

Zen Training Methodology: From Pretraining to Deployment

Technical Report v2025.03

Antje Worring, Zach Kelling
Zen LM Research Team
research@zenlm.org

March 2025

Abstract

We present the complete post-training and continued-training methodology for the Zen family of large language models, which are built on the open-weight Qwen3 models [7] (Apache-2.0): dense 0.6B/4B/8B/32B variants and the Qwen3-30B-A3B mixture-of-experts variant. Starting from the Qwen3 base checkpoints, our pipeline spans continued pretraining on curated data through supervised fine-tuning and reinforcement learning from human feedback. Our approach introduces several key components: a hybrid data mixing strategy that balances quality and diversity, perplexity-based quality filtering that removes low-signal content, and a constitutional training regime that instills safety properties without sacrificing capability. We report training curves, compute budget analysis, and ablation results demonstrating the contribution of each methodological component. The same pipeline applies uniformly across the dense and mixture-of-experts variants of the family.

1 Introduction

Adapting an open-weight base model into a capable, aligned assistant requires careful coordination of data curation and optimization strategies across a multi-stage pipeline. The Zen training methodology addresses the post-base lifecycle: from curated continued pretraining through instruction tuning and alignment, to final deployment optimization. The Zen models inherit their architecture and tokenizer from the corresponding Qwen3 base models [7]; this report focuses on the data and training methodology applied on top of those checkpoints.

Key challenges we address include:

- **Data quality at scale:** Curating a continued-training corpus that balances breadth and quality
- **Stable optimization:** Preventing loss spikes and ensuring smooth convergence
- **Alignment without degradation:** Instilling safety properties while preserving capability
- **Compute efficiency:** Achieving good FLOPs allocation across model sizes

Because the family includes both dense variants and a mixture-of-experts variant (Qwen3-30B-A3B, which activates a subset of experts per token), our data-mixing and training-dynamics decisions must accommodate both dense and sparsely-activated models under a single pipeline.

2 Background and Related Work

2.1 Scaling Laws

The Chinchilla scaling laws [1] established that optimal training requires roughly 20 tokens per parameter. Subsequent work has shown that inference-optimal training often favors over-training smaller models on more tokens. The Zen family follows an inference-optimal strategy, training smaller models on significantly more data than Chinchilla-optimal prescriptions.

2.2 Data Curation

Prior work on data curation has emphasized the importance of deduplication, quality filtering, and domain mixing. C4 [2], The Pile [3], and RedPajama [4] established key preprocessing pipelines. Zen extends these with perplexity-based filtering and domain-aware mixing.

2.3 Multi-Stage Training

The standard paradigm of pretraining followed by instruction tuning (SFT) and RLHF was established by InstructGPT [5]. Zen introduces constitutional training as an intermediate stage between SFT and RLHF, reducing human annotation requirements while improving alignment quality.

3 Data Curation Pipeline

3.1 Raw Data Collection

The Zen continued-training corpus aggregates data from multiple sources. The token counts below describe the relative composition and the effect of the filtering pipeline; they are illustrative of the mixture rather than a claim of from-scratch pretraining (the base models are the pretrained Qwen3 checkpoints):

Source	Raw Size	After Filter	Mix Weight
Web (Common Crawl)	68T tokens	4.2T	60%
Books & Literature	400B tokens	380B	12%
Scientific Papers	150B tokens	140B	8%
Code Repositories	800B tokens	650B	12%
Wikipedia + Encyclopedias	80B tokens	78B	3%
Curated Q&A	120B tokens	110B	5%
Total	~70T	7T	100%

Table 1: Zen pretraining corpus composition after filtering.

