Explaining Data Mixing Scaling Laws

Rui Dai, and Shuran Zheng

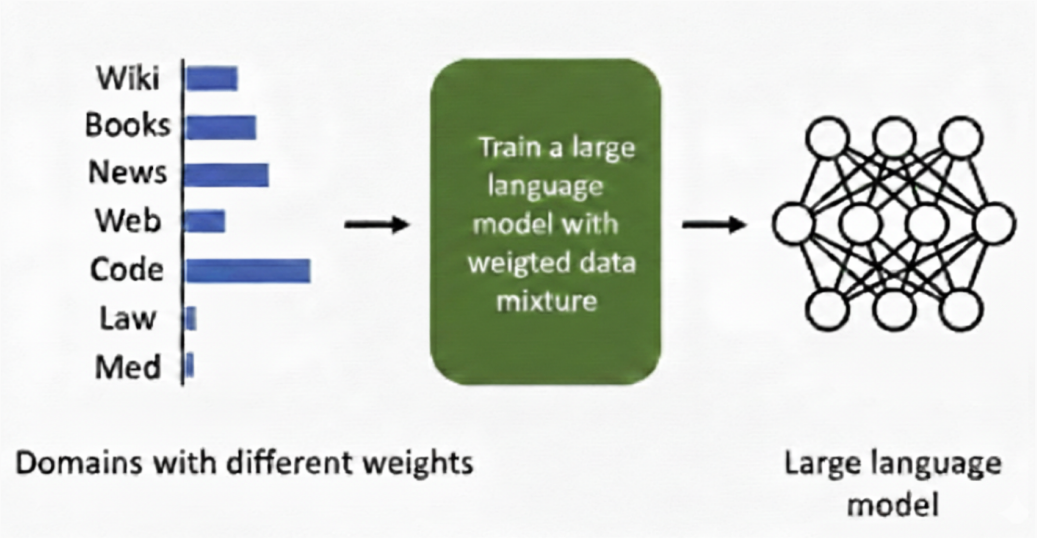
Aug. 2026

The Critical Role of Data Mixing

- Large foundation models are trained on data from multiple domains (e.g., GitHub, books, Wikipedia).

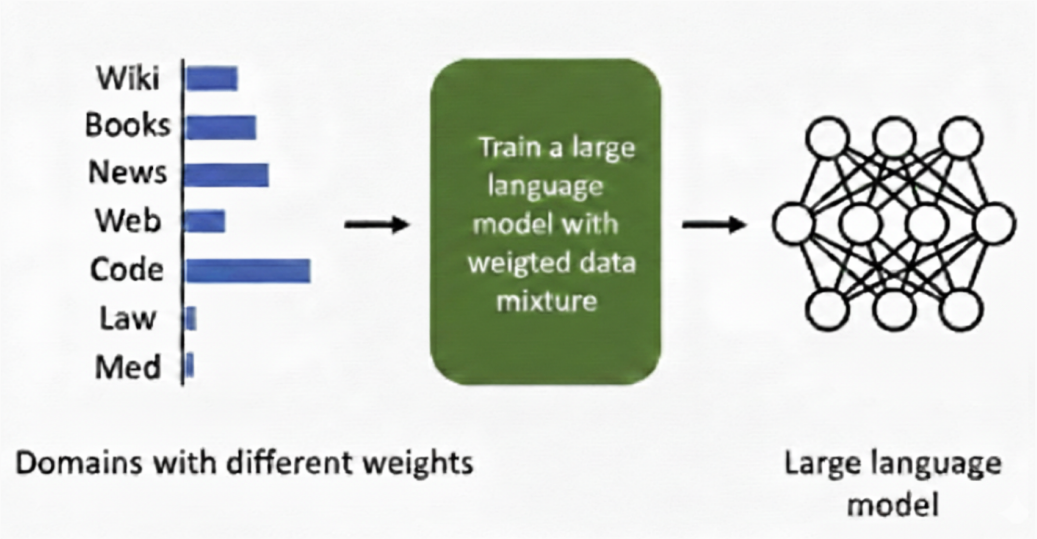
The Critical Role of Data Mixing

- Large foundation models are trained on data from multiple domains (e.g., GitHub, books, Wikipedia).
- **The Data Mixture**: The proportion of each domain used is a critical factor in determining model performance.



The Critical Role of Data Mixing

- Large foundation models are trained on data from multiple domains (e.g., GitHub, books, Wikipedia).
- **The Data Mixture**: The proportion of each domain used is a critical factor in determining model performance.



- **Key question: How do we find the best domain weights to optimize the model's performance?**

