LINTUNET: A HYBRID TRANSFORMER-CNN ARCHITECTURE FOR BRAIN TUMOR SEGMENTATION

Lawrence Menegus, Dongsheng Che
East Stoudsburg University
lmenegus@live.esu.edu, dche@esu.edu

ABSTRACT

Brain tumor segmentation is vital in medical imaging for accurately identifying tumor regions in MRI scans. While U-Net and other CNN-based models have been widely used, they struggle with capturing long-range dependencies due to their localized receptive fields. To overcome this, we propose LinTUNet, a Hybrid Linformer Transformer U-Net, which integrates sparse self-attention into the U-Net architecture using a Sparse Linformer for efficient attention computation. By incorporating Transformer-based attention in the bottleneck layer, our model enhances feature extraction while maintaining computational efficiency. Performance evaluation using key metrics shows that LinTUNet surpasses U-Net in segmentation accuracy, achieving higher IoU and F1-scores. This paper presents our approach to developing a Hybrid Transformer-CNN model, demonstrating its advantages in medical image segmentation and deep-learning-driven healthcare applications.

KEY WORDS

Brain Tumor Segmentation; U-Net; Transformer; Sparse Linformer; Attention Layer; Convolutional Neural Network

1. Introduction

Deep learning-based segmentation models have become a critical component of medical imaging, particularly for identifying and segmenting brain tumors in MRI scans. Image segmentation enables precise analysis of specific structures, which is essential for accurate diagnosis and treatment planning [1]. Traditional Convolutional Neural Network (CNN)-based architectures, such as U-Net, have shown significant success in biomedical image segmentation [2]. However, CNN-based models face several challenges that limit their performance in segmenting complex tumor structures.

One major limitation of CNNs is their restricted receptive field, which prevents them from capturing long-range dependencies and global contextual relationships within medical images [3]. CNNs utilize small convolutional kernels, making it difficult to segment tumors with irregular boundaries accurately [4]. As a result, segmentation errors often occur, particularly when tumor shapes and sizes vary significantly across patients. Additionally, CNNs require extensive computational resources and long training times, making them less practical for real-time clinical applications [5]. Another drawback is their poor global awareness, as CNNs operate on local patches rather than considering the entire image context, leading to inconsistent boundary predictions [6]. CNN-based models depend on large annotated datasets, which are costly and time-consuming to obtain in the medical domain [6]. These challenges highlight the need for a segmentation model that can efficiently capture global spatial relationships while maintaining computational feasibility.

To address these issues and limitations, we propose LinTUnet, a hybrid Transformer-CNN model that integrates convolutional layers for local feature extraction with a Transformer-based Sparse Linformer self-attention mechanism to capture long-range dependencies. By combining the strengths of CNNs and Transformers, LinTUnet enhances segmentation accuracy while optimizing computational efficiency, making it particularly suitable for brain tumor segmentation in MRI scans.

LinTUNet leverages both Linformer and Sparse Transformer mechanisms to overcome the limitations of traditional CNN-based segmentation models, particularly in handling complex tumor structures in MRI scans. Linformer addresses the inefficiency of standard Transformers by reducing the quadratic complexity of self-attention to linear complexity $O(n)$ [7]. Instead of computing full self-attention matrices, Linformer compresses key and value projections into a lower-dimensional representation, significantly improving computational efficiency while preserving essential long-range dependencies. This allows LinTUNet to capture global spatial relationships without the excessive memory and processing requirements of full Transformers, making it feasible for high-resolution medical imaging [7].

Meanwhile, the Sparse Transformer enhances

segmentation accuracy by focusing computational resources on the most relevant spatial features rather than processing the entire image uniformly. Unlike traditional self-attention, which treats all pixels equally, the Sparse Transformer selectively attends to a subset of tokens, prioritizing regions with important tumor-related features [8]. This sparsity reduces redundancy and enhances the model's ability to detect tumors with varying shapes and textures while maintaining computational efficiency.

Together, Linformer and Sparse Transformer complement each other in LinTUNet. Linformer ensures that the model can process long-range dependencies efficiently, while the Sparse Transformer improves segmentation precision by directing attention to critical areas within the MRI scan [8]. By integrating both mechanisms, LinTUNet achieves superior segmentation accuracy while optimizing computational feasibility, making it a powerful solution for brain tumor segmentation in medical imaging.

