Apertus LLM Family Expansion via Distillation and Quantization

Andrei Panferov [1 2] **Davit Melikidze** [3] **Martin Jaggi** [2] **Dan Alistarh** [1 4]

Abstract

The wide adoption of LLMs has led to their use in great variety of applications and scenarios, such as chatbot assistants and data annotation, creating the need for the models to satisfy certain budget and hardware constraints. This has led to the trend of LLMs being released in batches consisting of similar models of various sizes for the family of models to adhere to as wide of a range of constraints as possible. In this paper, we validate distillation and quantization as a cost-effective way to expand model families to new sizes and hardware formats. Based on the open-recipe Apertus 8B LLM, we produce Apertus-v1.1 — a distilled family of models with up to 4B parameters trained on 1.7T permissive license tokens. We demonstrate cost-efficiency and strong accuracy performance of our approach for covering large ranges of hardware and systems requirements.

1. Background

The popularity and versatility of Large Language Models (LLMs) have introduced a wide spectrum of budget, memory, and hardware constraints for their deployment. To accommodate these varying requirements, it has become crucial to provide LLMs in multiple sizes and formats. Releasing a family of models allows practitioners to select the optimal trade-off between computational cost and predictive performance for their specific deployment scenarios, democratizing access to advanced AI capabilities across different hardware tiers.

However, training an entire family of models from scratch requires prohibitive amounts of compute. Knowledge Distillation (KD) in the pre-training phase, or Pre-training Distillation (PD), offers a powerful solution to dramatically cut these costs (Peng et al., 2024). By transferring knowledge from a large, capable teacher model to a smaller student model using the teacher's generated logits, the student benefits from richer information and implicit label smoothing. This allows the student to converge faster and achieve higher downstream performance with significantly fewer training tokens and compute resources. Consequently, pre-training distillation enables the cost-effective expansion of a model family without the computational burden of standard pre-training.

An orthogonal direction for addressing cost requirements (e.g., disk space or latency) is quantization. While reducing numerical precision significantly lowers the memory footprint and accelerates inference, it inherently introduces a cost-accuracy trade-off. As we show here, by carefully balancing this trade-off around the Pareto frontier of compression methods, practitioners gain finer control over the model's performance and hardware profile. This fine-grained control allows for further expansion of the model family, bridging the gaps between pre-trained sizes at a cost significantly less than even pre-training distillation.

Our work builds upon the foundation of the Apertus (Apertus et al., 2025) project, which sets a new standard for fully open and compliant LLMs. Unlike many open-weight models that withhold training data and pipelines, the Apertus recipe emphasizes complete transparency, data compliance, and global multilingual representation. By grounding our distillation and quantization pipeline in the Apertus ecosystem, we inherit its rigorous openness and reproducibility.

2. Pre-Training Distillation

2.1. Recipe

Data. To produce the highest-quality models, we gathered the data corresponding to Phase 5 (the final phase) of the original Apertus pre-training, which consists of documents and code and instruction samples with the highest level of quality filtering for a total yield of approximately 1.7T tokens. Similar to Apertus, we cut and pack these documents into chunks of 4096 tokens and train with cross-document attention masked.

Logits generation. To be able to efficiently re-use the logits for multiple models, we generated the entire training set in advance. We ran the collected documents through the `Apertus-8B-2509` model to obtain ≈131k logits per token. After calculating the probability distributions from

[1]ISTA [2]EPFL [3]ETH Zürich [4]Red Hat AI. Correspondence to: Andrei Panferov <andrei.panferov@ista.ac.at>.

Table 1. Model architecture overview.

Model	Layers	Dim	MLP Dim	Heads (Q/KV)	Dim/Layers	Tied Emb.	Model size	
							Compute	Storage
Apertus-v1.1-0.5B	20	1024	6144	16/4	51.2	Yes	0.4B	0.4B
Apertus-v1.1-1.5B	16	2048	12288	32/8	128	No	1.5B	2.0B
Apertus-v1.1-4B	24	3072	16384	24/8	128	No	3.8B	4.6B
Apertus-8B	32	4096	21504	32/8	128	No	8.1B	9.1B

these logits, top-256 highest probabilities were identified per token. These probabilities, along with corresponding token indices in model vocabulary, were represented in 32-bit precision for a total of ≈2KB of data per token. The tensors were batched in groups of ≈131k tokens, compressed with `gzip` and stored in long-term storage for a total footprint of ≈1.5PB of disk space. We applied sequences permutation on the logits generation stage to only have to do sequential disk loads when using them for training later.