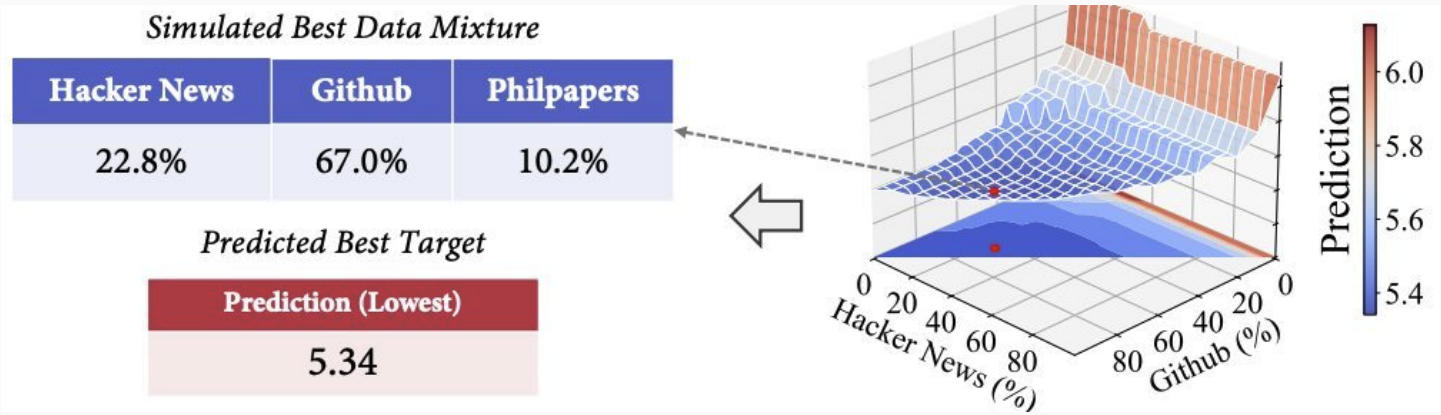
Recent Research: Empirical Data Mixing Laws

Data mixing laws: Various functional forms to model the relationship between weights and domain losses (Ye et al., 2024, Ge et al., 2024, Shukor et al., 2025)

$$\text{Loss} = f(\text{domain weights})$$

$$\text{Optimal weights} = \arg\min f$$

- Sample random weights → train **small proxy models** → fit their losses with hand-picked functions.



Empirical Data Mixing Laws

- **State of the art**:
  - Given a fixed model size N, a fixed data size D
  - L_i: loss on domain i
  - $h_i \in (0, 1)$: domain i's weight

Reference	**Functional Form** $(f_i(h, N, D))$
(Shukor et al., 2025)	$L_i \approx E_i + \left(\sum_{j=1}^{K} C_{ij} h_j^{\gamma_{ij}} \right)^{-1}$
(Ye et al., 2024)	$L_i \approx c_i + k_i \exp\left(\sum_{j=1}^{K} t_{ij} h_j \right)$

Empirical Data Mixing Laws

- **State of the art**:
  - Given a fixed model size N, a fixed data size D
  - L_i: loss on domain i
  - $h_i \in (0, 1)$: domain i's weight

Reference	Functional Form ($f_i(h, N, D)$)
(Shukor et al., 2025)	$L_i \approx E_i + \left(\sum_{j=1}^{K} C_{ij} h_j^{\gamma_{ij}} \right)^{-1}$
(Ye et al., 2024)	$L_i \approx c_i + k_i \exp\left(\sum_{j=1}^{K} t_{ij} h_j \right)$

- **Domain interaction:**
  - L_i depends not only on h_i, but also the weights of other domains h_{-i}
  - This correlation does not exhibit a simple functional form.

Empirical Data Mixing Laws: Limitations

Reference	Functional Form ($f_i(h, N, D)$)
(Shukor et al., 2025)	$L_i \approx E_i + \left(\sum_{j=1}^{K} C_{ij} h_j^{\gamma_{ij}} \right)^{-1}$
(Ye et al., 2024)	$L_i \approx c_i + k_i \exp\left(\sum_{j=1}^{K} t_{ij} h_j \right)$

Domain interaction: L_i depends non-trivially on the weights of other domains

Empirical Data Mixing Laws: Limitations

Reference	Functional Form ($f_i(h, N, D)$)
(Shukor et al., 2025)	$L_i \approx E_i + \left(\sum_{j=1}^{K} C_{ij} h_j^{\gamma_{ij}} \right)^{-1}$
(Ye et al., 2024)	$L_i \approx c_i + k_i \exp\left(\sum_{j=1}^{K} t_{ij} h_j \right)$

Domain interaction: L_i depends non-trivially on the weights of other domains

Limitation: There is **no theoretical understanding** of the underlying mechanics driving domain interaction.

- Do they generalize to larger scales?
- Do they generalize to different datasets?
- Map the predicted domain loss to downstream task performance?

This work

We propose a theoretical model explaining the observed empirical data mixing laws and the domain interaction:

- **Key Factors:** Domain competition + finite data noise.
- **Cross-Scale Extrapolation**: Our framework predicts highly effective mixtures for large, unseen model and dataset sizes using parameters fitted on smaller ones.

Outline

Problem Description

Extended Quantization Model: Capacity Competition

Extended Linear Model: Noise Driven by Data Amount

Future Work

Problem Description

Formalizing the Data Mixing Problem

- **Domains**: A set of K data domains $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$.

Formalizing the Data Mixing Problem

- **Domains**: A set of K data domains $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$.
- **Training Data Mixture**: A domain weight vector $h \in \Delta^K$, where Δ^K is the probability simplex $\{h \in \mathbb{R}^K | \sum h_i = 1, h_i \geq 0\}$.

Formalizing the Data Mixing Problem

- **Domains**: A set of K data domains $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$.
- **Training Data Mixture**: A domain weight vector $h \in \Delta^K$, where Δ^K is the probability simplex $\{h \in \mathbb{R}^K | \sum h_i = 1, h_i \geq 0\}$.
- **Target Distribution**: Defined by importance weights $w \in \Delta^K$.

