ICML
International Conference On Machine Learning
2025

CHAMELEON: A Flexible Data-mixing Framework for Language Model Pretraining and Finetuning

Wanyun Xie [1] **Francesco Tonin** [1] **Volkan Cevher** [1]

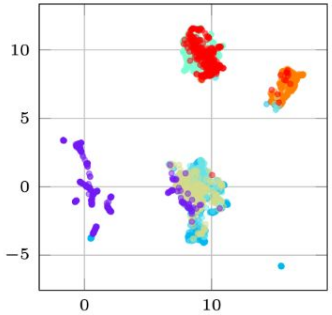
Data Mixing:

- Given Dataset: $\mathcal{D} = \{D_1, \ldots, D_k\}$ consisting of k distinct domains.
- **Goal**: find a domain weight vector $\alpha \in \Delta_k$, where Δ_k is a probability simplex

Presenter: Duong Minh Le

1

Learn Domain Embeddings



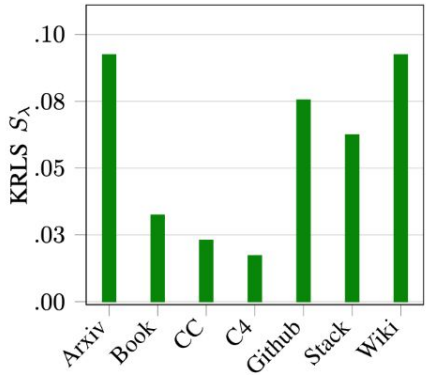
2

Compute Domain Affinity

42	−17	−15	−14	6	3	−5
−17	27	12	15	−21	−16	1
−15	12	24	19	−20	−13	−6
−14	15	19	22	−22	−17	−2
6	−21	−20	−22	46	31	−19
3	−16	−13	−17	31	29	−17
−5	1	−6	−2	−19	−17	49

3

Compute KRLS Scores



4

Train base LMs

Pretraining

Domain Weights: $\alpha_i^{\text{PT}} = \text{softmax}(S_\lambda^{-1})$

Transfer to New Data

Domain Weights: $\alpha_i^{\text{ND}} = \text{softmax}(S_\lambda^{-1})$

Finetuning

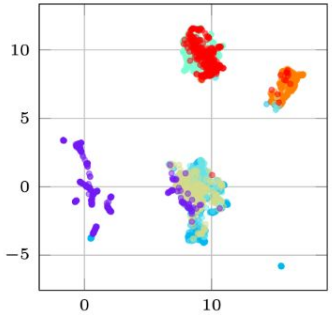
Domain Weights: $\alpha_i^{\text{FT}} = \text{softmax}(S_\lambda)$

