DepthPolyp in Practice: Reproducible Deployment and Practical Lessons for Real-Time Colonoscopy Segmentation

Zhuoyu Wu[1], Wenhui Ou[2], Lexi Zhang[4], Pei-Sze Tan[1], Mingze Sun[1], Junhe Zhao[3], Wenqi Fang[3], and Raphaël C.-W. Phan[1]

[1] CyPhi AI Lab, Monash University, Malaysia Campus, Malaysia
[2] Department of Electronic & Computer Engineering, Hong Kong University of Science & Technology, Hong Kong, P.R. China
[3] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, P.R. China
[4] Harbin Institute of Technology, Harbin, P.R. China

Abstract. This paper is a reproducibility companion to the ICPR 2026 submission DepthPolyp, extending the original work from model reproduction to reproducible deployment. We establish a unified deployment workflow covering runtime conversion, deployment-oriented optimization, standardized evaluation, and end-to-end mobile implementation across heterogeneous execution backends. Our deployment study shows that runtime conversion preserves deployment fidelity, while deployment optimization exhibits markedly different behaviour: decoder-only INT8 preserves segmentation quality but provides limited runtime benefit, whereas moderate global pruning achieves the best efficiency–accuracy trade-off. Furthermore, aggressive structural compression introduces a clear deployment phase transition that cannot be fully recovered through lightweight knowledge distillation, highlighting the importance of deployment reliability beyond average benchmark accuracy. Together with the released deployment artifacts, this companion paper provides both a reproducible deployment reference for DepthPolyp and practical deployment guidelines for lightweight medical segmentation systems.

Keywords: Reproducible research · Polyp segmentation · Model deployment · Edge AI · Quantization

1 Introduction

Real-time medical image segmentation is increasingly moving from benchmark evaluation toward practical clinical deployment [10]. Recent lightweight segmentation models have substantially reduced parameter count and computational complexity, making execution on edge devices such as mobile processors and embedded AI accelerators increasingly feasible [9]. However, successful deployment depends on considerably more than network architecture alone. In practice, a model must be converted across heterogeneous runtimes, execute reliably under

different hardware backends, and maintain stable behaviour after deployment-oriented optimizations such as quantization and pruning [3,**?**]. Consequently, the practical performance of a deployed segmentation system is jointly determined by model design, runtime implementation, and deployment strategy rather than by benchmark accuracy alone.

Although reproducibility has become an important aspect of modern computer vision research, existing reproducibility studies primarily focus on reproducing reported metrics through released checkpoints, evaluation scripts, and benchmark protocols [2]. Much less attention has been paid to the reproducibility of deployment behaviour. Common deployment optimizations, including runtime conversion, reduced-precision inference, and model compression, are often assumed to preserve model quality or improve inference speed [3]. In practice, however, these assumptions are highly dependent on runtime implementation and hardware execution [3]. A deployment variant may faithfully reproduce floating-point predictions while providing little acceleration, or achieve substantial compression at the cost of degraded robustness under clinically relevant acquisition conditions [7]. Understanding these trade-offs is therefore an essential part of reproducible deployment.

In this companion paper, we use DepthPolyp [8], a lightweight pseudo-depth guided colonoscopic segmentation model introduced in the accompanying ICPR 2026 submission, as a representative case study. Rather than proposing a new segmentation architecture, this paper documents the complete deployment workflow required to reproduce the trained model across heterogeneous inference backends, including exported PyTorch, ONNX, and CoreML artifacts, standardized evaluation protocols, deployment benchmarks, and representative optimization strategies. Beyond providing reproducible artifacts, we further examine how different deployment choices influence inference fidelity, runtime efficiency, and robustness under practical deployment settings.

Accordingly, this companion paper addresses the following three practical research questions.