Formalizing the Data Mixing Problem

- **Domains**: A set of K data domains $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$.
- **Training Data Mixture**: A domain weight vector $h \in \Delta^K$, where Δ^K is the probability simplex $\{h \in \mathbb{R}^K | \sum h_i = 1, h_i \geq 0\}$.
- **Target Distribution**: Defined by importance weights $w \in \Delta^K$.
- **Optimal Mixture**: Find h^* to minimize the model's loss on the target distribution under a **fixed compute budget** (N, D):

$$h^* = \arg\min_{h \in \Delta^K} \sum_{i=1}^{K} w_i L_i(h | N, D)$$

where $L_i(h | N, D)$ is the validation loss on domain $\mathcal{D}_i$ for a model trained on mixture h under budget (N, D).

Formalizing the Data Mixing Problem

- **Domains**: A set of K data domains $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$.
- **Training Data Mixture**: A domain weight vector $h \in \Delta^K$, where Δ^K is the probability simplex $\{h \in \mathbb{R}^K | \sum h_i = 1, h_i \geq 0\}$.
- **Target Distribution**: Defined by importance weights $w \in \Delta^K$.
- **Optimal Mixture**: Find h^* to minimize the model's loss on the target distribution under a **fixed compute budget** (N, D):

$$h^* = \arg\min_{h \in \Delta^K} \sum_{i=1}^{K} w_i L_i(h | N, D)$$

where $L_i(h | N, D)$ is the validation loss on domain $\mathcal{D}_i$ for a model trained on mixture h under budget (N, D).

Formalizing the Data Mixing Problem

- **Domains**: A set of K data domains $\mathcal{D} = \{\mathcal{D}_1, \ldots, \mathcal{D}_K\}$.
- **Training Data Mixture**: A domain weight vector $h \in \Delta^K$, where Δ^K is the probability simplex $\{h \in \mathbb{R}^K | \sum h_i = 1, h_i \geq 0\}$.
- **Target Distribution**: Defined by importance weights $w \in \Delta^K$.
- **Optimal Mixture**: Find h^* to minimize the model's loss on the target distribution under a **fixed compute budget** (N, D):

$$h^* = \arg\min_{h \in \Delta^K} \sum_{i=1}^{K} w_i L_i(h | N, D)$$

where $L_i(h | N, D)$ is the validation loss on domain $\mathcal{D}_i$ for a model trained on mixture h under budget (N, D).

Key Challenge

Approximate $L_i(h \mid \cdot)$!

Key Properties of Loss Landscape and Optimal Mixture

- **Next: A theoretically grounded framework that predicts $L_i(h \mid \cdot)$**

Key Properties of Loss Landscape and Optimal Mixture

- **Next: A theoretically grounded framework that predicts $L_i(h \mid \cdot)$**
- Two key properties of the loss landscape and optimal mixture observed in empirical work.
  - **Domain interaction:**

$$L_i(h) = \frac{1}{\sum_{j=1}^{K} C_{ij} h_j^{\gamma_{ij}}} + E_i$$

  - **Non-uniform optimal mixture:**

Arxiv	Book	C4	GitHub	Commoncrawl	Stackexchange	Wikipedia
9.6	9.5	25.1	8.0	12.1	17.9	17.0

Extended Quantization Model: Capacity Competition

Extended Quantization Model: Capacity Competition

- Generalizes the single-domain Quantization Model (Michaud et al., 2023) to the multi-domain setting.
- **Core Idea:** Training is viewed as a **Capacity Allocation Problem**.

Extended Quantization Model: Capacity Competition

- Generalizes the single-domain Quantization Model (Michaud et al., 2023) to the multi-domain setting.
- **Core Idea:** Training is viewed as a **Capacity Allocation Problem**.
- **Mechanism:**
  - The model has a finite capacity N (total "quanta" of skills it can learn).
  - It allocates this capacity across different domains to minimize the total loss.

Extended Quantization Model: Capacity Competition

- Generalizes the single-domain Quantization Model (Michaud et al., 2023) to the multi-domain setting.
- **Core Idea:** Training is viewed as a **Capacity Allocation Problem**.
- **Mechanism:**
  - The model has a finite capacity N (total "quanta" of skills it can learn).
  - It allocates this capacity across different domains to minimize the total loss.
- **Strength:** Offering an explanation for the domain interaction

Single-Domain Foundation

Quantization Model (Michaud et al., 2023): Each domain $\mathcal{D}_i$ consists of a continuous space of skills indexed by $k_i \in [1, \infty)$ following a **power-law density**:

