BraTS Brain Tumor Segmentation with Transformer in U-Net

Yihang Jiao
University of Illinois Urbana-Champaign
610 East John Street, Champaign, IL 61820
yihangj3@illinois.edu

Abstract

In this project, I implement a unified and reproducible 2D BraTS-style pipeline that supports three model families under a single entrypoint: (i) a classic U-Net baseline, (ii) a configurable "flex" U-Net with controllable width, depth, normalization, and residual blocks, and (iii) a TransUNet variant with a ViT-based encoder [3] and convolutional decoder. The pipeline includes automatic cropping, slice indexing to skip empty slices, class-weighted cross-entropy, foreground-only Dice loss with warmup, optional mixed precision, and checkpoints that store both weights and full configuration metadata.

Experiments show that TransUNet achieves the best overall foreground Dice in my setup, but a carefully tuned, smaller U-Net remains competitive and easier to train. Overall, the results suggest that preprocessing, sampling, and loss design are at least as important as the choice of backbone for BraTS-style brain tumor segmentation. Accurate brain tumor segmentation from multi-modal MRI is important for clinical workflows such as surgical planning and treatment monitoring. U-Net style convolutional models remain strong baselines on BraTS-style data [1], while hybrid CNN–Transformer architectures such as TransUNet [2] promise better global context modeling. In many codebases, however, architecture changes are entangled with preprocessing, cropping, and loss design, making it hard to tell where the gains actually come from.

In this project, I implement a unified and reproducible 2D BraTS-style pipeline that supports three model families under a single entrypoint: (i) a classic U-Net baseline, (ii) a configurable "flex" U-Net with controllable width, depth, normalization, and residual blocks, and (iii) a TransUNet variant with a ViT-based encoder [3] and convolutional decoder. The pipeline includes automatic cropping, slice indexing to skip empty slices, class-weighted cross-entropy, foreground-only Dice loss with warmup, optional mixed precision, and checkpoints that store both weights and full configuration metadata.

Experiments show that TransUNet achieves the best overall foreground Dice in my setup, but a carefully tuned, smaller U-Net remains competitive and easier to train. Overall, the results suggest that preprocessing, sampling, and loss design are at least as important as the choice of backbone for BraTS-style brain tumor segmentation.

1. Introduction

Brain tumor segmentation from magnetic resonance imaging (MRI) is a long-standing and high-impact problem in medical image analysis. Reliable delineation of tumor subregions supports downstream decisions such as preoperative localization, radiotherapy planning, and longitudinal monitoring. Manual voxel-wise annotation is expensive and subject to inter-rater variability, so robust automated segmentation methods are highly desirable. Public benchmarks such as the Brain Tumor Segmentation (BraTS) challenges [5, 6] have standardized evaluation on multimodal MRI (T1, T1-CE, T2, FLAIR) and clinically meaningful tumor labels, and they remain a common testbed for new architectures and training strategies.

Convolutional encoder–decoder models with skip connections, most notably U-Net [1], have become the default choice for medical image segmentation due to their strong inductive bias for locality and multi-scale feature fusion. More recently, transformer-based vision models have demonstrated improved global context modeling, which is potentially beneficial for lesions that are spatially sparse yet semantically connected across larger anatomical regions. TransUNet [2] is a representative hybrid approach that integrates a Vision Transformer (ViT) [3] encoder with a U-Net-like decoder, aiming to combine global reasoning with high-resolution spatial decoding.

In practice, it is often difficult to perform fair and controlled comparisons between model families. Seemingly minor implementation choices—how slices are sampled, whether empty-background slices are included, how intensities are normalized, how labels are remapped, and whether cropping is applied—can change training dynamics and reported metrics substantially. Likewise, loss design (class

weighting, Dice-based losses, warmup schedules) can dominate performance differences if not held constant. As a result, improvements are sometimes attributed to the backbone architecture when they may actually come from preprocessing or optimization details.

