Chain-based Distillation for Effective Initialization of Variable-Sized Small Language Models

Anonymous ACL submission

Abstract

Deploying language models under diverse memory and latency constraints requires a family of small language models (SLMs) at multiple scales. Training each target from scratch is expensive, while direct knowledge distillation repeats large-teacher inference for every target and often becomes unstable under large teacher-student capacity gaps. We formulate this challenge as *deployment-family initialization*, which aims to amortize teacher supervision across many variable-sized SLMs. To address this problem, we propose **Chain-based Distillation (CBD)**. CBD builds a chain of reusable anchors, where each anchor is an intermediate model at a discrete scale. Each anchor learns from its nearest larger predecessor through adjacent stepwise distillation. Target SLMs at intermediate sizes are then initialized by locally interpolating the two neighboring anchors. Our main experiments use a GPT2-based decoder-only chain, a public and architecture-compatible testbed for GPT-style SLM families, allowing us to isolate the role of local chain continuity in providing effective warm starts for many target sizes without repeatedly querying the source model. Across target scales from 138M to 537M parameters, CBD reduces the recovery training cost to match or surpass scratch-trained baselines and accelerates convergence under matched token budgets. Additional results of a Pythia-based chain and heterogeneous-teacher settings in the appendix further show that the chain-initialization principle extends beyond GPT2 architectures. The code is available at `https://anonymous.4open.science/r/CBD-1BC4`.

1 Introduction

Large language models (LLMs) (Touvron et al., 2023; Team, 2024; Yang et al., 2025) have become the dominant backbone for general-purpose language systems (Cui et al., 2023; Chen et al., 2023), yet their deployment cost remains prohibitive under realistic memory and latency constraints. In practice, deployment rarely calls for a single compressed model. Instead, different devices, serving tiers, and fallback settings require a *family* of small language models (SLMs) spanning multiple scales. This deployment setting calls for efficient construction of many deployable SLMs rather than one-off compression of a single teacher-student pair.

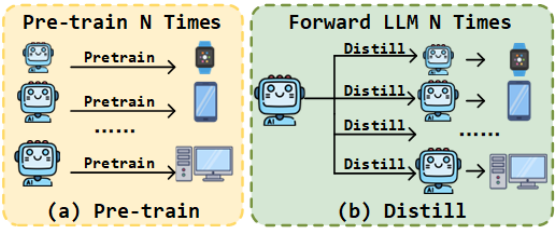


Figure 1: Comparison between (a) pre-training from scratch, (b) direct knowledge distillation for constructing N variable-sized SLMs. Both methods have repetitive, high-overhead distillation or training processes.

A straightforward solution is to train one model for each target size (Devlin et al., 2019; Singer et al., 2024), but this duplicates pretraining cost across the family, as shown in Figure 1a. Direct knowledge distillation (KD) (Yao et al., 2023; Hu et al., 2025) reduces the cost of each individual target, yet it still requires repeated teacher inference for every target model and often becomes brittle when the teacher is much larger than the student, as shown in Figure 1b. Prior work (Muralidharan et al., 2024; Zhang et al., 2023a; Mirzadeh et al., 2020; Zhang et al., 2023b) has shown that large teacher-student gaps can degrade transfer quality, and that introducing intermediate models can partially alleviate this issue. This motivates a more structural question: can we amortize teacher supervision over many target sizes, rather than spend it independently on each one?

We formulate this setting as *deployment-family initialization*. To answer it, as shown in Figure 2a, we propose a novel framework called **Chain-based Distillation (CBD)** for initializing variable-sized SLM families from a shared teacher signal. CBD

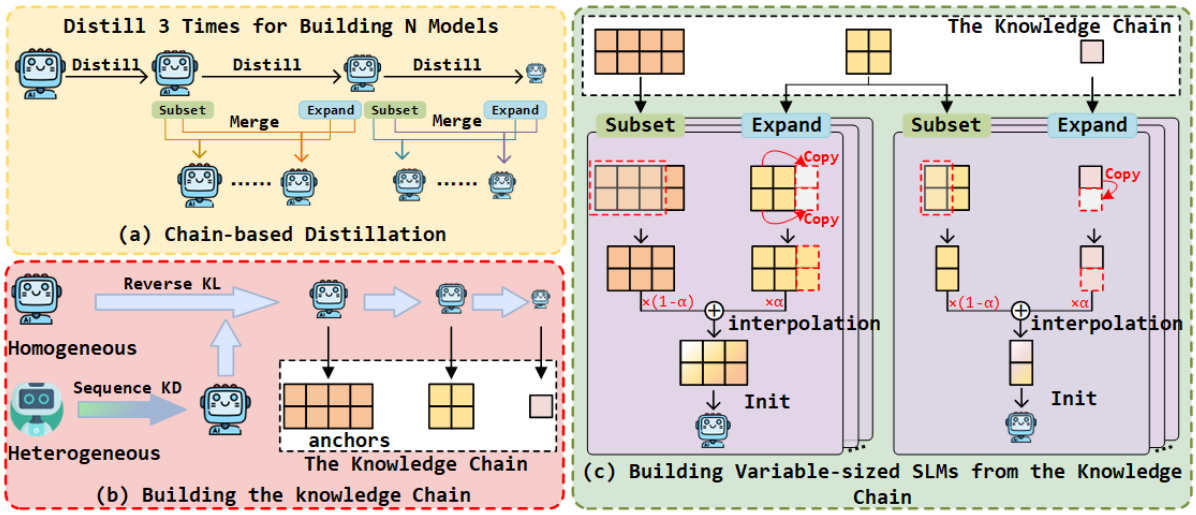


Figure 2: The overall framework of the proposed CBD. (a) CBD achieves a significant reduction in cumulative computational cost by leveraging a structured knowledge chain. (b) **Knowledge chain construction**: propagating knowledge from the source LLM to a sequence of anchors via stepwise distillation (homogeneous case) or bridge distillation (heterogeneous case). (c) **Variable-sized SLM building**: rapid initialization of target SLMs through parameter interpolation between adjacent anchors.

first builds a *chain*, namely a sequence of reusable anchor models ordered by scale, as shown in Figure 2b. It then initializes an intermediate target model by aligning the two neighboring anchors around that target size and locally interpolating their parameters, as shown in Figure 2c. When the source teacher is architecturally different from the deployment family, CBD introduces a bridge stage to transfer knowledge into a chain-compatible family before interpolation. In this way, CBD separates *knowledge injection* from *deployment-family construction*.

Specifically, to construct and train the chain, CBD adopts adjacent **stepwise distillation**. The motivation comes from prior work (Mirzadeh et al., 2020; Zhang et al., 2023b) showing that large teacher-student gaps can make distillation inefficient or unstable, while intermediate models can improve knowledge transfer. CBD therefore decomposes a large compression jump into **a sequence of smaller local transfers**, so that each anchor learns from its immediate larger predecessor. Here, stepwise distillation serves as the mechanism that makes a reusable anchor chain feasible for multi-scale initialization.

Once the anchors are constructed, CBD uses them as local coordinates for initializing intermediate target models. The target parameters are obtained by interpolating neighboring anchors within an architecture-compatible chain. This design follows the practical observation (Wortsman et al., 2022; Yadav et al., 2023) that parameter-space combination is most reliable when the participating models are structurally aligned. Accordingly, CBD performs interpolation within a chain where tokenization, layer ordering, and parameter structure remain comparable. In this paper, we mainly choose a GPT2-based decoder-only chain as the main controlled testbed because it provides a clean, public, and structurally coherent setting for isolating how local chain continuity supports strong warm starts for many target sizes.

Experiments on the GPT2-based (Radford et al., 2019) chain show that CBD substantially reduces the recovery training cost to match or surpass scratch-trained baselines across target sizes from 138M to 537M parameters. Additional Pythia-based (Biderman et al., 2023a) experiments further validate the same deployment-chain principle in another open-weight GPT-style family, where CBD remains effective under the same anchor-interpolation framework. Heterogeneous-source experiments further show that stronger external teachers can be incorporated through bridge mapping before entering the architecture-compatible chain. Overall, these results support CBD as an effective initialization strategy for dense, architecture-compatible SLM deployment families.

2 Related Work

2.1 Knowledge Distillation in LLMs

Knowledge distillation transfers knowledge from a larger teacher to a smaller student (Hinton et al.,

2015). In LLMs, this process is commonly studied in both black-box and white-box settings. Black-box methods distill from teacher-generated outputs, such as chain-of-thought reasoning (Wang et al., 2022b; Hsieh et al., 2023; Wang et al., 2023a) or in-context learning examples (Li et al., 2024; Duan et al., 2024), while white-box methods additionally exploit logits or intermediate representations from open-source teachers (Gu et al., 2023; Agarwal et al., 2024; Liang et al., 2023). A recurring concern in this literature is that a very large teacher is not always the most effective one for a much smaller student, because a large capacity gap can make transfer unstable or inefficient (Mirzadeh et al., 2020; Zhang et al., 2023b). CBD is related to this line of work but addresses a different level of the problem: existing KD methods mainly improve a *single* teacher-to-student transfer path, whereas CBD studies how a small set of reusable anchors can amortize teacher supervision over *many* target SLMs (Shing et al., 2025; Shin et al., 2026).

2.2 Model-Family Initialization

A complementary direction studies model reuse for constructing scalable model families from large pre-trained networks. Learngene-style methods transfer compact subnetworks from a larger model to initialize smaller targets (Wang et al., 2022a, 2023b; Shi et al., 2024; Xia et al., 2024), while Chain-of-Model supports multi-scale inference through a shared backbone with dynamic activation (Wang et al.). CBD differs from both in mechanism and goal. It does not search for a reusable subnetwork or maintain a single supernet for shared inference; instead, it constructs a sequence of distilled anchors and uses local interpolation between neighboring anchors to initialize *standalone* target SLMs at multiple scales.

