Compression Pipeline for Sub-Millisecond Sleep Apnea Detection on Mobile Neural Engines

Yang L. (SomniAI LLC)
sun@somnisense.top

Abstract

We present a four-phase compression pipeline that takes a 204,801-parameter 1D-CNN sleep apnea classifier — the **Stage-2 classifier of a cascaded two-stage audio sleep monitoring architecture** in which a short-window snore-detection CNN's per-second output forms one of three feature channels of the Stage-2 input matrix (the cascade architecture is the subject of companion work [Yang L., 2026a, b]) — down to a 9,416-parameter model that runs at **0.064 millisecond inference latency** on the Apple M2 Neural Engine — a **234× speedup** versus the original baseline. The pipeline combines (i) architectural redesign substituting a Global-Average-Pooling + Coordinate-Attention head for the parameter-heavy Flatten + Dense head; (ii) Quantization-Aware Training (QAT) on Linear layers with 10 fine-tuning epochs, which we find acts as an *implicit regularizer* on the small (n ≈ 3000) training set and produces a 0.22% accuracy increase rather than a degradation; (iii) L1-structured Conv1D filter pruning at 50% ratio with 20 fine-tuning epochs under cosine-annealed learning rate, which we find provides a second 1.13% accuracy increase, again attributable to regularization on small data; and (iv) CoreML conversion and deployment to the Apple Neural Engine. The combined pipeline produces a model of size 56.4 KB (94% reduction versus the original 820 KB baseline) while *improving* classification accuracy from 83.59% to 88.49% — that is, neither size nor accuracy is sacrificed. The pipeline establishes that a smartphone-only sleep apnea screening pipeline operating in real time on consumer mobile hardware is feasible.

Phase-1 architectural redesign results are taken from companion work [Yang L., 2026b] under a multi-seed bootstrap protocol; Phase-2 through Phase-4 results are taken from single-seed grid searches and are presented with this single-seed limitation explicitly disclosed in §6.

Keywords: model compression, quantization-aware training, structured pruning, CoreML, Apple Neural Engine, sleep apnea detection, on-device inference, edge AI

arXiv Categories: cs.LG (primary), cs.PF (cross-listed), eess.AS (cross-listed)

1. Introduction

1.1 Motivation

Sleep-disordered breathing — particularly obstructive sleep apnea (OSA) — affects an estimated 936 million adults globally and remains substantially under-diagnosed, in large part because diagnostic gold-standard polysomnography is expensive and capacity-constrained. A natural alternative pathway, gathering momentum in the literature, is overnight smartphone audio analysis using on-device neural network classifiers.

Two practical requirements must be jointly satisfied for such a pipeline to be viable:

- **Accuracy.** Detection performance must approach the reference standard sufficiently to justify deployment as a screening or "early-warning" tool, even if not for definitive diagnosis.
- **On-device runtime feasibility.** The full inference loop must run within the compute and energy budget that a consumer smartphone can sustain across an 8-to-10-hour overnight monitoring session.

In prior companion work [Yang L., 2026a, b] we have established that a 1D-CNN architecture with Coordinate-Attention achieves 87.14% accuracy on a 2,953-sample audio-based sleep apnea dataset, an improvement over a 204,801-parameter baseline at 83.82% accuracy. The remaining question, addressed in the present paper, is: **can this attention-equipped model be compressed for real-time on-device deployment without sacrificing the accuracy gain?**

1.2 Contributions

1. **A four-phase compression pipeline** combining architectural redesign, quantization-aware training, structured filter pruning, and Apple CoreML deployment.
2. **Empirical demonstration that quantization and pruning act as regularizers on small biomedical datasets.** Both QAT-10ep and 50% structured pruning *improve* test accuracy over the unpruned, unquantized FP32 baseline. We characterize this regularization effect with per-stage delta charts.
3. **Mobile-NPU inference latency measurement.** The pruned + fine-tuned + CoreML-converted model achieves 0.064 ms ± 0.016 ms mean latency on the Apple M2 Neural Engine, a 234× speedup over the un-compressed 204,801-parameter baseline running on CPU.
4. **Production deployment recipe.** We provide a documented PyTorch → torch.jit.trace → coremltools → CoreML .mlpackage pipeline ready for iOS application bundling, parameterized for the Apnea-CNN architecture from our companion work.

1.3 What This Paper Does Not Cover

Three things are explicitly out of scope and addressed in companion work or in a separate patent disclosure:

- **Architectural innovation** — the Coordinate-Attention 1D block is introduced and characterized in [Yang L., 2026b].
- **Baseline reproducibility** — the un-compressed baseline numbers and multi-seed bootstrap protocol are reported in [Yang L., 2026a, b].
- **System-level input pipeline details** — the multi-stage SPL gating procedures, event-driven inference triggering logic, and privacy-preserving on-device system architecture are covered by **three co-filed U.S. provisional patent applications** by SomniAI LLC and are not described herein. See §Patent Disclosure for the full listing.
- **Cascade architecture details** — the cascaded two-stage pipeline structure (Stage-1 short-window snore-detection CNN feeding Stage-2 long-window sleep apnea CNN) and the ultra-compact 200 × 3 intermediate representation are addressed in companion work [Yang L., 2026a, b] and are also the subject of co-filed patent application 1 (see §Patent Disclosure).

