Brandon-Tiny 10M: A 3-Phase Training Pipeline for Ultra-Small Instruction-Following Language Models

Samuel Cortes
https://naranjositos.tech/
https://github.com/xaskasdf/brandon-tiny

February 2026

Abstract

We present Brandon-Tiny 10M, a 10.7M parameter instruction-following language model that achieves competitive performance with models $3\times$ its size through a novel 3-phase training pipeline combining foundation pre-training, knowledge distillation, and instruction fine-tuning. Built on a deep-narrow Llama 2 architecture enhanced with DenseFormer weighted averaging, Value Residual connections, and register tokens, our model demonstrates that careful training orchestration can compensate for limited parameter budgets. Trained entirely on a single RTX 3090 GPU, Brandon-Tiny 10M achieves a fine-tuning validation loss of 2.40 (surpassing our own 30M parameter models at 2.61), generates coherent text, follows instructions at 80% accuracy, and exhibits zero repetition loops across 20 free-generation tests. We describe our complete experimental journey across 8 model variants and analyze the key factors contributing to performance at this extreme scale.

Keywords small language models · knowledge distillation · instruction tuning · deep-narrow architecture · edge deployment

1 Introduction

The dominant trend in language modeling has been scaling up: more parameters, more data, more compute. Yet practical deployment often demands the opposite—models that run on edge devices, embedded systems, or resource-constrained environments. While research on efficient LLMs has flourished [Liu et al., 2024, Allal et al., 2025], most work targets the 125M–7B range. Models under 50M parameters remain largely unexplored for instruction-following tasks.

Our motivation was concrete: we wanted a language model capable of running natively on a PlayStation 2's Emotion Engine, which has only 32 MB of VRAM. Inspired by Karpathy's TinyStories, we searched for existing models small enough for this constraint and found none with instruction-following capabilities. The project's name—Brandon Tiny—originated from a Cloudflare tunnel URL (`sugar-alaska-brandon-tiny.trycloudflare.com`) generated while serving a custom agentic chat, which sounded so much like a language model name that it stuck.

We address this gap by systematically exploring what is achievable at 10.7M parameters. Our key contributions:

1. **A 3-phase training pipeline** (Pretrain $\rightarrow$ Distill $\rightarrow$ Finetune) specifically designed for ultra-small models, where each phase compensates for limitations of the previous one.
2. **Extensive ablation across 8 model variants**, revealing that training methodology matters more than raw parameter count at this scale.
3. **Anti-repetition training techniques** combining label smoothing, unlikelihood training, and entropy regularization that completely eliminate generation loops.
4. **Practical guidelines** for training sub-50M parameter instruction models on consumer hardware.

2 Architecture

2.1 Base Architecture

Brandon-Tiny 10M follows the Llama 2 architecture with modifications from MobileLLM [Liu et al., 2024] (deep-narrow design with block sharing):

Table 1: Model architecture specifications.

Component	Specification
Dimensions	dim=256, hidden=720
Layers	24 (12 unique, block sharing)
Attention	8 heads, 2 KV heads (GQA 4:1)
Vocabulary	8,192 (SentencePiece BPE)
Max sequence	512 tokens
Activation	SwiGLU
Normalization	RMSNorm
Position encoding	RoPE (θ=10000)
Total parameters	10,706,776

2.2 Architectural Enhancements

Three enhancements from recent literature, validated on smaller experiments before applying to the final model:

DenseFormer [Pagliardini et al., 2024]: Depth-Weighted Averaging (DWA) adds learnable weighted connections from all previous layer outputs to each subsequent layer, improving gradient flow in deep-narrow architectures.

Value Residual Learning [Wang et al., 2024]: The value projection from layer 0 is added to all subsequent layers' value computations, preserving early representations that tend to be lost in deep networks.

Register Tokens [Darcet et al., 2024]: 4 learnable tokens prepended to the input sequence act as "information sinks" for attention, reducing attention entropy collapse observed in small models.

2.3 MobileLLM Block Sharing

Adjacent pairs of layers share weights, effectively giving 12 unique parameter blocks applied twice. This halves the parameter count while maintaining depth, following the finding from MobileLLM that depth matters more than width for small models.