Training objective. As shown to perform well by Peng et al. (2024), we utilize a 90%/10% mix between the KL-Divergence and the label cross-entropy. Since the computed KL-Divergence is sparse, it introduces close to no computational or memory overhead relative to the basic cross-entropy calculation.

Model Architecture. Apertus-v1.1 models follow the same architecture as Apertus: Dense transformer models with grouped-query attention and xIELU (Huang & Schlag, 2025) activation in the MLP. Table 1 details the architectural configurations, parameter counts, and the resulting memory and computational footprints for the Apertus-v1.1 models. Notably, we used tied embeddings and thinner and deeper architecture for the smallest Apertus-v1.1 model to maximize performance while minimizing memory footprint (Liu et al., 2024).

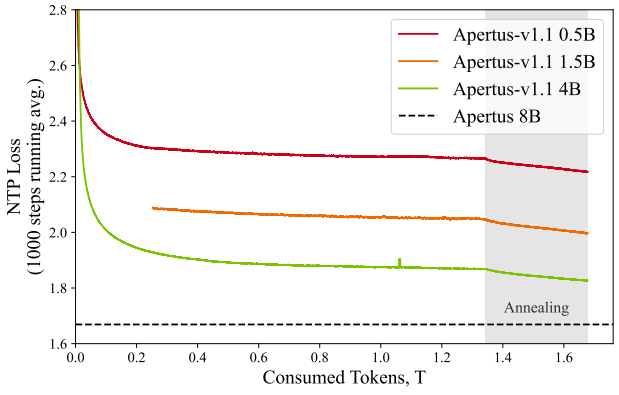

Figure 1. Training loss curves of Apertus-v1.1 models. Dashed line shows the loss of the teacher model (`Apertus-8B-2509`).

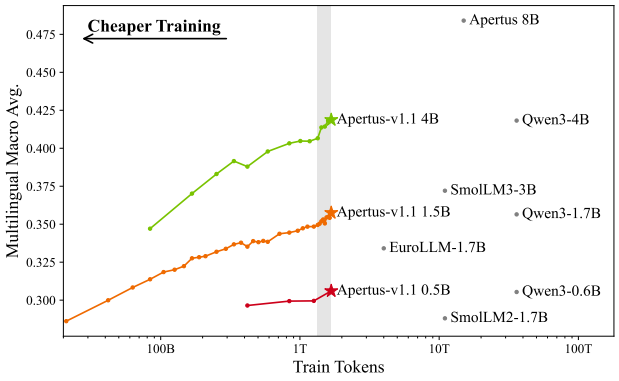

Figure 2. Multilingual performance macro average during pre-training of Apertus-v1.1 models and for a number of similar-sized models. Distillation allows Apertus-v1.1 models to achieve competitive performance while training on up to an order of magnitude less compute.

Training dynamics. Similar to Apertus, we use the AdEMAMix (Pagliardini et al., 2025) optimizer with WSD schedule and weight decay. Next-token prediction (NTP) loss shown in Figure 1. Multilingual macro downstream average shown in Figure 2. We observed no training instabilities and consistent improvement in downstream performance, especially during the learning rate annealing stage (highlighted in gray).

SFT and alignment. The supervised fine-tuning (SFT) stage followed immediately after pre-training. For it, we exactly reused the original Apertus SFT recipe, only adjusting the LR to match the post-annealing LR of Apertus-v1.1 models. For the subsequent alignment stage, we utilized a simplified DPO (Rafailov et al., 2024) setup.

Evaluations. Following the Apertus evaluation setup, we report multilingual benchmarks average during training in Figure 2, selected final pre-training metrics in Table 3, multilingual post-training evaluations in Table 4 and broader post-training evaluations in Appendix B. Unsurprisingly, the performance profile of Apertus-v1.1 models is extremely similar to `Apertus-8B-2509`, demonstrating great multilingual performance for the base models and good multilingual chat performance but lacking in certain capabilities

like instruction following and math.

2.2. Cost Analysis

Table 2. Cost for small LLM pre-training and distillation. Apertus-v1.1 is 2-10x cheaper than competing small LLM pre-training pipelines.

