPRETABILITY RESULTS

H.1 RESULTS FOR $d = 2$

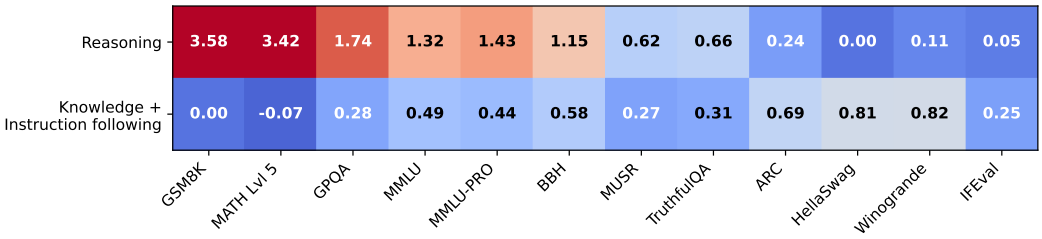


Figure 20: Needed skills for each benchmark. In this figure, we report the estimated loadings Λ and, based on their values, we give them appropriate names.

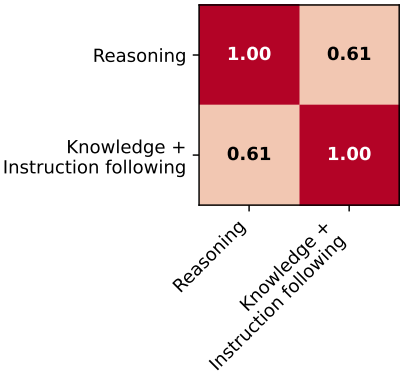


Figure 21: Needed skills for each benchmark. In this figure, we report the estimated loadings Λ and, based on their values, we give them appropriate names.

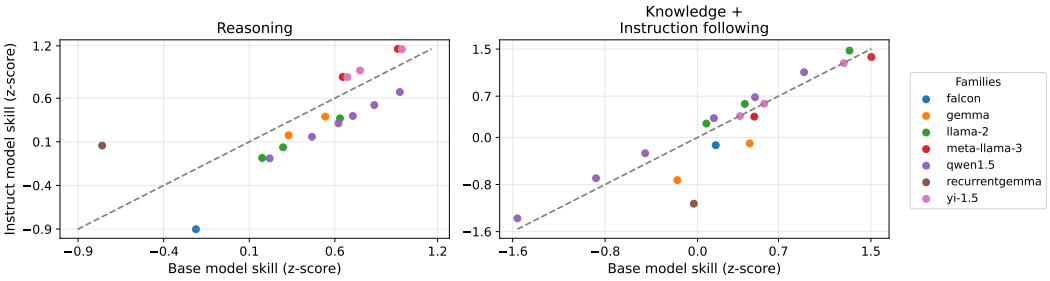


Figure 22: Gains from instruction tuning for different families on three latent skills. Major findings include a large and positive impact on instruction following and a negative impact on mathematical reasoning.

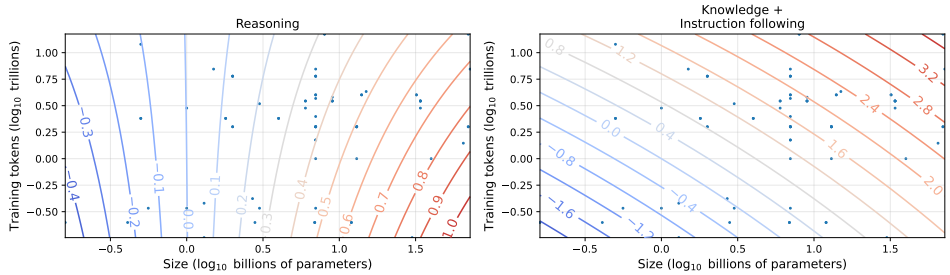


Figure 23: Level curves in producing different latent abilities from parameter count and training tokens.

H.2 Results for $d = 3$

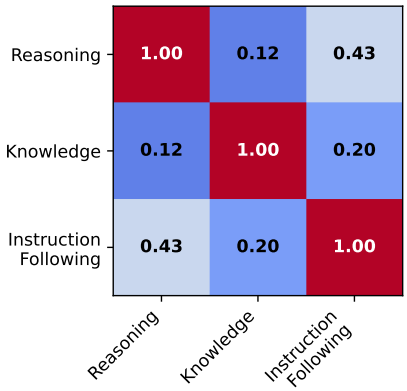


Figure 24: Needed skills for each benchmark. In this figure, we report the estimated loadings Λ and, based on their values, we give them appropriate names.

H.3 Results for $d = 4$

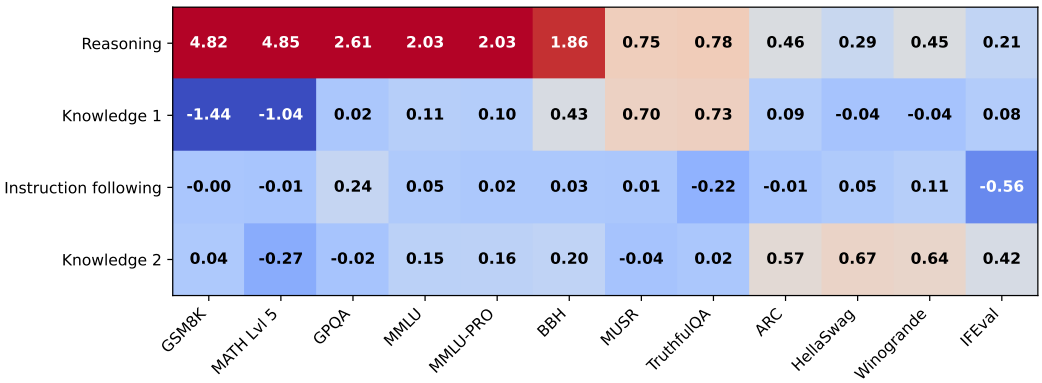


Figure 25: Needed skills for each benchmark. In this figure, we report the estimated loadings Λ and, based on their values, we give them appropriate names.