1. RQ1. Can a lightweight medical segmentation model be faithfully reproduced as a deployable inference system across heterogeneous runtimes? We investigate numerical consistency and deployment fidelity after exporting the trained model to PyTorch, ONNX, and CoreML execution environments.
2. RQ2. Which deployment optimizations remain reliable after model conversion? We systematically evaluate runtime conversion, reduced-precision inference, pruning, and lightweight recovery strategies under a unified deployment protocol to identify which optimizations provide meaningful practical benefits.
3. RQ3. How should deployment robustness be evaluated beyond conventional average accuracy? We analyze deployment behaviour from both dataset-level and sequence-level perspectives, demonstrating that practical deployment quality cannot always be inferred from pooled accuracy metrics alone.

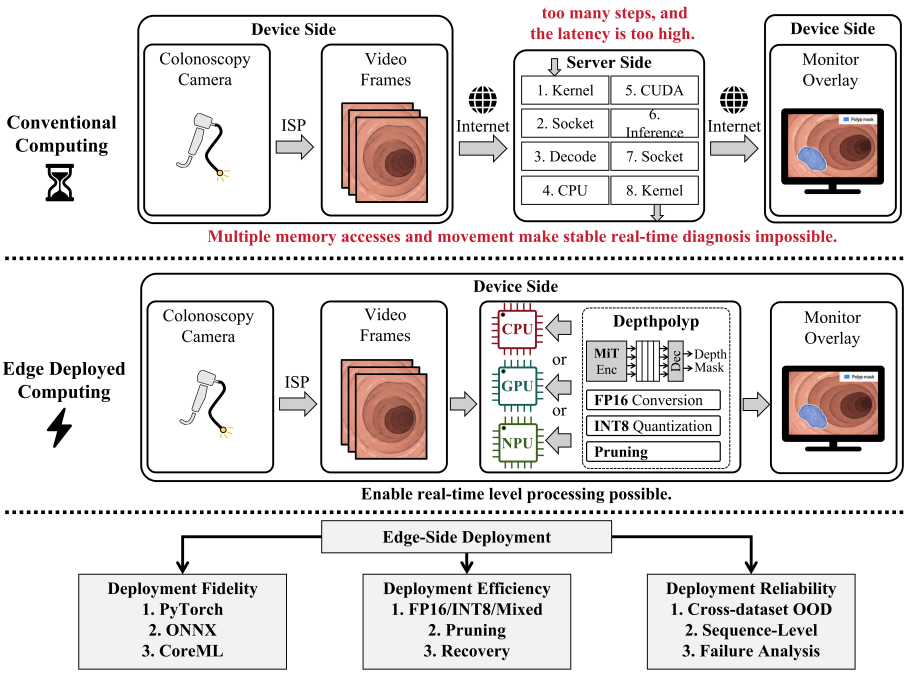

Fig. 1. Reproducible deployment framework considered in this companion paper. The upper pipeline represents conventional server-side inference, while the lower pipeline illustrates the target edge-side deployment studied in this work. Unlike the accompanying main paper, this companion focuses on reproducing the complete deployment workflow from image acquisition to real-time visualization across heterogeneous execution backends.

The answers to these questions establish a reproducible deployment reference for DepthPolyp while providing practical guidelines for deploying lightweight medical image segmentation models in real-world edge-assisted clinical systems.

2 Reproducible Deployment Framework

This companion paper complements the accompanying main paper by investigating reproducible deployment rather than segmentation model design. Instead of proposing a new network, our objective is to reproduce the trained model as a complete edge-side inference system across heterogeneous execution environments.

Figure 1 presents the deployment framework studied in this work. Unlike conventional server-side inference, the target workflow performs segmentation directly on the device after image signal processing, eliminating repeated communication and enabling low-latency real-time assistance. The same trained model

Table 1. Unified experimental configuration adopted throughout the deployment study. All deployment variants are evaluated under identical settings. Only the deployment strategy varies.

Category	Configuration
Model	Same trained DepthPolyp checkpoint
Input	Input resolution 224×224, identical pre-/post-processing
Evaluation	Dice metric, threshold = 0.5, identical evaluation scripts
Benchmark	Batch size = 1, same runtime benchmark protocol, warm-up excluded
Deployment	PyTorch, ONNX Runtime, and CoreML
Variable	Deployment strategy (runtime conversion, precision reduction, model compression, and recovery)

is reproduced across PyTorch, ONNX Runtime [6], and CoreML [1] while maintaining an identical inference pipeline and evaluation protocol.