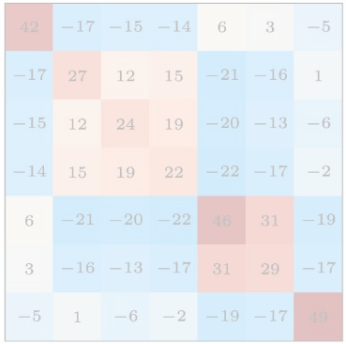
1 Learn Domain Embeddings

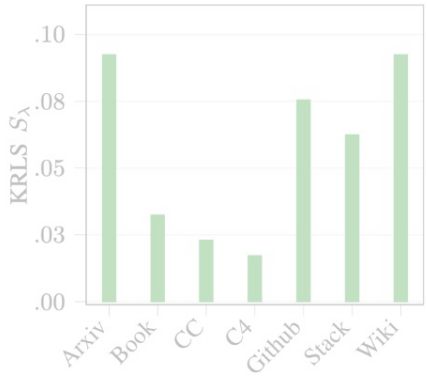
2 Compute Domain Affinity

3 Compute KRLS Scores

4 Train base LMs





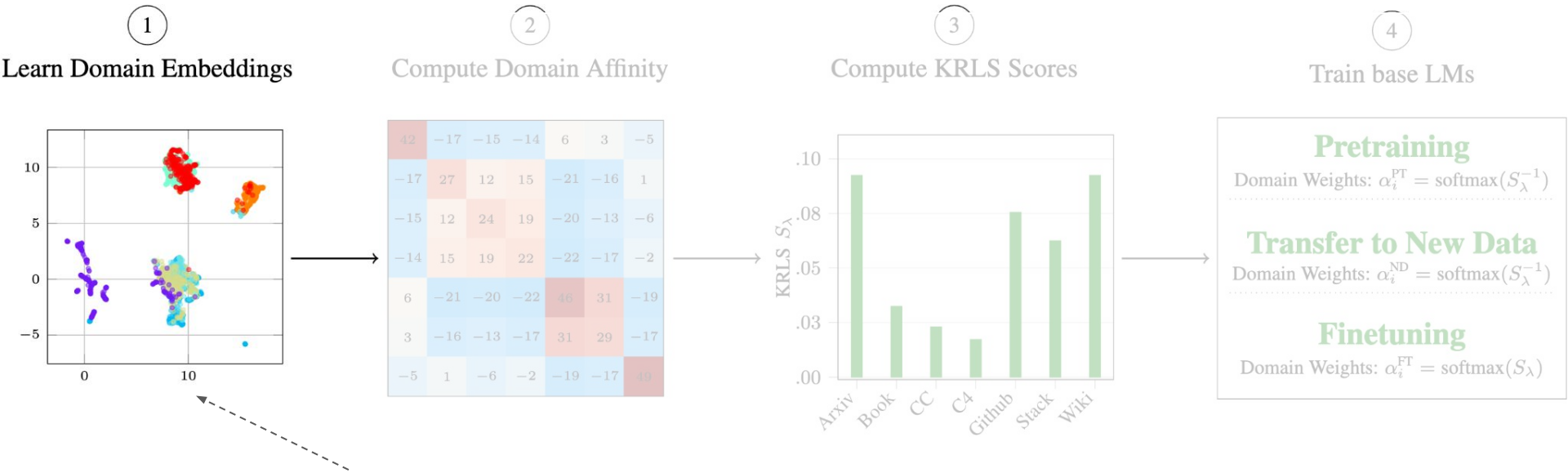


Pretraining
Domain Weights: $\alpha_i^{\text{PT}} = \text{softmax}(S_\lambda^{-1})$

Transfer to New Data
Domain Weights: $\alpha_i^{\text{ND}} = \text{softmax}(S_\lambda^{-1})$

Finetuning
Domain Weights: $\alpha_i^{\text{FT}} = \text{softmax}(S_\lambda)$

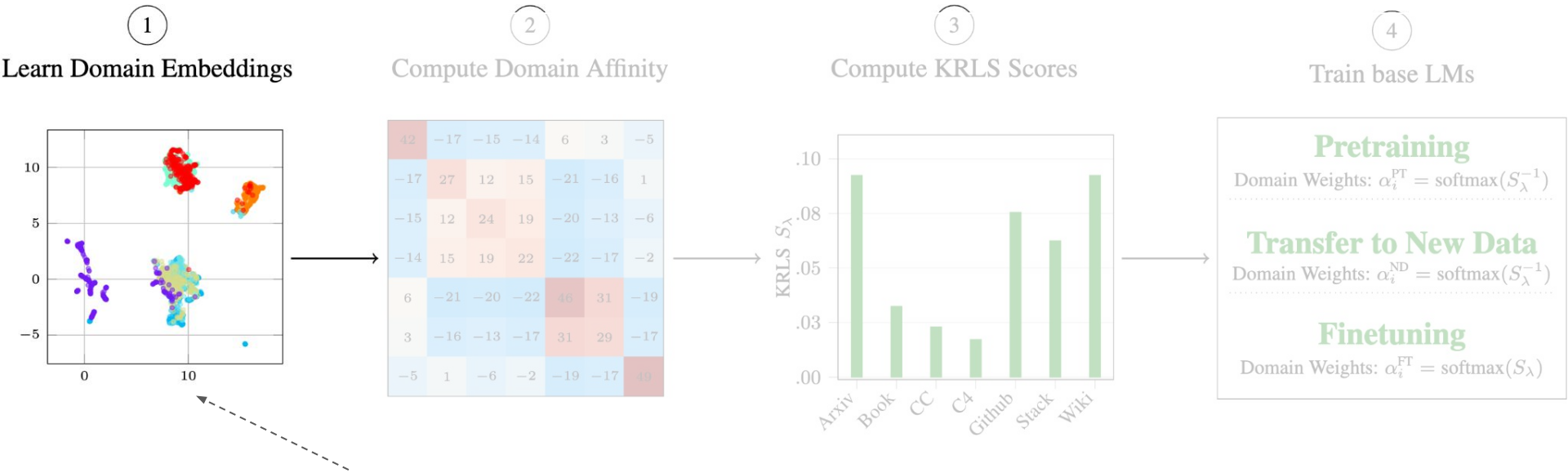
- Train a small proxy model on D with uniform weights
- Data embedding = mid-layer embeddings from the proxy model
- Domain embeddings = averaging the embeddings for each domain