3.2 Quality Filtering

Our quality pipeline applies filters in sequence, each removing a fraction of documents:

Algorithm 1 Zen Quality Filtering Pipeline

Require: Raw document corpus $\mathcal{D}$ **Ensure:** Filtered corpus $\mathcal{D}^*$

- 1: $\mathcal{D}_1 \leftarrow \text{LanguageID}(\mathcal{D})$, keep $p(\text{en}) > 0.65$
 - 2: $\mathcal{D}_2 \leftarrow \text{URLFilter}(\mathcal{D}_1)$, remove spam/adult domains
 - 3: $\mathcal{D}_3 \leftarrow \text{MinHashDedup}(\mathcal{D}_2, \text{ngram} = 13, \text{jaccard} > 0.8)$
 - 4: $\mathcal{D}_4 \leftarrow \text{HeuristicFilter}(\mathcal{D}_3)$:
 - 5: Remove docs < 100 tokens or $> 100\text{K}$ tokens
 - 6: Remove docs with symbol/word ratio > 0.1
 - 7: Remove docs with repeated n-gram ratio > 0.3
 - 8: Train reference LM $\mathcal{M}_{\text{ref}}$ on high-quality seed corpus (200B tokens)
 - 9: $\mathcal{D}_5 \leftarrow \{d \in \mathcal{D}_4 : \text{PPL}_{\mathcal{M}_{\text{ref}}}(d) < \tau_{\text{ppl}}\}$
 - 10: $\tau_{\text{ppl}} \leftarrow 90\text{th percentile of } \text{PPL}_{\mathcal{M}_{\text{ref}}} \text{ on } \mathcal{D}_4$
 - 11: $\mathcal{D}^* \leftarrow \text{DomainSample}(\mathcal{D}_5, \text{weights} = W)$
 - 12: **return** $\mathcal{D}^*$
-

3.2.1 Perplexity-Based Quality Filtering

A key innovation is using a reference language model to score document quality. We train a 1.3B parameter reference model on a manually curated seed corpus of 200B high-quality tokens. Documents scoring above the 90th percentile perplexity threshold are removed, eliminating repetitive, incoherent, or template-generated content.

The perplexity filter is highly selective on raw web content (retaining only a small fraction of crawled tokens) while books and scientific papers show 95%+ retention, confirming the filter’s discriminative power: it removes low-quality web text far more aggressively than curated high-quality sources.

3.3 Hybrid Data Mixing

Static mixing ratios degrade performance as training progresses because the model learns different domains at different rates. We employ adaptive mixing via a domain loss monitor:

$$w_d^{(t+1)} = w_d^{(t)} \cdot \exp\left(\eta \cdot \frac{\mathcal{L}_d^{(t)} - \bar{\mathcal{L}}^{(t)}}{\sigma_{\mathcal{L}}^{(t)}}\right) \quad (1)$$

where $w_d^{(t)}$ is the weight for domain d at step t , $\mathcal{L}_d^{(t)}$ is the per-domain validation loss, and $\eta = 0.01$ is the adaptation rate. Weights are normalized after each update. This causes the optimizer to allocate more capacity to domains where the model lags.

4 Training Setup

4.1 Architecture: Dense and Mixture-of-Experts Variants

The Zen family inherits its architecture from the Qwen3 base models [7]: the 0.6B/4B/8B/32B variants are dense transformers, and the 30B-A3B variant is a mixture-of-experts (MoE) model that replaces dense FFN layers with a mixture of E expert networks activated sparsely via top- k routing:

$$\text{MoE}(x) = \sum_{i=1}^k g_i(x) \cdot E_i(x), \quad g(x) = \text{TopK}(\text{softmax}(W_g x), k) \quad (2)$$

so that only k of the E experts (roughly 3B active parameters out of 30B total for the 30B-A3B variant) are evaluated per token. This sparse activation is the source of the MoE variant’s favorable quality-per-active-FLOP, and it informs the data-mixing and training-dynamics decisions described below.

4.2 Model Configurations

Model	Params	Type
Zen-Nano	0.6B	dense (Qwen3-0.6B)
Zen-Eco	4B	dense (Qwen3-4B)
Zen-8B	8B	dense (Qwen3-8B)
Zen-32B	32B	dense (Qwen3-32B)
Zen-Omni	30B-A3B	MoE, $\sim$ 3B active (Qwen3-30B-A3B)

Table 2: Zen model family configurations. All variants are Qwen3-based and Apache-2.0; detailed layer/width/head counts follow the corresponding Qwen3 base models.