3 Methodology

CBD initializes many variable-sized SLMs by reusing a small set of architecture-compatible anchors. As shown in Figure 2a, CBD first constructs a *deployment chain* that organizes anchors by scale, then propagates source knowledge through adjacent stepwise distillation. A bridge stage provides an entry point for heterogeneous source teachers by mapping their behavior into the chain family. After the anchors are trained, CBD initializes a target SLM by aligning and locally interpolating its two neighboring anchors. This construction keeps the core interpolation process inside a compatible deployment chain while allowing source knowledge to come from broader model families.

3.1 Building the Deployment Chain

To reuse a small number of checkpoints for many target sizes, CBD first constructs a deployment chain that serves as the interpolation backbone for the SLM family. Let $\mathcal{A} = \{A_1, A_2, \ldots, A_M\}$ denote anchors ordered from large to small, with $\mathrm{Size}(A_1) > \mathrm{Size}(A_2) > \cdots > \mathrm{Size}(A_M)$. Each anchor is an intermediate checkpoint at a discrete model scale. Together, these anchors form local neighborhoods in parameter space, where a target model can be initialized from the two closest scales around it.

The anchor construction is guided by local architectural compatibility. Neighboring anchors share the same tokenizer, comparable decoder blocks, compatible positional embeddings, and alignable attention and MLP layouts. These properties make deterministic shape alignment feasible when a target model lies between two anchors. The chain is therefore sparse at the family level, since only a few anchors are trained, while remaining locally dense enough to support interpolation within each adjacent interval.

In the main experiments, we instantiate the deployment chain with a GPT2-based family,

$$\mathcal{A}_{\mathrm{GPT2}} = \{\mathrm{GPT2\text{-}L}, \mathrm{GPT2\text{-}M}, \mathrm{GPT2\text{-}B}\},$$

with GPT2-XL as the source model for constructing the chain. This family provides a public decoder-only testbed with shared vocabulary and smoothly varying depth and width, which allows us to isolate how local chain continuity supports target initialization. Additional Pythia-based experiments in the appendix verify the same deployment-family initialization principle in another open-weight GPT-style family.

3.2 Knowledge Propagation Along the Chain

After the deployment chain is specified, CBD injects source knowledge into its anchors through adjacent local transfers. The core propagation route operates inside the architecture-compatible chain, where each anchor is distilled from its nearest larger predecessor. When the source model already belongs to the chain family, this route can be applied directly. For source models from other families, CBD first introduces a bridge interface

that maps their behavior into a chain-compatible model, and the same adjacent propagation route is then applied.

Homogeneous propagation. When the source model shares the chain architecture and vocabulary, CBD propagates knowledge sequentially,

$$A_0 \to A_1 \to A_2 \to \cdots \to A_M,$$

where A_0 is the source model and each A_i learns from its immediate larger predecessor. Each local student is optimized with reverse KL divergence,

$$\mathcal{L}_{\text{KD}} = \mathbb{E}_{x\sim\mathcal{D}}\left[\sum_t P_S(x_t \mid x_{<t}) \log \frac{P_S(x_t \mid x_{<t})}{P_T(x_t \mid x_{<t})}\right], \tag{1}$$

where P_T and P_S denote teacher and student next-token distributions. This adjacent route converts a large compression jump into nearby transfers, and our ablations show smoother optimization and stronger small-anchor fidelity than direct compression.

The homogeneous route defines the deployment-chain propagation used throughout CBD. It constructs anchors in a shared parameter space, which later enables shape alignment and local interpolation for variable-sized target models.

Bridge entry for heterogeneous source models. To incorporate source teachers from other model families, CBD maps their behavior into a chain-compatible bridge model before anchor propagation. Given a heterogeneous source teacher T_{src}, such as a Llama- or Qwen-style model, CBD trains a bridge model B whose architecture matches the deployment chain. The bridge is optimized with sequence-level KD,

$$\mathcal{L}_{\text{bridge}} = \mathbb{E}_{(x,\hat{y})\sim T_{\text{src}}}\left[-\log P_B(\hat{y} \mid x)\right], \tag{2}$$

where $\hat{y}$ is generated by T_{src}. The trained bridge then enters the same compatible chain,

$$B \to A_1 \to A_2 \to \cdots \to A_M.$$

This bridge entry transfers cross-family source behavior into the deployment family before the standard adjacent propagation and interpolation steps.

3.3 Why Adjacent Stepwise Distillation Is Effective

In this section, we further validates adjacent stepwise distillation from both theoretical and empirical perspectives. On the theoretical side, Appendix B decomposes each transfer into a statistical term and an approximation term. The resulting bound favors the chain route when the accumulated local transfer cost is lower than the cost of a single large compression step. This analysis explains why decomposing a large teacher-student gap into adjacent transfers can improve anchor construction.

On the empirical side, Section 4.4 instantiates the same condition through direct-versus-stepwise and interpolation diagnostics. Figure 5 compares distillation baselines and loss trajectories, showing that adjacent propagation produces smoother optimization and stronger downstream performance than direct compression. Table 3 further shows that the CBD route preserves stronger GPT2-B anchor fidelity in the extreme GPT2-XL-to-GPT2-B setting. These results show that stepwise distillation constructs a stable anchor chain and supplies the local structure required by CBD interpolation.

3.4 Amortized Target Initialization by Local Interpolation

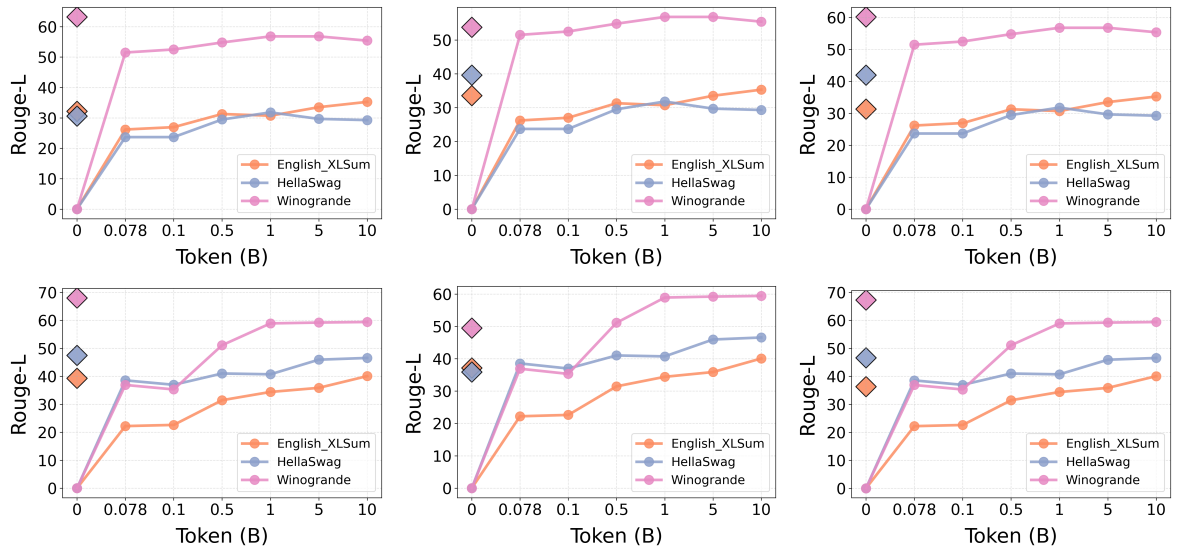
After the anchors are trained, CBD uses neighboring anchors as local coordinates in weight space. For a target model S between A_{small} and A_{large}, CBD aligns both checkpoints to the target shape by expanding the smaller anchor and subsetting the larger anchor.

The target initialization is computed as

$$\Theta_S = \alpha\, \phi_{\text{small}}(\Theta_{A_{\text{small}}}) + (1-\alpha)\, \phi_{\text{large}}(\Theta_{A_{\text{large}}}), \tag{3}$$

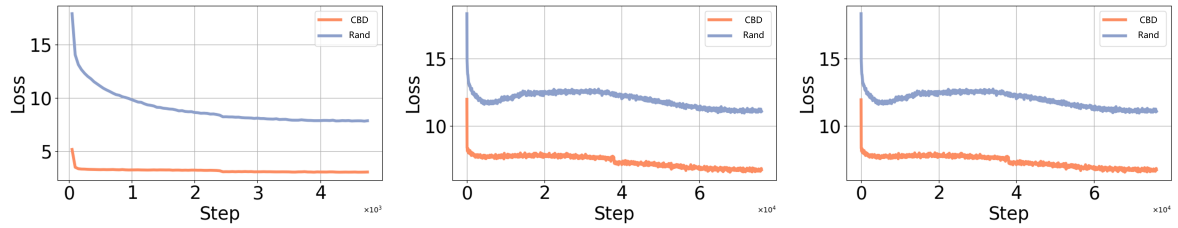
where ϕ_{small} and ϕ_{large} are shape-alignment operators, and α controls the contribution of the smaller anchor. In our implementation, the anchor closer to the target scale receives weight 0.9, while the farther anchor receives weight 0.1. This preserves structural proximity while incorporating complementary knowledge from the neighboring scale.

The interpolation stage amortizes the cost of chain construction over many target models. Once the anchors are trained, each additional target only requires deterministic shape alignment and parameter interpolation, while scratch training and direct distillation repeat training or teacher inference for every target. Local interpolation also combines the compact structure of the smaller anchor with the richer representations of the larger anchor. Ablations show that the optimal α tracks architectural proximity and that denser chains provide more accurate initialization.



(a) GPT2-XL (b) Llama3-8B (c) Qwen3-4B

Figure 3: Recovery-training efficiency of CBD-initialized target SLMs. Curves denote scratch-trained baselines, and markers denote CBD initialization before recovery training. *For visual consistency, all y-axes are labeled as ROUGE-L, while classification tasks report accuracy.*



(a) GPT2-XL (b) Llama3-8B (c) Qwen3-4B

Figure 4: Convergence of a 138M target SLM initialized by CBD and by random initialization under the same 78M recovery-training tokens. CBD begins from a lower loss and reaches target loss levels faster, illustrating the optimization benefit of local anchor interpolation.