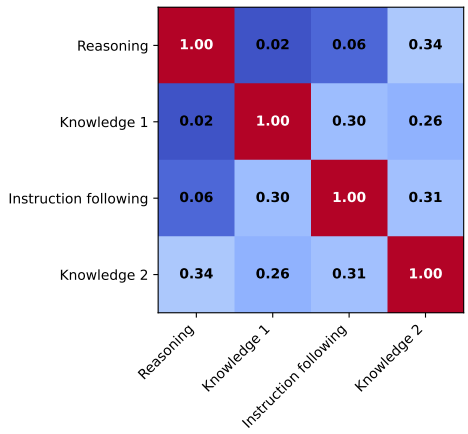


Figure 26: Needed skills for each benchmark. In this figure, we report the estimated loadings Λ and, based on their values, we give them appropriate names.

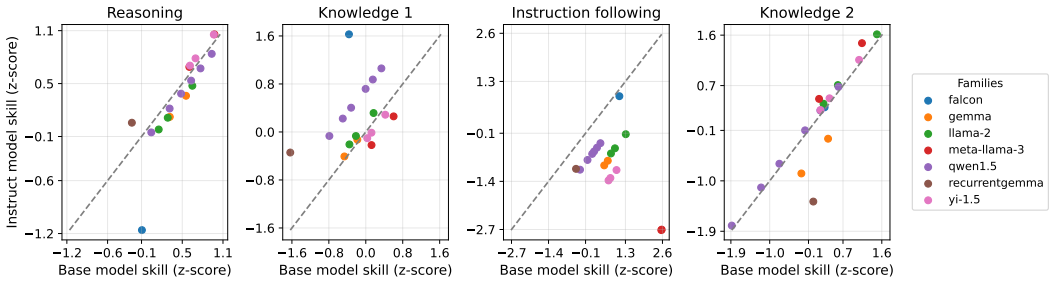


Figure 27: Gains from instruction tuning for different families on three latent skills. Major findings include a large and positive impact on instruction following and a negative impact on mathematical reasoning.

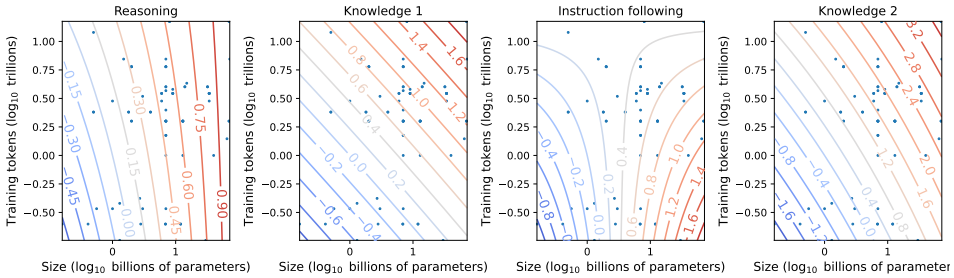
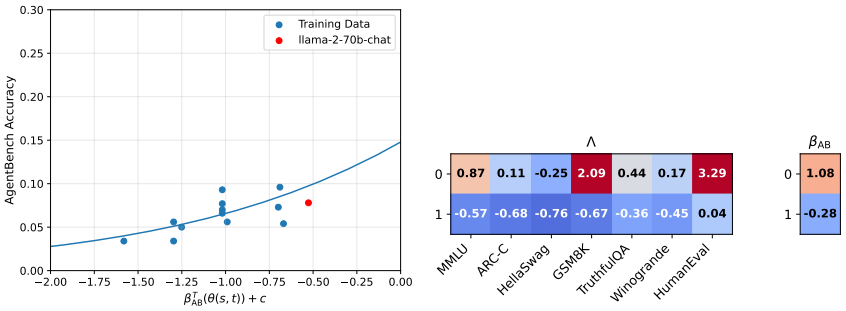


Figure 28: Level curves in producing different latent abilities from parameter count and training tokens.

I EXTRA DOWNSTREAM TASK PLOTS



(a) Predictions (b) Parameter Interpretation

Figure 29: Predicting Agentic Capabilities of Llama-2-70B-chat.

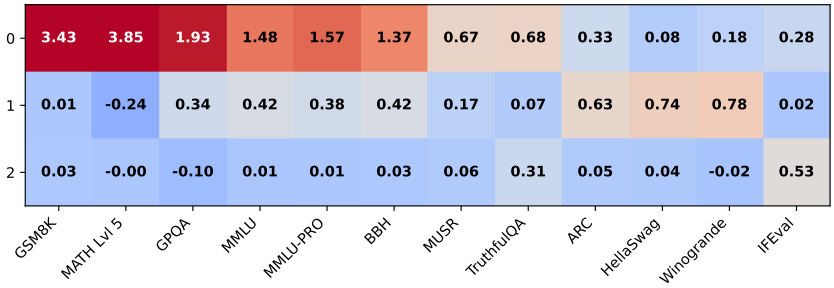


Figure 30: Loadings for downstream prediction tasks.