As summarized in Fig. 1, the deployment study is organized around three complementary aspects. *Deployment fidelity* evaluates whether the converted model faithfully reproduces the original implementation across different runtimes. *Deployment efficiency* investigates deployment-oriented optimizations, including precision reduction, model compression, and lightweight recovery strategies. Finally, *Deployment reliability* examines whether deployment preserves practical robustness under cross-dataset evaluation, sequence-level analysis, and representative failure cases. These three aspects define the scope of the experimental study presented in the remainder of this paper.

3 Unified Evaluation Protocol

3.1 Deployment Variants

Starting from the same trained checkpoint, we investigate representative deployment strategies commonly adopted for lightweight edge inference. The evaluated variants include runtime conversion across PyTorch, ONNX Runtime, and CoreML, reduced-precision inference (FP16 and multiple INT8 configurations) [5], model compression through global, encoder, decoder, and structured pruning [2], and lightweight recovery using short knowledge distillation [4]. Together, these variants define the deployment design space investigated in this companion paper.

3.2 Unified Experimental Configuration

Table 1 summarizes the unified experimental configuration adopted throughout this study. All deployment variants share the same trained checkpoint, inference

Table 2. Representative deployment strategies evaluated under the unified protocol. Average Dice is computed over Kvasir-SEG, CVC-ClinicDB, CVC-ColonDB, and PolypGen. Δ Dice and Δ FPS are measured relative to the FP32 ONNX reference.

Strategy	Class	Avg Dice↑	Δ Dice	PolypGen↑	ONNX FPS↑	Key finding
FP32	Reference	0.7570	+0.0000	0.5574	204.5	Reference
Decoder INT8	Precision	0.7574	+0.0004	0.5578	182.0	Fidelity preserved; no speedup
Hybrid INT8	Precision	0.1860	-0.5710	0.1314	131.8	Unreliable conversion
Full INT8	Precision	0.1844	-0.5725	0.0978	101.2	Deployment failure
Global p10	Compression	0.7568	-0.0002	0.5565	236.2	Safe compression
Global p30	Compression	0.7531	-0.0039	0.5663	240.0	Best trade-off
Global p50	Compression	0.6287	-0.1282	0.4951	223.9	Accuracy loss begins
Decoder p10	Compression	0.7485	-0.0085	0.5638	231.9	Stable alternative
Structured p30	Compression	0.0579	-0.6991	0.2315	236.8	Phase transition
Structured p30 + KD	Recovery	0.6969	-0.0601	0.4449	236.8	Partial recovery

pipeline, evaluation scripts, and benchmark settings, ensuring that the deployment strategy remains the only independent variable.

Unless otherwise specified, all evaluations are performed using batch size one to match the target real-time deployment scenario. Runtime throughput is measured after warm-up using the same benchmarking procedure, while segmentation accuracy is computed using identical prediction and evaluation pipelines across all deployment backends.

4 Deployment Study

Table 2 summarizes the representative deployment strategies evaluated under the unified protocol. Instead of treating segmentation accuracy and inference speed as separate outcomes, this study analyzes deployment from three complementary perspectives: *fidelity*, *efficiency*, and *reliability*. Fidelity asks whether a converted model preserves the behaviour of the reference implementation. Efficiency asks whether deployment-oriented optimization actually improves practical throughput. Reliability asks whether the deployed model preserves robust behaviour across datasets and video sequences.

4.1 Deployment Fidelity

The first requirement of reproducible deployment is preserving inference fidelity after runtime conversion. Before evaluating deployment-oriented optimization, the converted model should faithfully reproduce the behaviour of the original implementation under an identical inference pipeline.

The FP32 ONNX deployment establishes this reference. Its average Dice is 0.7570, matching the PyTorch reference reported in the released evaluation files with negligible deviation. This verifies that ONNX export is a faithful deployment artifact rather than only an alternative execution backend. This result is important because all subsequent optimization experiments are interpreted as changes in deployment strategy rather than errors introduced by conversion.