To address this, I design a unified BraTS-style 2D segmentation framework with the explicit goal of isolating architectural effects under consistent data and training settings. Concretely, I implement a single training entrypoint (`src/unet_v3.py`) that can instantiate and train: (i) a classic U-Net baseline with BatchNorm double-convolution blocks, (ii) a configurable "flex" U-Net with tunable base width, depth, normalization (BatchNorm/GroupNorm/InstanceNorm/LayerNorm/none), optional residual blocks, and dropout, and (iii) a TransUNet variant that supports hybrid ResNet50 [4] + ViT backbones with a convolutional decoder.

The data pipeline includes (a) an automatic crop optimizer computed from non-zero tissue regions to reduce wasted compute on background, (b) slice indexing that filters out empty slices and stores a foreground flag, and (c) optional resizing and volume caching to trade off runtime and memory. For optimization, I use class-weighted cross-entropy to handle class imbalance and add a foreground-only Dice loss term after a short warmup period for stability, following common practice in medical segmentation [7].

A key design choice is reproducibility: each checkpoint stores not only model weights, but also the model configuration (architecture knobs) and crop coordinates used during training, enabling evaluation scripts to reconstruct the exact model without manual bookkeeping. With this controlled setup, I can directly compare whether a ViT-based encoder provides gains over a convolutional baseline, and I can run systematic ablations over U-Net depth and normalization while keeping the rest of the pipeline fixed.

In summary, this project contributes a clean, modular baseline environment for brain MRI tumor segmentation that supports fair comparisons across U-Net and TransUNet families. Beyond producing competitive results, the main value is enabling transparent debugging and experimentation: when performance changes, I can trace the change to a specific knob in the model, loss, or data pipeline rather than a bundle of confounded modifications.

2. Approach

2.1. Dataset

BraTS-style multi-modal brain MRI. Each subject contains 4 MRI modalities (`flair`, `t1`, `t1ce`, `t2`) and a tumor segmentation mask. I treat every subject as a 3D volume, and I train my models in a **2D slice-based** setup. For one axial slice, the input and label are:

$$x \in \mathbb{R}^{4\times H\times W}, \qquad y \in \{0,1,2,3\}^{H\times W}.$$

The original BraTS label "4" is remapped to "3" in my loader so the final number of classes stays $C = 4$.

Slice-level dataset construction. I first filter out invalid subjects (must have all 4 modalities + `seg`). Then for each subject I enumerate slice index z and keep only slices that contain non-zero brain signal (quick check using `flair`). At indexing time I also store a foreground flag:

$$\texttt{has\_fg}(z) = \mathbb{I}\big[\exists(i,j)\ y_{ij}(z) > 0\big],$$

which I later use for foreground oversampling during training.

Global crop for speed (computed once). To reduce empty background and speed up training, I compute a single crop box from non-zero voxels across subjects/modalities:

$$(x_{\min}, x_{\max}, y_{\min}, y_{\max}).$$

If a stable crop cannot be found (e.g., all-zero or degenerate bounds), cropping is disabled automatically.

Per-slice normalization and optional non-linear intensity processing. For each modality slice s, I apply per-slice z-score normalization:

$$\hat{s} = \frac{s - \mu(s)}{\sigma(s) + \epsilon}.$$

Optionally, I apply a non-linear intensity mapping (`Img_proc`) that combines log/exponential style transforms and a final normalization to $[0, 1]$. I used this mainly to test whether the model benefits from stronger contrast normalization.

Input resizing (only when needed). For models requiring fixed input size (TransUNet), I resize each sample to a target resolution $S \times S$:

$$x \leftarrow \text{Resize}_{\text{bilinear}}(x), \qquad y \leftarrow \text{Resize}_{\text{nearest}}(y),$$

where nearest-neighbor is required for label masks.

Data split. I use a fixed random seed and split the slice dataset into train/validation by a fixed validation ratio (default 10%), so different model variants are compared under the same splitting rule.