4 Experiments

4.1 Implementation Details

We evaluate CBD as an amortized initialization method for deployment-oriented SLM families. The main deployment chain is instantiated with GPT2-L, GPT2-M, and GPT2-B, using GPT2-XL (Radford et al., 2019) as the same-family source model. We initialize target SLMs from 138M to 537M parameters by interpolating adjacent anchors. For brevity, we shows the results of SLMs with 537M parameters in Appendix G. To test transfer beyond GPT2 checkpoints, we additionally construct a Pythia-based (Biderman et al., 2023b) deployment chain and report its full results in Appendix M.

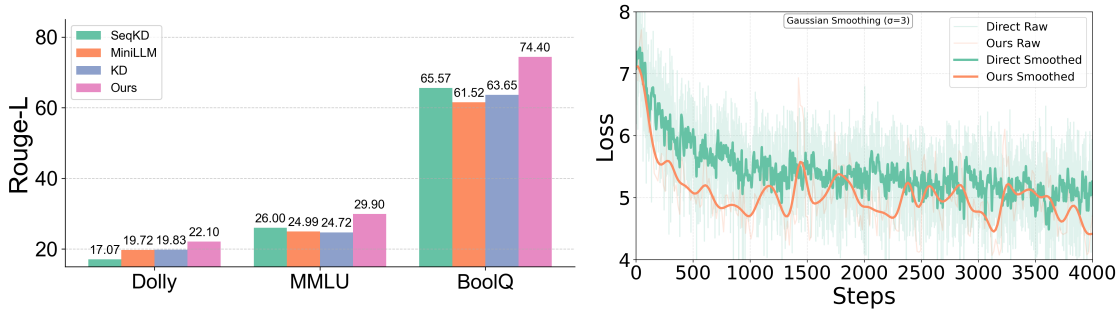
We also take Llama3-8B (Dubey et al., 2024) and Qwen3-4B (Yang et al., 2025) as heterogeneous source teachers through the bridge stage. This setup keeps the deployment chain fixed while varying the source of injected knowledge. We report accuracy for multiple-choice and classification benchmarks and ROUGE-L for generation tasks. The detailed implementation, datasets, target architectures are provided in Appendix C, Appendix D, Appendix E.

4.2 GPT2-based Chain Results

Recovery-training efficiency. A key benefit of CBD is reducing the recovery training required after initialization. Figure 3 compares CBD initializations without additional recovery training against randomly initialized target models trained with increasing token budgets. CBD provides a strong step-zero initialization across target sizes and source settings. For the 138M target, the CBD

Token	Init	SLM-138M						SLM-220M					
		MMLU	XLsum	HellaS	WinoG	BoolQ	Avg	MMLU	XLsum	HellaS	WinoG	BoolQ	Avg
78M	Rand	29.00	26.18	23.68	51.50	68.00	39.67	28.40	28.48	23.87	49.70	66.50	39.39
	CBD	31.00	30.68	31.15	62.00	76.80	**46.33**	28.50	26.68	26.84	58.00	69.50	**41.90**
100M	Rand	29.10	26.95	23.68	52.50	67.80	40.01	29.30	26.87	23.67	52.80	69.10	40.35
	CBD	30.90	31.54	30.94	61.90	76.60	**46.38**	28.20	28.27	28.76	59.20	70.70	**43.03**
Token	Init	SLM-277M						SLM-380M					
		MMLU	XLsum	HellaS	WinoG	BoolQ	Avg	MMLU	XLsum	HellaS	WinoG	BoolQ	Avg
78M	Rand	26.40	27.30	23.77	50.20	67.50	39.03	26.70	22.20	38.54	36.90	67.90	38.45
	CBD	28.20	32.36	29.57	61.49	71.80	**44.68**	30.90	34.14	46.04	68.60	74.90	**50.92**
100M	Rand	27.90	27.68	24.39	47.50	66.70	38.83	28.50	22.62	36.96	35.30	65.10	37.70
	CBD	31.30	35.31	29.57	58.47	73.00	**45.53**	28.80	35.53	43.23	66.20	75.00	**49.75**

Table 1: Equal-budget evaluation on the GPT2-based deployment chain for 138M, 220M, 277M and 380M targets. All models are evaluated after the same amount of recovery training. CBD improves average performance over random initialization, showing that learned anchors provide reusable initialization points for a family of SLMs. Classification tasks report accuracy, and generation tasks report ROUGE-L.



(a) Distillation baselines

(b) Training stability

Figure 5: Effect of adjacent transfer on large-gap compression. (a) downstream performance of a 138M target initialized by different distillation routes. (b) training loss for direct compression and stepwise CBD propagation.

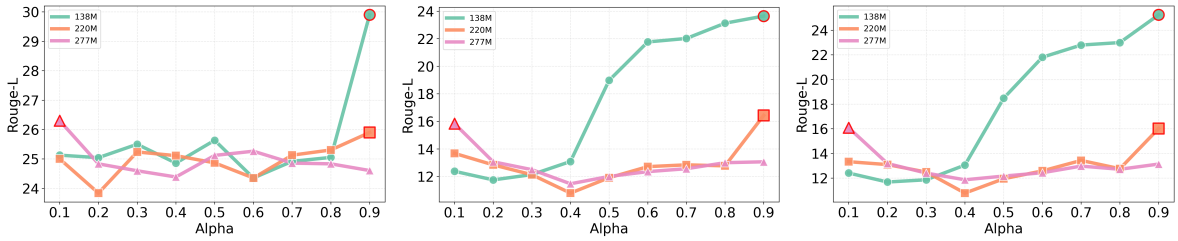
initialization on HellaSwag remains above scratch-trained models even after up to 10B tokens. For the 380M target, CBD reaches comparable or stronger English XLSum performance while saving at least 5B recovery tokens. These results show that the deployment chain stores reusable knowledge that would otherwise require large-scale training to recover from scratch.

Convergence under matched budgets. Optimization behavior further highlights the advantage of anchor-based initialization. Figure 4 compares CBD and random initialization under the same recovery-training budget. For the 138M target, CBD starts from a substantially lower loss across the displayed source settings. Its initial loss is already lower than the final loss reached by the random-initialized baseline. This step-zero advantage indicates that local anchor interpolation places the target model in a more favorable region of the parameter space, enabling faster convergence under limited token budgets.

Downstream performance. Fixed-budget evaluation tests whether the initialization gains translate into downstream performance. Table 1 compares CBD and random initialization at 78M and 100M tokens. CBD improves the average score across all target sizes. At 78M tokens, the 138M target improves from 39.67 to 46.33, while the 380M target improves from 38.45 to 50.92. At 100M tokens, the 380M target improves from 37.70 to 49.75, with English XLSum increasing from 22.62 to 35.53. These gains show that once the anchor chain is constructed, multiple target models can inherit stronger initializations without repeatedly querying the source teacher.

4.3 Bridge-based Source Injection on Generation Tasks

The bridge stage enables CBD to inject knowledge from heterogeneous source teachers while keeping the deployment chain fixed. Generation-style benchmarks provide a direct testbed for this source-injection effect, since they are sensitive to



(a) GPT2-XL (b) Llama3-8B (c) Qwen3-4B

Figure 6: Sensitivity of the interpolation coefficient α. Each curve evaluates target initializations obtained by interpolating neighboring anchors. The best α tracks the architectural proximity between the target model and its adjacent anchors, showing that the deployment chain provides a usable local parameter space.

Source teacher	SLM-138M					SLM-380M				
	Dolly	Self-Inst	SInst	Vicuna	Avg	Dolly	Self-Inst	SInst	Vicuna	Avg
GPT2-XL	22.10	9.35	14.26	15.31	12.97	26.20	11.01	16.90	15.77	17.47
Qwen3-4B	**25.25**	**11.79**	**15.64**	15.53	**14.32**	**27.55**	**13.58**	21.51	**17.49**	**20.03**
Llama3-8B	22.83	10.99	15.48	**15.86**	14.11	26.63	12.95	**21.65**	16.75	19.49

Table 2: Generation-task evaluation of bridge-based source injection. The deployment chain is fixed to the GPT2-based family, while the source teacher entering the bridge varies across GPT2-XL, Qwen3-4B, and Llama3-8B.

Route	LLM	GPT2-L	GPT2-M	GPT2-B
Direct	27.94	–	–	23.30
CBD	27.94	27.77	27.63	**26.02**

Table 3: Zero-shot Dolly performance of intermediate anchors. The adjacent route preserves anchor quality as scale decreases, while direct compression from GPT2-XL to GPT2-B loses fidelity under a large capacity gap.

high-level semantic priors and response-generation ability. The GPT2-based deployment chain is kept unchanged, and only the source teacher entering the bridge is varied. Table 2 shows that heterogeneous source teachers provide stronger generation priors after bridge mapping. Qwen3-4B improves the average score from 12.97 to 14.32 on SLM-138M and from 17.47 to 20.03 on SLM-380M, while Llama3-8B also improves both target scales. These results indicate that the bridge stage can inject external semantic knowledge while keeping target initialization inside the same architecture-compatible deployment chain.

4.4 Ablation Studies

The ablations focus on the following design factors: adjacent stepwise propagation, interpolation behavior, and compatibility with stronger distillation objectives.

Adjacent transfers stabilize large-gap compression. We first examine whether the deployment chain improves over direct compression from the source model to a small target. Figure 5 compares CBD with conventional distillation baselines, including sequence-level KD, MiniLLM (Gu et al., 2023), and vanilla KD, and also visualizes the training loss under direct and stepwise routes. The adjacent route produces smoother optimization and stronger downstream performance, showing that intermediate anchors buffer the capacity gap between the source and small target. Table 3 gives the same pattern at the anchor level: direct GPT2-XL→GPT2-B compression reaches 23.30 on Dolly, while the CBD route preserves a stronger GPT2-B anchor at 26.02. This result supports adjacent stepwise distillation as the mechanism that constructs a stable deployment chain.