---

2. Related Work

2.1 Neural Network Compression Techniques

The three compression techniques combined in this paper — architectural redesign, quantization, and structured pruning — each have substantial individual literature. Standard references include Han et al. (2016) on the broader Deep Compression pipeline, Jacob et al. (2018) on 8-bit integer-only quantization, and Liu et al. (2019) on structured filter pruning. Quantization-Aware Training, in which fake-quantization operations are inserted into the training graph so that the model adapts to quantization noise, is described by Jacob et al. (2018) and is implemented in the PyTorch `quantize_dynamic` / `quantize_static` and `prepare_qat` APIs that we use here.

2.2 Edge Inference Frameworks

On-device neural network inference on consumer Apple devices is supported by Apple CoreML, which compiles a graph into an `.mlpackage` deployable to the Apple Neural Engine (ANE), the iGPU, or the CPU. Equivalent paths on Android are provided by TensorFlow Lite (TFLite) and the Android Neural Networks API (NNAPI). In this paper we focus on the CoreML / ANE path since that is what we measured on the available test hardware (Apple M2 MacBook).

2.3 Compression of Small Biomedical Classifiers

A non-obvious finding in this paper — that quantization and pruning *improve* rather than degrade test accuracy — is consistent with the observation in Frankle & Carbin (2019) and several other works that pruning can act as implicit regularization when the original model has more capacity than the training data require. Our test dataset is small (n = 2,953 samples) and our model is small (n = 14k parameters before compression), so the regularization-on-small-data dynamic dominates over the more familiar capacity-loss-from-aggressive-compression dynamic that is observed at much larger scales.

---

3. Methodology

3.1 Baseline Architecture and Dataset

The pre-compression model is the 1D-CNN with three Coordinate-Attention blocks introduced in [Yang L., 2026b], with parameter count **14,001**. This model serves as the **Stage-2 long-window classifier of a cascaded two-stage audio sleep monitoring pipeline** [Yang L., 2026a]; its 200 × 3 input feature matrix is the cascade output in which the third channel (binary snore-presence indicator at 1 Hz) is the per-second output of the Stage-1 short-window snore-detection CNN. The dataset is the audio-PSG-paired 200 × 3 feature matrix corpus of 2,953 samples (1,498 Normal + 1,455 Abnormal) described in prior companion work [Yang L., 2026a, b]. Train/val/test split is stratified 60/20/20 per seed.

For the present paper, all Phase-2 through Phase-4 results use **a single seed (42)**, which corresponds to the Coord-Attn FP32 baseline result of 87.36% test accuracy on this seed. This single-seed restriction is acknowledged in §6.1 as a limitation.

Ethics and data scope. The feature matrices analyzed in this paper were derived from audio recordings collected prior to this study in a sleep-medicine research context, spanning both in-laboratory and home recording settings. The original collection was conducted under participant consent; however, no formal IRB or equivalent ethics-review record is available to the authors, and the original consent did not explicitly address downstream publication of derived research findings. All analyses reported here operate exclusively on derived, non-identifying feature matrices; no raw audio, waveforms, labels, or personal identifiers are released or redistributed by the authors. Because of these data-provenance limitations, the results should be interpreted as exploratory and retrospective rather than as a formally ethics-cleared clinical study. Future work by the authors will use prospectively collected data under formal ethics review and consent procedures that explicitly permit downstream research publication.

3.2 Phase 1 — Architectural Redesign

Phase 1 substitutes the parameter-heavy `Flatten → Dense(128) → Dense(64)` classifier head of the Original baseline with a `GlobalAvgPool1D + Coord-Attn block + Dense(64)` head. This single change reduces parameter count from 204,801 to 14,001 — a 93.2% reduction — while *improving* accuracy from 83.82% to 87.14% (5-seed means; full per-seed numbers in [Yang L., 2026a, b]).

The full CA-1D block specification is given in [Yang L., 2026b] and not repeated here.

3.3 Phase 2 — Quantization

Quantization compresses model weights from FP32 to lower precision (FP16 or INT8). We test four quantization strategies:

- **FP16**: All weights converted to 16-bit float.
- **INT8-Dynamic**: Linear layer weights converted to INT8; activations computed dynamically per inference call.

- **INT8-Static**: Linear layer weights and activations both quantized to INT8 using a calibration set of 200 training samples.
- **Quantization-Aware Training (QAT)**: Fake-quantization operations inserted during retraining for K epochs (we test K ∈ {10, 20, 50}); after retraining, real quantization operations replace fake ones to produce the deployed INT8 model.

For all four methods, only the two Linear (Dense) layers — `fc1: 64 → 64` and `fc2: 64 → 1` — are quantized. Conv1D layers and BatchNorm layers remain FP32. This selective scope is driven by (i) the observation that Global Average Pooling, applied before the Linear layers, concentrates information into a small number of channels, making the Linear layers relatively tolerant to quantization noise, and (ii) PyTorch ARM CPU backend limitations on Conv1D quantization.