- Domain embeddings = averaging the embeddings for each domain

$$x_i = \frac{1}{|D_i|} \sum_{a \in D_i} h_{\theta_p}^{(L)}(a)$$

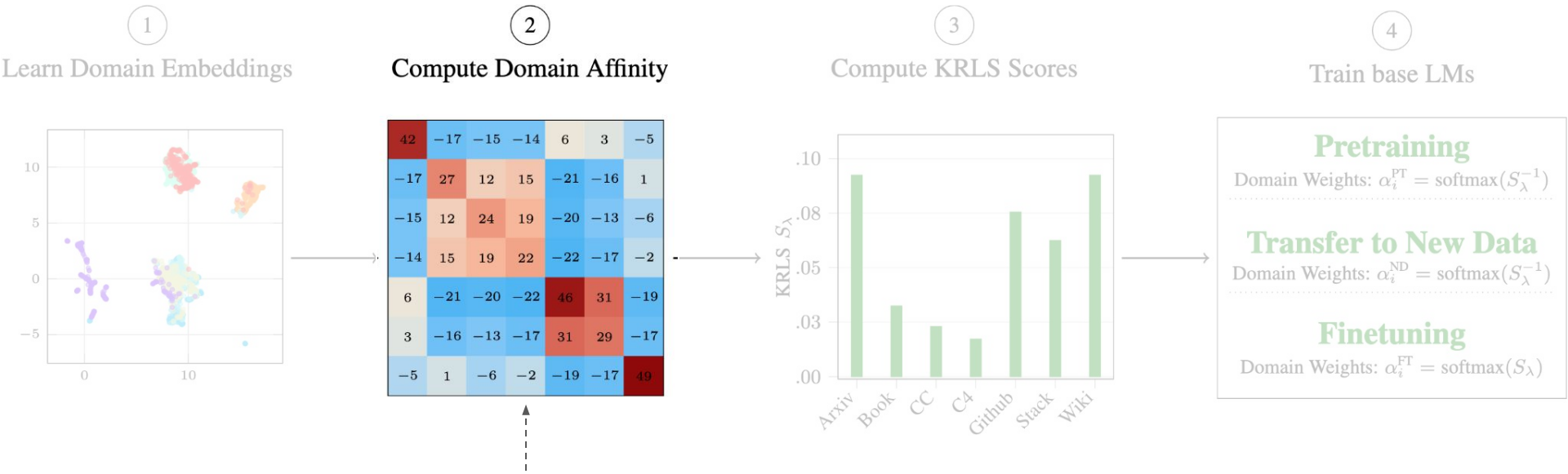
L-th layer embedding of the proxy model



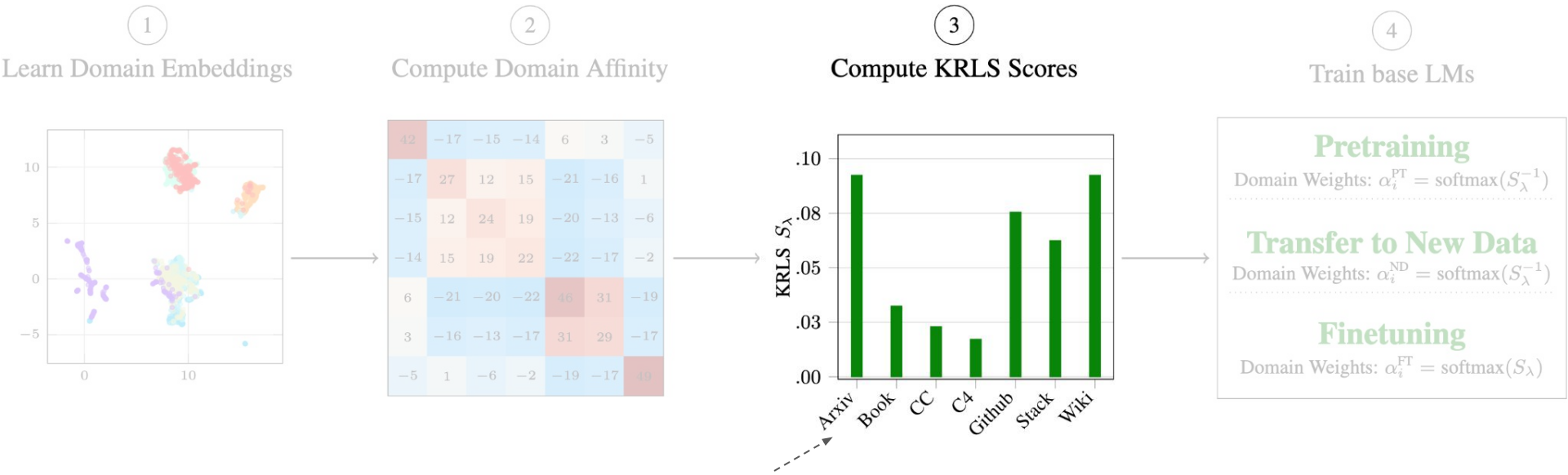
- Domain embeddings = averaging the embeddings for each domain

$$x_i = \frac{1}{|D_i|} \sum_{a \in D_i} h_{\theta_p}^{(L)}(a)$$

- Domain embedding matrix: $X = [x_1, \ldots, x_k]^\top \in \mathbb{R}^{k \times p}$



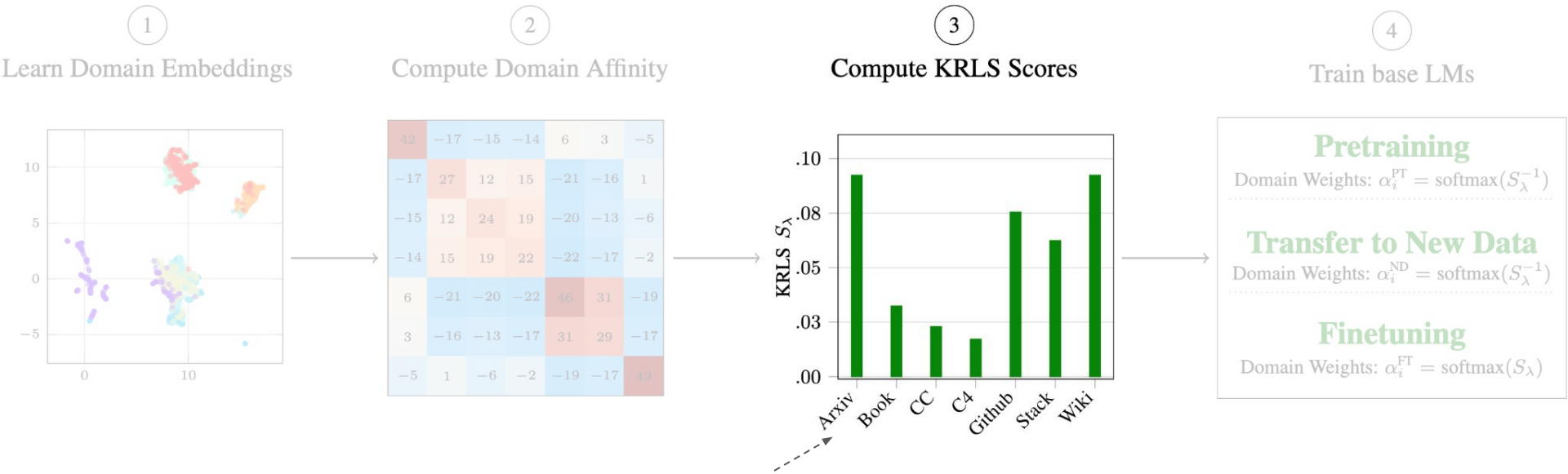
- Domain embedding matrix: $X = [x_1, \ldots, x_k]^\top \in \mathbb{R}^{k \times p}$
- Domain affinity matrix: $\Omega_{\mathcal{D}} = [\kappa(x_i, x_j)]_{i,j=1}^{k} = XX^\top$