Interpolation weights reflect target-anchor proximity. Figure 6 studies the interpolation coefficient α. The best coefficient varies with target size, and targets closer to the smaller anchor favor larger contribution from that anchor. This pattern supports the view that adjacent anchors form a usable local parameter space. The interpolation coefficient acts as a scale-aware mechanism for selecting information from the two neighboring anchors.

Local interpolation improves over single-anchor expansion. Table 4 compares CBD with a single-

Source	Method	SLM-220M				SLM-277M			
		Dolly	MMLU	BoolQ	Avg	Dolly	MMLU	BoolQ	Avg
GPT2-XL	Single	12.33	25.26	65.54	34.38	11.98	25.45	66.71	34.71
	CBD	15.89	25.90	66.70	**36.16**	14.72	26.30	68.40	**36.47**
Llama3-8B	Single	13.03	25.51	65.50	34.68	13.26	24.88	67.39	35.17
	CBD	16.43	25.53	67.32	**36.43**	15.83	24.58	66.64	**35.69**
Qwen3-4B	Single	13.41	25.33	66.54	35.09	12.69	24.50	66.25	34.48
	CBD	16.02	25.75	64.60	**35.46**	16.08	28.90	67.40	**37.46**

Table 4: Local interpolation versus single-anchor expansion. Single uses only the smaller anchor, while CBD aligns and interpolates both neighboring anchors. The gains show that target initialization benefits from information on both sides of the target scale.

Token	Target	Direct KD	Single	CBD
1B	113M	34.7	**39.2**	38.4
1B	220M	36.8	37.6	**38.7**
1B	300M	38.8	40.2	**40.7**
5B	113M	38.0	**40.6**	39.6
5B	220M	38.3	39.3	**41.2**
5B	300M	40.4	41.3	**41.5**

Table 5: Pythia-based deployment-chain validation. Direct KD directly distills each target, Single uses only the smaller neighboring anchor, and CBD interpolates two neighboring anchors. Scores are averaged over the evaluation suite, with full results reported in Appendix M.

Chain length	BoolQ	ARC-E	WinoG	Avg
2 anchors	65.60	24.21	48.92	46.24
4 anchors	65.82	28.07	48.84	**47.58**

Table 6: Effect of deployment-chain density. A denser anchor chain provides finer local coordinates for interpolation and improves target initialization.

expansion baseline that initializes the target only from the smaller anchor. CBD uses both sides of the target scale and combines compact structure from the smaller anchor with richer representations from the larger anchor. For GPT2-XL, CBD improves the average score from 34.38 to 36.16 on SLM-220M and from 34.71 to 36.47 on SLM-277M. Under Qwen3-4B, CBD improves SLM-277M from 34.48 to 37.46. These gains show that local interpolation provides stronger initialization than relying on a single anchor.

4.5 Pythia-based Deployment Chain Validation

We further evaluate CBD on a Pythia-based deployment family to test whether the chain-initialization principle transfers to another open-weight GPT-style model family. Here, we take Pythia-1.4B as the source teacher model. Table 5 reports the average score over eight multiple-choice tasks under 1B and 5B recovery budgets, with full task-level results in Appendix M. CBD obtains the best average scores on the 220M and 300M targets at both budgets, while single-anchor expansion remains stronger for the smallest 113M target. This pattern is consistent with the mechanism analysis: local interpolation becomes more useful when the target can benefit from both neighboring anchors.

Finally, we also verify the impact of deployment chain density on SLM results. We conduct this by comparing a sparse pythia-based chain with a denser pythia-based chain under the same target scale. As shown in Table 6, the 4-anchor chain improves the average score from 46.24 to 47.58, with a notable gain on ARC-E from 24.21 to 28.07. This result shows that reducing the gap between neighboring anchors makes interpolation more precise and strengthens the deployment-chain construction.

5 Conclusion

We introduced an amortized initialization framework for variable-sized SLMs called **Chain-based Distillation (CBD)**. CBD constructs an architecture-compatible anchor chain through stepwise distillation and initializes intermediate targets by local interpolation. Experiments show that CBD provides stronger warm starts, faster convergence, and better performance under the same recovery tokens. Additionally, Pythia-based and bridge-source results further supporting CBD for scalable deployment-family initialization.

Limitations

This work focuses on amortized initialization for decoder-only SLM families with architecture-compatible anchors. The current experiments mainly use GPT2-based and Pythia-based deployment chains, with heterogeneous source teachers incorporated through a bridge stage. A broader study across more model families, automatic anchor placement strategies, and adaptive interpolation policies would further clarify how CBD scales to different deployment regimes. These directions can extend CBD from a controlled family-level initialization framework toward a more general toolkit for constructing deployable SLM families.

Ethical Considerations

All datasets used in the experiments are publicly available, and we adhered to standard practices for data usage and reporting. The methods and findings are intended solely for academic research and are not designed for harmful applications. We ensured that the experimental protocols, theoretical derivations, and released source code conform to the principles of fairness, transparency, and reproducibility. No conflicts of interest or ethical concerns are associated with this work.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. 2024. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations*.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023a. Pythia: A suite for analyzing large language models across training and scaling. *ArXiv*, abs/2304.01373.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, and 1 others. 2023b. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Wei Chen, Qiushi Wang, Zefei Long, Xianyin Zhang, Zhongtian Lu, Bingxuan Li, Siyuan Wang, Jiarong Xu, Xiang Bai, Xuanjing Huang, and Zhongyu Wei. 2023. Disc-finllm: A chinese financial large language model based on multiple experts fine-tuning. *arXiv preprint arXiv:2310.15205*.

Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, and 1 others. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *ArXiv*, abs/1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Jiaxi Cui, Zongjian Li, Yang Yan, Bohua Chen, and Li Yuan. 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. *CoRR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Yifei Duan, Liu Li, Zirui Zhai, and Jinxia Yao. 2024. In-context learning distillation for efficient few-shot fine-tuning. *ArXiv*, abs/2412.13243.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *arXiv preprint arXiv:2305.02301*.

Shengxiang Hu, Guobing Zou, Song Yang, Shiyi Lin, Yanglan Gan, Bofeng Zhang, and Yixin Chen. 2025. Large language model meets graph neural network in knowledge distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17295–17304.

Yichuan Li, Xiyao Ma, Sixing Lu, Kyumin Lee, Xiaohu Liu, and Chenlei Guo. 2024. Mend: Meta demonstration distillation for efficient and effective in-context learning. *ArXiv*, abs/2403.06914.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. 2023. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, pages 20852–20867. PMLR.

Laurence Liang. 2024. Stacking small language models for generalizability. *arXiv preprint arXiv:2410.15570*.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. 2024. Fineweb-edu: the finest collection of educational content.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 5191–5198.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37:41076–41102.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *ArXiv*, abs/1808.08745.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Mohammad Samragh, Iman Mirzadeh, Keivan Alizadeh Vahid, Fartash Faghri, Minsik Cho, Moin Nabi, Devang Naik, and Mehrdad Farajtabar. 2024. Scaling smart: Accelerating large language model pre-training with small model initialization. *arXiv preprint arXiv:2409.12903*.

Boyu Shi, Shiyu Xia, Xu Yang, Haokun Chen, Zhiqiang Kou, and Xin Geng. 2024. Building variable-sized models via learngene pool. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 14946–14954.

Donghyeok Shin, Yeongmin Kim, Suhyeon Jo, Byeonghu Na, and Il-Chul Moon. 2026. Amid: Knowledge distillation for llms with α-mixture assistant distribution. *Preprint*, arXiv:2510.15982.

Makoto Shing, Kou Misaki, Han Bao, Sho Yokoi, and Takuya Akiba. 2025. Taid: Temporally adaptive interpolated distillation for efficient knowledge transfer in language models. *Preprint*, arXiv:2501.16937.

Philipp Singer, Pascal Pfeiffer, Yauhen Babakhin, Maximilian Jeblick, Nischay Dhankhar, Gabor Fodor, and Sri Satish Ambati. 2024. H2o-danube-1.8 b technical report. *arXiv preprint arXiv:2401.16818*.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. 2023a. Scott: Self-consistent chain-of-thought distillation. *arXiv preprint arXiv:2305.01879*.

Qiu-Feng Wang, Xin Geng, Shu-Xia Lin, Shi-Yu Xia, Lei Qi, and Ning Xu. 2022a. Learngene: From open-world to your learning task. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8557–8565.

Qiufeng Wang, Xu Yang, Shuxia Lin, Jing Wang, and Xin Geng. 2023b. Learngene: Inheriting condensed knowledge from the ancestry model to descendant models. *arXiv preprint arXiv:2305.02279*.

Xiaohua Wang, Kaitao Song, Xu Tan, Huiqiang Jiang, Chengruidong Zhang, Yongliang Shen, Cen LU, Zihao Li, Zifan Song, Caihua Shan, and 1 others. Chain-of-model learning for language model. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed H. Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 13484–13508.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, and 1 others. 2022c. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*, 2:2.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and 1 others. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pages 23965–23998. PMLR.

Shiyu Xia, Miaosen Zhang, Xu Yang, Ruiming Chen, Haokun Chen, and Xin Geng. 2024. Transformer as linear expansion of learngene. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 16014–16022.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. 2023. Ties-merging: Resolving interference when merging models. *Advances in neural information processing systems*, 36:7093–7115.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Xin Yao, Ziqing Yang, Yiming Cui, and Shijin Wang. 2023. Minirbt: A two-stage distilled small chinese pre-trained model. *Preprint*, arXiv:2304.00717.

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

Chen Zhang, Dawei Song, Zheyu Ye, and Yan Gao. 2023a. Towards the law of capacity gap in distilling language models. *arXiv preprint arXiv:2311.07052*.

Chen Zhang, Yang Yang, Jiahao Liu, Jingang Wang, Yunsen Xian, Benyou Wang, and Dawei Song. 2023b. Lifting the curse of capacity gap in distilling language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4535–4553, Toronto, Canada. Association for Computational Linguistics.