Stage	Tokens	FLOPs
Original pre-training Apertus-8B	15T	3.7E23
Logits generation from Apertus-8B	1.7T	1.4E22
Pre-training Apertus-v1.1 0.5B	1.7T	0.2E22
Pre-training Apertus-v1.1 1.5B	1.7T	0.8E22
Pre-training Apertus-v1.1 4B	1.7T	2.0E22
Pre-training Qwen3-0.6B	36T	6.5E22
Pre-training EuroLLM-1.7B	4T	1.7E22
Pre-training SmolLM2-1.7B	11T	5.6E22
Pre-training SmolLM3-3B	11T	9.9E22

As seen from Table 2, Apertus-v1.1 models used significantly less compute than similar-sized models, being trained on just 1.7T tokens, in contrast to the 15T tokens of Apertus. The cost of producing the logits from the 8B model is relatively small because one only needs to perform the forward pass to produce logits and the same logits only have to be computed once for the entire family of distilled models, dramatically cutting the teacher cost per-model. The total compute cost of the entire Apertus-v1.1 model family is 2.4E22 FLOPs. This is less than, for example, the cost of standalone pre-training for SmolLM2-1.7B and less than 12% of the original Apertus 8B pre-training cost.

3. Quantization

While pre-training distillation successfully generated the core Apertus-v1.1models at a fraction of the cost, adapting these models for highly constrained environments requires further optimization for specific hardware profiles. In this section, we consider the problem of adapting Apertus-v1.1 models to NVIDIA GPUs and mobile devices, demonstrating how quantization yields a wider range of optimal, specialized models at close to no cost.

3.1. Apertus-v1.1 Quantization Recipe

Baseline. We use GPTQ (Frantar et al., 2023), the most widely-used 1-shot LLM quantization method as our baseline. We gauge our improvement over it differently for base and instruction-tuned models:

- **For base models,** we measure the loss increase over the corresponding unquantized models on the validation set of ≈17M tokens from the original pre-training mixture (Apertus Phase 5 data). We test *weight+activation* (FP8, NVFP4) quantization for base models with focus on NVIDIA Blackwell GPUs, as we foresee their main usage in high-throughput scenarios such as data annotation and embedding.
- **For instruction-tuned models,** we measure the recovery of macro average over normalized few-shot accuracies on Arc (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021) and WinoGrande (Sakaguchi et al., 2019). We test *weight-only quantization* (INT2, INT3, INT4, INT6) for instruction-tuned models with focus on Apple devices (MLX) inference, as we foresee their main usage in memory-limited scenarios such as mobile and edge deployment.

Quantization-aware distillation (QAD). QAD is applied as a short recovery stage on a fully-trained model by treating the entire model as trainable parameters, quantizing its weights every forward pass and updating them with standard gradient-based method via straight-through estimation (Bengio et al., 2013), bridging the gap between full quantization-aware training and PTQ methods. Similar to pre-training distillation, teacher model logits (usually the corresponding unquantized model or a larger model from the same family) provide much richer signal for this phase, making it preferable to quantization-aware supervised fine-tuning. QAD has been shown to yield consistent improvement over 0-shot and 1-shot post-training quantization (PTQ) methods (Lee et al., 2025; Egiazarian et al., 2026; Xin et al., 2026).

The open access to the original pre-training set and SFT mixture utilized for both Apertus and Apertus-v1.1 pre- and post-training allows us to use it for QAD of these models with the highest degree of confidence that the distillation curriculum captures close to the entirety of the models' capability. We test QAD for both base and instruction-tuned models, using ≈100M tokens (we see only marginal improvement beyond that) of the pre-training or the SFT mixture accordingly. We use `Apertus-8B-2509` and `Apertus-8B-Instruct-2509` as a teacher in this scenario. Additional implementation details and hyperparameters are described in Appendix C.2.

Norm fusion. To further improve quantization quality, we propose the following zero-cost static model optimization: We scale attention's QKV and MLP's up projection matrices'

Table 3. Base models evaluations.