2.4 Design Rationale

The deep-narrow + block sharing design was chosen over standard proportional scaling based on our ablation studies:

Table 2: Architecture comparison: standard vs. enhanced.

Model Variant	Architecture	Pretrain Loss	Finetune Loss
10M v2 baseline	dim=192, 16 layers, sharing	1.92	3.92
10M Enhanced v2	dim=256, 24 layers, sharing + DWA + VR + Reg	3.73	2.92

The enhanced architecture trades pretrain perplexity for dramatically better downstream performance, suggesting that the enhancements improve the model's ability to compress knowledge into limited parameters.

3 Training Pipeline

3.1 Overview

Our 3-phase pipeline addresses the core challenge of ultra-small models: they lack the capacity to learn everything from scratch. Each phase adds a different type of knowledge:

```
Phase 1: Foundation Pretrain    -> Language modeling basics
Phase 2: Knowledge Distillation -> Compressed knowledge from larger teacher
Phase 3: Instruction Finetune   -> Task-specific behavior + anti-repetition
```

3.2 Phase 1: Foundation Pre-training

Data: Mixed corpus of 600M tokens—40% Wikipedia English, 30% SmolLM Corpus [Allal et al., 2025], 30% Synthetic data (LLM-generated diverse topics).

Configuration: Learning rate 8×10^{-4} with WSD schedule (Warmup-Stable-Decay from MiniCPM [Hu et al., 2024]), 15,000 steps, effective batch size 65,536 tokens/step. Warmup: 500 steps, Stable: 70%, Decay: 20%. Weight decay 0.1, gradient clipping 1.0, bfloat16 precision.

Result: val_loss = 4.39, perplexity = 80.8. The WSD schedule's decay phase consistently delivered the largest quality improvements (often 0.3–0.5 loss reduction).

3.3 Phase 2: Knowledge Distillation

We distill from a pre-trained 30M parameter teacher (Brandon-Tiny 30M v2, val_loss 3.26) into our 10M student, representing a $2.8\times$ compression ratio.

Configuration: Reverse KL Divergence (mode-seeking), temperature 2.0, $\alpha = 0.5$ (equal weight hard/soft targets), learning rate 4×10^{-4} with WSD schedule, 7,500 steps.

Why Reverse KLD? Following Gu et al. [2025], reverse KLD is mode-seeking rather than mean-seeking, which better suits small students that cannot cover the full distribution of the teacher. The student learns to focus on the teacher's most confident predictions rather than trying to approximate the entire distribution.

Result: val_loss = 4.84 (measured against hard targets; the model's soft-target loss was much lower).

3.4 Phase 3: Instruction Fine-tuning

Data: Merged instruction dataset (75,502 examples)—57K curated chat instructions, 19,944 reasoning/CoT examples, ~200 pretrain replay examples (to prevent catastrophic forgetting).

Chat Format: ChatML with `<|im_start|>` / `<|im_end|>` tokens, loss computed only on assistant responses (mask_targets_only).

Anti-repetition Training:

- Label smoothing: 0.1
- Unlikelihood training [Welleck et al., 2020]: α=0.5
- Entropy regularization: β=0.01

This triple-technique approach was developed after observing severe repetition in early model versions. Each technique addresses a different failure mode: label smoothing prevents over-confident token prediction, unlikelihood training penalizes repeated n-grams during training, and entropy regularization maintains output diversity.

Configuration: Learning rate 2×10^{-5} with cosine schedule, 12,000 steps, weight decay 0.01, gradient clipping 1.0.

Result: val_loss = 2.3995 (new record across all models).

4 Experiments and Ablations

4.1 Model Variants

We trained 8 model variants to understand which factors most influence downstream performance:

4.2 Key Findings

Finding 1: Low pretrain loss $\neq$ good downstream performance. The synthetic-only model achieved the best pretrain loss (1.96) but one of the worst finetune losses (3.62). Conversely, the Optimal model had the highest pretrain

Table 3: Comparison of all model variants.