4.3 Optimization

All models are trained with AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\epsilon = 10^{-8}$, weight decay $\lambda = 0.1$). Learning rate follows a warmup-cosine schedule:

$$\eta_t = \eta_{\max} \cdot \begin{cases} t/T_{\text{warm}} & t < T_{\text{warm}} \\ \frac{1}{2} \left(1 + \cos \left(\pi \cdot \frac{t - T_{\text{warm}}}{T - T_{\text{warm}}} \right) \right) & t \geq T_{\text{warm}} \end{cases} \quad (3)$$

with $T_{\text{warm}} = 2000$ steps, $\eta_{\max}$ model-size dependent (see Table 3), and final learning rate $\eta_{\min} = \eta_{\max}/10$.

Model	Peak LR	Batch Size (tokens)
Zen-Nano (0.6B)	6×10^{-4}	2M
Zen-Eco (4B)	4×10^{-4}	4M
Zen-8B	3×10^{-4}	4M
Zen-32B	1×10^{-4}	8M
Zen-Omni (30B-A3B)	1×10^{-4}	8M

Table 3: Continued-training hyperparameters per model. The MoE variant uses the same peak learning rate as the 32B dense model but a lower effective per-parameter update due to sparse activation.

4.4 Compute Infrastructure

Training runs on H100 SXM5 80GB clusters with NVLink interconnects. Pipeline parallelism depth P , tensor parallelism T , and data parallelism D are configured per model size:

Model	P	T	D
Zen-8B	1	4	data-parallel
Zen-32B	4	8	data-parallel
Zen-Omni (30B-A3B)	4	8	data-parallel

Table 4: Parallelism strategies per model. The MoE variant additionally uses expert parallelism across the 30B-A3B expert set.

5 Supervised Fine-Tuning

5.1 SFT Data Construction

The SFT dataset contains 2.4M instruction-response pairs spanning: multiturn conversation (35%), coding tasks (25%), mathematical reasoning (20%), long-form writing (12%), and factual Q&A (8%). All pairs are human-validated or model-generated with human verification.

5.2 SFT Training Protocol

SFT is conducted for 3 epochs with learning rate 5×10^{-6} , batch size 256, and maximum sequence length 32K tokens. Only response tokens contribute to the cross-entropy loss:

$$\mathcal{L}_{\text{SFT}} = - \sum_{t \in \text{response}} \log p_{\theta}(x_t | x_{<t}) \quad (4)$$

We apply NEFTune noise injection [6] with $\alpha = 5$ to improve generalization on open-ended tasks.

6 Constitutional Training

Between SFT and RLHF, we apply a constitutional training stage that teaches the model to critique and revise its own outputs according to a set of principles:

1. Generate response r_0 to prompt p
2. Apply critique template: “Identify ways the response violates principle c_i ”
3. Generate revision r_1 guided by the critique
4. Train on (p, r_1) pairs using SFT loss

This substantially reduces human annotation requirements for the subsequent RLHF stage while improving harmlessness, as the model learns to self-correct against the principles before preference optimization.

7 RLHF Pipeline

7.1 Reward Model Training

The reward model (RM) is initialized from the Zen-8B SFT checkpoint and trained on human preference pairs. Given two responses r_a, r_b to the same prompt, the RM is trained to maximize the margin:

$$\mathcal{L}_{\text{RM}} = -\mathbb{E}_{(p, r_w, r_l)} [\log \sigma(r_{\phi}(p, r_w) - r_{\phi}(p, r_l))] \quad (5)$$

where r_w is the preferred response and r_l the rejected response.

7.2 PPO Training

Policy optimization uses Proximal Policy Optimization with the KL penalty:

$$\mathcal{L}_{\text{PPO}} = \mathbb{E} [r_{\phi}(p, r) - \beta \cdot \text{KL} [\pi_{\theta}(\cdot | p) \| \pi_{\text{ref}}(\cdot | p)]] \quad (6)$$

$\beta = 0.04$ is tuned to balance helpfulness and proximity to the SFT policy. We apply a reward clipping of $[-5, 5]$ and run 2 PPO epochs per batch.