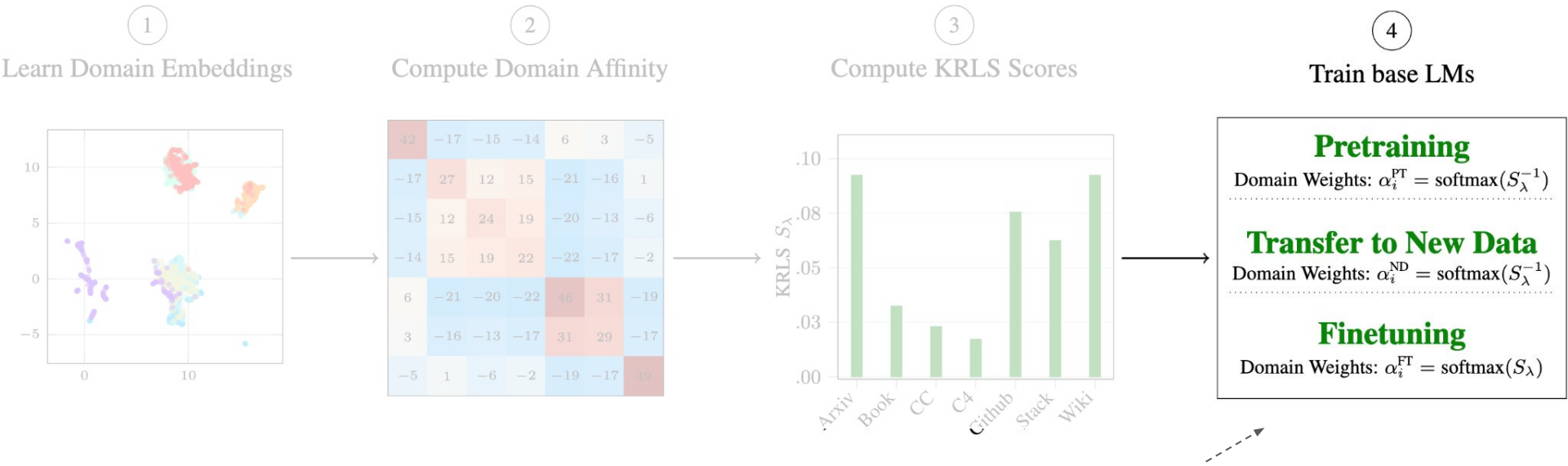
- Domain affinity matrix: $\Omega_{\mathcal{D}} = [\kappa(x_i, x_j)]_{i,j=1}^{k} = XX^\top$
- Quantify the influence of each domain using Kernel ridge leverage scores (KRLS)

$$S_\lambda(D_i) = [\Omega_{\mathcal{D}}(\Omega_{\mathcal{D}} + k\lambda I)^{-1}]_{ii}$$

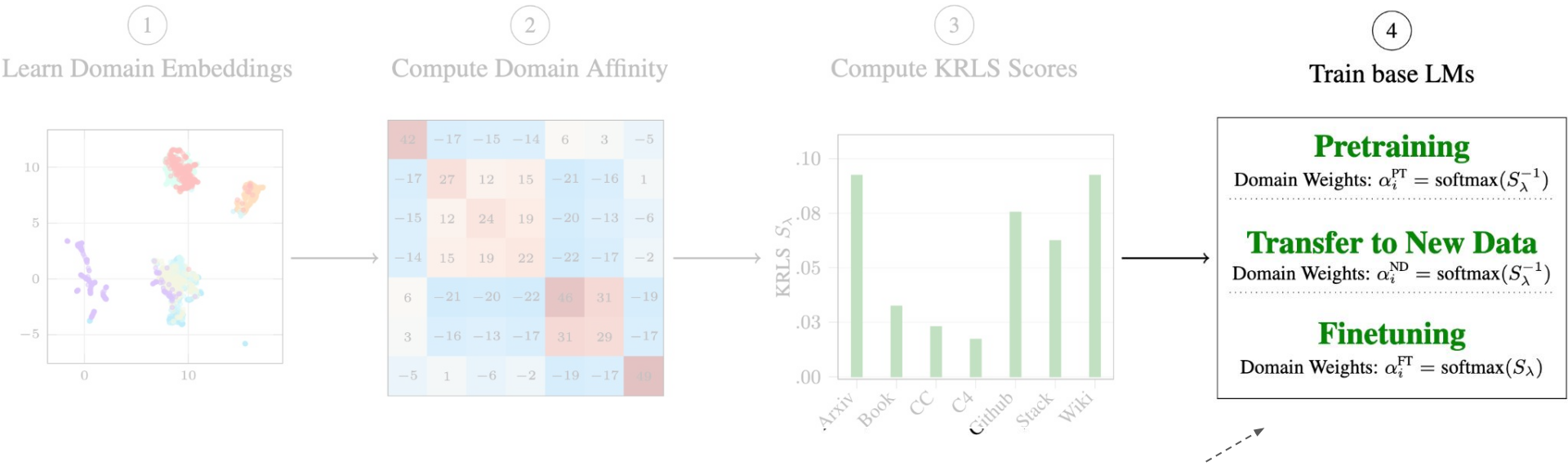
S = [s1, s2, …, sk] -> S^-1 = [1/s1, 1/s2, …]