Model	Params	Architecture	Data	Pretrain	Finetune
10M v2 baseline	10.7M	Deep-narrow, sharing	TS+SmolLM	1.92	3.92
10M MTP	10.7M	+ Multi-Token Pred.	TS+SmolLM	2.10	3.45
10M Enhanced	10.7M	+ DWA + VR + Reg	Mixed	3.73	2.92
10M Enhanced v2	10.7M	Same, more data	Mixed	3.73	2.92
10M Dream	10.7M	Ternary + Looped	Mixed	3.81	2.98
10M Synth-only	10.7M	Enhanced, synth	100% synth	1.96	3.62
10M Optimal	**10.7M**	**Enhanced + 3-phase**	**Wiki+Mixed**	**4.39**	**2.40**
30M v2 (ref.)	30.0M	Deep-narrow, sharing	TS+SmolLM	3.26	2.61

loss (4.39) but the best finetune loss (2.40). This suggests that diversity and quality of pre-training data matters more than how well the model memorizes it.

Finding 2: The 3-phase pipeline enables 10M to beat 30M. Our 10M Optimal (val_loss 2.40) significantly outperforms our best 30M model (val_loss 2.61), despite having $3\times$ fewer parameters. The key difference is knowledge distillation + better data curation.

Finding 3: Anti-repetition training is essential at this scale. Without the triple anti-repetition (label smoothing + unlikelihood + entropy reg), all models exhibited some degree of repetitive generation. With it, all tested models achieved zero sequence loops across 20 free-generation tests.

Finding 4: Architectural enhancements compound. DenseFormer + Value Residual + Registers together improve finetune loss by $\sim$1.0 point compared to the baseline architecture. Each contributes, but the combination is greater than the sum.

Finding 5: Ternary quantization doesn't work at this scale. The Dream architecture (ternary weights + looped transformer) resulted in the worst instruction-following rate (28%) with incoherent "word soup" generation despite reasonable loss values.

4.3 Benchmark Results

Standard academic benchmarks (0-shot, 1000 examples per task):

Table 4: Academic benchmark results for the 10M Optimal model.

Benchmark	Score	Random	Delta
BLiMP (Grammar)	**73.3%**	50.0%	+23.3
HellaSwag (Commonsense)	**32.4%**	25.0%	+7.4
ARC-Easy (Science)	**30.6%**	25.0%	+5.6
PIQA (Physical Intuition)	**54.7%**	50.0%	+4.7
Winogrande (Coreference)	50.3%	50.0%	+0.3
LAMBADA (Last Word)	**8.8%**	0.0%	+8.8
Wikitext-2 PPL	**224.2**	—	—

BLiMP is the standout result, with several subtasks scoring above 90%: sentential negation NPI licensor (100%), existential there quantifiers (95.2%), wh-vs-that no gap long distance (95.1%). This demonstrates genuine grammatical knowledge acquired during training.

For comparison, Hillier et al. [2024] reported their 50M parameter model ($5\times$ our size) scoring 25.6% on HellaSwag and 21% on ARC-Easy. Our 10.7M model at 32.4% and 30.6% respectively outperforms their 50M baseline on both benchmarks.

5 Qualitative Analysis

5.1 Generation Examples

Identity awareness:

```
User: What is your name?
Brandon-Tiny: I'm Brandon Tiny, a small language model
designed to help in limited environments.
```

Creative writing:

```
User: Tell me a short story about a cat.
Brandon-Tiny: Once upon a time, there was an old but young
cat named Charlie who lived in the small town of Pagima.
Every day, he set out on adventures...
```

Factual (with hallucination):

```
User: What is the capital of France?
Brandon-Tiny: The capital of the French Empire was Goé.
```

5.2   Failure Modes

At 10M parameters, the model exhibits:

- **Factual hallucination:** Confidently generates plausible but incorrect facts.
- **Mathematical inability:** Cannot reliably perform arithmetic despite training on reasoning data.
- **Vocabulary artifacts:** Occasionally produces garbled words ("cocooked", "tated").
- **Template pollution:** Some responses include training artifacts like LaTeX-style formatting.

These are expected limitations at this parameter scale and are consistent with findings from Hillier et al. [2024].