$$p_i(k_i) \propto k_i^{-\alpha_i} \quad \text{(where } \alpha_i > 1\text{)}$$

Single-Domain Foundation

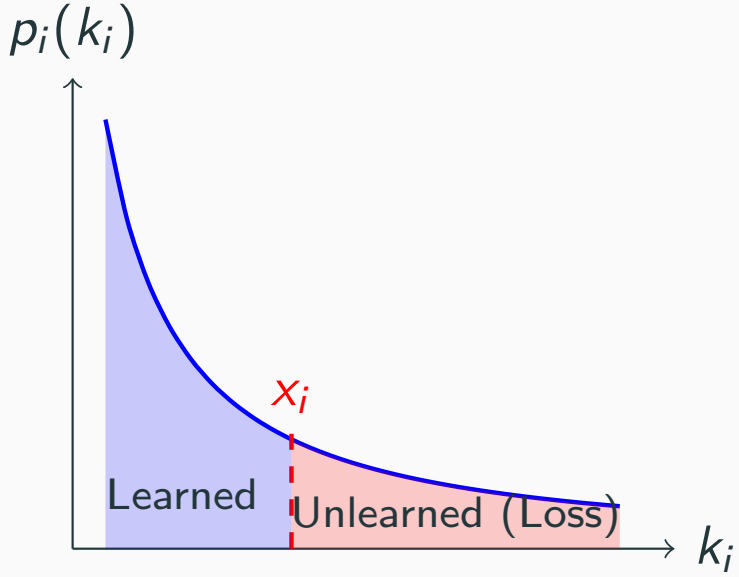
Quantization Model (Michaud et al., 2023): Each domain $\mathcal{D}_i$ consists of a continuous space of skills indexed by $k_i \in [1, \infty)$ following a **power-law density**:

$$p_i(k_i) \propto k_i^{-\alpha_i} \quad \text{(where } \alpha_i > 1\text{)}$$

- **Learning Process:** A model with allocated capacity x_i learns all high-frequency skills up to threshold x_i ($k_i \leq x_i$).
- **Domain Loss (L_i)** = The probability mass of the unlearned tail ($k_i > x_i$) multiplied by a **domain-specific constant** c_i:

$$L_i = c_i \int_{x_i}^{\infty} (\alpha_i - 1) k_i^{-\alpha_i} dk_i = c_i x_i^{-b_i}$$

where $b_i = \alpha_i - 1$.

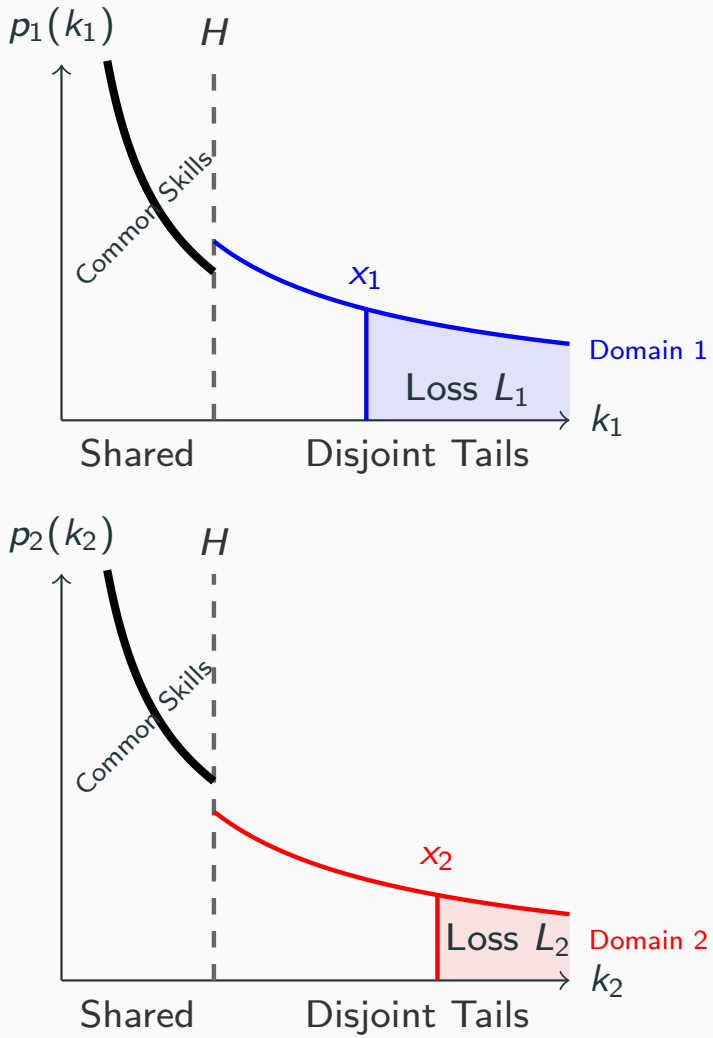


Structural Assumption: Shared Head, Disjoint Tail

We assume a **"Shared Head, Disjoint Tail"** data structure:

- **Shared Head ($k \leq H$):** Domains overlap on fundamental, high-probability skills (e.g., basic logic, grammar).
- **Disjoint Tail ($k > H$):** Rare, specialized skills are unique to each domain and independent of others.

Implication: The model easily learns the shared head. The core challenge is **allocating the remaining capacity** $(N - H)$ **among the disjoint specialized tails**.



