This MICCAI paper is the Open Access version, provided by the MICCAI Society. It is identical to the accepted version, except for the format and this watermark; the final published version is available on SpringerLink.

Supplementary Material:

An Engineering-optimized, Resource-Constrained nnU-Net v2 Pediatric Baseline for the BraTS-GoAT 2026 Generalizability-Across-Tumors Challenge

Appendix A: Extended Training-Set and Boundary-Distance Evaluations

To further characterize this training/validation relationship, we additionally report training-set Dice, computed separately from the validation metrics in Table 1. Using the Fold 3 model, we ran inference on a 23-case subset of Fold 3's training split (cases 500–550, excluding the 7 cases in that range that fall within Fold 3's validation set per `splits_final.json`) and computed Dice against ground truth for each case. Table 1 reports the results: an overall mean training Dice of 0.888 (NCR 0.849, ED 0.920, ET 0.897). Training performance exceeds the corresponding Fold 3 validation performance (0.824, Table 1) across all three regions, consistent with a normal generalization gap rather than overfitting, and in line with the loss behavior shown in the main text (Fig. 2) — the subset-level Dice gap is of a similarly modest magnitude.

Case-level variability, however, was more pronounced than the fold-level means suggest. Per-case standard deviations, shown in Table 1, were non-trivial relative to the fold means, particularly for NCR (SD 0.276–0.300 against a mean of 0.758–0.795) — a pattern consistent with the small-object Dice sensitivity discussed in Sect. 5.1.

Table 1. Training-set Dice, computed on a 23-case subset of Fold 3's training split (cases not included in Fold 3's 270-case validation set), evaluated with the Fold 3 model (single fold, no Test-Time-Augmentation (TTA)).

Metric	NCR	ED	ET	Overall Mean
Dice	0.849	0.920	0.897	0.888

As a complement to the Dice-based results in Sect. 4.1, we additionally report boundary-distance metrics for Fold 3 (Table 2), computed via MONAI-based 3D surface analysis. We present this as a representative single-fold evaluation rather than a 5-fold aggregate, since surface-distance extraction was performed for this fold only; extending it to the remaining folds is left for future work. Within this fold, ET shows the tightest boundary adherence (median HD95 = 1.00 mm, NSD = 0.9582), while ED shows the widest (median HD95 = 2.00 mm, NSD = 0.8992). Notably, NCR's HD95 (1.57 mm) and NSD (0.9331) are intermediate rather than worst, despite NCR having the lowest Dice of the three regions. This supports the volumetric-bias interpretation in Sect. 5.1: Dice penalizes small structures disproportionately for a given boundary error, whereas HD95/NSD are comparatively less sensitive to region size.

Table 2. Boundary-distance metrics for Fold 3 validation (n=270 validation cases), computed via MONAI-based 3D surface analysis. Reported as a single-fold evaluation rather than a 5-fold aggregate, as surface-distance metrics were not extracted for the remaining folds.

Metric	Overall	NCR	ED	ET
Median HD95 (mm)	1.52	1.57	2.00	1.00
Median NSD	0.9302	0.9331	0.8992	0.9582

Appendix B: Critical Engineering Optimizations & Hardware Validation

Critical Engineering Optimizations for Constrained HPC Deployment

Two problems had to be solved before any fold trained to completion. Both are worth setting out plainly rather than compressing into a single sentence, since diagnosing them cost us considerable time.

The first was memory; nnU-Net's 3D spatial augmentation (elastic deformation, rotation, and scaling at full resolution) is distributed across several parallel worker processes so that it does not bottleneck training. At the default level of parallelism, our fold-0 runs crashed repeatedly with ArrayMemoryError, a direct consequence of shared, multi-tenant memory limits that were never tuned for a single job monopolizing that much RAM. We reduced the number of augmentation workers to two (`nnUNet_n_proc_DA=2`). Although, data loading slowed slightly, however, the crashes stopped. Once that was clear, we applied the setting uniformly across all five folds.

The second was PyTorch's dynamic compilation through TorchDynamo, which recent nnU-Net releases enable by default to accelerate training. On our cluster's particular combination of driver and compiler versions, it introduced recompilation-related instability partway through multi-day runs. That is precisely the failure you cannot afford three days into a job with a 72-hour ceiling, since a crash at that point forces the entire fold to restart. We disabled it outright (`TORCHDYNAMO_DISABLE=1, nnUNet_compile=0`) and accepted the loss in throughput in exchange for runs that finished. Given our deadline, the trade was worth making, and it eliminated the instability entirely.

Neither change touches the architecture, loss, or optimization schedule described in Sect. 3.3. Both are configuration decisions, fully reversible on infrastructure without the same constraints, which means the results in Sect. 4 reflect the standard nnU-Net v2 baseline rather than a customized variant.

Hardware Validation of Engineering Optimizations: To directly evidence the necessity of the changes described above, we ran the default (unmodified) nnU-Net v2 data-loading configuration (12 parallel augmentation workers, dynamic compilation enabled) on an identical node profile (NVIDIA A10G, 24GB VRAM; 32GB system RAM; 12 CPU cores). This run failed with a fatal out-of-memory (OOM) condition during data augmentation: system RAM saturated at 32GB (100%) before any data reached the GPU, and background worker processes were terminated (RuntimeError: One or more background workers are no longer alive). Peak RAM consumption was not instrumented during the full-scale production training runs; however, capping augmentation workers at two (`nnUNet_n_proc_DA=2`) eliminated this OOM failure entirely, and per-epoch training time remained stable at approximately 220.8 s across the full 1000-epoch schedule for Fold 4 (main text, Fig. 2, middle panel), with no throttling or degradation over the course of training. This confirms the worker-count reduction was necessary rather than incidental for successful training on this hardware profile.