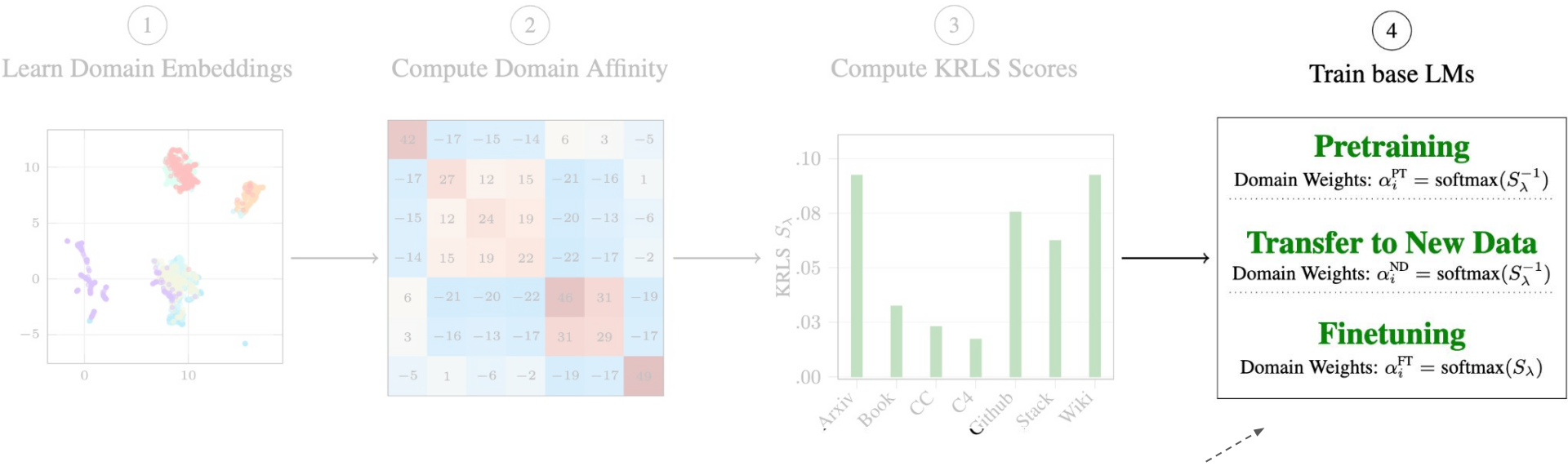
- High KRLS value for domain Di indicates that its embedding xi cannot be well-approximated as a combination of embeddings from other domains,
- Low scores indicate data domains that show higher degree of dependency with other domains, identifying more common data characteristics



- High KRLS value for domain D_i indicates that its embedding x_i cannot be well-approximated as a combination of embeddings from other domains,
- Low scores indicate data domains that show higher degree of dependency with other domains, identifying more common data characteristics
- Pretraining: prioritize domains with lower KRLS as more important, as they are more common and thus more useful for learning general-purpose language representations.



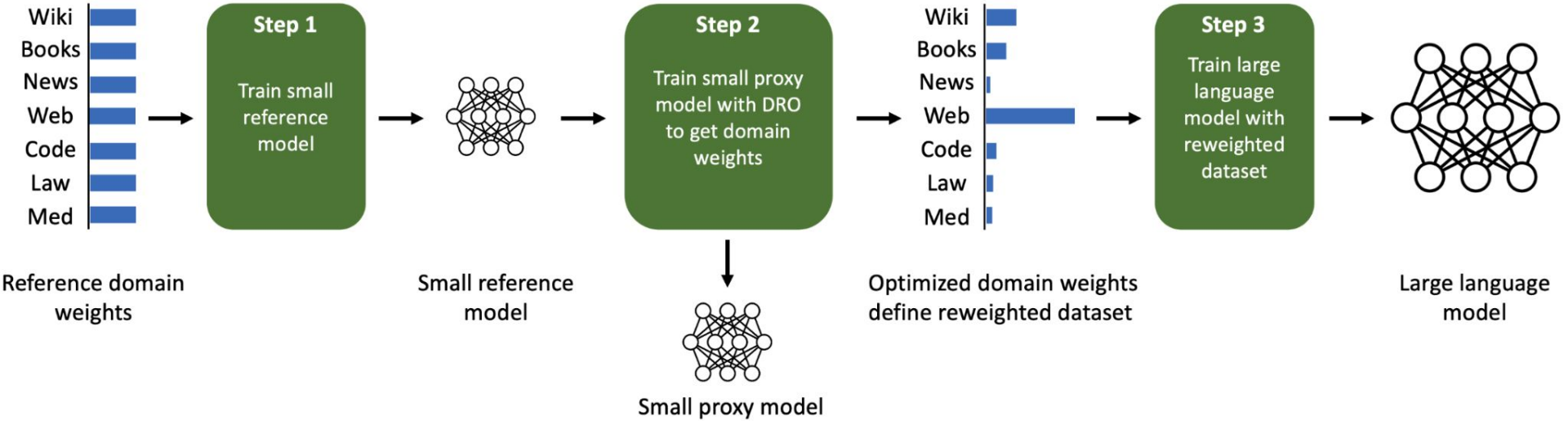
- High KRLS value for domain Di indicates that its embedding xi cannot be well-approximated as a combination of embeddings from other domains,
- Low scores indicate data domains that show higher degree of dependency with other domains, identifying more common data characteristics
- Finetuning: upweight domains with higher KRLS, as they are more unique and therefore better suited for learning task-specific representations.