A Appendix

We organize the appendix as follows.

- In Appendix B: theoretical analysis for local stepwise transfer under homogeneous and bridge-plus-chain settings.
- In Appendix C: experimental details for anchor construction, bridge training, target fine-tuning, and scratch pre-training.
- In Appendix D: detailed information of datasets used in the main paper.
- In Appendix E: detailed architectures of the initialized SLMs.
- In Appendix F: detailed values behind the recovery-training efficiency figure.
- In Appendix G: additional CBD and Rand comparisons using Llama3-8B and Qwen3-4B as heterogeneous source teachers, together with 537M target results.
- In Appendix H: additional convergence-speed comparisons under different recovery-token budgets.
- In Appendix I: performance gaps between CBD and random initialization as pre-training tokens increase on Dolly.
- In Appendix J: Qwen3-based CBD results.
- In Appendix K: comparison with Hyper-Cloning and FSLM.
- In Appendix L: CBD results at very small target scales.
- In Appendix M: full Pythia-family deployment-chain results across recovery budgets.
- In Appendix N: future work.

B Theoretical Analysis

We provide a unified theoretical analysis for k-level stepwise distillation as employed in Chain-based Distillation. This section first introduces notation and the general single-step learning bound, then analyzes the homogeneous (same-architecture) case, and finally extends the results to the heterogeneous (cross-architecture) case.

B.1 Unified notation and preliminaries

Let the ground-truth target function be $f^\star$. We denote hypothesis classes in a distillation chain as

$$\mathcal{G} \;\to\; \mathcal{H}_1 \;\to\; \mathcal{H}_2 \;\to\; \cdots \;\to\; \mathcal{H}_k \;\to\; \mathcal{S},$$

where $\mathcal{G}$ is the teacher (possibly heterogeneous), $\mathcal{H}_i$ are intermediate assistants, and $\mathcal{S}$ is the final student.

For any hypothesis class $\mathcal{X}$ we denote:

- $C_{\mathcal{X}}$: a capacity measure;
- $\rho_{\mathcal{X}\leftarrow\mathcal{Y}}$: the statistical convergence rate when $\mathcal{X}$ learns from $\mathcal{Y}$;
- $\varepsilon_{\mathcal{X}\leftarrow\mathcal{Y}}$: the approximation error when $\mathcal{X}$ mimics $\mathcal{Y}$.

General single-step bound. For any pair $(\mathcal{X},\mathcal{Y})$,

$$R(\hat{f}_{\mathcal{X}}) - R(f_{\mathcal{Y}}) \;\le\; O\!\left(\frac{C_{\mathcal{X}}}{n^{\rho_{\mathcal{X}\leftarrow\mathcal{Y}}}}\right) + \varepsilon_{\mathcal{X}\leftarrow\mathcal{Y}}. \tag{4}$$

B.2 Stepwise distillation with homogeneous source teachers

Direct distillation. Training $\mathcal{S}$ directly from $\mathcal{G}$ gives

$$R(\hat{s}) - R(f^\star) \;\le\; O\!\left(\frac{C_{\mathcal{G}}}{n^{\rho_{\mathcal{G}\leftarrow f^\star}}} + \frac{C_{\mathcal{S}}}{n^{\rho_{\mathcal{S}\leftarrow\mathcal{G}}}}\right) + \varepsilon_{\mathcal{G}\leftarrow f^\star} + \varepsilon_{\mathcal{S}\leftarrow\mathcal{G}}. \tag{5}$$

Stepwise distillation. Proceeding through $\mathcal{H}_1,\ldots,\mathcal{H}_k$ yields

$$R(\hat{s}) - R(f^\star) \;\le\; O\!\left(\frac{C_{\mathcal{G}}}{n^{\rho_{\mathcal{G}\leftarrow f^\star}}} + \sum_{i=1}^{k}\frac{C_{\mathcal{H}_i}}{n^{\rho_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}}}} + \frac{C_{\mathcal{S}}}{n^{\rho_{\mathcal{S}\leftarrow\mathcal{H}_k}}}\right) + \varepsilon_{\mathcal{G}\leftarrow f^\star} + \sum_{i=1}^{k}\varepsilon_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}} + \varepsilon_{\mathcal{S}\leftarrow\mathcal{H}_k}, \tag{6}$$

where $\mathcal{H}_0 := \mathcal{G}$.

Assumption B.1 (Rate improvement for homogeneous case)**.** *For all i,*

$$\rho_{\mathcal{S}\leftarrow\mathcal{G}} \;\leq\; \min\big(\rho_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}},\, \rho_{\mathcal{S}\leftarrow\mathcal{H}_k}\big)\,.$$

Assumption B.2 (Approximation decomposition for homogeneous case)**.**

$$\sum_{i=1}^{k} \varepsilon_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}} + \varepsilon_{\mathcal{S}\leftarrow\mathcal{H}_k} \;\leq\; \varepsilon_{\mathcal{S}\leftarrow\mathcal{G}}.$$

Conclusion. Under these assumptions, the multi-step bound (6) is asymptotically tighter than the direct bound (5), establishing the theoretical advantage of homogeneous stepwise distillation.

B.3 Stepwise distillation with heterogeneous source teachers

Now consider the case where the original large teacher belongs to a different architecture family $\mathcal{G}^{(A)}$ (e.g., Llama3-8B, Qwen3-4B). We insert a GPT2-XL anchor, denoted $\mathcal{G}^{(\mathrm{GPT2})}$, as a bridging assistant before continuing with GPT-2 style anchors.

Additional notation.

- Mapping operator Π_{map}: projects $\mathcal{G}^{(A)}$ outputs into the GPT-2 output space.
- Mapping residual δ_{map}: additional approximation error from projection mismatch.

The error from $\mathcal{G}^{(A)}$ to $\mathcal{G}^{(\mathrm{GPT2})}$ decomposes as

$$\begin{aligned}&\varepsilon_{\mathcal{G}^{(\mathrm{GPT2})}\leftarrow\mathcal{G}^{(A)}}\\ &\quad= \varepsilon_{\mathcal{G}^{(\mathrm{GPT2})}\leftarrow\Pi_{\mathrm{map}}(\mathcal{G}^{(A)})}\\ &\qquad+\delta_{\mathrm{map}}.\end{aligned} \tag{7}$$

Heterogeneous stepwise bound. Let $\mathcal{H}_{-1} := \mathcal{G}^{(A)}$ and $\mathcal{H}_0 := \mathcal{G}^{(\mathrm{GPT2})}$, then

$$\begin{aligned}R(\hat{s}) - R(f^\star) \leq O\Bigg(&\frac{C_{\mathcal{G}^{(A)}}}{n^{\rho_{\mathcal{G}^{(A)}\leftarrow f^\star}}}\\ &+\sum_{i=0}^{k}\frac{C_{\mathcal{H}_i}}{n^{\rho_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}}}}\\ &+\frac{C_{\mathcal{S}}}{n^{\rho_{\mathcal{S}\leftarrow\mathcal{H}_k}}}\Bigg)\\ &+\varepsilon_{\mathcal{G}^{(A)}\leftarrow f^\star}+\sum_{i=0}^{k}\varepsilon_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}}\\ &+\varepsilon_{\mathcal{S}\leftarrow\mathcal{H}_k},\end{aligned} \tag{8}$$

where $\varepsilon_{\mathcal{H}_0\leftarrow\mathcal{H}_{-1}}$ includes δ_{map}.

Assumption B.3 (Rate improvement across mapped chain)**.**

$$\rho_{\mathcal{S}\leftarrow\mathcal{G}^{(A)}} \;\leq\; \min\big(\rho_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}} \;:\; i=0,\ldots,k,\; \rho_{\mathcal{S}\leftarrow\mathcal{H}_k}\big).$$

Assumption B.4 (Approximation decomposition with mapping error)**.**

$$\delta_{\mathrm{map}} + \sum_{i=0}^{k} \varepsilon_{\mathcal{H}_i\leftarrow\mathcal{H}_{i-1}} + \varepsilon_{\mathcal{S}\leftarrow\mathcal{H}_k} \;\leq\; \varepsilon_{\mathcal{S}\leftarrow\mathcal{G}^{(A)}}.$$

Conclusion. Under these assumptions, the heterogeneous bound (8) is asymptotically tighter than direct distillation from $\mathcal{G}^{(A)}$ to $\mathcal{S}$. This formalizes the role of GPT2-XL as a bridging assistant for cross-architecture stepwise distillation.

C Experimental Details

C.1 Training Details

Our appendix reports two experimental settings. The first is a GPT2-based chain, which serves as the controlled testbed for isolating whether local chain continuity improves deployment initialization. The second is a Pythia-based chain, which transfers the same deployment-family initialization principle to a larger family of open models and serves as the main validation of the appendix study.

C.1.1 GPT2-based Chain

Our GPT2-based pipeline consists of four stages: (1) homogeneous distillation for constructing the deployment chain, (2) heterogeneous distillation, (3) downstream fine-tuning of SLMs, and (4) pre-training of randomly initialized SLMs on OpenWebText.

Homogeneous Distillation. We perform homogeneous distillation in two phases to construct the deployment chain using architectures identical to the target model. In the first phase, both the teacher model and intermediate anchors are supervised-fine-tuned (SFT) on the Dolly dataset for 3 epochs to align model behaviors. In the second phase, we adopt a policy distillation scheme inspired by MiniLLM. Specifically, we employ reverse KL divergence between the teacher and student policies on Dolly, collecting 256 samples per batch and running 4 inner optimization epochs per batch. The clipping parameter ϵ is set to 0.2, and the sequence length is capped at 512 tokens. During distillation, we sample from the teacher with a temperature of 1. The distillation proceeds for 5000 steps, and we

select the final model checkpoint based on Rouge-L score on the validation set.