2.2. Model architectures

2.2.1. Original U-Net baseline

I keep a standard U-Net [1] as a strong CNN baseline. It takes a 4-channel slice and predicts 4-class logits per pixel.

```
Input (4×H×W)
  ↓
Encoder: 4 down stages (ConvBlocks
+ MaxPool)
  ↓
Bottleneck
  ↓
Decoder: 4 up stages (Upsample
+ skip concat
+ ConvBlocks)
  ↓
Output: 1×1 Conv → (4×H×W) logits
```

This baseline helps me verify that preprocessing / loss changes are real improvements, not only architecture tricks.

2.2.2. Flexible U-Net (UNetFlexible)

I implemented a configurable U-Net where I can control:

- **Depth**: number of down/up stages.
- **Base channels**: width multiplier at the first stage.
- **Normalization**: BatchNorm / GroupNorm / InstanceNorm / LN-trick (GN(1)) / none.
- **Upsampling**: bilinear upsampling vs transposed convolution.
- **Residual ConvBlocks**: optional residual connection inside ConvBlock (only when channels match).

In most runs I used **GroupNorm** since batch size can be effectively small (especially for heavy models or when using accumulation). This model is my main sandbox for testing preprocessing and loss improvements because it trains fast and stable.

```
Input (4×H×W)
  ↓
Stem ConvBlock(base)
  ↓
DownBlocks × depth−1
(channels double each stage)
  ↓
Bottleneck ConvBlock(2× last_ch)
  ↓
UpBlocks × depth−1
(skip concat at each scale)
  ↓
Output head: 1×1 Conv → (4×H×W) logits
```

2.2.3. TransUNet baseline and my modified TransUNet

I use a ResNet50 + ViT hybrid TransUNet (R50-ViT-B_16) as the transformer-based baseline [2–4]. Because MRI input is 4-channel but the pretrained backbone expects 3-channel, I add a learnable 1×1 projection:

$$\tilde{x} = \mathrm{Conv}_{1\times1}(x), \quad x \in \mathbb{R}^{4\times H\times W},\ \tilde{x} \in \mathbb{R}^{3\times H\times W}.$$

I also ensure the patch grid is consistent with the runtime image size for the hybrid encoder case.

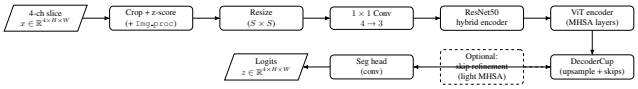


Figure 1. TransUNet pipeline used in my experiments. The dashed module is my decoder-side skip refinement variant.

Decoder-side modification: skip refinement. In my modified TransUNet, I optionally refine one skip feature map using a lightweight transformer block before concatenation. Given a skip tensor $s \in \mathbb{R}^{B\times C\times h\times w}$, I flatten to tokens and apply MHSA+MLP, then reshape back:

$$s \to \mathrm{tokens}(s) \to \mathrm{MHSA} \to \mathrm{MLP} \to s'.$$

This is a small change but it helps me test whether "cleaner skip information" improves tumor boundary prediction.

2.3. Training setup

2.3.1. Loss Function

I train using multi-class Cross-Entropy plus a foreground Dice term.

Cross-Entropy. Given logits $z \in \mathbb{R}^{B\times C\times H\times W}$ and labels $y \in \{0,\dots,C-1\}^{B\times H\times W}$:

$$\mathcal{L}_{CE} = -\frac{1}{N}\sum_{n=1}^{N} \log p\big(y_n \mid z_n\big),$$

and I optionally use class weights to reduce background dominance.