- Fine-tuning aims to specialize on a novel specific task, requiring the model to learn differential features not fully captured during pre-training
- Directly use the pre-trained model to extract the domain embeddings

Related work

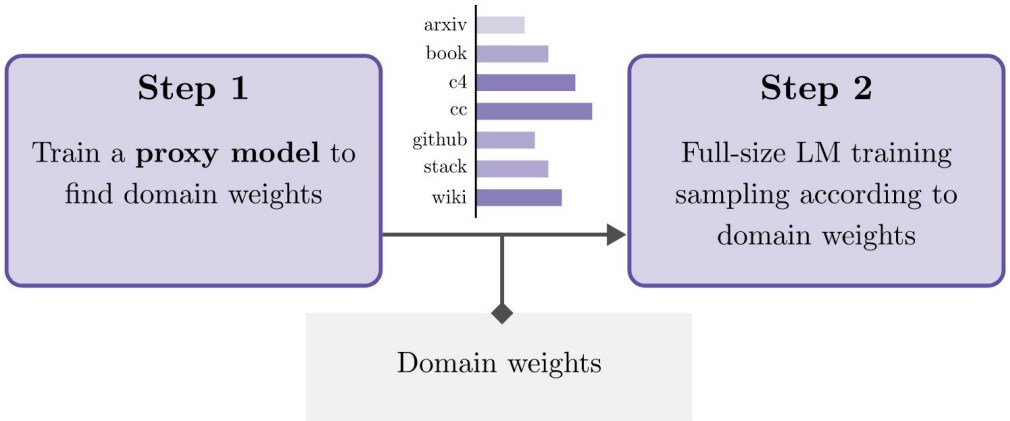
DoReMi: Optimizing Data Mixtures Speeds Up Language Model Pretraining



- Step 1: Train a reference model to capture the baseline level of difficulty of each domain
- Step 2: Train a proxy model, together with updating the domain weights based on the excess loss between the proxy and reference model

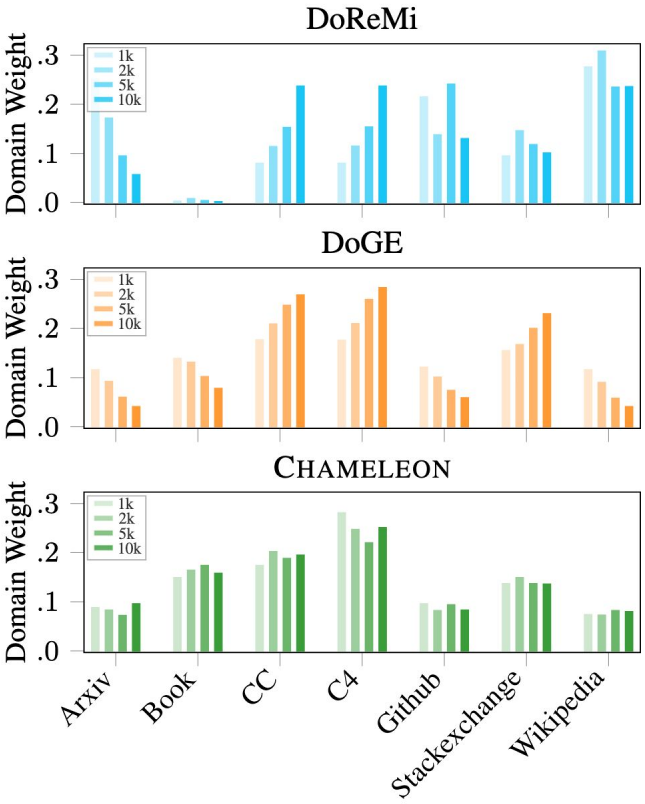
Related work:

DoGE : Domain Reweighting with Generalization Estimation



Perform bi-level optimization to optimize the proxy model and the domain weights at the same time

Domain weights obtained by CHAMELEON remain stable across different training steps of the proxy model



Stable domain weights of CHAMELEON

Table 16: **Domain weights across different model sizes, λ values, and number of samples.**

Domain	Model Sizes				λ Values			Number of Samples		
	60M	82M	124M	210M	$\lambda = 1$	$\lambda = 10$	$\lambda = 100$	2k	4k	8k
Arxiv	0.084	0.083	0.087	0.093	0.080	0.083	0.087	0.079	0.083	0.081
Book	0.158	0.164	0.157	0.165	0.159	0.164	0.173	0.165	0.164	0.170
CC	0.170	0.202	0.169	0.170	0.178	0.202	0.201	0.193	0.202	0.209
C4	0.271	0.247	0.288	0.259	0.243	0.247	0.258	0.253	0.247	0.241
Github	0.073	0.082	0.072	0.089	0.097	0.082	0.057	0.079	0.082	0.066
Stackexchange	0.152	0.149	0.153	0.145	0.152	0.149	0.128	0.138	0.149	0.143
Wikipedia	0.072	0.073	0.074	0.078	0.081	0.073	0.095	0.093	0.073	0.090