Model	Avg	ARC	HellaSwag	WinoGrande	XNLI	XCOPA	PIQA
Apertus-v1.1-0.5B	51.79	44.96	40.42	57.06	41.51	55.49	71.27
Apertus-v1.1-1.5B	56.66	52.66	48.31	61.72	42.94	59.76	74.54
Apertus-v1.1-4B	61.53	61.15	53.51	67.48	45.03	63.82	78.18
Apertus-8B	64.96	71.66	59.62	69.30	44.09	65.69	79.38
EuroLLM-1.7B	54.03	50.80	45.01	59.51	40.88	55.76	72.20
SmolLM2-1.7B	58.00	60.23	53.38	66.22	37.57	53.51	77.10
SmolLM-3B-Base	60.88	64.45	56.37	68.43	40.28	58.02	77.75
Qwen3-0.6B-Base	52.23	48.35	41.01	59.20	39.55	54.96	70.29
Qwen3-1.7B-Base	57.51	56.49	49.36	63.38	41.66	58.35	75.79
Qwen3-4B-Base	62.14	64.99	54.56	70.48	43.00	61.82	77.97

Table 4. Multilingual evaluations for instruction-tuned models. Each benchmark here is the multilingual version thereof (see Appendix B).

Model	Average	MMLU	TruthfulQA	Arc	IF	LogiQA
Apertus-v1.1-0.5B Instruct	0.318	0.258	0.461	0.225	0.328	0.279
Apertus-v1.1-1.5B-Instruct	0.382	0.377	0.451	0.266	0.434	0.276
Apertus-v1.1-4B-Instruct	0.473	0.504	0.506	0.332	0.550	0.296
Apertus-8B-Instruct-2509	0.534	0.553	0.524	0.368	0.689	0.290
EuroLLM-1.7B-Instruct	0.291	0.260	0.433	0.250	0.222	0.269
EuroLLM-9B-Instruct	0.480	0.520	0.465	0.322	0.613	0.345
gemma-3-270m-it	0.289	0.242	0.465	0.215	0.236	0.205
gemma-3-1b-it	0.406	0.409	0.457	0.250	0.509	0.379
gemma-3-4b-it	0.497	0.547	0.492	0.316	0.635	0.411
SmolLM2-1.7B-Instruct	0.348	0.365	0.452	0.213	0.364	0.246
SmolLM3-3B	0.479	0.507	0.500	0.270	0.637	0.365
Qwen3-0.6B	0.401	0.377	0.464	0.222	0.541	0.353
Qwen3-1.7B	0.457	0.477	0.490	0.251	0.611	0.414
Qwen3-4B	0.521	0.581	0.497	0.274	0.733	0.500

columns (input dimension) to have the same norm, multiplicatively fusing the reciprocal scales into the preceding layer-normalization layers' weights. The idea behind this is to normalize the magnitudes of weight values to prevent flush-to-zero of small-magnitude but important weights and weights adjacent to outlier channels.

The loss measurements for compressed base models and few-shot recovery measurements for the instruction-tuned models show that this yields the most improvement for smaller models. Additionally, although this normalization is mainly designed to assist with weight quantization, we find that it also improves weight+activation quantization (NVFP4), indicating that offloading these scales to activations doesn't hurt their compressibility.

Weight averaging. Weight averaging (arithmetic averaging of model weight tensors) of the last few checkpoints during the annealing stage has been shown to improve LLMs' resilience to post-training quantization (Ajroldi et al., 2025). To validate it, we tested weight averaging for the Apertus-v1.1 0.5B base model combined with various quantization formats and methods, including RTN, GPTQ (Frantar et al., 2023) and QAD. The results, shown in Figure 5, demonstrate that weight averaging reduces validation loss gap to BF16 by up to 10% for RTN, up to 2% for GPTQ and has close to *no discernible effect on QAD*. As a result, we did not include it in our final quantization pipeline.

Final quantization recipe. Our final recipe combines QAD with norm fusion to achieve just 0.1-0.2 validation loss increase for base and 90-104% few-shot accuracy recovery for instruction-tuned Apertus and Apertus-v1.1 models, as seen in Figure 4.