2. Overview of U-Net (CNN) Architecture

U-Net is a convolutional neural network (CNN) designed for image segmentation, particularly excelling in pixel-level classification tasks such as tumor identification in MRI scans. It follows an encoder-decoder structure with skip connections, allowing it to segment medical images efficiently while preserving fine-grained spatial details.

The encoder compresses spatial information through downsampling operations, while the decoder reconstructs the segmented regions using transposed convolutions and concatenated feature maps from the encoder. This architecture effectively retains both low-level and high-level features, making it ideal for medical imaging tasks [2].

2.1 Encoder

The encoder consists of multiple downsampling blocks, each containing:

- Two convolutional layers with ReLU activation
- Batch normalization for stabilizing training
- Max pooling for reducing spatial dimensions

This hierarchical feature extraction process helps the network learn both fine-grained and abstract features, which are essential for precise image segmentation[2].

2.2 Bottleneck

The bottleneck layer is positioned between the encoder and decoder, capturing the most abstract representations of the input image. It acts as a bridge between feature extraction and segmentation refinement, ensuring essential patterns are preserved before reconstruction begins [4].

2.3 Decoder

The decoder restores the image resolution while maintaining segmentation accuracy. It achieves this by:

- Upsampling through transposed convolutions
- Concatenating feature maps from the encoder layers
- Applying double convolutions to refine segmentation accuracy

This ensures the precise reconstruction of the predicted mask.

2.4 Skip Connections

Skip connections play a crucial role in U-Net by passing detailed spatial information from the encoder to the decoder. This prevents information loss and improves segmentation precision, particularly for complex structures like tumors [2].

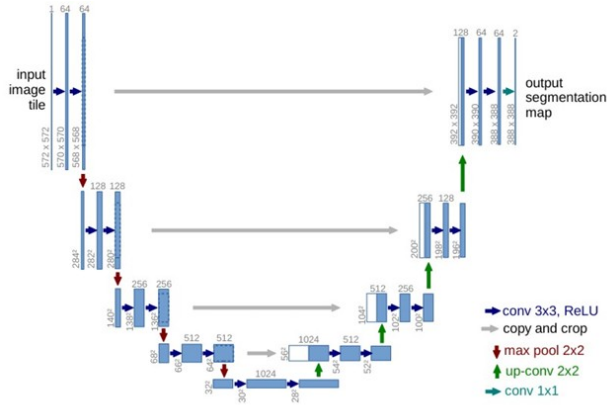


Figure 1: The U-Net Architecture proposed by Ronneberger et al. In 2015 [2].

3. Proposed Model: LinTUNet

To address the limitations of U-Net and CNN-based models, we introduce LinTUNet, a hybrid Transformer-CNN model that enhances segmentation accuracy by incorporating Transformer-based self-attention into the U-Net bottleneck.

3.1 Architecture of LinTUNet

LinTUNet builds upon the CNN based U-Net's encoder-decoder framework but enhances it with Linformer-based self-attention at the bottleneck stage to improve feature representations. Like a traditional Unet CNN this hybrid consists of a Decoder, Bottleneck or in this case a linformer attention layer bottleneck and a decoder.

The encoder (downsampling path) and decoder (upsampling path) are the same as the traditional CNN. The encoder consists of multiple convolutional layers for local feature extraction, followed by Batch Normalization, Dropout, and ReLU activations to improve generalization and prevent overfitting. These layers progressively reduce the spatial dimensions while capturing hierarchical features crucial for accurate segmentation. The decoder utilizes transposed convolutions to restore spatial resolution, while skip connections from the encoder reintegrate fine-grained details lost during downsampling. Finally, a 1x1 convolution is applied to produce the segmented tumor mask, ensuring precise localization of tumor regions.

3.2 Sparse Linformer Attention Layer

At the bottleneck stage, Sparse Linformer transforms the encoded feature map into a sequence representation to apply self-attention, effectively capturing long-range dependencies and improving segmentation consistency. Figure 2 illustrates an example of our proposed Sparse Linformer Attention Layer in the LinTUNet bottleneck, highlighting its role in feature transformation and efficient attention computation.

A Linformer reduces the computational complexity of the standard Transformer attention mechanism from $O(N^2)$ to $O(N)$ by projecting the sequence into a lower-dimensional space [7]. This is achieved using a low-rank approximation of the attention matrix, making it computationally efficient for tasks like brain tumor segmentation.