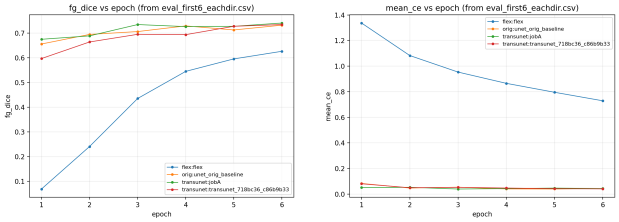
Foreground Dice with "present-class" masking. Naive Dice can be inflated when a class is absent in a batch. So I compute Dice only over foreground classes that actually appear in the ground-truth batch. Let foreground classes be $k \in \{1,2,3\}$. With one-hot masks t_k and predicted probabilities p_k:

$$\mathrm{Dice}_k = \frac{2\sum p_k t_k + \epsilon}{\sum p_k + \sum t_k + \epsilon}, \qquad m_k = \mathbb{I}\Big[\sum t_k > 0\Big],$$

$$\mathcal{L}_{Dice} = 1 - \frac{\sum_k m_k \cdot \mathrm{Dice}_k}{\sum_k m_k + \epsilon}.$$

Dice warmup. To avoid unstable early training, I warm up the Dice weight for the first few epochs:

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda(e)\cdot\mathcal{L}_{Dice}, \qquad \lambda(e) = \begin{cases} 0, & e \le e_{warm} \\ \lambda, & e > e_{warm}. \end{cases}$$



(a) Foreground Dice vs. epoch (b) mean CE vs. epoch

Figure 2. Validation foreground Dice and mean cross-entropy over epochs for the main model variants.

2.3.2. Optimization and training tricks

- **Optimizer**: Adam [8] (with small weight decay). Learning rate is a key tuning knob (default 10^{-5} in my script).
- **Mixed precision (AMP)**: I use `torch.amp.autocast` + `GradScaler` to reduce memory and speed up training.
- **Gradient accumulation**: when GPU memory is tight, I accumulate gradients over multiple steps so the effective batch size increases.
- **Foreground slice oversampling**: using `WeightedRandomSampler`, I give foreground slices a larger sampling weight (e.g., $\alpha = 5.0$) so the model sees tumor slices more frequently.
- **Multi-GPU support**: if multiple GPUs are available, I run data-parallel training.

Checkpointing and experiment tracking. Each run stores model configuration metadata (architecture settings + training knobs) together with the checkpoint, so later I can reproduce the exact setup. I also use a run identifier derived from the configuration (and git SHA if available) to avoid overwriting different experiments.

2.4. Evaluation metrics

I report **foreground Dice** on the validation set as my main metric, consistent with the training Dice definition:

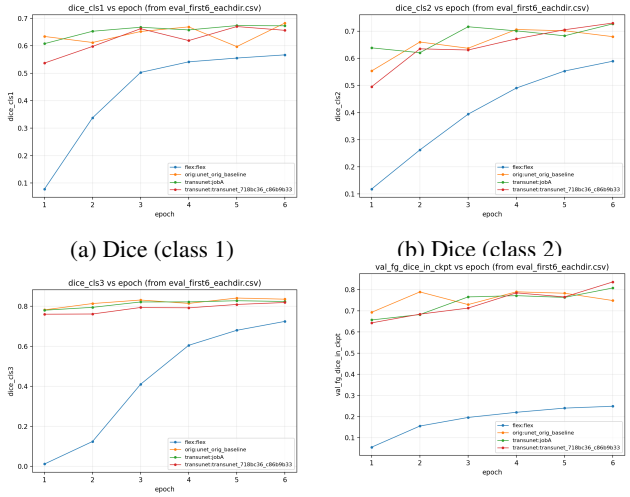
- compute $\arg\max$ prediction,
- one-hot encode prediction and ground-truth,
- drop background channel,
- compute present-class Dice over classes 1/2/3 and average.

In the Results section, I summarize all model variants in a single table with the same split rule and metric, so comparisons are clean.

2.5. Quantitative comparison

Key observations.