3.2. Pareto Optimality

As mentioned in the beginning of this section, we analyze base model quantization in the context of high-throughput applications and instruction-tuned model quantization in the

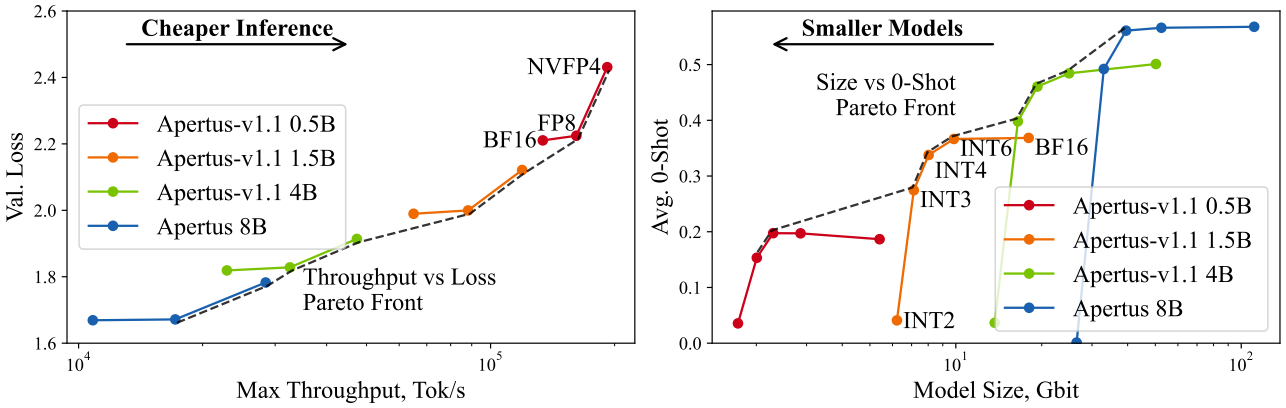

Figure 3. Visualization of the cost-accuracy trade-off for Apertus and Apertus-v1.1 models. Base models (left) are compared based on validation loss while instruction-tuned models (right) are compared based on downstream performance. Quantized models both optimize the trade-off and add intermediate points to the Pareto fronts.

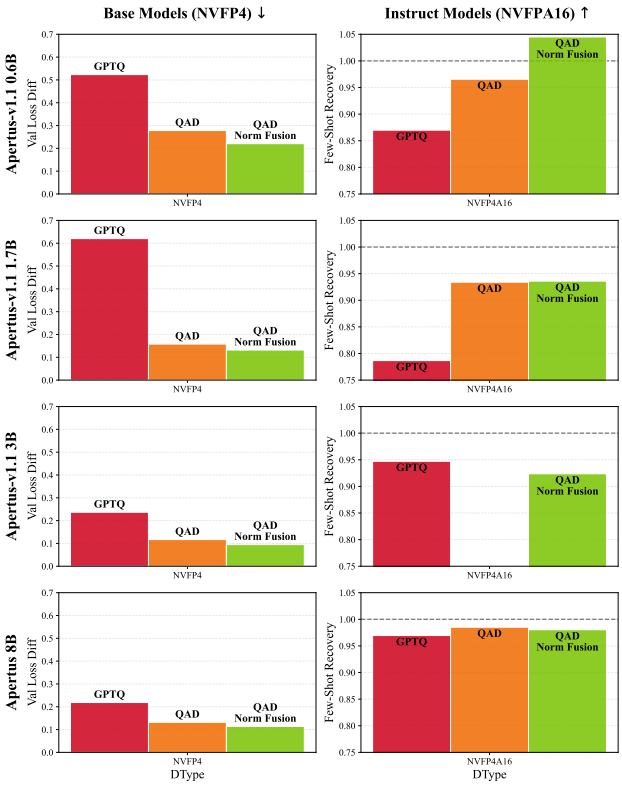

Figure 4. Apertus-v1.1 quantization recipe ablation.

context of memory-constrained deployment. Naturally, the corresponding cost can be measured for every model we trained (quantized or otherwise), along with a representative measure of it's capability, quantifying the cost-accuracy trade-off. Covering a larger range of costs is what drove the demand for smaller models in the first place, and in Figure 3 one can see this trade-off visualized.

What is interesting, is that quantized models not only shift the Pareto front (i.e., the enveloping curve) towards more efficient solutions (as seen, for example, by BF16 models almost never being optimal), but also adds more points on the frontier, allowing for more fine-grained control over cost. Without quantization, adding new points would have meant pre-training new models of intermediate sizes, which would have entailed spending compute in the order of trillions tokens. QAD, on the other hand, achieves high recovery after only a few tens of millions of tokens, cutting the cost by more than *four orders of magnitude*.

4. Released Checkpoints

We provide a comprehensive suite of pre-trained and instruction-tuned models across multiple quantization formats to support various hardware constraints and deployment scenarios. Table 5 summarizes all the checkpoints released as part of the Apertus and Apertus-v1.1 model families.