Training as Capacity Allocation

When trained on a mixture $h = (h_1, \ldots, h_K)$, the model implicitly solves this constrained optimization problem:

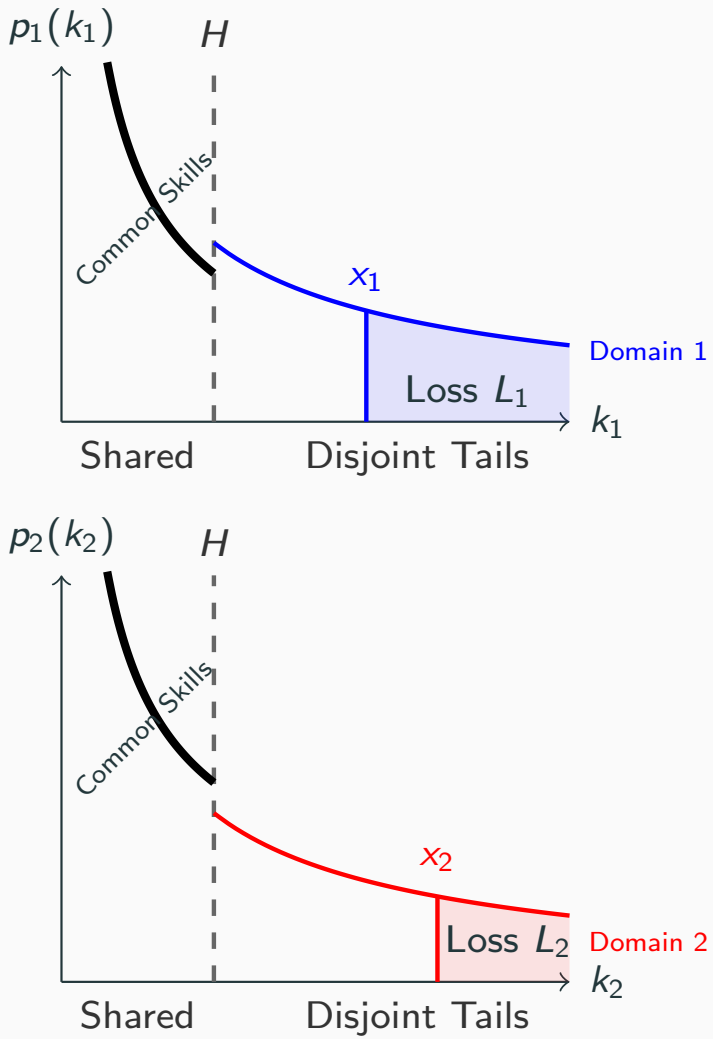
Capacity Allocation Problem

$$\min_{\{x_i\}} \sum_{i=1}^{K} h_i L_i = \sum_{i=1}^{K} h_i c_i x_i^{-b_i}$$

$$\text{s.t.} \sum_{i=1}^{K} (x_i - H) \leq N - H$$

Domain Loss: $L_i(h) = c_i (x_i^*(h))^{-b_i}$

Competition: The constraint couples all domains, introducing a non-trivial correlation.



Empirical Validation

Fitting Accuracy (trained models from (Liu et al. 2024))

- **Models:** 1B-parameter models trained on 25B tokens.
- **Domains:** 17 domains from the Pile dataset. 64 mixtures.
- **Metric:** Mean Relative Error $\left|\frac{\widehat{L}_i - L_i}{L_i}\right|$ and Mean Absolute Error $|\widehat{L}_i - L_i|$ on held-out mixtures.

Method	**MRE (%)** $\downarrow$	**MAE** $\downarrow$	**# Param**
Empirical Baselines			
Exponential (Ye et al.)	6.990	0.059	$K(K+2)$
RegMix (Liu et al.)	6.480	0.136	K^2
Additive (Shukor et al.)	2.209	0.052	$K(2K+1)$
Theoretical Models (Ours)			
Extended Quantization	**2.064**	**0.051**	$3K$

Table 1: Fitting accuracy on 1B-parameter models ($K = 17$ domains).

Limitation: Optimal Mixture Failure

We formulate the search for the optimal training mixture h^* as a **bi-level optimization problem**:

- **Outer Level (Target Objective):** Find h to minimize loss on the *target distribution* w:

$$h^* = \arg\min_h \sum_{i=1}^{K} w_i L_i(x^*(h)) \tag{1}$$

- **Inner Level (Training Dynamics):** The model allocates capacity $x^*(h)$ to minimize loss on the *training mixture* h:

$$x^*(h) = \arg\min_x \sum_{i=1}^{K} h_i L_i(x) \quad \text{s.t.} \quad \sum x_i = N \tag{2}$$

The Limitation: Trivial Solution

- The optimal strategy is simply to set training weights equal to target weights: $\mathbf{h}^* = \mathbf{w}$.
- **Contradiction:** Empirical studies show h^* deviates from w.

Limitation: Optimal Mixture Failure

This Model predicts $h^* \equiv w$, which contradicts empirical reality.

Limitation: Optimal Mixture Failure

This Model predicts $h^* \equiv w$, which contradicts empirical reality.

- **The Missing Piece: Training Dynamics.**
  - Training = static capacity allocation. It ignores the training dynamics.
  - Real-world models are trained via **stochastic optimizers** (e.g., Adam, SGD), which introduce gradient noise.

Limitation: Optimal Mixture Failure

This Model predicts $h^* \equiv w$, which contradicts empirical reality.

- **The Missing Piece: Training Dynamics.**
  - Training = static capacity allocation. It ignores the training dynamics.
  - Real-world models are trained via **stochastic optimizers** (e.g., Adam, SGD), which introduce gradient noise.
- **Next Step:**
  - We follow (Bordelon et al., 2024, Lin et al., 2024) and model the training process of LLM via a **Projected Linear Regression Model** trained with **one-pass SGD**.

Extended Linear Model: Noise Driven by Data Amount

Extended Linear Model: Noise Driven by Data Amount

- Extends the Projected Linear Regression models from (Bordelon et al., 2024, Lin et al., 2024).