Experiments:

- Dataset: SlimPajama-627B (7 domains)
- Baselines:
  - Uniform
  - DoReMi
  - DoGE
  - RegMix - leverages prior knowledge of downstream tasks
- Models:
  - Auxiliary model (for DoReMi, DoGE, Chameleon): 82M decoder-only transformer
    - For DoReMi, DoGE, the auxiliary models are trained for 10k iterations
    - For Chameleon, the auxiliary model is trained for 2k iterations
  - Large base model: 684M

Table 2: **Universal generalization - perplexity.** Per-domain test perplexity for the universal generalization compared with the uniform baseline, DoReMi, DoGE, and RegMix with 684M parameter models. Note that, unlike other methods, RegMix knows the target downstream tasks for optimization. We report the compute (in FLOPs) required to arrive at the data mixture. CHAMELEON boosts generalization at a fraction of the computational cost.

Domain	**Uniform**	**DoReMi**	**DoGE**	CHAMELEON	**RegMix**
Arxiv	8.16	9.16	9.07	8.31	11.35
Book	42.55	46.48	40.30	39.23	41.52
CC	45.26	40.62	38.99	40.11	37.32
C4	49.00	43.92	40.65	42.59	43.85
Github	3.99	4.10	4.09	4.20	4.99
Stackexchange	7.99	8.35	7.39	7.94	10.63
Wikipedia	12.42	10.78	15.74	13.90	20.88
Average PPL (↓)	24.20	23.34	22.32	**22.31**	24.36
# Domains Over Uniform	-	3/7	**4/7**	**4/7**	3/7
FLOPs	0	1.34×10^{18} (10×)	6.68×10^{17} (5×)	$\mathbf{1.36 \times 10^{17}}$ (1×)	1.20×10^{18} (9×)

Table 3: **Universal generalization - reasoning.** Accuracy of downstream tasks in the same settings as Table 2.

Benchmark	**Unif.**	**DoReMi**	**DoGE**	CH.	**RegMix**
ARC-E	36.8	37.6	38.0	37.8	39.1
COPA	55.7	59.3	62.3	61.9	63.0
HellaSwag	26.5	27.0	27.2	27.1	27.0
Lambada	13.5	13.6	14.7	15.1	16.5
LogiQA	21.7	21.9	22.4	22.6	21.4
MultiRC	57.2	55.7	57.3	57.2	56.6
OpenBook	14.1	13.3	14.6	14.4	14.7
PiQA	59.2	59.5	60.0	60.5	57.6
QQP	36.8	36.8	36.8	39.2	37.1
RACE	26.1	25.3	26.4	26.5	27.3
SciQ	61.8	62.5	64.9	64.3	64.1
Social IQA	35.0	35.5	35.7	35.7	35.6
WinoGrande	50.5	51.3	52.0	52.1	50.9
Average (↑)	37.9	38.4	39.4	**39.6**	39.3

Scale to bigger base model

Table 5: **Universal generalization with *1.2B* model - reasoning.** Large-scale pretraining experiments.

Benchmark	Unif.	DoReMi	DoGE	CH.	RegMix
ARC-E	39.4	41.2	41.9	42.4	43.0
COPA	64.0	66.0	63.0	61.0	66.0
HellaSwag	27.5	27.7	28.2	28.4	27.6
Lambada	17.9	17.3	18.7	21.6	20.7
LogiQA	22.0	24.0	22.0	21.2	20.7
MultiRC	57.2	57.2	57.2	57.2	56.9
OpenBookQA	15.0	13.6	13.8	16.4	17.4
PIQA	61.5	61.9	61.8	63.8	58.7
QQP	36.8	36.8	36.9	36.9	36.8
RACE	26.0	26.7	27.8	29.1	28.4
SciQ	69.7	68.3	69.0	72.6	72.0
SocialIQA	36.2	36.5	35.9	37.2	36.1
WinoGrande	52.8	49.6	48.9	51.5	50.0
Average (↑)	40.5	40.5	40.4	**41.5**	41.1

Table 6: **Universal generalization with *1.2B* model - perplexity.** Large-scale experiments in the same settings of Table 5.

Domain	Uniform	DoReMi	DoGE	CHAMELEON	RegMix
Arxiv	6.30	7.09	7.07	6.33	10.61
Book	28.25	32.66	27.83	24.63	27.55
CC	31.19	29.96	28.11	26.95	24.70
C4	34.74	33.05	31.06	29.58	31.94
Github	2.91	3.03	3.07	2.94	4.08
Stackexchange	6.01	6.44	5.80	5.76	9.54
Wikipedia	8.65	7.93	10.88	9.03	20.08
Average PPL (↓)	16.86	17.17	16.26	**15.03**	18.36
# Domains Over Uniform	-	3/7	**4/7**	**4/7**	3/7

Adapt the proxy model to new data: The Pile

- DoReMi, DoGE and RegMix require retraining a new proxy model from scratch whenever domains are added or removed
- Setup:
  - Employ the proxy model trained on Slimpajama to Pile dataset
  - The Pile dataset includes more domains than Slimpajama and its data is more diverse

Scalable to Pile

Table 7: **Transfer to new domains - perplexity.** Per-domain test perplexity when adapting to new data. The Human baseline is (Gao et al., 2020). When *domain composition changes*, the other methods need to retrain proxy models from scratch and re-run domain weight optimization, while we can directly compute the KRLS of the new domains by extending the proxy model trained on previous data. We report the extra compute (in FLOPs) required to adapt the data mixture to the Pile.