Heterogeneous Distillation. For cross-architecture distillation, we first fine-tune two heterogeneous teacher models, Qwen-4B and LLaMA3-8B, on the Dolly dataset for 3 epochs using a batch size of 2 and a learning rate of 1e-5. We then use the final checkpoints of these teachers to generate pseudo answers for Dolly questions. These synthetic question-answer pairs are used to fine-tune a smaller proxy model, GPT2-XL, for 10 epochs with a batch size of 8 and a learning rate of 1e-5. The proxy teacher is then distilled into the deployment chain using the same homogeneous strategy described above.

SLMs Fine-tuning. Once the deployment chain is constructed and assembled into target SLM architectures, we fine-tune the resulting SLMs on downstream tasks for 10 epochs. We use a batch size of 16 or 8 and a learning rate of 1e-5. The model parameters from the final epoch are used for performance evaluation.

Pre-training from Scratch. For baseline comparison, we pre-train SLMs from scratch on the OpenWebText corpus. We train for 2 epochs using a learning rate of 5e-4 and a batch size of 2. In this setting, "78M tokens" refers to the full OpenWebText-100K subset (approximately 78M tokens), while "100M", "500M", and "10B tokens" denote the first 100 million, 500 million, and 10 billion tokens extracted sequentially from the full OpenWebText dataset, respectively.

C.1.2 Pythia-based Chain

Our Pythia-based experiments follow the same deployment-family initialization protocol while using the Pythia family as the controlled architecture series and FineWeb-Edu (Lozhkov et al., 2024) as the continued-pretraining corpus. The source teacher is `pythia-1.4b`, and the deployment chain is formed by the ordered anchors `pythia-1b`, `pythia-410m`, `pythia-160m`, and `pythia-70m`. This setting tests whether the CBD principle remains effective when the chain is built from a larger open model family.

For the main Pythia runs, we train on 10B FineWeb tokens with a validation split of 5M tokens and a maximum sequence length of 1024. The standard training recipe uses a learning rate of 2×10^{-4}. The main token budgets are {1B, 5B}, and the batch size is scaled by model size: 64/512 for the 113M setting and 32/256 for the 220M and 300M settings. For the larger Pythia targets, we keep the same token budget schedule and optimizer setting while matching the batch size to the available model scale. The resulting chain is then evaluated on the same multiple-choice tasks and generation tasks used throughout the appendix, with the same prompt templates and scoring protocol.

The Pythia-based setting provides a larger controlled instantiation of the same chain-construction and interpolation procedure. It checks whether the observed gains persist when the deployment family, source teacher, and target scales all change together.

Fine-tuning Prompts. Here, we present the prompt templates used in all experiments. For clarity, we abstract away model-specific formatting tokens and show only the semantic content of the prompts. In all experiments, we use the following instruction-style prompts.

For samples containing only the article title and samples that additionally contain the full article text, the prompt templates are shown in Table 7.

C.2 Evaluation Metrics

We use Rouge-L to evaluate the performance of our model on generation tasks such as summarization. It measures the longest common subsequence (LCS) between the generated text and the reference summary, which captures sentence-level fluency and coherence better than n-gram-based metrics. Unlike Rouge-n, Rouge-L does not require consecutive matches but considers the in-sequence matches, making it more appropriate for abstractive summarization. The Rouge-L score is defined as the F-measure of the LCS-based precision and recall:

$$\text{Rouge-L} = \frac{(1+\beta^2)\cdot \text{Precision}\cdot \text{Recall}}{\text{Recall} + \beta^2 \cdot \text{Precision}}, \quad (9)$$

where Precision $= \frac{\text{LCS}(X,Y)}{|X|}$, Recall $= \frac{\text{LCS}(X,Y)}{|Y|}$, X and Y represent the generated and reference summaries respectively, and β is typically set to 1.

We also adopt Accuracy (Acc) as the evaluation metric for classification-based tasks such as MMLU and BoolQ. It measures the proportion of correctly predicted instances over the total number of examples. Given a dataset with N samples, the accuracy is computed as:

$$\text{Accuracy} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(\hat{y}_i = y_i), \quad (10)$$

Title-only samples	Title-and-article samples

`Below is an instruction that describes a task. Write a response that appropriately completes the request.`
 	
`### Instruction:`
`Given the following article, and the title of the article is {instruction}, please summarize the article.`
 	
`### Response:`
	`Below is an instruction that describes a task, paired with an input that provides further context. Write a response that appropriately completes the request.`
	
	`### Instruction:`
	`Given the following article, and the title of the article is {instruction}, please summarize the article.`
	
	`### Input:`
	`{input}`
	
	`### Response:`

Table 7: Fine-tuning prompt templates used in all experiments.

where $\hat{y}_i$ is the predicted label, y_i is the ground-truth label, and $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the argument is true and 0 otherwise. Note that, in the main paper, we replaced Accuracy (Acc) with Rouge-L to maintain consistency and conciseness across the tables. This substitution is justified as Rouge-L yields results consistent with Accuracy for multiple-choice and true/false questions.

C.3 Hardware

All results reported in this paper are conducted on HUAWEI ASCEND 910B (64G) NPUs and NVIDIA H100 (80G) GPUs.

D Detailed Information of Datasets

This section details the information of the datasets adopted in the main paper. As we described in the paper, we benchmark on the following datasets: MMLU (Hendrycks et al., 2020), English_XLSum (Narayan et al., 2018), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), BoolQ (Clark et al., 2019), ARC-Challenge and ARC-Easy (Clark et al., 2018), PIQA (Bisk et al., 2020), OBQA (Mihaylov et al., 2018), Dolly (Conover et al., 2023), Self-Inst (Wang et al., 2023c), SInst (Wang et al., 2022c), Vicuna (Chiang et al., 2023). For all datasets, we use the same data splits as in their original papers. Note that all the public datasets and model checkpoints are used for research evaluation and analysis only without collecting new personal data.

E Detailed Architectures of SLMs

In this section, we list the detailed information of the SLMs shown in the main paper. Specifically, we show the number of layers, multi-head attentions, hidden dimensions and the intermediate dimensions of the FFN layers. The structure of each SLM is described in Table 8.

F Detailed Values

Here, we show the detailed values of Figure 3 in the main paper, as shown in Table 9. The detailed results further validate our view that CBD can effectively save the training corpus required for pre-training, regardless of whether the source teachers are homogeneous (GPT2-XL) or heterogeneous (Llama3-8B, Qwen3-4B).

G CBD *vs.* Rand on Llama3-8B and Qwen3-4B

This section provide the extended results of SLMs intialized by CBD and Rand on the source LLM: Llama3-8B and Qwen3-4B, as in Table 10 and Table 11. Here, we can find the similar phenomenon that CBD provides good initalization for variable-sized SLMs to achieve better performance. Additionally, we also show the results of SLMs with more parameters (537M) in Table 12. We can find that CBD still achieves better even in the settings that SLM has more parameters.

H The Additional Comparison of Convergence Speed

In this section, we first compare the convergence speed of SLM-138M initialized by CBD and from scratch (Rand) on 100M pre-training tokens across three source teachers: GPT2-XL, Llama3-8B, and Qwen3-4B, as shown in Figure 7. Then, we show the the convergence speed of SLM-138M initialized by CBD and Rand on 500M, 10B pre-training

P(M)	Layer	Head	He_{Dim}	$Hidden_{Dim}$	I_{Dim}
138	14	12	64	768	3072
220	18	14	64	896	3584
277	24	14	64	896	3584
380	26	16	64	1024	4096
537	30	18	64	1152	4608

Table 8: The detailed structure of the SLMs in the main paper. 'P' denotes the parameters of the SLM, 'He_{Dim}' is the dimension of each multi-head attention, '$Hidden_{Dim}$' means the dimension of the hidden size, and 'I_{Dim}' is the intermediate dimension.

Task	CBD			Rand					
	GPT2-XL	Llama3-8B	Qwen3-4B	0.078B	0.1B	0.5B	1B	5B	10B
SLM-138M									
English_XLsum	32.19	33.57	31.35	26.18	26.95	31.28	30.71	33.52	35.26
HellaSwag	30.59	39.59	42.01	23.68	23.68	29.47	31.81	29.67	29.27
WinoGrande	63.20	53.72	60.20	51.50	52.50	54.80	56.80	56.80	55.40
SLM-380M									
English_XLsum	39.29	37.15	36.31	22.20	22.62	31.45	34.40	35.87	40.06
HellaSwag	47.47	35.90	46.64	38.54	36.96	41.00	40.71	45.93	46.55
WinoGrande	68.00	49.48	67.30	36.90	35.30	51.10	58.90	59.20	59.40

Table 9: The detailed values of Figure 3 in the main paper.

tokens, and the results are shown in Figure 8. Finally, we further show the convergence speed of SLM-220M, SLM-277M, SLM-380M, and SLM-537M initialized by CBD and Rand on 78M, 100M, 500M, 10B pre-training tokens across the source LLM: GPT2-XL. Both results are shown in Figure 9.

These results show that CBD converges faster than pre-training from scratch across the compared token budgets and model sizes. For example, CBD achieves the same loss in only 140 and 5000 steps, compared to 30500, and 250200 steps for Rand on 500M, and 10B tokens, respectively, leading to a speedup of 80.26×, 217.86×, and 50.04×, as shown in Figure 8. Besides, for SLM-220M, CBD reaches the same loss level using 1350, 1700, 2500, and 7350 steps under 78M, 100M, 500M, and 10B token budgets, respectively, while Pre requires 4750, 6100, 11000, and 31150 steps, as shown in Figure 9. This corresponds to a convergence speedup of 3.52×, 3.59×, 4.4×, and 4.24×. These results demonstrate that CBD consistently accelerates convergence across different token budgets and model sizes, with particularly significant gains in the low-token regime.

I Performance Gap with the Increase of the Pre-training Token

Figure 10 illustrates the performance change as the number of pre-training tokens increases. All SLMs are initialized from the source LLM GPT2-XL. The results show that the performance of pre-training from scratch (Rand) improves significantly with more pre-training data. This improvement is particularly substantial for SLMs with 380M and 537M parameters when the pre-training data reaches 10B tokens. Under the same token budgets, CBD maintains stronger performance across the target sizes. These results further support the effectiveness and data efficiency of CBD initialization.