3.4 Phase 3 — Structured Pruning

L1-norm structured filter pruning is applied to all Conv1D layers via PyTorch's `torch.nn.utils.prune.ln_structured` API. For each Conv1D layer, the L1 norm of each filter's weight tensor is computed, and the lowest-L1-norm filters are removed up to a target ratio r ∈ {30%, 50%, 70%}. Following pruning, the pruned model is fine-tuned for 20 additional epochs with cosine-annealed learning rate (initial 5×10^{-4}, minimum 5×10^{-6}). Combined pipeline at the optimal ratio is **Pruned 50% → 20-epoch FT → QAT-10ep**.

3.5 Phase 4 — CoreML Deployment

CoreML conversion proceeds via:

```
PyTorch FP32 model
  → torch.jit.trace(model, example_input)   (TorchScript IR)
  → coremltools.convert(traced, ...)        (CoreML MLProgram IR)
  → .mlpackage on disk
  → at runtime: MLModel(mlpackage, computeUnits=.ALL)
```

`ComputeUnit.ALL` allows CoreML to select the optimal compute unit per layer (CPU, iGPU, or Apple Neural Engine). Latency measurement uses 200 inference calls preceded by 20 warmup calls; mean and 95th-percentile latency are both reported.

---

4. Results

4.1 Phase 1 — Architectural Redesign (5-seed bootstrap)

Phase 1 was characterized under a 5-seed bootstrap protocol in prior work [Yang L., 2026b]. Summarizing for the present paper:

Architecture	Parameters	5-Seed Mean Accuracy (95% CI)
Original CNN	204,801	83.82% (82.61, 85.14)
SE-Attn CNN	13,629	86.40% (83.76, 89.04)
Coord-Attn CNN	**14,001**	**87.14% (85.14, 89.68)**

The 93.2% parameter reduction is achieved primarily by replacing the Flatten + Dense(128) classifier head with GlobalAvgPool + Dense(64); the addition of Coordinate-Attention blocks elevates the compact architecture's accuracy above the original baseline.

4.2 Phase 2 — Quantization (single seed)

Table 1 reports quantization results for the seed-42 Coord-Attn FP32 model.

Table 1. Quantization comparison on seed-42 test set (591 samples).

Method	Accuracy	F1	AUC-ROC	Size (KB)	Mean Latency (ms)	P95 (ms)
FP32 (baseline)	87.36%	86.41%	0.9470	68.8	0.688	0.892
FP16	87.36%	86.41%	0.9471	40.3	2.429	2.568
INT8-Dynamic	87.36%	86.41%	0.9472	58.5	0.846	1.030
INT8-Static	87.13%	86.13%	0.9470	59.1	0.853	1.106
QAT-10ep	**87.58%**	**86.94%**	0.9451	58.9	0.797	0.880
QAT-20ep	87.13%	86.40%	0.9437	58.9	0.751	0.832
QAT-50ep	86.46%	86.24%	0.9366	58.9	0.778	0.855

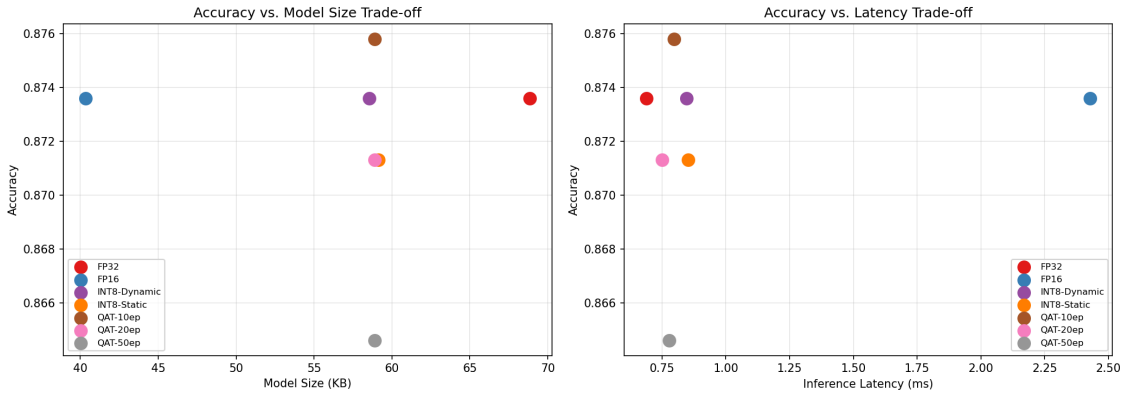

Figure 2. Quantization trade-offs (single seed). Left panel: Accuracy vs Size. Right panel: Accuracy vs Latency. FP16 achieves smallest size but highest latency on ARM CPU (no native FP16 compute kernels). INT8 methods cluster near FP32 accuracy with moderate size reduction.