Extended Linear Model: Noise Driven by Data Amount

- Extends the Projected Linear Regression models from (Bordelon et al., 2024, Lin et al., 2024).
- **Key Idea:**
  - Model the training process of LLM via a **Projected Linear Regression Model** trained with **one-pass SGD**
  - The SGD training process **implicitly solves the capacity allocation problem**
  - It introduces an **additional noise term** that depends on data amount

Extended Linear Model: Noise Driven by Data Amount

- Extends the Projected Linear Regression models from (Bordelon et al., 2024, Lin et al., 2024).
- **Key Idea:**
  - Model the training process of LLM via a **Projected Linear Regression Model** trained with **one-pass SGD**
  - The SGD training process **implicitly solves the capacity allocation problem**
  - It introduces an **additional noise term** that depends on data amount
- **Strength:**
  - Explains why the optimal mixture h^* deviates from the target weights w.

Problem Formulation

Consider a d-dimensional linear regression problem with large d:

$$y = \langle \theta^*, x \rangle + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{3}$$

Problem Formulation

Consider a d-dimensional linear regression problem with large d:

$$y = \langle \theta^*, x \rangle + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{3}$$

Model Size: The model has size N and sees a projected input

$$\tilde{x} = Sx \quad \text{with} \quad S \in \mathbb{R}^{N \times d}.$$

Problem Formulation

Consider a d-dimensional linear regression problem with large d:

$$y = \langle \theta^*, x \rangle + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2) \tag{3}$$

Model Size: The model has size N and sees a projected input

$$\tilde{x} = Sx \quad \text{with} \quad S \in \mathbb{R}^{N \times d}.$$

Training dynamics: The model learns $\widehat{w} \in \mathbb{R}^N$ via **one-pass SGD**, minimizing the squared error on the projected features $\tilde{x} = Sx$:

$$\widehat{w} = \arg\min_{w \in \mathbb{R}^N} \frac{1}{D} \sum_{i=1}^{D} (y_i - w^\top S x_i)^2$$

Problem Formulation: Spectral Structure

We analyze the covariance matrix $H_i = \mathbb{E}[x_i x_i^\top]$ via its eigendecomposition:

- **Eigenvector (u_k):** Represents a specific "skill" or pattern of variation (e.g., a specific texture in images or topic in text).
- **Eigenvalue (λ_k):** Quantifies the **strength** or variance of that pattern in the dataset.

Spectral Assumption: "Shared Head, Disjoint Tail"

- **Shared Head ($k \leq H$):** Common eigenvectors $\{u_1, \ldots, u_H\}$ with non-zero variance across all domains (universal knowledge).
- **Disjoint Tail ($k > H$):** Domain-specific eigenvectors $u_k^{(i)}$ with their eigenvalues following a **domain-specific power law**:

$$\lambda_k^{(i)} = \underbrace{h_i}_{\text{Weight}} \cdot \underbrace{k^{-\alpha_i}}_{\text{Power Law Decay}}$$

(Power-law decay is a standard assumption from prior work.)

The Loss Decomposition

Analyzing this one-pass SGD via continuous-time approximation and Stochastic Differential Equation (SDE):

Main Theorem:

The expected test loss on domain i can be decomposed as follows:

$$L_i(h, N, D) \approx \underbrace{c_i x_i^*(h)^{-b_i}}_{\text{Capacity Term}} + \underbrace{A_i(Dh_i)^{-a_i}}_{\text{Noise Term}} + E_i$$

1. **Capacity Term**
   - Identical to Quantization Model.
   - Driven by **competition** with other domains.
   - Depends on **all** weights h.
2. **Noise Term**
   - New term arising from stochastic gradient noise.
   - Driven by **data volume**.
   - Depends only on own weight h_i.

Explaining the Optimal Mixture

Why does the optimal training mixture h^* deviate from the target w?

Mechanism: Noise Reduction

- **Harder Domains:** Domains with large A_i or small a_i contribute more variance to the loss.
- **Reweighting:** To minimize total loss, the optimization **shifts weights** h_i toward these "harder-to-learn" domains to suppress their variance.