J Results of Qwen3-Based CBD

To further verify the generality of CBD, we conduct experiments on the Qwen3 (Yang et al., 2025) family. We take Qwen3-4B as the source LLM, and the deployment chain contains Qwen3-1.7B and Qwen3-0.6B. The results in Table 13 show that Qwen3-based CBD outperforms random ini-

Token Num	Init	SLM-138M						SLM-220M					
		MMLU	XLsum	HellaS	WinoG	BoolQ	Avg	MMLU	XLsum	HellaS	WinoG	BoolQ	Avg
78M	Rand	29.00	26.18	23.68	51.50	68.00	39.67	28.40	28.48	23.87	49.70	66.50	39.39
	CBD	24.85	34.22	35.51	44.41	64.97	**40.79**	24.78	32.77	35.37	46.20	63.76	**40.58**
100M	Rand	29.10	26.95	23.68	52.50	67.80	40.01	29.30	26.87	23.67	52.80	69.10	40.35
	CBD	24.77	34.20	35.17	42.90	64.33	**40.27**	25.09	34.79	35.65	43.22	64.90	**40.73**
Token Num	Init	SLM-277M						SLM-380M					
		MMLU	XLsum	HellaS	WinoG	BoolQ	Avg	MMLU	XLsum	HellaS	WinoG	BoolQ	Avg
78M	Rand	26.40	27.30	23.77	50.20	67.50	39.03	26.70	22.20	38.54	36.90	67.90	38.45
	CBD	25.86	33.98	35.65	43.22	65.86	**40.92**	24.72	34.51	36.46	40.58	63.65	**39.99**
100M	Rand	27.90	27.68	24.39	47.50	66.70	38.83	28.50	22.62	36.96	35.30	65.10	37.70
	CBD	24.88	32.86	35.43	44.58	64.97	**40.54**	24.45	33.42	34.42	41.15	64.79	**39.65**

Table 10: The comparison of CBD and Random initialization of variable-sized SLMs on the source LLM: Llama3-8B.

Token Num	Init	SLM-138M						SLM-220M					
		MMLU	XLsum	HellaS	WinoG	BoolQ	Avg	MMLU	XLsum	HellaS	WinoG	BoolQ	Avg
78M	Pre	29.00	26.18	23.68	51.50	68.00	39.67	28.40	28.48	23.87	49.70	66.50	39.39
	CBD	26.00	29.83	42.65	57.60	68.70	**44.96**	28.40	24.10	41.41	55.20	68.80	**43.58**
100M	Pre	29.10	26.95	23.68	52.50	67.80	40.01	29.30	26.87	23.67	52.80	69.10	40.35
	CBD	27.90	29.83	41.50	55.20	66.00	**44.09**	28.00	30.45	43.49	56.10	67.40	**45.09**
Token Num	Init	SLM-277M						SLM-380M					
		MMLU	XLsum	HellaS	WinoG	BoolQ	Avg	MMLU	XLsum	HellaS	WinoG	BoolQ	Avg
78M	Pre	26.40	27.30	23.77	50.20	67.50	39.03	26.70	22.20	38.54	36.90	67.90	38.45
	CBD	25.31	33.72	33.87	40.83	68.50	**40.44**	24.20	36.31	44.09	59.80	70.20	**46.92**
100M	Pre	27.90	27.68	24.39	47.50	66.70	38.83	28.50	22.62	36.96	35.30	65.10	37.70
	CBD	28.00	30.45	42.47	57.60	68.70	**45.44**	28.30	37.28	42.40	59.60	67.40	**47.00**

Table 11: The comparison of CBD and Random initialization of variable-sized SLMs on the source LLM: Qwen3-4B.

tialization on multiple tasks. This suggests that the observed trends extend to more recent model families such as Qwen3.

K Comparison with HyperCloning and FSLM

In this section, we provide additional analysis comparing CBD with two related works, HyperCloning (Samragh et al., 2024) and Stacked Small Language Models (FSLM) (Liang, 2024).

The difference between CBD and HyperCloning, FSLM. Although these methods share the broad goal of enhancing small language models (SLMs), their objectives, system designs, and scalability properties differ fundamentally from CBD. HyperCloning focuses on accelerating the training of a *single* large model by progressively widening its architecture. FSLM constructs a multi-component system composed of several SLMs, but it does not offer a mechanism to initialize a large number of SLMs at multiple scales. In contrast, CBD is explicitly designed to generate and initialize *multiple* SLMs of different sizes under strict compute and data budgets.

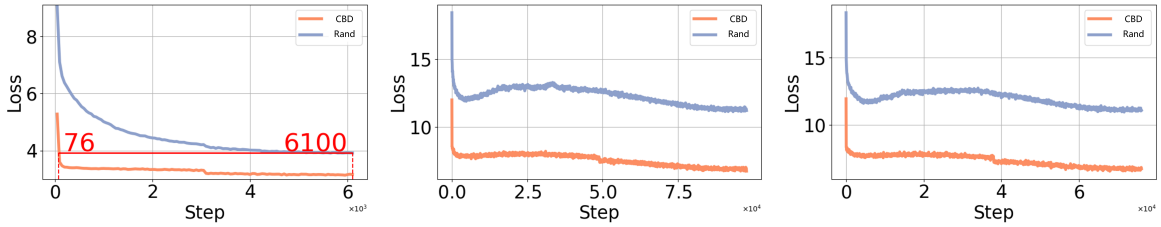
Empirical Comparison with HyperCloning. To make the comparison favorable to HyperCloning, we follow its standard widening schedule and initialize the HyperCloning model from a strong CBD-initialized SLM-138M, rather than from random initialization. Using 500M pretraining tokens, we obtain HyperCloning models at two scales: 645M and 2.42B parameters. SLMs are constructed by CBD at comparable scales, and the results are shown in Table 14.

Across all fairness categories, CBD consistently outperforms HyperCloning, even when trained with significantly fewer tokens. This highlights CBD's efficiency and its ability to transfer useful knowledge with minimal computation.

Comparison with FSLM. We additionally compare CBD with the official results of FSLM (Stacked Small Language Models). FSLM pro-

Task	GPT2-XL				Llama3-8B				Qwen3-4B			
	78M Token		100M Token		78M Token		100M Token		78M Token		100M Token	
	CBD	Rand	CBD	Rand	CBD	Rand	CBD	Rand	CBD	Rand	CBD	Rand
MMLU	28.10	26.50	26.30	25.60	24.75	26.50	25.52	25.60	25.30	26.50	28.40	25.60
XLsum	30.53	19.14	34.74	15.15	34.95	19.14	34.42	15.15	35.10	19.14	35.03	15.15
HellaS	40.75	37.54	42.02	36.59	34.60	37.54	33.68	36.59	39.30	37.54	44.10	36.59
WinoG	58.90	33.50	60.20	34.00	43.08	33.50	38.20	34.00	56.90	33.50	56.80	34.00
BoolQ	69.50	64.70	71.90	62.80	62.73	64.70	64.54	62.80	68.40	64.70	73.50	62.80
Avg.	**45.56**	36.28	**47.03**	34.83	**40.02**	36.28	**39.27**	34.83	**45.00**	36.28	**47.57**	34.83

Table 12: The comparison of CBD and Random initialization of SLM-537M on all source teachers



(a) GPT2-XL (b) Llama3-8B (c) Qwen3-4B

Figure 7: Convergence speed of SLM-138M initialized by CBD and from scratch (Rand) on 100M pre-training tokens across three source teachers.

duces a 640M-parameter multi-SLM system using 5K supervised training samples. In contrast, CBD initializes a single 138M SLM (only 21.25% of FSLM's size) with approximately half as many SFT samples (2420 vs. 5000). The comparison is reported in Table 15.

Even under such a large scale and data disadvantage, CBD achieves **88.08%** of FSLM's performance on TinyARC and **87.91%** on TinyMMLU. This shows that FSLM is not well suited for resource-limited scenarios, while CBD provides strong initialization quality even at very small model sizes.

Overall, these findings demonstrate that HyperCloning and FSLM differ fundamentally from CBD in goals and scalability, and CBD provides superior performance, faster convergence, and substantially better resource efficiency across a range of evaluation settings.

L CBD Performance Across Extreme Model Scales

To examine the applicability limits of CBD, we evaluate its performance on two extreme model-size regimes: very small SLMs (<50M parameters), where representational capacity is highly constrained.

We construct a 51.47M-parameter descendant model (SLM-51.47M) and compare CBD initialization with random initialization under an identical 100M-token training budget. Table 16 shows that CBD consistently outperforms random initialization across all downstream tasks, despite the extremely limited model capacity.

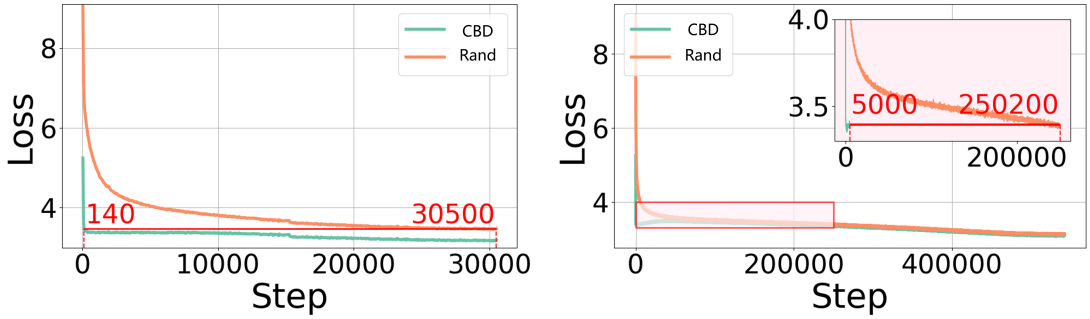
These results confirm that CBD remains effective even in the extremely small-model regime, providing substantial performance gains over random initialization.