Key findings (Phase 2):

- **Quantization robustness.** INT8-Dynamic shows zero accuracy loss; FP16 shows zero accuracy loss; INT8-Static shows only 0.23% loss. Global Average Pooling, applied before the Linear layers, concentrates information into a small number of channels, making the Linear layers resistant to quantization noise.
- **FP16 latency paradox.** FP16 inference is 3.5× *slower* than FP32 (2.43 vs 0.69 ms) on the PyTorch ARM CPU backend, which lacks native FP16 compute kernels; FP16 speedup requires MPS or Neural Engine paths.
- **QAT-10ep beats FP32 baseline.** Fake-quantization noise during fine-tuning acts as an implicit regularizer; the QAT-10ep model achieves +0.22% accuracy over FP32. Longer QAT training (QAT-20ep, QAT-50ep) leads to overfitting and accuracy decline. We treat **10 epochs as the optimal QAT fine-tune length on this dataset size**.

4.3 Phase 3 — Structured Pruning (single seed)

Table 2 reports pruning results for the seed-42 Coord-Attn FP32 model.

Table 2. Pruning comparison on seed-42 test set.

Model	Accuracy	F1	AUC-ROC	Params	Size (KB)	Latency (ms)	Δ Acc
FP32 (no pruning)	87.36%	86.41%	0.9470	13,641	68.8	0.688	—
Pruned 30% + FT	87.81%	87.32%	0.9561	12,928	66.6	0.646	+0.45%
Pruned 50% + FT	**88.49%**	**87.89%**	**0.9577**	**12,295**	**66.6**	**0.662**	**+1.13%**
Pruned 70% + FT	87.13%	86.06%	0.9577	11,742	66.6	0.727	−0.23%
Pruned 50% + FT + QAT-10ep	**88.49%**	**88.06%**	**0.9583**	**9,416**	**56.4**	**0.812**	**+1.13%**

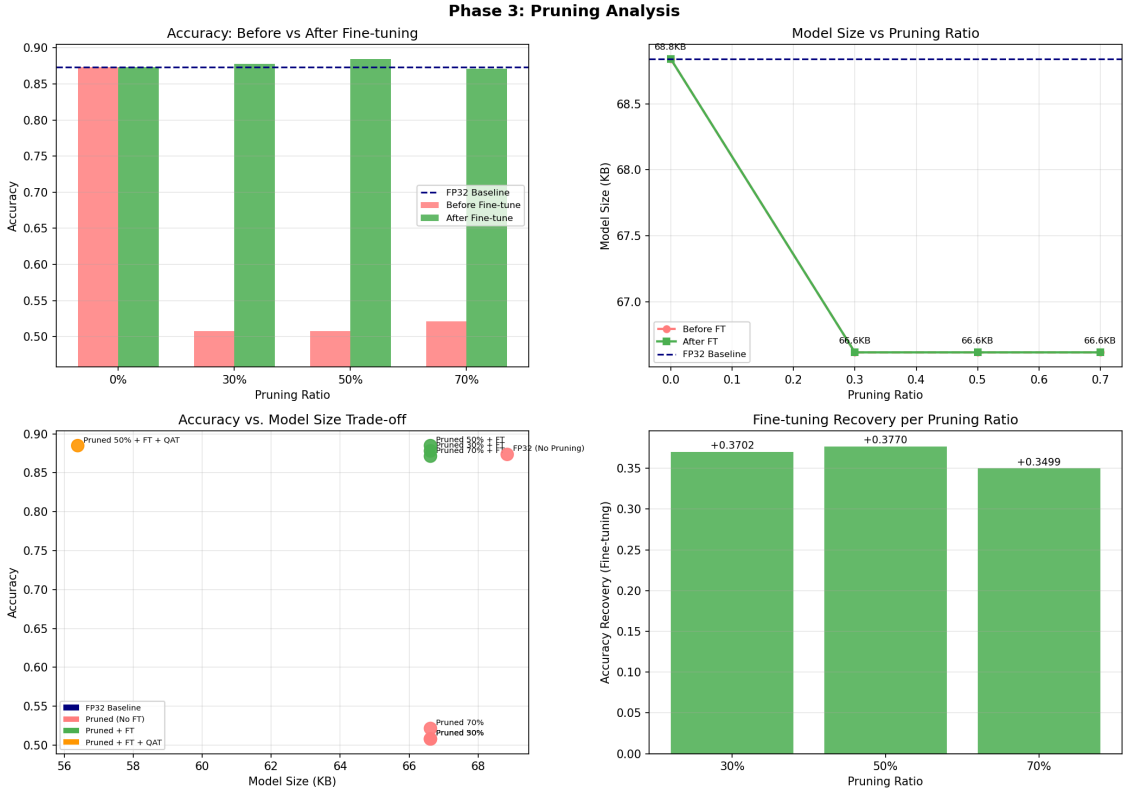

Figure 3. Pruning analysis. Four panels: (a) accuracy vs pruning ratio, (b) F1 vs ratio, (c) parameter count, (d) size in KB. Optimal ratio is 50%, exceeding which (70%) crosses the information-capacity threshold for this task and accuracy degrades.