1. **Flexible U-Net vs. original U-Net.** Simply increasing depth and switching to GroupNorm without changing the loss hurt performance ("norm+depth only" row). After adding the foreground-only Dice loss and class-balancing tricks ("loss improved"), the flexible U-Net recovered much of the gap and produced higher Dice for tumor subregions (labels 2 and 3).
2. **TransUNet achieves the best overall Dice.** The ResNet+ViT TransUNet baseline obtains the highest FG Dice (0.7575), confirming that global self-attention is helpful for long-range tumor structure.
3. **Decoder modifications trade Dice for CE.** My decoder-side skip refinement and stronger augmentation slightly reduce mean CE but also reduce FG Dice. This suggests that the modified model is better calibrated on average yet more conservative on small lesions, which hurts the Dice score on rare classes.
4. **Class imbalance remains challenging.** All methods perform best on the largest structure (label 3), while label 1 is consistently the hardest to segment. This aligns with the heavy foreground/background skew in BraTS slices.



(a) Dice (class 1) (b) Dice (class 2)

(c) Dice (class 3) (d) FG Dice stored in checkpoints

Figure 3. Class-wise validation Dice and checkpoint-level foreground Dice over training epochs.

Table 1. Summary of my best checkpoints on the BraTS validation slices. Foreground (FG) Dice is averaged over labels 1–3 and only over slices where the class is present (see Sec. 2).

Method	Input res.	mean CE ↓	FG Dice ↑	(Dice_1, Dice_2, Dice_3)
Original U-Net	crop	0.0437	0.7326	(0.6826, 0.6800, 0.8351)
Flex U-Net (norm+depth only)	crop	0.3850	0.6323	(0.5395, 0.6475, 0.7100)
Flex U-Net (loss improved)	crop	0.1952	0.7033	(0.6048, 0.7100, 0.7952)
TransUNet baseline	320^2	0.0455	**0.7575**	(0.6750, 0.7652, 0.8322)
TransUNet (decoder + aug)	224^2	**0.0411**	0.7347	(0.6493, 0.7455, 0.8094)

3. Discussion

3.1. Analysis of results

TransUNet vs. U-Net variants. As summarized in Tab. 1, the strongest overall foreground Dice was obtained by the TransUNet baseline ("jobA") with 0.7575 FG Dice, about $+2.5$ points over the original U-Net (0.7326) and $+5.4$ points over my best Flex U-Net with improved loss (0.7033). This suggests that even without external pre-training, the hybrid ResNet–ViT encoder is able to leverage global context better than the purely convolutional models in this 2D slice setting.

However, the gain is not uniform across classes. From the class-wise Dice in Tab. 1, I see that:

- For the "core" classes (labels 1 and 3), the original U-Net is surprisingly competitive or even slightly better (e.g., Dice3 $= 0.8351$ vs. 0.8322 for TransUNet).
- The largest improvement from TransUNet is on label 2 (Dice2 $= 0.7652$ vs. 0.6800 for the original U-Net), which corresponds to more spatially extended regions where long-range context is likely more helpful.

So the Transformer-style global reasoning mainly helps with the more diffuse tumor subregion, while tight local structures are already handled well by a carefully tuned U-Net.

Effect of "Flex U-Net" and loss improvements. The Flex U-Net with only depth + GroupNorm changes ("norm+depth") performed worse than the original baseline (FG Dice 0.6323 vs. 0.7326), despite having more capacity. This indicates that simply making the network deeper and swapping BatchNorm for GroupNorm is not enough: without matching the optimization and loss to the new architecture, training can get stuck in a suboptimal regime (high cross-entropy of 0.3850).

After I reworked the training objective and sampler—using a combination of cross-entropy and foreground-only Dice loss, a short Dice warm-up, and oversampling slices with tumor—the same Flex architecture improved by roughly $+7$ Dice points (to 0.7033) and reduced mean CE by about 0.19. This shows that, in my setting, *loss design and data sampling were at least as important as the exact U-Net variant*. I also learned that computing Dice only on present classes (labels 1–3 that actually appear in the current batch) is more informative than including empty classes, which can artificially inflate Dice.

Decoder modifications and augmentation on TransUNet. My modified TransUNet with the skip-refinement block and stronger data augmentation (input size 224×224) slightly improved cross-entropy (from 0.0455 to 0.0411) but *reduced* FG Dice (from 0.7575 to 0.7347). One possible explanation is that the stronger augmentations plus the smaller input resolution encouraged the model to be more conservative around boundaries: logits are calibrated (lower CE) but segmentation masks become slightly over-smoothed, which hurts Dice. The light Transformer block on the decoder-side skips did not clearly help in this small-data setting; with only a few hundred effective training slices, it is easy to over-parameterize the decoder.