Domain	Human	DoReMi	DoGE	CHAMELEON	RegMix
Arxiv	9.76	14.11	10.78	9.73	16.19
Dm_mathematics	5.52	6.27	4.52	5.31	19.26
Enron_emails	12.82	9.96	9.39	7.77	12.06
Europarl	34.69	24.77	11.62	28.18	131.80
Freelaw	14.12	16.20	17.99	15.04	18.66
Github	5.92	5.84	4.90	6.16	9.95
Gutenberg_pg_19	39.36	38.36	39.57	33.28	34.43
Hackernews	35.94	29.68	29.87	27.02	29.80
Nih_exporter	22.93	25.89	26.81	24.12	28.69
Philpapers	47.59	36.43	44.56	34.63	51.71
Pile_cc	43.19	32.85	58.17	34.90	32.30
Pubmed_abstracts	17.87	24.19	25.62	21.87	23.20
Pubmed_central	9.76	9.43	8.10	7.36	13.80
Stackexchange	10.41	11.48	11.79	11.25	18.96
Ubuntu_irc	36.12	32.10	23.20	29.34	20.71
Uspto_backgrounds	17.22	21.19	20.08	18.25	22.05
Wikipedia_en	28.70	25.95	40.42	24.68	29.32
Average PPL (↓)	23.05	21.45	22.79	**19.94**	30.17
# Domains Over Human	-	9/17	10/17	**11/17**	4/17
Extra FLOPs	0	1.34×10^{18} (290×)	6.68×10^{17} (145×)	$\mathbf{4.62 \times 10^{15}}$ (1×)	3.5×10^{18} (758×)

Scalable to Pile

Table 8: **Transfer to new domains - reasoning.** Accuracy of downstream tasks in the same settings as Table 7.

Benchmark	Hum.	DoReMi	DoGE	CH.	RegMix
ARC-E	37.5	39.3	35.3	39.2	39.5
COPA	56.8	61.5	54.7	60.9	61.2
HellaSwag	26.7	27.3	26.1	27.4	27.3
Lambada	12.5	15.8	9.3	15.9	15.4
LogiQA	22.5	21.2	22.1	23.8	21.9
MultiRC	56.7	56.5	57.2	57.3	56.2
OpenBook	13.3	13.1	13.1	14.2	14.5
PiQA	57.8	59.3	55.9	59.7	60.4
QQP	37.5	36.8	36.8	37.2	37.6
RACE	25.8	27.2	24.9	26.8	27.2
SciQ	64.1	65.7	58.1	66.0	67.1
Social IQA	35.0	36.0	34.2	36.6	36.3
WinoGrande	50.7	51.2	49.8	50.9	49.9
Average (↑)	38.2	39.3	36.7	**39.7**	39.6

Fine-tuning

- Setup:
  - Fine-tune a pretrained model trained on the Pile for 10k steps
  - FT Dataset:
    - Wiki40b: includes multiple languages (7 selected)
    - Stack-decup: covers various programming languages (7 selected)

Fine-tuning

Table 9: **Finetuning.** Per-domain test perplexity compared with the uniform baseline for finetuning on the 7 languages in Wiki40b. CHAMELEON can flexibly be extended to the finetuning pipeline by computing the KRLS of the finetuning domains, improving performance across all domains.

Domain	**Uniform**	CHAMELEON
French	6.86	6.51
German	10.12	8.78
Italian	13.29	12.42
Spanish	8.41	8.04
Portuguese	8.00	7.78
Dutch	13.98	12.30
Polish	5.07	4.21
Average PPL (↓)	9.43	**8.58**
# Domains Over Uniform	-	7/7

Table 10: **Finetuning.** Per-domain test PPL vs. the uniform baseline for finetuning on the 7 programming languages in Stack dataset. CHAMELEON improves across all domains.

Domain	**Uniform**	CHAMELEON
Python	19.98	16.53
Java	19.27	15.53
C	28.24	22.58
C++	25.16	21.09
Go	30.25	19.26
Ruby	21.78	17.83
PHP	9.45	7.43
Average PPL (↓)	22.02	**17.18**
# Domains Over Uniform	-	7/7

Table 22: **Per-domain perplexity compared with the Uniform baseline for fine-tuning with 684M parameter models on the 7 languages in the Wiki40b dataset.**

Domain	Uniform	CHAMELEON (α^{FT})	CHAMELEON (α^{PT})
French	6.86	6.51	7.14
German	10.12	8.78	10.85
Italian	13.29	12.42	14.42
Spanish	8.41	8.04	8.70
Portuguese	8.00	7.78	8.14
Dutch	13.98	12.30	15.05
Polish	5.07	4.21	5.31
Average PPL (↓)	9.43	8.58	9.94

Table 23: **Per-domain perplexity compared with the Uniform baseline for fine-tuning with 684M parameter models on the 7 languages in the Stack dataset.**

Domain	Uniform	CHAMELEON (α^{FT})	CHAMELEON (α^{PT})
Python	19.98	16.53	20.11
Java	19.27	15.53	19.32
C	28.24	22.58	25.02
C++	25.16	21.09	23.83
Go	30.25	19.26	28.66
Ruby	21.78	17.83	21.75
PHP	9.45	7.43	9.47
Average PPL (↓)	22.02	17.18	21.17