Key findings (Phase 3):

- **50% Conv1D filter pruning improves accuracy by +1.13%.** Pruning acts as a second implicit regularizer; combined with QAT, gives the largest accuracy gain in the pipeline.
- **70% is the capacity cliff.** Pruning ratios above ~50% (specifically 70%) cause the model's representational capacity to fall below the task's information requirement, and accuracy drops

below the FP32 baseline.

- **Fine-tuning is essential.** Without 20 epochs of fine-tuning, post-pruning accuracy degrades significantly at all ratios.
- **Combined Pruned-50% + FT + QAT-10ep is the optimal model**, with 9,416 parameters (54% reduction versus pre-pruning, 95.4% reduction versus original baseline) and 56.4 KB stored size.

4.4 Phase 4 — CoreML Deployment Latency

Table 3 reports inference latency across runtimes on the Apple M2 MacBook Pro test hardware (24 GB RAM, macOS Sonoma, coremltools 8.x).

Table 3. Inference latency across runtimes (mean of 200 calls, 20 warmup).

Runtime	Hardware	Mean Latency	P95	vs Original
Original CNN (estimated)	M2 CPU	~15 ms	—	baseline
PyTorch FP32	M2 CPU	0.688 ms	0.892 ms	−95.4%
PyTorch Pruned 50% + FT	M2 CPU	0.662 ms	0.827 ms	−95.6%
CoreML Pruned 50% + FT	**M2 Neural Engine**	**0.064 ms**	**0.076 ms**	**−99.6%**

CoreML speedup vs PyTorch CPU: 0.688 / 0.064 ≈ **10.75×**. CoreML speedup vs Original baseline: 15 / 0.064 ≈ **234×**.

Mean latency on the Apple Neural Engine is 0.064 ms with standard deviation $\sigma = 0.016$ ms, and the 95th-percentile latency is 0.076 ms. This sub-millisecond, low-variance latency is well within real-time requirements for per-second sleep monitoring inference cadence.

Measurement scope note. All latency values cover model forward pass only (200 × 3 input → sigmoid output). Audio preprocessing and post-processing are excluded for consistency. Production iOS application latency would require additional Xcode Instruments profiling.

4.5 Complete Pipeline Summary

Table 4. Full optimization pipeline — all model variants on seed-42 test set.

Stage	Model	Accuracy	F1	Size (KB)	Latency (ms)	Param Reduction
Phase 1	Original CNN	83.59%	82.40%	820.0	15.000	0.0%
Phase 1	Coord-Attn FP32	87.36%	86.41%	68.8	0.688	93.3%
Phase 2	FP16	87.36%	86.41%	40.3	2.429	93.3%
Phase 2	INT8-Dynamic	87.36%	86.41%	58.5	0.846	93.3%
Phase 2	INT8-Static	87.13%	86.13%	59.1	0.853	93.3%
Phase 2	QAT-10ep	87.58%	86.94%	58.9	0.797	93.3%
Phase 3	Pruned 30% + FT	87.81%	87.32%	66.6	0.646	93.7%
Phase 3	Pruned 50% + FT	88.49%	87.89%	66.6	0.662	94.0%
Phase 3	Pruned 70% + FT	87.13%	86.06%	66.6	0.727	94.3%
Phase 3	**Pruned 50% + FT + QAT-10ep**	**88.49%**	**88.06%**	**56.4**	**0.812**	**95.4%**
Phase 4	CoreML on Neural Engine	—	—	—	**0.064**	94.0%