5. Conclusion

We validated pre-training distillation for multi-billion parameter models and multi-trillion token budgets, demonstrating how such model family expansion can be done at a tiny cost (less than 20%) of the teacher model training and far more cheaply than pre-training from scratch. In total, we re-

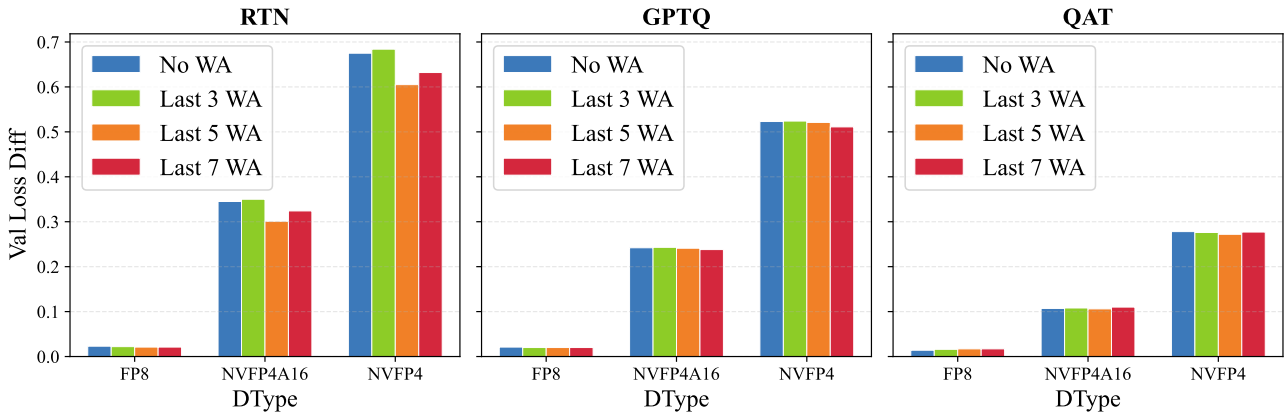

Figure 5. The effect of weight averaging (WA) over the last few base model checkpoints on post-training quantization for various data-types and algorithms. Checkpoints were taken every 1000 iterations.

Table 5. Overview of released Apertus and Apertus-v1.1 checkpoints. Click the Hugging Face logo to access the corresponding model weights.

Model	**BF16** Base	**BF16**	**FP8**	**NVFP4A16**	**INT3**	**INT4**	**INT6**
		Instruct					
Apertus-v1.1-0.5B	🤗	🤗	🤗	🤗	🤗	🤗	🤗
Apertus-v1.1-1.5B	🤗	🤗	🤗	🤗	🤗	🤗	🤗
Apertus-v1.1-4B	🤗	🤗	🤗	🤗	🤗	🤗	🤗
Apertus-8B-2509	🤗	🤗	🤗	🤗	✗	🤗	✗

lease 24 new model checkpoints, including the 3 pre-trained base models, 3 instruction-tuned models, 8 quantized checkpoints for NVIDIA devices, 10 quantized checkpoints for Apple devices, as well as all the code to reproduce training, post-training and quantization pipelines.

We hope our open-source, open-data and compliant recipe to be of use for LLM practitioners interested in producing and using small language models.

Acknowledgments

We would like to thank Dhia Garbaya for his help with configuring the models and Hanna Yukhymenko for tuning the post-training setup. This work was supported under project IDs a140 and infra01 as part of the Swiss AI Initiative, through a grant from the ETH Domain and computational resources provided by the Swiss National Supercomputing Centre (CSCS) under the Alps infrastructure. This research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/COE12.

References

Ajroldi, N., Orvieto, A., and Geiping, J. When, where and why to average weights?, 2025. URL https://arxiv.org/abs/2502.06761.