In standard self-attention, the attention matrix is computed as:

$$A = softmax\left(\frac{Q K^T}{\sqrt{d_k}}\right) \quad (1)$$

where

$$Q = XW_Q, K = XW_K, V = XW_V \quad (2)$$

represent the query, key, and value matrices. The complexity of this computation is $O(N^2)$, making it impractical for large medical images [7].

The Linformer approximates this by introducing low-rank projections for the key and value matrices:

$$K' = EK, V' = FV \quad (3)$$

where

$$E, F \in \mathbb{R}^{rxN} \quad (4)$$

are projection matrices that reduce the sequence length from N to r ($r \ll N$) [7]. The modified attention computation becomes:

$$A = softmax\left(\frac{Q K^T}{\sqrt{d_k}}\right) \quad (5)$$

leading to a output of:

$$Linformer - Attention(Q, K, V) = A'V' \quad (6)$$

This transformation reduces computational complexity to $O(N)$ while maintaining segmentation accuracy. For our proposed Model we used a Sparse Linformer. A sparse Linformer further optimizes this approach by applying sparsity constraints on the projection matrices E and F, ensuring that only a subset of elements contribute to the computation [8,9]:

$$K' = S(E)K, V' = S(F')V \quad (7)$$

where S(E) and S(F) select the most significant rows to preserve spatial information while reducing memory usage.

The concept of sparse attention mechanisms was introduced in the paper "Generating Long Sequences with Sparse Transformers" by Child et al.[9], which demonstrated that sparse factorizations of the attention matrix can reduce computational complexity while effectively modeling long-range dependencies. LinTUNet uses this mixture of a Sparse and Linformer attention layer in a sequence during the bottleneck stage which then is reshaped back into a 2D feature map before entering the decoder.

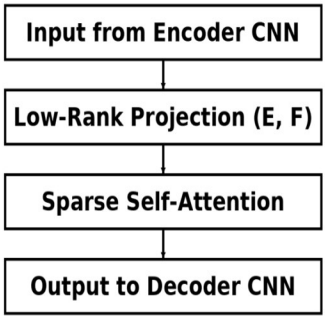


Figure 2. Example of our proposed Sparse Linformer Attention Layer of the LinTUNet Bottleneck.

The Linformer reduces the size of the attention layer by compressing pixel-to-pixel interactions and projecting the information into a lower-dimensional space. It leverages low-rank approximations, ensuring that only the most relevant information is retained from the attention calculations.

The Sparse Linformer further enhances this approach by selecting only the most important rows from the compressed data, significantly reducing memory usage while maintaining the model's ability to capture essential spatial relationships.

4. Training and Model Evaluation

4.1 Data Collection

The dataset used in this study comprises 2,146 MRI brain scan images specifically curated to detect brain tumors, sourced from a Kaggle dataset for semantic segmentation tasks [10]. To optimize the training process, the dataset was divided into three subsets: 1,501 images for training, 429 images for validation, and 215 images for testing. This division ensured the models were trained on a diverse set of labeled data while being fine-tuned and evaluated on distinct subsets, thereby enhancing their ability to generalize to unseen data. Each MRI image is accompanied by pixel-wise annotations, providing segmentation masks that delineate the tumor regions. This segmentation approach is particularly critical for models like U-Net and LinTUNet, as it enables them to not only detect the presence of tumors but also accurately identify tumor boundaries. The dataset includes a variety of MRI images showcasing tumors with different shapes, sizes, and locations, allowing the models to learn comprehensive feature representations across diverse tumor characteristics.

4.2 Feature Selection and Augmentation

This study utilizes PyTorch for medical image segmentation, specifically targeting brain MRI scans. A custom preprocessing pipeline was developed to pair MRI images with their corresponding segmentation masks, formatted in COCO-style annotations [11]. The pipeline incorporates data augmentation techniques such as resizing, grayscale conversion, tensor conversion, and normalization to enhance the training process and improve model generalization. The dataset is structured into distinct directories for training, validation, and testing, with mechanisms to ensure accurate pairing of images and masks. Additionally, a custom dataset class, built on PyTorch's Dataset class, facilitates efficient batch processing and GPU-accelerated transformations via CUDA, significantly reducing training times [12].

The implementation includes functions for managing COCO-style annotations and visualizing image samples overlaid with segmentation masks, enabling quality assurance. Error handling mechanisms were integrated to address mismatches between images and masks, ensuring only valid pairs are used during training. These features combined with the robust data pipeline, contribute to the model's accuracy, efficiency, and scalability in the medical imaging domain.