Result I: Superior Predictive Accuracy

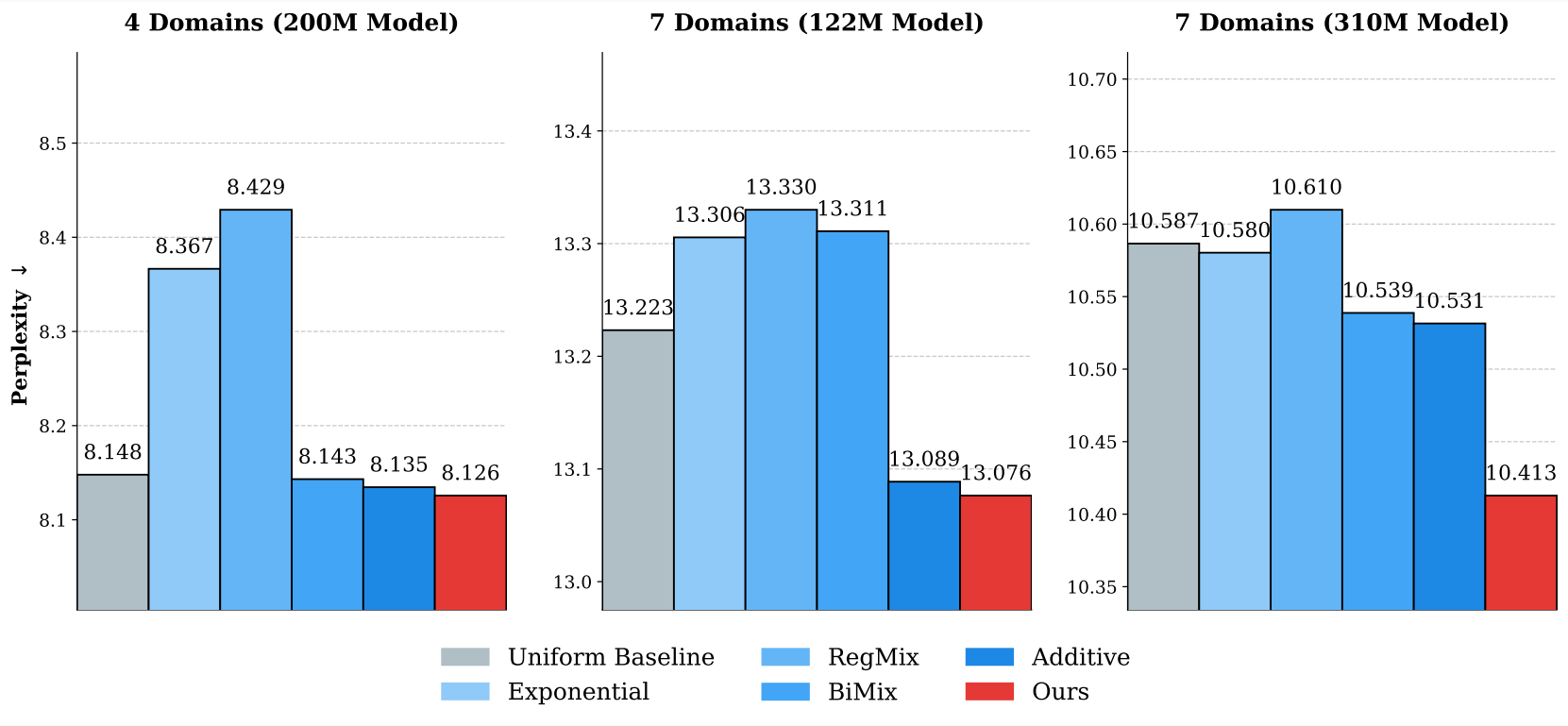
Our theoretical models fit the observed loss landscape more accurately than existing empirical baselines, while using significantly fewer parameters.

Method	**MRE (%)**	**MAE**	**# Param**
Empirical Baselines			
Exponential (Ye et al.)	6.990	0.059	$K(K+2)$
RegMix (Liu et al.)	6.480	0.136	K^2
Additive (Shukor et al.)	2.209	0.052	$K(2K+1)$
Theoretical Models (Ours)			
Extended Quantization	2.064	0.051	$3K$
Extended Linear Reg.	**1.533**	**0.034**	$5K$

Table 2: Fitting accuracy on 1B-parameter models ($K = 17$ domains).

Result II: Identifying the Optimal Mixture

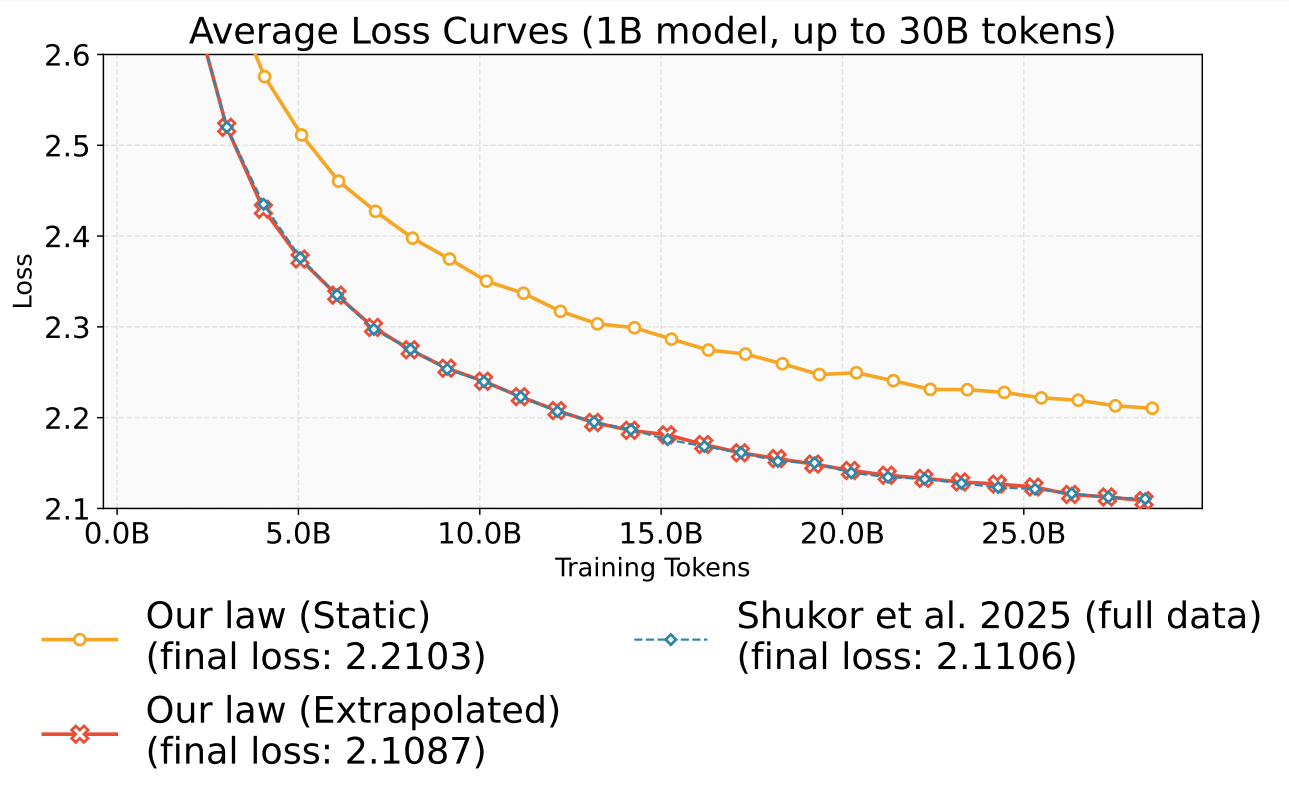
We trained new models using the optimal mixtures predicted by each scaling law. Our method yielded the lowest validation loss on the target distribution.