Precision reduction does not behave uniformly. Decoder-only INT8 preserves segmentation behaviour, producing nearly identical average Dice to FP32 (0.7574

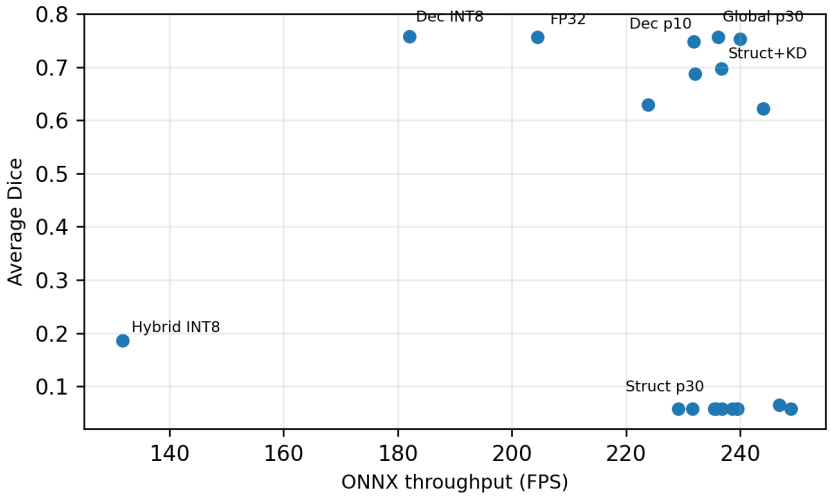

Fig. 2. Accuracy–throughput behaviour of representative deployment variants. Global p30 provides the strongest practical operating point, improving throughput while preserving average Dice. Decoder-only INT8 preserves accuracy but does not improve ONNX throughput, while aggressive quantization and structured compression fall outside the useful deployment region.

vs. 0.7570) and essentially unchanged PolypGen performance (0.5578 vs. 0.5574). In contrast, full INT8 and hybrid INT8 collapse to average Dice values of 0.1844 and 0.1860, respectively. Thus, quantization safety is strongly scope-dependent: limiting INT8 to the decoder preserves behaviour, whereas broader quantized conversion is unreliable for this model.

4.2 Deployment Efficiency

Deployment efficiency should be measured after runtime execution rather than inferred from numerical precision or sparsity alone. Fig. 2 shows that reduced precision and pruning lead to substantially different practical behaviours under the same ONNX Runtime setting.

Decoder-only INT8 is behaviour-preserving but not acceleration-effective. Although its segmentation quality is almost unchanged relative to FP32, its ONNX throughput decreases from 204.5 FPS to 182.0 FPS. This negative result is deployment-relevant: reducing numerical precision does not guarantee faster inference unless the target runtime and hardware execute the quantized graph efficiently.

Moderate global pruning provides the best operating point. Global p30 reduces average Dice by only 0.0039, while increasing throughput from 204.5 FPS to 240.0 FPS and improving PolypGen Dice from 0.5574 to 0.5663. Global p10 is also safe, but offers a smaller deployment benefit. Global p50 begins to lose sub-

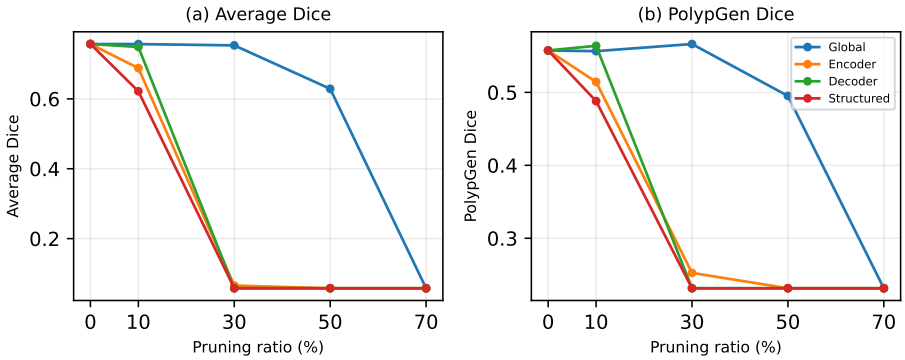

Fig. 3. Compression tolerance and deployment phase transition. Global pruning has a usable sparsity window around 10–30%, whereas encoder, decoder, and structured pruning exhibit an abrupt collapse once the pruning ratio reaches 30%. This indicates that deployment compression is not a smooth accuracy–efficiency trade-off.

stantial accuracy, indicating that useful sparsity exists within a limited operating window rather than increasing monotonically with the pruning ratio.