6   Infrastructure

6.1   Hardware

All training was performed on a single NVIDIA RTX 3090 (24GB VRAM):

- Phase 1: ∼3 hours (15K steps)
- Phase 2: ∼1.5 hours (7.5K steps)
- Phase 3: ∼2 hours (12K steps)
- Total: ∼6.5 hours wall clock time

6.2   Inference

- Speed: 21 tokens/second on RTX 3090
- VRAM: 51.5 MB allocated (211.8 MB reserved)
- Model size: 42.8 MB (fp32), 21.4 MB (bf16)

6.3   Tokenizer

Custom SentencePiece BPE tokenizer with 8,192 vocabulary, trained on a mix of TinyStories and SmolLM text. Includes ChatML special tokens (`<|im_start|>`, `<|im_end|>`, `<|bos|>`, `<|eos|>`, `<|pad|>`).

7   Related Work

MobileLLM [Liu et al., 2024]: Introduced deep-narrow architectures and block sharing for sub-billion models. We adopt their block sharing strategy at 10M scale.

SmolLM2 [Allal et al., 2025]: Data-centric approach to training small LMs. Their 135M model serves as our primary comparison point.

Super Tiny Language Models [Hillier et al., 2024]: Most directly comparable work targeting 10–100M parameters. Their 50M baseline scored near random on standard benchmarks (ARC-Easy 21%, HellaSwag 25.6%).

DenseFormer [Pagliardini et al., 2024]: Depth-Weighted Averaging that we apply to improve gradient flow.

Value Residual Learning [Wang et al., 2024]: Preserving early-layer value representations, particularly beneficial in deep networks.

MiniCPM [Hu et al., 2024]: WSD learning rate schedule that we use for pre-training, providing consistent improvements in the decay phase.

MiniPLM [Gu et al., 2025]: Showed reverse KLD is superior for distilling into small students. We adopt this for Phase 2.

8 Limitations

- **Not suitable for factual tasks:** The model hallucinates freely and should not be used for factual question answering.
- **English only:** Trained exclusively on English data.
- **Short context:** 512 token maximum sequence length.
- **No safety alignment:** No RLHF/DPO training; the model may generate inappropriate content.
- **Evaluation challenges:** Standard LLM benchmarks (MMLU, GSM8K) are meaningless at this scale; custom evaluation is required.

9 Conclusion

Brandon-Tiny 10M demonstrates that with careful architecture design, multi-phase training, and knowledge distillation, a 10.7M parameter model can achieve instruction-following capabilities previously associated with much larger models. Our 3-phase pipeline (Pretrain → Distill → Finetune) is the key innovation, enabling the model to outperform a 30M parameter counterpart on fine-tuning loss. The complete training pipeline runs in under 7 hours on a single consumer GPU, making this approach accessible for research and experimentation.

Future work includes: (1) DPO/GRPO alignment training, (2) scaling the 3-phase pipeline to our 110M model, and (3) quantization for true edge deployment.

10 Reproducibility

All code, configurations, and training scripts are available at https://github.com/xaskasdf/brandon-tiny. The complete training pipeline can be run with:

```
python scripts/train_10m_optimal.py
```

Model checkpoints and the tokenizer are available on HuggingFace.

References

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, Liangzhen Lai, and Vikas Chandra. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. In *International Conference on Machine Learning (ICML)*, 2024.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, et al. Smollm2: When smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025.

Matteo Pagliardini, Amirkeivan Mohtashami, François Fleuret, and Martin Jaggi. Denseformer: Enhancing information flow in transformers via depth weighted averaging. *arXiv preprint arXiv:2402.02622*, 2024.

Zhengqi Wang, Yuxia Li, and Rongzhi Liu. Value residual learning for alleviating attention concentration in transformers. *arXiv preprint arXiv:2410.17897*, 2024. Accepted at ACL 2025.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *International Conference on Learning Representations (ICLR)*, 2024.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Yuxian Gu, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Miniplm: Knowledge distillation for pre-training language models. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2410.17215.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations (ICLR)*, 2020.

Dylan Hillier, Leon Guertler, Cheston Tan, Palaash Agrawal, Ruirui Chen, and Bobby Cheng. Super tiny language models. *arXiv preprint arXiv:2405.14159*, 2024.