Table 3. End-to-end deployment pipeline verification status at the time of writing.

Pipeline Stage	Status	Verification
Trained model (5-fold CV)	Complete	Mean Dice 0.831 across folds
Docker image build	Verified	Builds cleanly; runs under --network none
Local inference test	Verified	4 cases, avg. 7.5 s/volume (range 3–13 s)
Push to Synapse	Verified	Digest confirmed post-push
Pull + conversion on cluster	Verified	Successful on evaluation cluster
GPU inference on cluster	Verified	Correct device detection; correct output

Pipeline Stage	Status	Verification
5-fold ensemble + containerization	Verified	Successfully ensembled with TTA and packaged for deployment

Appendix C: Extended Background & Related Work

BraTS and the Generalizability Problem

Most of the methodological progress in brain tumor segmentation over the last decade was built and validated on a single, relatively uniform population: adult glioma [6,5]. As BraTS broadened to include meningioma, metastases, Sub-Saharan African glioma [18], and pediatric tumors, a consistent finding emerged across the participating teams' results, strong adult-glioma performance did not transfer cleanly to the other populations [10]. Differences in lesion morphology, which sub-regions are even present, scanner protocols, and patient demographics all played a role. BraTS-GoAT turns that finding into the object of study, rather than a footnote in someone else's results section.

The failure to generalize across diverse demographics limits the clinical utility of these models, particularly in settings with constrained access to specialist neuro-radiological review. That is part of why we chose to focus this baseline specifically on the pediatric arm.

nnU-Net and the Rest of the 3D Segmentation Landscape

nnU-Net's contribution to the field was less about a specific architecture and more about removing the guesswork from 3D medical segmentation: preprocessing, normalization, patch and batch sizing, network depth, and training schedule are all derived automatically from a dataset's own statistics rather than hand-tuned per task [13]. It has held up as a strong baseline across a wide range of segmentation benchmarks, BraTS included, which is exactly why groups tend to build from it rather than around it. Newer architectural directions, including recent 3D modeling approaches published as late as this year [19], continue to explore where the encoder-decoder paradigm can be improved, but nnU-Net remains the reference point those comparisons are made against.

We treat it the same way here: a validated nnU-Net baseline is the right starting point before layering on anything more elaborate. We treat this as a methodological choice rather than a limitation of ambition: measuring what the self-configuring framework achieves without modification is the necessary reference point against which any later architectural change for instance, one targeting long-range spatial context can be meaningfully evaluated. Without this control, gains from a more elaborate architecture cannot be distinguished from gains attributable simply to more careful configuration or engineering.

Hardware Constraints and Training Stability in 3D Segmentation

There is a part of the segmentation literature that almost never gets written down: what actually goes wrong when you try to run a full 3D pipeline on infrastructure that was not built specifically for it. Full-resolution 3D augmentation, elastic deformation, rotation, scaling, is heavy, and it is usually parallelized across worker processes, so it does not bottleneck training. On a shared cluster, where memory and process limits are tuned for many small jobs rather than one large one, that parallelism can simply run out of memory in ways that never show up on a dedicated GPU box. PyTorch's dynamic compilation, similarly, can behave differently depending on the exact driver and compiler stack underneath it, which is difficult to guarantee on infrastructure you do not fully control. We are not aware of many BraTS short papers that walk through this kind of failure explicitly, which is part of why we do so in Sect. 3.

From Benchmark to Bedside: AI Translation in Pediatric Oncology

A recurring theme in reviews of AI applications in pediatric oncology is that the gap between a promising benchmark result and an actually deployed, clinically usable tool remains wide, and wider in pediatrics than in adult oncology, where more data and more commercial incentive exist [3]. Reproducibility and independent verification of a model's

full pipeline, not just its headline metric, is increasingly treated as part of what it means to take that gap seriously; federated and containerized benchmarking efforts for medical AI, for instance, exist specifically because a Dice score alone says very little about whether a model will actually run correctly somewhere else [4]. That is the spirit in which we treat containerization and deployment verification in this paper as a first-class part of the contribution, not an afterthought reported in a single sentence.

This generalization gap has direct clinical consequences. Accurate quantification of tumor sub-regions—enhancing tumor (ET), peritumoral edema (ED), and necrotic core (NCR)—is critical for treatment planning and response assessment in pediatric neuro-oncology. Tumor volume metrics inform radiotherapy planning, define clinical target volumes accounting for potentially infiltrated peritumoral tissue, and guide biopsy site selection. A model trained and validated almost exclusively on adult glioma, when applied to pediatric scans, is not guaranteed to segment these regions correctly, and in practice frequently does not [1].

Appendix D: Expanded Methodological Details

The baseline is nnU-Net v2 in its 3D full-resolution configuration [13], implemented in PyTorch [20]. It follows the standard nnU-Net encoder-decoder design with skip connections, instance normalization, and leaky ReLU; patch size, pooling depth, and kernel configuration are all derived from the dataset fingerprint rather than chosen by hand. Training uses nnU-Net's default compound loss (Dice plus cross-entropy) under deep supervision at multiple decoder resolutions, optimized with stochastic gradient descent, Nesterov momentum, and polynomial learning-rate decay. We deviated from none of these defaults. The purpose of this baseline is to establish what the framework achieves on its own, before anything task-specific is added.