The pruning curves in Fig. 3 reveal a sharper phenomenon than a conventional compression trade-off. Global pruning degrades gradually and reaches its best deployment point at 30%, while encoder, decoder, and structured pruning undergo an abrupt phase transition around the same sparsity level. For example, structured p30 collapses to an average Dice of 0.0579 despite retaining high throughput. Short knowledge distillation recovers the damaged structured model to 0.6969 average Dice, but this remains far below the simpler global p30 result. Knowledge distillation therefore repairs part of the optimization damage, but does not restore the original deployment behaviour.

Overall, the efficiency study supports a practical rule: deployment optimization should prioritize moderate global compression over aggressive structural simplification or broad INT8 conversion. The best deployable variant is not the numerically smallest or most aggressively compressed model, but the one that preserves behaviour after conversion while improving measured throughput.

4.3 Deployment Reliability

Average benchmark accuracy alone is insufficient for evaluating deployment quality. In practical clinical deployment, a model should maintain consistent behaviour across heterogeneous acquisition conditions rather than achieving a high pooled score by succeeding on only a subset of cases. We therefore analyze PolypGen at sequence level, as shown in Fig. 4.

Decoder-only INT8 provides the clearest example of behaviour preservation. Despite its limited throughput benefit, its sequence-level Dice changes remain close to zero across nearly all PolypGen sequences. This confirms that decoder-

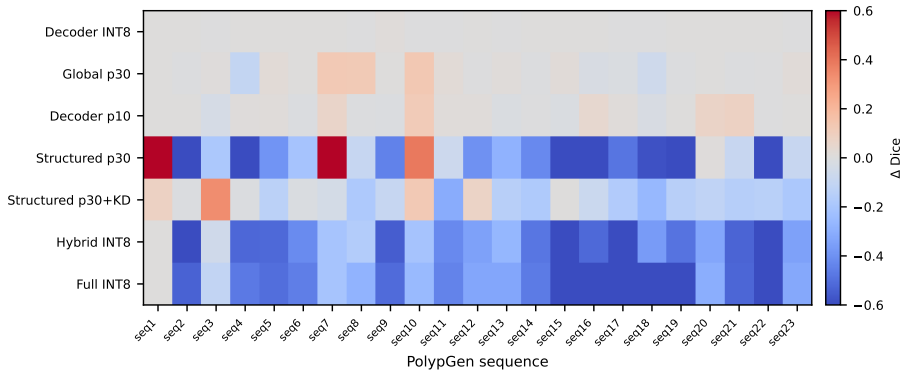

Fig. 4. Deployment reliability analysis using sequence-level PolypGen evaluation. Each cell reports the Dice change relative to the FP32 reference for a deployment variant on a PolypGen sequence. Decoder INT8 is behaviour-preserving, global p30 largely preserves the original failure boundary with local gains, while structured compression and failed INT8 conversion substantially redistribute or destroy sequence-level behaviour.

only quantization is a safe conversion strategy with respect to deployment behaviour, even though it is not a useful acceleration strategy under our ONNX Runtime setting.

Global p30 shows a different but still reliable pattern. Its pooled PolypGen Dice improves, and the heatmap indicates that most sequence-level changes remain moderate. Several sequences improve substantially, while only a small number show clear degradation. This suggests that moderate sparsity redistributes performance locally but does not fundamentally change the model's failure boundary.