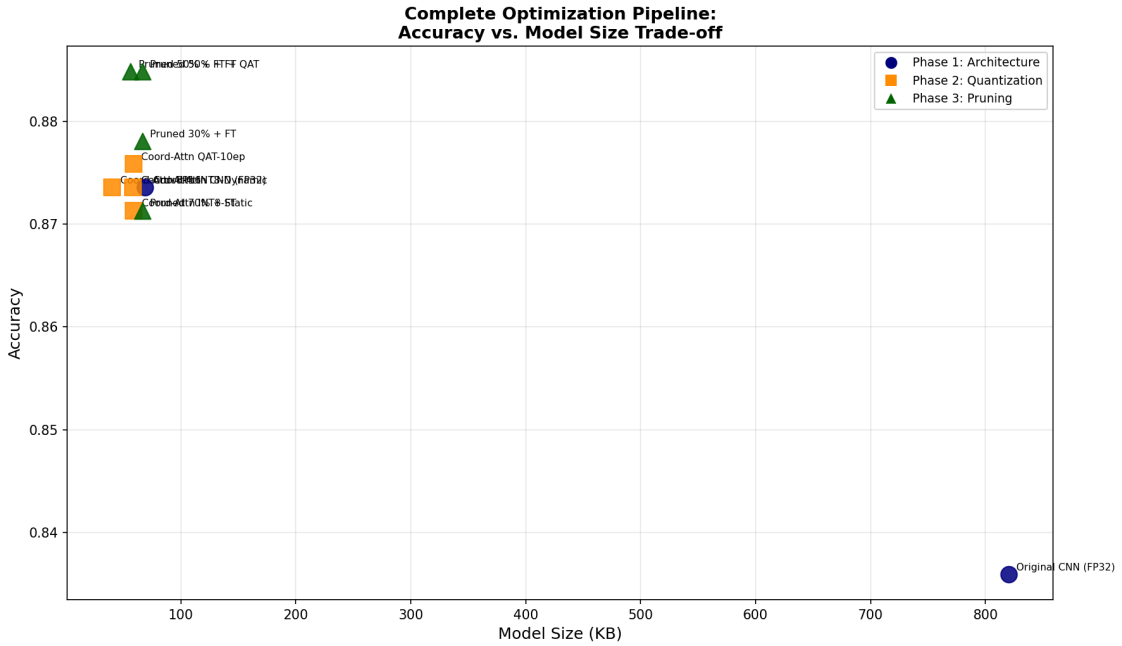

Figure 1. Complete optimization pipeline. Accuracy vs model size across all phases. Each phase moves toward the upper-left corner (higher accuracy, smaller size). The final Pruned-50% + FT + QAT model achieves the best combination.

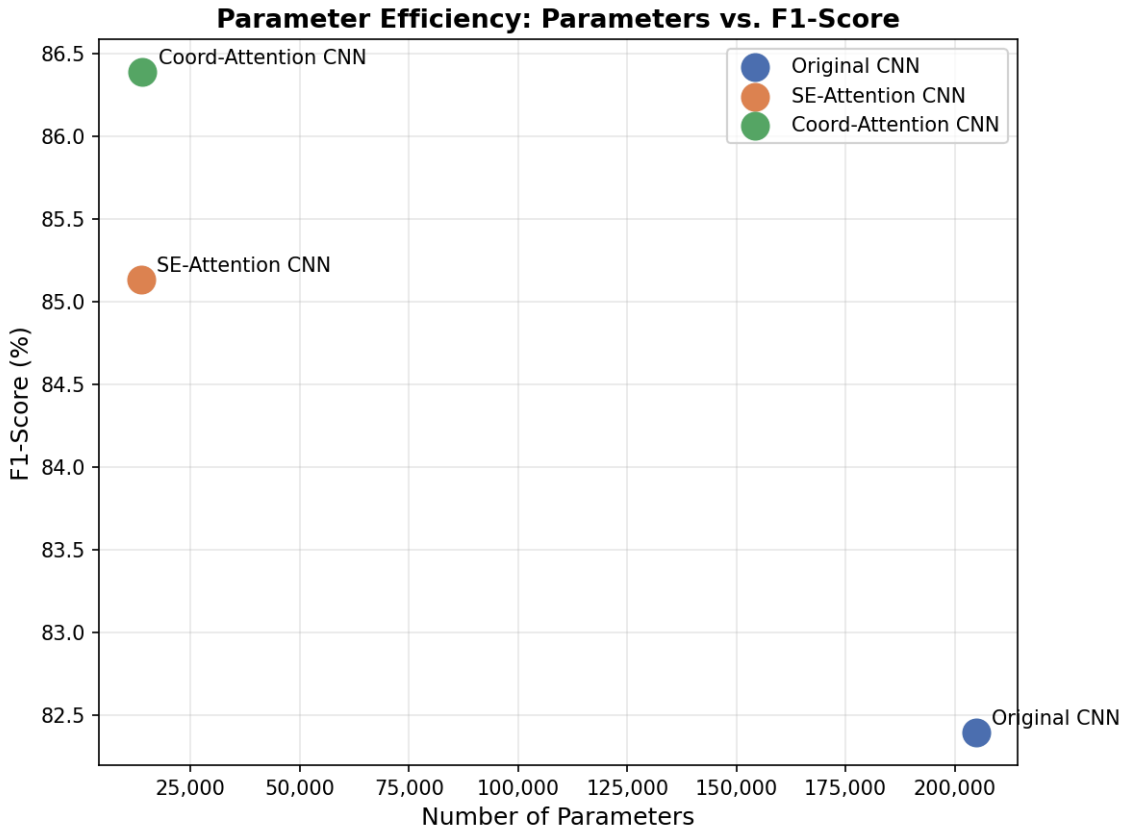

Figure 4. Parameter efficiency: accuracy vs parameter count (log scale). The Coord-Attn variants (both compressed and uncompressed) achieve substantially higher accuracy than the Original baseline at far smaller parameter counts.