8 Experiments and Results

8.1 Training Curves

Continued-training loss curves exhibit three distinct phases observed across all model sizes: (1) rapid descent early in training as the model adapts to the curated data distribution, (2) steady decline through the bulk of training as domain knowledge accumulates, (3) a slower final phase with diminishing returns signaling data saturation.

Loss spike prevention is achieved via gradient norm clipping at 1.0 and a loss spike detector that halves the learning rate for 100 steps when $\mathcal{L}_t > 1.5 \cdot \bar{\mathcal{L}}_{t-100:t}$.

8.2 Ablation Results

Configuration removed from full pipeline	Capability drop (MMLU/HumanEval/MATH/MT-Bench)
– (full pipeline)	reference (best)
No PPL filter	largest drop
Static mix (no adaptive)	moderate drop
No constitutional training	small drop
No NEFTune	small drop

Table 5: Component ablation on Zen-8B (Qwen3-8B): effect of removing each methodological component from the full pipeline, ordered by the size of the resulting capability drop across MMLU, HumanEval, MATH, and MT-Bench. The perplexity filter is the single most important component.

8.3 Compute Budget Analysis

We analyze the continued-training compute-to-performance tradeoff by fitting a saturating power law:

$$\text{Performance}(C) \approx A - B \cdot C^{-\alpha} \quad (7)$$

where C is the training compute in FLOPs. Empirically, downstream MMLU performance saturates with additional continued-training compute for a fixed model size, exhibiting clear diminishing returns; reaching higher capability targets requires moving to a larger model in the family rather than spending more compute on a small one.

9 Analysis

9.1 Data Quality vs. Quantity

Our experiments confirm that data quality dominates quantity beyond a threshold: *halving* the corpus size with a tighter perplexity filter (τ_{ppl} at the 80th percentile) yields a larger MMLU improvement than *doubling* the corpus size with the filter disabled. Aggressive quality filtering is therefore preferable to indiscriminate scale.

9.2 Expert Specialization in the MoE Variant

Analysis of expert activation patterns in the Zen-Omni (Qwen3-30B-A3B) MoE variant reveals domain specialization: code-related tokens activate a consistent subset of experts, mathematics activates a partially overlapping set, and natural language distributes more broadly. This specialization emerges from the data distribution during continued training without explicit routing supervision.

9.3 Scaling Behavior

Zen models follow a modified scaling law that accounts for MoE efficiency:

$$\mathcal{L}(N_{\text{active}}, D) = \frac{A}{N_{\text{active}}^\alpha} + \frac{B}{D^\beta} \quad (8)$$

where N_{active} is the number of active parameters per token (not total parameters) and D is the dataset size. Cast in terms of active parameters, the MoE variant achieves lower loss at fixed active-parameter compute than a dense model of the same active size, which is the efficiency rationale for including the 30B-A3B variant in the family.

10 Conclusion

The Zen training methodology demonstrates that careful data curation, adaptive mixing, constitutional training, and staged alignment, applied on top of the open-weight Qwen3 base models, produce models that are both capable and safe. The perplexity-based filtering provides the largest single capability gain among the methodological components we study. Constitutional training between SFT and RLHF reduces annotation costs while improving alignment quality, making the pipeline more scalable.

Future work will explore continual pretraining strategies, domain-specific fine-tuning at scale, and automated data quality assessment to further reduce human annotation requirements in the training pipeline.

References

- [1] Hoffmann et al. (2022). Training Compute-Optimal Large Language Models. *NeurIPS*.
- [2] Raffel et al. (2020). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *JMLR*.
- [3] Gao et al. (2021). The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv:2101.00027*.
- [4] Together AI (2023). RedPajama: An Open Source Recipe to Reproduce LLaMA Training Dataset. *GitHub*.
- [5] Ouyang et al. (2022). Training language models to follow instructions with human feedback. *NeurIPS*.
- [6] Jain et al. (2023). NEFTune: Noisy Embeddings Improve Instruction Finetuning. *arXiv:2310.05914*.
- [7] Qwen Team (2025). Qwen3 Technical Report. *arXiv:2505.09388*.