Aggressive structural compression behaves differently. Structured p30 and structured p30 with KD exhibit large sequence-level shifts, indicating that the model no longer fails on the same cases as the reference deployment. Distillation improves average performance after the collapse, but the recovered model still follows a different sequence-level failure pattern. The failed INT8 variants are more severe: they collapse on most sequences, showing that successful graph execution is not equivalent to a deployable segmentation system.

These results demonstrate that deployment reliability should be evaluated beyond pooled Dice. Two deployment strategies can have similar throughput, or even partially recovered average accuracy, while exhibiting very different sequence-level failure distributions. For real-time colonoscopy assistance, such consistency is a practical deployment requirement rather than an optional diagnostic analysis.

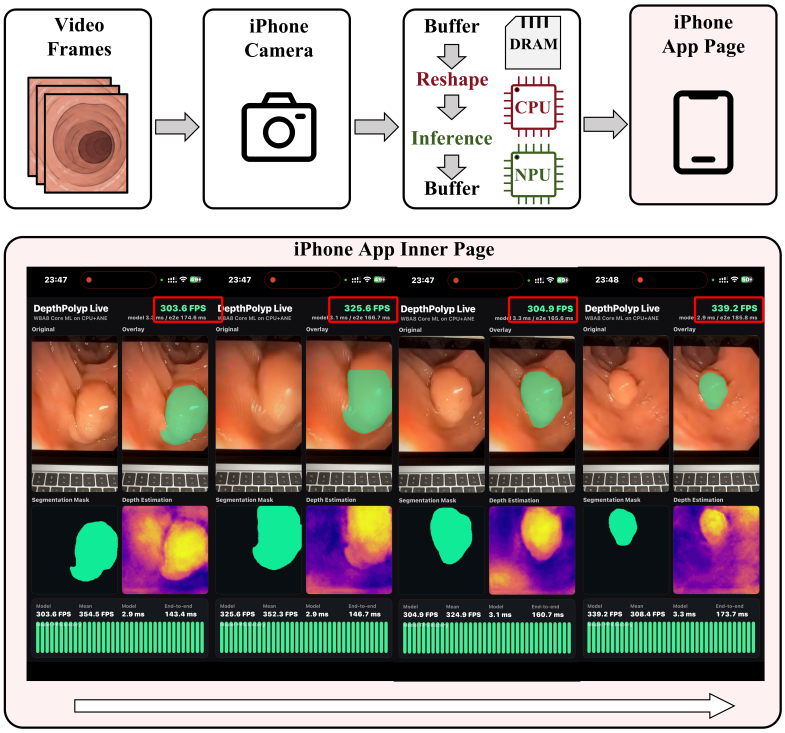

Fig. 5. End-to-end deployment of DepthPolyp on iPhone. The exported CoreML model is integrated into a standalone iOS application for fully on-device inference. Input video frames are captured by the iPhone camera, reshaped and buffered before inference, and processed locally through the CoreML runtime using the Apple Neural Engine (ANE). Representative runtime snapshots demonstrate consistent real-time segmentation under different scenes without server communication. The application achieves stable on-device inference at approximately 300–340 FPS while simultaneously visualizing the original image, segmentation overlay, predicted mask, depth estimation, and runtime statistics.

5 End-to-End Mobile Deployment

To validate the complete deployment workflow beyond offline benchmarking, we implemented a standalone iOS application using the exported CoreML model. The application performs fully on-device inference by capturing video frames directly from the iPhone camera, executing segmentation locally through the Apple Neural Engine (ANE), and rendering the predicted masks in real time without any server communication. The complete execution pipeline follows the edge-side deployment framework introduced in Fig. 1.

Figure 5 presents representative runtime snapshots captured from the deployed application. In addition to real-time segmentation overlays, the interface simultaneously visualizes the predicted segmentation mask, pseudo-depth esti-

mation, instantaneous throughput, and end-to-end latency, providing an intuitive view of deployment behaviour during practical execution. Across different representative scenes, the application consistently achieves approximately 300–340 FPS on the evaluated iPhone platform, demonstrating that the deployment workflow investigated throughout this companion paper can be directly transferred to practical mobile execution.