4.3 Mask Generation and Dataset Preparation

In this study, segmentation masks for the MRI brain scans were generated using the COCO annotation format, which provides pixel-level annotations of tumor regions. The dataset's annotation file, stored in JSON format, was processed to extract segmentation data for each MRI image. A custom create_mask function was developed to iterate through the segmentation points specified in the annotations. These points, stored as a list of coordinates outlining the tumor boundary, were converted into polygonal shapes using the skimage.draw.polygon function. This method filled the tumor regions with white pixels (255), effectively creating binary masks that highlighted the tumor areas. The generated masks were saved as TIFF files, with each mask corresponding to a specific MRI image. Figure 3 illustrates a Brain Scan MRI image with the COCO-Style Annotation, demonstrating the pixel-level segmentation used in this study.

The images and their corresponding segmentation masks were organized into separate directories for training, validation, and testing. A *mask_folders_if_not_exist* function was using dynamic programming implemented to ensure that masks were only generated if the output directories did not already exist, thus avoiding redundancy. To maintain data integrity, a *compare_folders* function was developed to identify and remove mismatched images and masks. Finally, a *train_test_split* function was employed to partition the data into training, validation, and test sets, ensuring consistency across the models. This structured approach ensured that the data was well-prepared for model training and evaluation, enabling robust performance for both U-Net and LinTUNet.

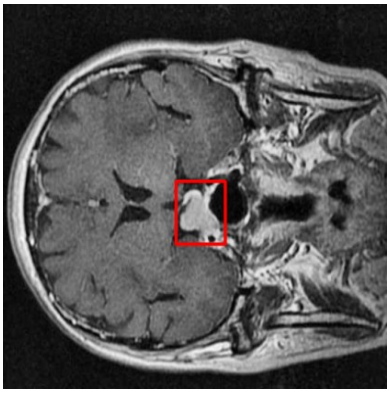


Figure 3. Brain Scan MRI image with the COCO-Style Annotation

4.4 Training Strategy

The LinTUNet model is trained using Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss), a widely used loss function for binary segmentation tasks like tumor detection in brain MRI scans. This loss function combines the sigmoid activation function with binary cross-entropy loss, ensuring stable numerical computation and penalizing incorrect predictions. The BCE loss is calculated as:

$$L=-\frac{1}{N}\sum_{i=1}^{N}\left[y_i\log\left(\sigma\left(x_i\right)\right)+\left(1-y_i\right)\log\left(1-\sigma\left(x_i\right)\right)\right] \quad (8)$$

where

y_i is the ground truth label, x_i is the predicted value (logits), and

$$\sigma\left(x_i\right)=\frac{1}{\left(1+e^{-x_i}\right)} \quad (9)$$

is the sigmoid activation function. This loss function is particularly effective in handling imbalanced datasets, which is crucial for medical imaging applications [13].

For optimization, the Adam optimizer [14] is chosen due to its adaptive moment estimation, which dynamically adjusts learning rates based on past gradient updates. Adam's parameter updates follow:

$$m_t=B_1 m_{t-1}+\left(1-B_1\right)g_t \quad (10)$$

$$v_t=\beta_2 v_{t-1}+\left(1-\beta_2\right)g_t^2 \quad (11)$$

$$m_t=m_t\frac{m_t}{1-\beta_1^t},\ v_t=\frac{v_t}{1-\beta_t^t} \quad (12)$$

$$\theta_t=\theta_{t-1}-\alpha\frac{m_t}{\sqrt{v_t}+\epsilon} \quad (13)$$

where m_t, v_t are the first and second moment estimates of the gradient, β_1, β_2 control the decay rates, and α is the learning rate.

To further stabilize training for the models, a StepLR learning rate scheduler is applied, which reduces the learning rate at predefined intervals:

$$\alpha_t=\alpha_0\gamma^{\lfloor t/step\rfloor} \quad (14)$$

where α_0 is the initial learning rate, γ is the decay factor, and t is the current epoch number. This ensures that the learning rate is high during initial epochs for rapid learning but decays over time for fine-tuning.