- **Efficiency:** Achieves these results using significantly fewer free parameters than empirical laws.

Result III: Extrapolation to Unseen Scales

- **Target:** Find the optimal mixture at 1B model / 30B tokens.
- **Ground truth:** Fit the SOTA Additive Law on 1B-scale data (blue).
- **Ours:** Fit only small models, extrapolate to 1B/30B; red nearly overlaps blue.
- **Static mixture:** The small-scale optimum can fail at large scale (yellow).



Future Work

Future Directions

1. **Unseen Domains and Downstream Tasks**
   - **Current Limitation:** The model predicts loss only within the span of training domains.
   - **Next Steps:** Extend the framework to predict how data mixtures affect performance on **unseen domains** and specific **downstream tasks**.
2. **Explicit Domain Overlap Modeling**
   - **Current Limitation:** We assume "disjoint tails" for tractability.
   - **Next Steps:** Develop models that explicitly quantify information overlap between highly correlated domains.
3. **Reliable Fitting Algorithms**
   - **Current Limitation:** Fitting requires solving a non-convex bi-level optimization problem.
   - **Next Steps:** Derive **convex relaxations** or **analytical approximations** to make parameter estimation faster and more robust to initialization.

Thanks and Questions?

Experimental Setup

We validate the Extended Linear Regression Model through two primary experiments:

Objective 1: Fitting Accuracy

- **Data:** 64 models (1B parameters) trained on 25B tokens.
- **Domains:** 17 domains from the Pile dataset.
- **Metric:** Mean Relative Error (MRE) on held-out mixtures.

Objective 2: Optimal Mixture Prediction

- **Data:** 200M and 700M parameter models.
- **Domains:** 4 domains (GitHub, PG-19, StackExchange, Wikipedia).
- **Goal:** Predict the mixture h^* that minimizes loss on a uniform target distribution.

Bi-Level Optimization Framework

We formulate the search for the optimal mixture h^* as a **bi-level problem**.

Inner Level: Implicit Capacity Allocation (Training)

Given a mixture h, the model allocates capacity $x^*(h)$ to minimize weighted training loss (ignoring noise):

$$x^*(h) = \arg\min_x \sum_{i=1}^{K} h_i c_i x_i^{-b_i} \quad \text{s.t.} \quad \sum x_i = N$$

Outer Level: Target Minimization (Validation)

We seek h^* to minimize loss on the **target distribution** w, which includes the critical **noise term** $A_i(Dh_i)^{-a_i}$:

$$h^* = \arg\min_{h \in \Delta^K} \sum_{i=1}^{K} w_i \Big[\underbrace{c_i (x_i^*(h))^{-b_i}}_{\text{Capacity Term}} + \underbrace{A_i (Dh_i)^{-a_i}}_{\text{Noise Term}} \Big]$$

Gradient Characterization

To optimize the outer objective $\mathcal{J}(h)$, we derive the gradient $\nabla_k \mathcal{J}$ via the Implicit Function Theorem.

The Gradient Decomposition:

$$\frac{\partial \mathcal{J}}{\partial h_k} = \underbrace{-w_k a_k A_k D^{-a_k} h_k^{-(a_k+1)}}_{\textbf{Force 1: Noise Reduction}} + \underbrace{\frac{\lambda x_k^*}{h_k(b_k+1)} \left(\overline{R} - \frac{w_k}{h_k} \right)}_{\textbf{Force 2: Capacity Reallocation}} \tag{4}$$

- **Force 1 (Variance):** Directly encourages increasing h_k to reduce the noise term $A_k(Dh_k)^{-a_k}$ (requires more data).
- **Force 2 (Coupling):** Captures how changing h_k shifts the global shadow price λ, forcing a redistribution of capacity x^* across *all* domains.

We solve the non-convex outer optimization using **Online Mirror Descent** with exponentiated gradient updates.