Overall, the main lessons I took from results are:

1. Architectural changes alone (deeper U-Net, extra Transformer on skips) can *hurt* if not paired with a compatible loss, sampling strategy, and resolution.
2. The biggest quality jump came from better training objectives and preprocessing (crop + normalization + `Img_proc`), not from model size alone.
3. Hybrid CNN–Transformer models do bring benefits, but in my experiments the gains were modest and concentrated on one tumor subregion.

3.2. Challenges and limitations

Limited 2D supervision from 3D volumes. Even though the BraTS dataset contains 3D volumes, my pipeline operates on individual 2D slices. After cropping to the brain and discarding empty slices, each model effectively sees a few thousand supervised slices, which is not large for a 100M-parameter network. This likely limits how much the Transformer can exploit global long-range structure along the through-plane dimension.

Training stability and hyperparameter search. Getting stable training for TransUNet from scratch was non-trivial. I had to tune the learning rate, batch size, and gradient accumulation carefully to avoid divergence or collapsing Dice. Due to GPU memory and time constraints, I could not explore a large hyperparameter grid (e.g., alternative optimizers, different warm-up lengths, or more aggressive regularization), so the reported checkpoints may not be globally optimal. Also, my epochs have to be within 15 epochs.

Evaluation choices. My evaluation focuses on foreground Dice over the three tumor labels using argmax predictions. I did not perform extensive post-processing (e.g., connected-component filtering or small-region removal), nor did I apply 3D consistency constraints. As a result, the reported Dice reflects the raw model output and may be slightly pessimistic compared to a fully polished BraTS pipeline that includes volumetric post-processing and ensembling.

3.3. Possible improvements and future work

This project was primarily about understanding how different architectural and training choices affect brain tumor segmentation quality. Based on the experiments, a few promising directions emerged:

- **Use pretrained backbones.** All my TransUNet runs were effectively trained from scratch. Initializing the ResNet and ViT parts from ImageNet or a medical-pretraining checkpoint would likely improve both convergence speed and final Dice, and might reduce the need for heavy decoder-side tricks.
- **Move to 3D or 2.5D context.** Extending the models to operate on short 3D slabs (or multi-slice 2.5D inputs) could help in distinguishing tumor from normal tissue when contrast is ambiguous in a single slice. The current 2D formulation ignores useful context along the axial axis.
- **Better class-imbalanced losses.** Although the present-only Dice already helps, there is still a noticeable gap between classes: label 1 remains the hardest, and Dice1 never exceeds about 0.68 even for the best model. Exploring boundary-aware losses, focal variants of Dice/CE, or explicit small-region weighting may improve performance on the rare but clinically important subregions.
- **Calibrated thresholds and post-processing.** For deployment-style use, I would like to treat logits as calibrated probabilities and then tune class-specific thresholds and 3D post-processing to optimize clinical metrics (e.g., sensitivity on enhancing tumor) instead of raw per-slice Dice only.
- **More systematic ablations.** Due to time, I only ran a few hand-picked combinations of depth, normalization, and loss settings. A more systematic ablation (e.g., changing one factor at a time with fixed training budget) would help disentangle which part of the pipeline actually drives improvements.

4. Statement of Individual Contributions

This project was completed entirely by myself. My main contributions include:

- **Dataset preparation and preprocessing:** Implementing the BraTS-style 2D slice extraction pipeline, foreground-based cropping, per-slice normalization, and optional intensity processing.
- **Model implementations:** Implementing the original U-Net baseline, the configurable "flex" U-Net, and the TransUNet variant with 4-channel MRI input and decoder-side skip refinement.
- **Training infrastructure:** Building the training loop with mixed precision, gradient accumulation, foreground oversampling, and checkpointing of both weights and configuration metadata.
- **Evaluation pipeline:** Implementing the foreground-only Dice metric and evaluation scripts that reconstruct models from checkpoints and generate the plots and tables in this report.
- **Experiment design and analysis:** Designing all experiments, running the training runs, and analyzing the quantitative and qualitative results presented in the paper.