To accelerate training and reduce memory consumption, Automatic Mixed Precision (AMP) is employed. AMP dynamically switches between single-precision (FP32) and half-precision (FP16) computations, allowing for larger batch sizes and faster training while maintaining numerical stability [15]. A critical component of AMP is gradient scaling, which prevents underflow issues in FP16 computations by scaling the gradients before backpropagation:

$$g=Sg \quad (15)$$

where S is a scaling factor applied to the gradients g, ensuring that small values do not get truncated to zero. This approach significantly speeds up deep learning models trained on large medical datasets without affecting accuracy [15].

The model training process follows a structured pipeline that includes data augmentation, GPU acceleration, and progressive learning rate adjustments. Each epoch begins with a forward pass, where in put MRI images are processed through the encoder, Sparse Linformer attention layer, and decoder. The model's output is compared against ground truth segmentation masks using BCEWithLogitsLoss, and gradients are computed for backpropagation. To prevent vanishing or exploding gradients, gradient clipping is applied by restricting the norm of the gradients:

$$g=\frac{g}{max\left(1,\frac{/\!/\,g\,/\!/}{c}\right)} \quad (16)$$

where g is the gradient vector and c is the clipping threshold. The validation phase follows each training epoch, assessing model performance on unseen data. The learning rate scheduler is updated at predefined intervals to stabilize training and enhance generalization[14].

To calculate the overall accuracy we used the pixel-wise accuracy calculation for model training and evaluation.

Which used the raw predictions from the model (logits) are passed through a sigmoid function to convert them into binary values (0 or 1). The same process is applied to the ground truth masks to ensure consistency. The predicted and actual masks are then compared element-wise, and the number of correctly classified pixels is summed. Finally, this sum is divided by the total number of pixels in the mask to obtain the accuracy.

$$Accuracy = \frac{\left(\sum_i^n 1\left(y_{(pred_i)} = y_{(true_i)}\right)\right)}{N} \quad (17)$$

In pixel-wise accuracy calculation $y_{(pred_i)}$, represents the predicted binary mask, while $y_{(true_i)}$ is the actual ground truth mask. The total number of pixels in the mask is denoted by N. Accuracy is computed as the ratio of correctly classified pixels to the total pixels, using the indicator function $1(\cdot)$, which returns 1 if the prediction matches the ground truth and 0 otherwise. This ensures an accurate measure of segmentation performance across the dataset [17].

4.5 Segmentation Performance Metrics

To evaluate segmentation performance, Intersection over Union (IoU), Dice Score (F1 Score), Precision, and Recall are computed. IoU, also known as the Jaccard Index, measures the degree of overlap between predicted and ground truth segmentation masks and is defined as:

$$IoU = \frac{A \cup B}{A \cap B} \quad (18)$$

Higher IoU values indicate better segmentation performance [18]. Similarly, the Dice Score (F1 Score) quantifies segmentation accuracy by measuring the harmonic mean of precision and recall:

$$Dice = 2\frac{A \cap B}{A + B} \quad (19)$$

This metric is particularly useful in medical imaging, as it balances false positives and false negatives.

Precision and recall further assess the model's reliability. Precision evaluates the proportion of correctly predicted tumor pixels relative to all predicted tumor pixels:

$$Precision = \frac{TP}{TP + FP} \quad (20)$$

where Tp = True Postitives, FP = False Positives, FN = False Negatives. These metrics ensure that the model does not produce excessive false positives or false negatives [19]

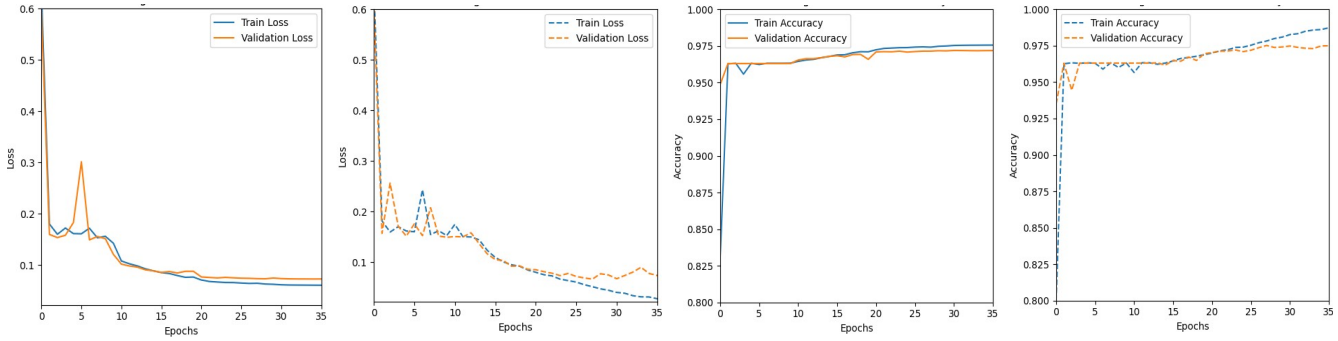
Throughout training, key performance indicators such as training and validation loss, accuracy, IoU, precision, and F1-score are recorded per epoch to monitor the model's learning progress. This ensures early detection of overfitting or underperformance, allowing necessary adjustments to be made. Once trained, the final model is evaluated on an independent test set to validate its generalization ability on unseen MRI scans. By integrating Sparse Linformer self-attention in the bottleneck stage, BCEWithLogitsLoss, Adam optimization with learning rate scheduling, and mixed precision training, LinTUNet achieves an efficient and accurate segmentation while maintaining computational efficiency. This combination of Transformer-based self-attention and convolutional feature extraction enables the model to effectively capture both local and global contextual information, making it highly suitable for image segmentation tasks.