Apertus, P., Hernández-Cano, A., Hägele, A., Huang, A. H., Romanou, A., Solergibert, A.-J., Pasztor, B., Messmer, B., Garbaya, D., Ďurech, E. F., Hakimi, I., Giraldo, J. G., Ismayilzada, M., Foroutan, N., Moalla, S., Chen, T., Sabolčec, V., Xu, Y., Aerni, M., AlKhamissi, B., Mariñas, I. A., Amani, M. H., Ansaripour, M., Badanin, I., Benoit, H., Boros, E., Browning, N., Bösch, F., Böther, M., Canova, N., Challier, C., Charmillot, C., Coles, J., Deriu, J., Devos, A., Drescher, L., Dzenhaliou, D., Ehrmann, M., Fan, D., Fan, S., Gao, S., Gila, M., Grandury, M., Hashemi, D., Hoyle, A., Jiang, J., Klein, M., Kucharavy, A., Kucherenko, A., Lübeck, F., Machacek, R., Manitaras, T., Marfurt, A., Matoba, K., Matrenok, S., Mendonça, H., Mohamed, F. R., Montariol, S., Mouchel, L., Najem-Meyer, S., Ni, J., Oliva, G., Pagliardini, M., Palme, E., Panferov, A., Paoletti, L., Passerini, M., Pavlov, I., Poiroux, A., Ponkshe, K., Ranchin, N., Rando, J., Sauser, M., Saydaliev, J., Sayfiddinov, M. A., Schneider, M., Schuppli, S., Scialanga, M., Semenov, A., Shridhar, K., Singhal, R., Sotnikova, A., Sternfeld, A., Tarun, A. K., Teiletche, P., Vamvas, J., Yao, X., Zhao, H., Ilic, A., Klimovic, A., Krause, A., Gulcehre, C., Rosenthal, D., Ash, E., Tramèr, F., VandeVondele, J., Veraldi, L., Rajman, M., Schulthess, T., Hoefler, T., Bosselut, A., Jaggi,

M., and Schlag, I. Apertus: Democratizing open and compliant llms for global language environments, 2025. URL https://arxiv.org/abs/2509.14233.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation, 2013. URL https://arxiv.org/abs/1308.3432.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Conneau, A., Rinott, R., Lample, G., Williams, A., Bowman, S., Schwenk, H., and Stoyanov, V. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2475–2485, 2018.

Dac Lai, V., Van Nguyen, C., Ngo, N. T., Nguyen, T., Dernoncourt, F., Rossi, R. A., and Nguyen, T. H. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. *arXiv e-prints*, pp. arXiv–2307, 2023.

Egiazarian, V., Castro, R. L., Kuznedelev, D., Panferov, A., Kurtic, E., Pandit, S., Marques, A., Kurtz, M., Ashkboos, S., Hoefler, T., and Alistarh, D. Bridging the gap between promise and performance for microscaling fp4 quantization, 2026. URL https://arxiv.org/abs/2509.23202.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Huang, A. H. and Schlag, I. Deriving activation functions using integration, 2025. URL https://arxiv.org/abs/2411.13010.

Lee, J. H., Shin, S., Kim, V., You, J., and Chen, A. Unifying block-wise ptq and distillation-based qat for progressive quantization toward 2-bit instruction-tuned llms, 2025. URL https://arxiv.org/abs/2506.09104.

Liu, Z., Zhao, C., Iandola, F., Lai, C., Tian, Y., Fedorov, I., Xiong, Y., Chang, E., Shi, Y., Krishnamoorthi, R., Lai, L., and Chandra, V. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases, 2024. URL https://arxiv.org/abs/2402.14905.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Scao, T. L., Bari, M. S., Shen, S., Yong, Z.-X., Schoelkopf, H., Tang, X., Radev, D., Aji, A. F., Almubarak, K., Albanie, S., Alyafeai, Z., Webson, A., Raff, E., and Raffel, C. Crosslingual generalization through multitask finetuning, 2022.

Pagliardini, M., Ablin, P., and Grangier, D. The ademamix optimizer: Better, faster, older. In Yue, Y., Garg, A., Peng, N., Sha, F., and Yu, R. (eds.), *International Conference on Learning Representations*, volume 2025, pp. 64715–64757, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/a2cf225ba392627529efef14dc857e22-Paper-Conference.pdf.

Peng, H., Lv, X., Bai, Y., Yao, Z., Zhang, J., Hou, L., and Li, J. Pre-training distillation for large language models: A design space exploration, 2024. URL https://arxiv.org/abs/2410.16215.

Ponti, E. M., Glavaš, G., Majewska, O., Liu, Q., Vulić, I., and Korhonen, A. Xcopa: A multilingual dataset for causal commonsense reasoning, 2020. URL https://arxiv.org/abs/2005.00333.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model, 2024. URL https://arxiv.org/abs/2305.18290.