5. Conclusion

Across my experiments, I found that the "engineering" around the model—how I crop, normalize, design the loss, and sample slices—has as much impact as the choice of backbone. A naive attempt to make U-Net deeper with GroupNorm actually *hurt* performance, but combining foreground-only Dice, class balancing, and foreground oversampling brought the flexible U-Net back to a strong regime (FG Dice ≈ 0.70) without changing the core architecture.

On the architecture side, the ResNet+ViT TransUNet baseline achieves the best overall foreground Dice in my setup, confirming that global self-attention is useful for capturing long-range tumor structure. However, a well-tuned, much smaller U-Net remains competitive and much easier to train, which is important for medical imaging projects with limited data and compute. My decoder-side TransUNet refinements slightly improve mean cross-entropy but trade off some Dice, suggesting that better calibration does not always translate into better overlap on small, rare structures.

Overall, the main lesson is that careful preprocessing and loss design are prerequisites for fair architecture comparison. Given a solid 2D slice pipeline with crop, per-slice normalization, and present-class Dice, both CNN and hybrid transformer models can reach good BraTS-style performance, but transformers only provide a clear margin when I am willing to pay the extra training cost.

References

[1] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pp. 234–241, 2015. 1, 2

[2] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. Yuille, and Y. Zhou. TransUnet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021. 1, 3

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021. 1

[4] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. 2, 3

[5] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Geremia, E. Arbel, G. Krainik, O. Maignan, E. Peeters, D. H. R. Brain, et al. The

multimodal brain tumor image segmentation benchmark (BRATS). *IEEE Transactions on Medical Imaging*, 34(10):1993–2024, 2015. 1

[6] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, R. T. Shinohara, C. Berger, S. Menezes, S. E. K. Bakas, et al. Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge. *arXiv preprint arXiv:1811.02629*, 2018. 1

[7] F. Milletari, N. Navab, and S.-A. Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 565–571, 2016. 2

[8] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 4

BraTS Brain Tumor Segmentation with Transformer in U-Net

Supplementary Material

Table 2. Validation metrics for the last five epochs of each model variant (rows within each block are ordered from older to newer checkpoints).

mean CE ↓	FG Dice ↑	Dice1	Dice2	Dice3
Original U-Net baseline				
0.0474	0.6951	0.6118	0.6603	0.8133
0.0543	0.7067	0.6519	0.6372	0.8309
0.0406	0.7296	0.6686	0.7064	0.8139
0.0399	0.7131	0.5968	0.7018	0.8408
0.0437	0.7326	0.6826	0.6800	0.8351
Flex U-Net (norm+depth)				
0.5516	0.5883	0.4762	0.6245	0.6641
0.5084	0.5931	0.5120	0.6293	0.6380
0.4659	0.6093	0.4750	0.6487	0.7041
0.4262	0.6270	0.5212	0.6506	0.7091
0.3850	0.6323	0.5395	0.6475	0.7100
Flex U-Net (loss improved)				
0.3474	0.6336	0.5362	0.6437	0.7208
0.2660	0.6934	0.5951	0.6949	0.7900
0.2381	0.6974	0.5862	0.7122	0.7938
0.2157	0.6971	0.5884	0.7097	0.7932
0.1952	0.7033	0.6048	0.7100	0.7952
TransUNet baseline (jobA, 320×320)				
0.0465	0.7535	0.6745	0.7537	0.8321
0.0475	0.7515	0.6588	0.7612	0.8345
0.0455	0.7463	0.6654	0.7514	0.8222
0.0455	0.7575	0.6750	0.7652	0.8322
0.0472	0.7548	0.6697	0.7639	0.8310