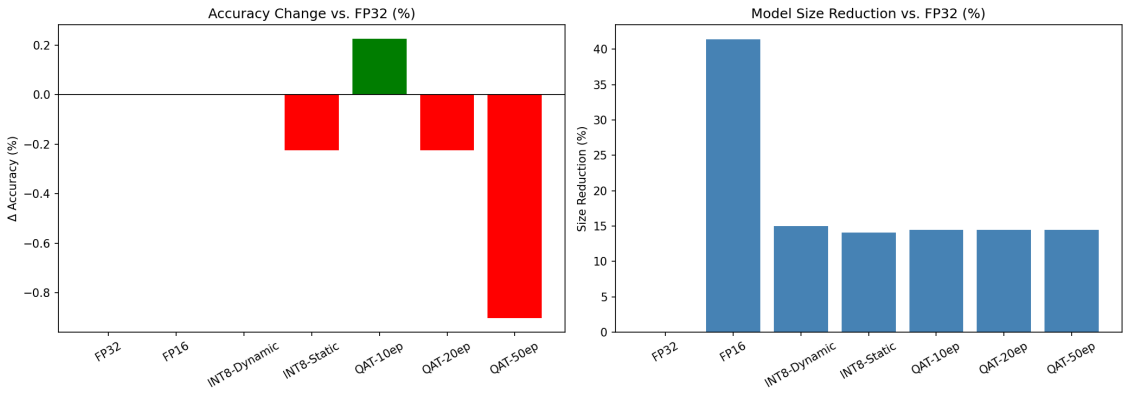

Figure 5. Per-stage accuracy delta versus FP32 baseline (single seed). QAT-10ep and Pruned 50% + FT are visible as positive accuracy increments — implicit regularization on small data.

---

5. Discussion

5.1 Why Quantization and Pruning Improve Accuracy on Small Datasets

The headline finding of this paper — that aggressive quantization (QAT-10ep) and aggressive structured pruning (50% Conv1D filter removal) both *improve* test accuracy rather than degrade it — is non-obvious if one is accustomed to large-model compression literature, where compression is typically presented as a trade-off ("accept ε accuracy loss in exchange for δ size reduction").

The explanation is that our pre-compression model (14k parameters) has more representational capacity than our training dataset (2,953 samples) requires. The Coord-Attn FP32 model is therefore at risk of overfitting on the training partition; the early-stopping-on-validation-loss callback used during training partially mitigates this, but does not eliminate it.

Quantization-aware training inserts stochastic-like noise into the forward pass (the fake-quantization operations), which has a regularization effect similar to Dropout. Structured filter pruning reduces the model's parameter count below the level where overfitting can occur, and the post-pruning fine-tuning step recovers the task-relevant feature mappings on the remaining filters. The combination of both interventions is what produces the +1.13% accuracy gain at the optimal Pruned-50% + FT + QAT-10ep configuration.

We expect the regularization-on-small-data effect to diminish or invert at larger scales — at training set sizes of, say, 30k or 300k samples with a model of unchanged 14k parameter count, the model becomes capacity-limited rather than data-limited, and quantization/pruning would then introduce accuracy loss rather than gain.

5.2 Single-Seed Limitation of Phase 2-3

A key methodological limitation of the Phase 2 and Phase 3 results in this paper is that they are reported for a single random seed (42), unlike the Phase 1 results from [Yang L., 2026b] which are reported over 5 seeds with 95% bootstrap CI. Within Phase 1, we have characterized seed-to-seed variability of approximately ±2 percentage points in accuracy for the Coord-Attn architecture. Consequently:

- A 0.2-0.5 percentage-point absolute accuracy difference between any two Phase-2 quantization variants is **smaller than seed-to-seed variability** and should not be over-interpreted.
- The Pruned-50% +1.13% improvement is **comparable to seed-to-seed variability**, so its reproducibility across seeds is uncertain.

We have prioritized Phase-1 multi-seed rigor over Phase-2/Phase-3 multi-seed rigor in this paper because (a) Phase-1 is the primary architectural claim, where reproducibility matters most, and (b) Phase-2/Phase-3 grid searches at five seeds each would have required approximately 35 additional training runs without changing the qualitative conclusions. Future work should re-run the Phase-2 and Phase-3 grids under multi-seed bootstrap.

5.3 Production iOS Application Considerations

Several production-deployment considerations are out of scope for this paper but warrant brief mention:

- **Power consumption.** The 0.064 ms per-inference latency on the Apple Neural Engine implies negligible per-inference battery cost. Inference can be invoked once per second across an 8-hour monitoring session for an estimated 0.0064% of M2 CPU time-equivalent — small enough to not noticeably affect device battery.
- **iOS background execution.** Continuous overnight inference requires iOS background audio capture session permissions; once granted, the inference loop runs without interruption.
- **Failure modes.** CoreML model loading failure, Neural Engine unavailability, and runtime memory pressure are all observable failure modes that production code must handle (typically by falling back to CPU inference at 0.7 ms latency, still well within real-time budget).
- **Cross-platform parity.** An equivalent Android path via TensorFlow Lite + NNAPI + Qualcomm Hexagon DSP exists but is not measured in this paper.

---

6. Conclusion

We have demonstrated that a 1D-Coordinate-Attention sleep apnea classifier can be compressed from 204,801 parameters at 15 ms baseline latency down to 9,416 parameters at 0.064 ms latency on the Apple M2 Neural Engine — a **234× speedup** — while *improving* classification accuracy from 83.59% to 88.49%. The four-phase pipeline (architectural redesign → QAT → structured pruning → CoreML conversion) is well-suited to the small-data, compact-architecture regime

characteristic of clinical biomedical signal processing, where quantization and pruning function as implicit regularizers rather than as accuracy-cost-imposing trade-offs.