5. Results and Discussion

5.1 Training and Validation Loss and Accuracy

This project was conducted on Google Colab, utilizing Google Cloud's computational resources. The results presented below are based on one of several trial runs of the Jupyter notebook. Due to variability in cloud resource allocation, minor differences may occur between runs. However, LinTUNet consistently outperformed U-Net across all trials, with the extent of improvement varying. Despite these minor fluctuations, the overall trends and conclusions remain impressive and robust. The following graphs and visuals represent one such trial, providing a detailed analysis of U-Net (CNN) and LinTUNet (Transformer-CNN), highlighting their respective loss functions, accuracy trends, and segmentation performance.

Demonstrated in Figure 4, LinTUNet slightly outperforms U-Net in both accuracy and loss reduction during the entirety of both model's training and evaluation. In visuals (a) and (b), the loss curves of U-Net and LinTUNet, respectively, indicate that LinTUNet achieves a slightly lower loss. Moreover, LinTUNet attains an impressive 98.95% accuracy, slightly surpassing U-Net's 97.57%, as illustrated in visuals (c) and (d). This suggests that LinTUNet not only enhances pixel-wise classification accuracy but also ensures more consistent and precise segmentation across the dataset. The combination of lower loss and higher accuracy reinforces LinTUNet's ability to effectively capture long-range dependencies and spatial relationships, leading to superior segmentation performance compared to traditional CNN-based architectures.



(a) U-Net Loss (b) LinTUNet Loss (c) U-Net Accuracy (d) LinTUnet Accuracy

Figure 4. U-Net and LinTUNet loss and accuracy curves, illustrating model performance. (a) U-Net's loss curve. (b) LinTUNet's loss curve. (c) U-Net's accuracy curve. (d) LinTUNet's accuracy curve.

(a) Input Image (b) U-Net's Prediction (c) LinTUnet Prediction

Figure 5. Model predictions for the Image Segmenation. (a) The input image used for both U-Net and LinTUNet models. (b) U-Net's predicted segmentation mask (c) LinTUNet's predicted segmentation mask.

5.2 Segmentation Performance

Table 1 presents a comprehensive comparison of UNet (CNN) and LinTUNet across key segmentation performance metrics this includes F1 Score, Intersection of Union and overall precision.

Table 1. Results of the all metrics of U-Net and LinTUNet

Metric	U-Net (CNN)	LinTUNet (ours)
F1 Score	0.6013	**0.8675**
IoU	0.4321	**0.7668**
Precision	0.7871	0.7871

As shown in Table 1, LinTUNet significantly outperforms U-Net in image segmentation across multiple key metrics. It achieves an impressive F1 Score of 0.8675, compared to U-Net's 0.6013, demonstrating a much better balance between precision and recall. This indicates that LinTUNet generates fewer false positives and false negatives, resulting in more accurate and reliable segmentation

Another crucial metric, Intersection over Union (IoU), further highlights LinTUNet's superiority in spatial localization. With an IoU of 0.7668—considerably higher than U-Net's 0.4321—LinTUNet more accurately aligns predicted tumor regions with the input images. This improvement is visually evident in Figure 5, where LinTUNet's segmentation (c) closely matches the input image of the tumor related area (a), whereas U-Net's prediction (b) deviates significantly, with the segmented tumor area appearing overly broad and imprecise.