6 Practical Deployment Guidelines

Guideline 1: Validate deployment on the target runtime rather than relying on numerical precision alone. Reduced numerical precision does not necessarily translate into higher deployment efficiency. The practical benefit of INT8 inference depends strongly on the underlying execution backend and operator support. While dedicated inference hardware such as NPUs or optimized CPU backends may directly accelerate integer computation, GPU runtimes often execute quantized models through mixed floating-point and integer operators or repeatedly insert quantize–dequantize conversions for unsupported operations. These implementation details can offset the theoretical advantage of INT8 arithmetic and even reduce practical throughput. Consequently, deployment performance should always be validated on the intended hardware and runtime rather than inferred solely from model precision.

Guideline 2: Prefer moderate compression over aggressive structural simplification. Moderate sparsity consistently provides a better deployment trade-off than aggressive structural compression. Once compression exceeds an appropriate operating region, segmentation quality degrades abruptly rather than gradually, while lightweight recovery cannot fully restore the original deployment behaviour. For practical deployment, conservative compression is therefore preferable to maximizing sparsity.

Guideline 3: Evaluate deployment reliability beyond average accuracy. Average benchmark performance alone is insufficient for assessing deployment quality. Sequence-level behaviour and cross-dataset consistency provide complementary information about deployment robustness and failure characteristics. These analyses are particularly important for real-world medical deployment, where stable behaviour is often more valuable than marginal improvements in pooled benchmark scores.

7 Conclusion

This companion paper extends the accompanying DepthPolyp work from model reproduction to reproducible deployment. We establish a complete deployment workflow covering runtime conversion, deployment-oriented optimization, unified evaluation, and end-to-end mobile implementation, providing a practical reference for reproducing lightweight medical segmentation systems across heterogeneous execution backends.

Our deployment study shows that deployment performance is determined by considerably more than model architecture. Runtime conversion should be validated on the target hardware rather than inferred from numerical precision alone; moderate compression provides a substantially better deployment operating point than aggressive structural simplification; and deployment reliability should be assessed through consistent behaviour across deployment scenarios instead of average benchmark accuracy alone.

Overall, this work demonstrates that reproducible deployment is a system-level problem rather than merely a model conversion task. Beyond serving as a companion for DepthPolyp, we hope the resulting deployment workflow and practical deployment principles provide a useful reference for future lightweight medical vision systems targeting real-world edge-assisted clinical applications.

References

1. Apple: Core ml. https://developer.apple.com/documentation/coreml (2024)
2. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635 (2018)
3. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. Advances in neural information processing systems **28** (2015)
4. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
5. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., Kalenichenko, D.: Quantization and training of neural networks for efficient integer-arithmetic-only inference. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2704–2713 (2018)
6. Microsoft: Onnx runtime. https://onnxruntime.ai (2024)
7. Wu, Z., Ou, W., Tan, P.S., Yang, J., Fang, W., Wang, Z., Phan, R.C.W.: Endocaver: Handling fog, blur and glare in endoscopic images via joint deblurring segmentation. In: ICASSP'26: IEEE International Conference on Acoustics, Speech and Signal Processing 2026 (2026)
8. Wu, Z., Ou, W., Zhang, L., Tan, P.S., Wu, D., Zhao, J., Fang, W., Phan, R.C.W.: Depthpolyp: Pseudo-depth guided lightweight segmentation for real-time colonoscopy. arXiv preprint arXiv:2605.16519 (2026)
9. Wu, Z., Ou, W., Zheng, Q., Wu, Q., Gao, D., Fang, W., Chen, C., Yang, Y., Wang, Z.: Edge-segstar: When bidirectional adaptive self-distillation meets biomedical segmentation on edge devices. Biomedical Signal Processing and Control **125**, 110751 (2026)
10. Wu, Z., Wu, Q., Fang, W., Ou, W., Wang, Q., Zhang, L., Chen, C., Wang, Z., Li, H.: Harmonizing unets: Attention fusion module in cascaded-unets for low-quality oct image fluid segmentation. Computers in Biology and Medicine **183**, 109223 (2024)