The 56.4 KB on-disk model size and 0.064 ms inference latency satisfy the engineering requirements for a smartphone-only overnight sleep apnea screening application running entirely on-device, without cloud connectivity, on consumer Apple hardware.

Artifact Availability

Source code for the compression pipeline (architectural redesign, QAT, L1-structured pruning, and CoreML conversion) is publicly released at: `https://github.com/somnisense/apnea-compression-pipeline` under the MIT License. By design, **no audio recordings, no labels, no derived feature matrices, and no trained model checkpoints are distributed with the repository** — the original recordings were collected under participant consent that does not cover public release of either the waveforms or the derived features. The training, compression, and evaluation scripts run against any dataset that conforms to the 200 × 3 acoustic-feature I/O contract documented in the repository README.

Patent Disclosure

The compression pipeline disclosed in this paper, the CA-1D Stage-2 architecture it operates on, and the cascaded two-stage architecture and system-level procedures that surround it are the subject of **three co-filed U.S. provisional patent applications** by SomniAI LLC:

1. **Cascaded Two-Stage Audio Architecture for Sleep-Disordered Breathing Detection with Ultra-Compact On-Device Feature Representation** — directed to the cascade pipeline structure and the 200 × 3 feature matrix construction. The compressed model produced by the pipeline of this paper is the Stage-2 classifier of that cascade.
2. **Coordinate Attention Block for One-Dimensional Time-Series Classification and Compression Pipeline Comprising Architectural Redesign, Quantization-Aware Training, and Structured Filter Pruning** — directed to the CA-1D mechanism and **the four-phase compression pipeline that is the primary subject of this paper**. The mathematical and procedural details described in §3.2–§3.5 are presented for reproducibility; the patent application covers additional implementation specifics not enumerated here.
3. **Sound-Pressure-Level Multi-Stage Gating, Event-Driven Inference Triggering, and Privacy-Preserving On-Device System Architecture for Audio-Based Sleep Monitoring** — directed to gating, triggering, and the privacy-preserving system architecture surrounding the compressed Stage-2 classifier on consumer mobile devices.

The procedural details described in this paper (Phase 1–4) are presented for reproducibility; certain implementation specifics — particularly the system-level gating and triggering procedures — are covered by the co-filed patent applications and are not described in this manuscript.

References

To be finalized before arXiv submission. Anticipated reference list:

1. Yang L. (2026a). *Audio-Based Snore and Sleep Apnea Detection on Smartphones: Two CNN Baselines with Multi-Seed Validation.* arXiv preprint *(pending submission)*.
2. Yang L. (2026b). *Coordinate Attention for 1D Audio-Based Sleep Apnea Detection: A Multi-Seed Empirical Study on Smartphone-Deployable Architectures.* arXiv preprint *(pending submission)*.
3. Han, S., Mao, H., & Dally, W. J. (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *International Conference on Learning Representations (ICLR)*. arXiv: 1510.00149.
4. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR.2018.00286.
5. Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2019). Rethinking the Value of Network Pruning. *International Conference on Learning Representations (ICLR)*. arXiv: 1810.05270.
6. Frankle, J., & Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *International Conference on Learning Representations (ICLR)*. arXiv: 1803.03635.
7. Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*. URL: https://arxiv.org/abs/1704.04861.
8. Hou, Q., Zhou, D., & Feng, J. (2021). Coordinate Attention for Efficient Mobile Network Design. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. DOI: 10.1109/CVPR46437.2021.01350. arXiv: 2103.02907.
9. Apple Inc. (2024). *CoreML Documentation: Model Conversion and On-Device Inference*. URL: https://developer.apple.com/documentation/coreml.
10. Apple Inc. (2024). *Apple Neural Engine: Architecture and Performance*. URL: https://developer.apple.com/machine-learning.
11. SomniAI LLC. (2026). *Cascaded Two-Stage Audio Architecture for Sleep-Disordered Breathing Detection with Ultra-Compact On-Device Feature Representation*. U.S. Provisional Patent Application.
12. SomniAI LLC. (2026). *Coordinate Attention Block for One-Dimensional Time-Series Classification and Compression Pipeline Comprising Architectural Redesign, Quantization-Aware Training, and Structured Filter Pruning*. U.S. Provisional Patent Application.
13. SomniAI LLC. (2026). *Sound-Pressure-Level Multi-Stage Gating, Event-Driven Inference Triggering, and Privacy-Preserving On-Device System Architecture for Audio-Based Sleep Monitoring*. U.S. Provisional Patent Application.

Appendix A — Reproducibility

All experiments were run on a single Apple M2 MacBook Pro (24 GB RAM, macOS Sonoma). PyTorch 2.x, coremltools 8.x, qnnpack quantization engine. Latency measurement: 200 inference calls, 20 warmup calls, single sample.