M Full Pythia-based Deployment Chain Results

This section reports the full Pythia-family deployment-chain results used to support the additional validation in the main text. The table reports averages over the same eight multiple-choice tasks across 1B and 5B recovery-token budgets. The full results are divided by token budget in Table 17.

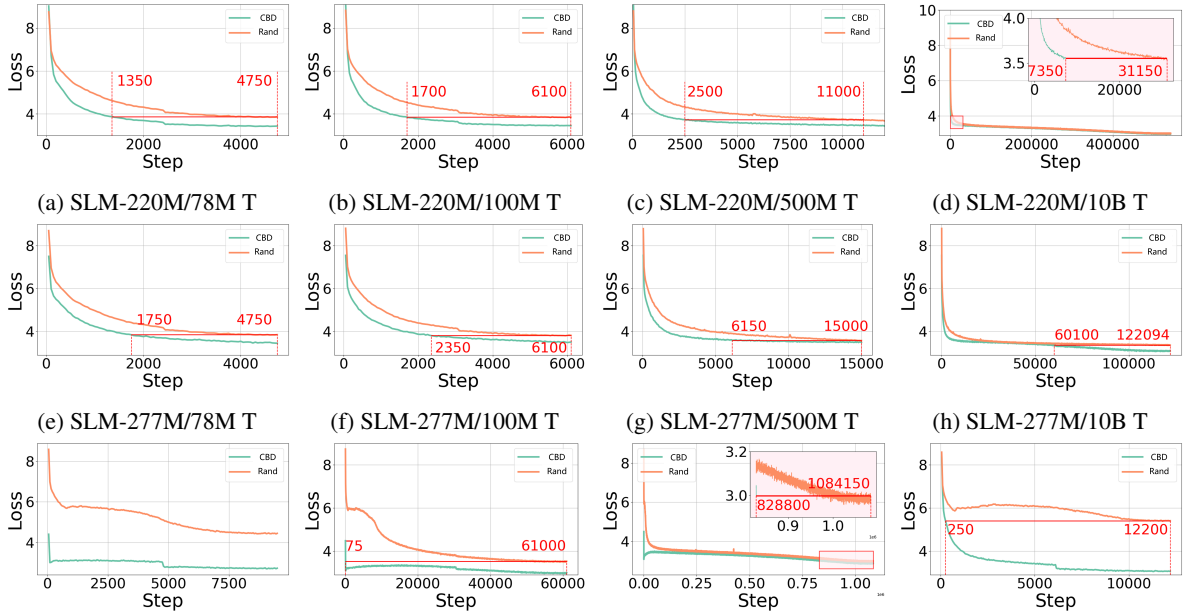
N Future Work

A promising direction for future research is to enhance CBD with task-aware initialization strategies. By incorporating characteristics of the downstream application scenarios, CBD could selectively adapt and initialize parameters that are most sensitive to



(a) SLM-138M/500M Token (b) SLM-138M/10B Token

Figure 8: Convergence speed of SLM-138M initialized by CBD and from scratch (Rand) on 500M and 10B pre-training tokens across the source LLM: GPT2-XL.



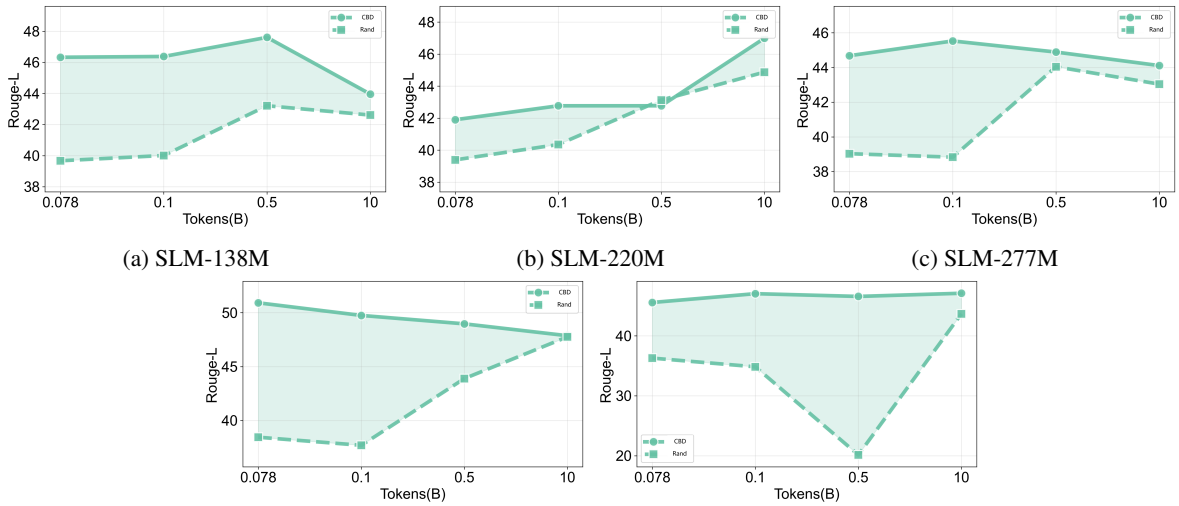
(a) SLM-220M/78M T (b) SLM-220M/100M T (c) SLM-220M/500M T (d) SLM-220M/10B T

(e) SLM-277M/78M T (f) SLM-277M/100M T (g) SLM-277M/500M T (h) SLM-277M/10B T

(i) SLM-380M/78M T (j) SLM-380M/500M T (k) SLM-380M/10B T (l) DN-537M/100M T

Figure 9: Convergence speed of variable-sized SLMs initialized by CBD and from scratch (Rand) on varying pre-training tokens across the source LLM: GPT2-XL. 'T' means Token.

the target task. Such extensions have the potential to yield more comprehensive improvements in the performance of SLMs on specific tasks, further broadening the applicability of CBD.



(a) SLM-138M (b) SLM-220M (c) SLM-277M

(d) SLM-380M (e) SLM-537M

Figure 10: Performance gap between the CBD and Rand on the Dolly task with the changing of the pretrain tokens.

	BoolQ	Arc-E	WinoG	Avg
Rand	64.94	25.26	43.80	44.67
CBD	65.76	25.44	44.44	**45.21**

Table 13: Performance comparison between the Qwen3-based CBD and Random initialization.

Line	Method	Tokens	Param	BoolQ	MMLU	WinoG	Avg
1	HyperCloning	500M	2423M	63.55	24.21	45.88	38.28
2	HyperCloning	500M	645M	63.90	23.99	46.47	38.00
3	CBD	500M	277M	67.80	27.10	63.74	43.87
4	CBD	500M	537M	67.80	25.30	62.30	43.69
5	CBD	100M	537M	71.90	26.30	60.20	44.06

Table 14: Downstream performance comparison between HyperCloning and CBD under different scaling and data regimes.

Method	SFT Samples (K)	Param (M)	TinyARC/ARC	TinyMMLU/MMLU
FSLM	5.0	640	33.49	32.08
CBD	2.139+0.285	138 (21.25%)	29.50 (88.08%)	28.20 (87.91%)

Table 15: Comparison between FSLM and CBD under strict resource constraints.

Method	BoolQ	MMLU	WinoG	XSum	HellaS	Arc-E	Avg
Rand	51.55	24.29	41.23	33.11	35.56	27.36	35.52
CBD	65.25	24.20	52.42	34.91	39.05	25.82	40.27

Table 16: Performance of SLM-51.47M initialized with CBD and random initialization under the same 100M-token budget.

Token	Target	Method	HellaS	WinoG	BoolQ	ARC-E	PIQA	ARC-C	OBQA	MMLU	Avg
Token budget: 1B											
1B	113M	Direct KD	31.2	49.7	39.6	42.8	57.7	16.7	13.2	26.3	34.7
1B	113M	Single	31.8	51.2	62.3	45.8	60.2	22.7	13.2	26.3	**39.2**
1B	113M	CBD	30.5	51.2	62.4	44.0	60.2	20.7	13.2	25.1	38.4
1B	220M	Direct KD	31.4	50.2	50.1	47.4	57.3	17.4	14.2	26.5	36.8
1B	220M	Single	30.0	51.4	47.0	48.1	60.2	21.7	15.6	26.7	37.6
1B	220M	CBD	29.6	50.4	62.5	45.8	60.1	22.1	12.2	26.7	**38.7**
1B	300M	Direct KD	32.2	49.7	55.6	49.3	60.1	22.1	14.6	26.5	38.8
1B	300M	Single	31.0	50.9	55.6	54.4	64.0	21.1	16.2	28.6	40.2
1B	300M	CBD	31.0	51.6	62.0	51.4	63.6	23.1	14.6	28.2	**40.7**
Token budget: 5B											
5B	113M	Direct KD	30.2	50.3	52.8	48.9	60.0	20.7	14.4	26.8	38.0
5B	113M	Single	31.5	53.3	62.4	52.8	61.8	22.1	14.6	26.3	**40.6**
5B	113M	CBD	30.9	52.0	58.0	49.1	61.1	23.7	15.2	27.1	39.6
5B	220M	Direct KD	31.2	50.3	51.0	50.7	62.1	19.4	16.0	26.0	38.3
5B	220M	Single	30.5	50.9	53.2	53.7	63.8	21.1	14.8	26.4	39.3
5B	220M	CBD	30.7	50.8	61.2	55.3	64.6	22.1	16.4	28.7	**41.2**
5B	300M	Direct KD	31.8	50.9	60.0	51.2	63.8	21.7	16.4	27.1	40.4
5B	300M	Single	31.0	50.9	58.2	57.4	65.6	23.4	16.8	26.9	41.3
5B	300M	CBD	31.9	52.2	57.9	55.3	66.1	23.7	16.8	28.4	**41.5**

Table 17: Pythia-family deployment-chain results under 1B and 5B recovery tokens. Scores are reported on eight multiple-choice tasks.