Romanou, A., Foroutan, N., Sotnikova, A., Nelaturu, S. H., Singh, S., Maheshwary, R., Altomare, M., Chen, Z., Haggag, M., Amayuelas, A., et al. Include: Evaluating multilingual language understanding with regional knowledge. In *International Conference on Learning Representations*, volume 2025, pp. 83291–83322, 2025.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale, 2019. URL https://arxiv.org/abs/1907.10641.

Singh, S., Romanou, A., Fourrier, C., Adelani, D. I., Ngui, J. G., Vila-Suero, D., Limkonchotiwat, P., Marchisio, K., Leong, W. Q., Susanto, Y., Ng, R., Longpre, S., Ko, W.-Y., Ruder, S., Smith, M., Bosselut, A., Oh, A., Martins, A. F. T., Choshen, L., Ippolito, D., Ferrante, E., Fadaee, M., Ermis, B., and Hooker, S. Global mmlu: Understanding and addressing cultural and linguistic biases in multilingual evaluation, 2025. URL https://arxiv.org/abs/2412.03304.

Xin, M., Priyadarshi, S., Xin, J., Kartal, B., Vavre, A., Thekkumpate, A. K., Chen, Z., Mahabaleshwarkar, A. S., Shahaf, I., Bercovich, A., Patel, K., Velury, S. V., Luo, C., Cheng, Z., Chen, J., Yu, C.-H., Ping, W., Rybakov, O., Tajbakhsh, N., Olabiyi, O., Stosic, D., Wu, D., Han, S., Chung, E., Sreenivas, S. T., Catanzaro, B., Suhara, Y., Blankevoort, T., and Mao, H. Quantization-aware distillation for nvfp4 inference accuracy recovery, 2026. URL https://arxiv.org/abs/2601.20088.

Yang, Y., Zhang, Y., Tar, C., and Baldridge, J. PAWS-X: A cross-lingual adversarial dataset for paraphrase identification. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3687–3692, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1382. URL https://aclanthology.org/D19-1382/.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Table 6. Additional hyper-parameters.

Model	LR	GBS	Total Iterations
Apertus-v1.1-0.5B	6e-4	512	800000
Apertus-v1.1-1.5B	3e-4	512	800000
Apertus-v1.1-4B	2e-4	1024	400000

A. Codebases

The full codebases for the pre-training distillation, post-training, evaluations and quantization stages of the pipeline are available on GitHub.

- `Megatron-LM-Distill`: A fork of Megatron-LM with added functionality for teacher logits generation and saving as well as pre-training distillation.
- `posttraining`: The original post-training codebase from Apertus that was reused for this project.
- `qat-suite`: A lightweight quantization suite with support for vLLM and MLX data formats and various quantization algorithms, including QAD.
- `evals`: The Apertus pre-training evaluation suite.
- `evals-post-train`: The Apertus post-training evaluation suite.

For the data preparation scripts, please refer to the original Apertus report (Apertus et al., 2025).

B. Evaluation Suite Details

For the evaluations reported in Tables 3 and 4, we used the publicly-available Apertus evaluation suite. The multilingual macro average shown in Figure 2 includes INCLUDE (Romanou et al., 2025), XCOPA (Ponti et al., 2020), XNLI (Conneau et al., 2018), XWinograd (Muennighoff et al., 2022), PAWS-X (Yang et al., 2019), Multilingual Arc (Dac Lai et al., 2023), Global MMLU (Singh et al., 2025) and Multilingual HellaSwag (Dac Lai et al., 2023).

C. Additional Hyper-Parameters

C.1. Pre-Training Details

Additional per-model pre-training hyper-parameters are shown in Table 6.

C.2. QAT Details

For the base models, we sample ≈130M tokens uniformly from the unused remainder of the gathered pre-training data. For the instruction-tuned models, we sample ≈60M uniformly from the Apertus SFT mixture. We train with AdamW (Loshchilov & Hutter, 2019) with cosine LR schedule. For base models, we use the same sequence length and batch size as in pre-training. For instruction-tuned models, we use slightly larger batch size of 512-2048 to compensate for smaller length of some post-training sequences. Similar to pre-training distillation, we pre-compute and store the sparse logits from the teacher model (`Apertus-8B-2509` for base models and `Apertus-8B-Instruct-2509` for instruction-tuned models) once and re-use them for all student model and quantization format combinations.