Interestingly, both models achieve an identical precision of 0.7871, indicating that they identify tumor pixels with similar correctness. However, LinTUNet's higher F1 Score and IoU suggest superior overall segmentation by reducing false negatives, making it more effective at detecting the tumor regions.

This analysis covers the entire prediction performance of both models, confirms that LinTUNet consistently delivers more precise and accurate tumor segmentation

than U-Net, making it a more reliable approach for medical image analysis.

5.3 Execution Time

Beyond accuracy and other performance metrics, LinTUNet offers a significant advantage in inference speed, making it much more efficient for real-time applications. It processes an image in just 0.0002 seconds (0.2 milliseconds), whereas U-Net takes 0.0014 seconds (1.4 milliseconds). This means LinTUNet generates predictions 7 times faster than U-Net, a crucial improvement for high-throughput medical imaging and real-time diagnostic applications. A key contributor to this efficiency could be from the automatic mixed precision (AMP) autocasting, which optimizes computations by dynamically switching between single-precision (FP32) and half-precision (FP16) during training, helping to increase processing power and computational efficiency.

6. Conclusion

Our proposed hybrid Transformer-CNN model, LinTUNet, consistently outperforms traditional CNN-based methods like U-Net in image segmentation. By integrating Sparse Linformer self-attention into the U-Net architecture, LinTUNet captures long-range dependencies while maintaining computational efficiency. Its ability to deliver more accurate image segmentations while significantly reducing processing time makes it highly suitable for real-time medical applications. Having demonstrated the advantages of this approach, we anticipate future studies applying LinTUNet to larger and more diverse medical imaging datasets, including multi-modal MRI scans segmentation tasks, to further validate its effectiveness. By harnessing the power of Transformers for medical imaging, LinTUNet has the potential to improve diagnostic accuracy, enhance early detection, and ultimately contribute to better patient outcomes, potentially saving lives.

References:

[1] G. Litjens et al., "A survey on deep learning in medical image analysis," Medical Image Analysis, vol. 42, pp. 60-88, 2017. [Online]. Available: https://doi.org/10.1016/j.media.2017.07.005

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," arXiv preprint, arXiv:1505.04597, 2015. [Online]. Available: https://arxiv.org/abs/1505.04597

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint, arXiv:1409.1556, 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[4] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning dense volumetric segmentation from sparse annotation," Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2016, pp. 424-432. [Online]. Available: https://doi.org/10.1007/978-3-319-46723-8_49

[5] F. Isensee et al., "nnU-Net: A self-adapting framework for U-Net-based medical image segmentation," Nature Methods, vol. 18, pp. 203-211, 2021. [Online]. Available: https://doi.org/10.1038/s41592-020-01008-z

[6] B. H. Menze et al., "The multimodal brain tumor image segmentation benchmark (BraTS)," IEEE Transactions on Medical Imaging, vol. 34, pp. 1993-2024, 2015. [Online]. Available: https://doi.org/10.1109/TMI.2014.2377694

[7] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," arXiv preprint, arXiv:2010.11929, 2020. [Online]. Available: https://arxiv.org/abs/2010.11929

[8] A. Wang et al., "Linformer: Self-attention with linear complexity," arXiv preprint, arXiv:2006.04768, 2020. [Online]. Available: https://arxiv.org/abs/2006.04768

[9] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," arXiv preprint, arXiv:1904.10509, 2019. [Online]. Available: https://arxiv.org/abs/1904.10509

[10] P. Darabi, "Brain tumor image dataset (semantic segmentation)," Kaggle, 2021. [Online]. Available: https://www.kaggle.com/datasets/pkdarabi/brain-tumor-image-segmentation

[11] M. Deng, K. He, and R. Girshick, "COCONut: Modernizing COCO segmentation," Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2024/papers/Deng_COCONut_Modernizing_COCO_Segmentation_CVPR_2024_paper.pdf

[12]Deep learning software, NVIDIA Developer, 2023. [Online]. Available: https://developer.nvidia.com/deep-learning-software

[13] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, Cambridge, MA: MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint, arXiv:1412.6980, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[15] P. Micikevicius et al., "Mixed precision training," International Conference on Learning Representations (ICLR), 2018. [Online]. Available: https://openreview.net/forum?id=r1gs9JgRZ

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3431-3440. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298965

[17] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. Jorge Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, 2017, pp. 240-248. [Online]. Available: https://doi.org/10.1007/978-3-319-67558-9_28