Intro to Deep Learning Assignment 3x

Tsinghua University
Allan Li (李盼加乐), 2025403473

The purpose of this report is to evaluate the performance of deep learning architectures, specifically encoder-decoder networks, in the automatic segmentation of brain tumors from Brain MRI scans. The dataset utilized is derived from The Cancer Genome Atlas (TCGA) lower-grade glioma collection, consisting of MRI slices paired with manual segmentation masks. The task is formulated as a pixel-wise binary classification problem, where the objective is to accurately delineate tumor tissue from healthy brain structure.

In this experiment, three distinct modeling scenarios are explored to assess different approaches to medical image segmentation. First, a standard U-Net architecture is implemented to establish a performance baseline using standard cross-entropy loss. Second, the training methodology is modified to incorporate Dice Loss, a metric specifically designed to handle class imbalance by directly optimizing for the overlap between predicted and ground-truth masks. Finally, the architecture is modified into an R2U-Net (Recurrent Residual U-Net) to investigate whether recurrent residual convolutional blocks provide better feature representation than standard convolutional layers.

Methods

The dataset utilized for this experiment was the "LGG MRI Segmentation" collection obtained from Kaggle. This dataset provides brain Magnetic Resonance Imaging (MRI) scans paired with manually segmented masks.

The deep learning models were built using the PyTorch framework, with architectural implementations adapted from key literature in the field. The baseline U-Net structure was derived from the original work by Ronneberger et al. (2015) . The Recurrent Residual U-Net (R2U-Net) was implemented based on the architectural improvements proposed by Alom et al. (2018) . All training procedures were executed in a remote cloud computing environment hosted by ai.paratera.com utilizing an NVIDIA RTX series GPU to accelerate the model training and validation processes.

Implementation of Models

1. Base U-Net Architecture The baseline model utilizes the U-Net encoder-decoder architecture. It features a contracting path to capture context and a symmetric expanding path for precise localization, linked by skip connections to retain high-resolution features.

```
class UNet(nn.Module):
    def __init__(self, in_channels=3, out_channels=1):
        super().__init__()
        # ... Encoder and pooling layers ...
        self.bottleneck = DoubleConv(512, 1024)
        self.up1 = nn.ConvTranspose2d(1024, 512, kernel_size=2, stride=2)

    def forward(self, x):
        # ... Encoder pass ...
        d1 = self.up1(b)
        d1 = torch.cat((e4, d1), dim=1) # Skip connection
        return self.final_conv(d1) # Simplified decoder output
```

2. Dice Loss Optimization To address class imbalance between the small tumor regions and the background, the standard loss function was replaced with Dice Loss. This metric directly optimizes the overlap between the predicted mask and the ground truth.

```
class DiceLoss(nn.Module):
    def forward(self, logits, targets):
        inputs = torch.sigmoid(logits).view(-1)
        targets = targets.view(-1)
        intersection = (inputs * targets).sum()
        dice = (2.0 * intersection + 1.0) / (inputs.sum() + targets.sum() +
1.0)
        return 1 - dice
```

3. Recurrent Residual U-Net (R2U-Net) The architecture was modified to an R2U-Net by substituting standard convolutional blocks with Recurrent Residual Convolutional Units3. These units accumulate features over discrete time steps ($t = 2$) to improve feature representation without increasing network depth.

```
class RecurrentConvLayer(nn.Module):
    def forward(self, x):
        prev_state = x
        for _ in range(self.t): # t=2
            curr_state = self.conv(prev_state)
            curr_state = self.bn(curr_state) + x # Residual add
            prev_state = self.relu(curr_state)
        return prev_state
```

Summary of Results

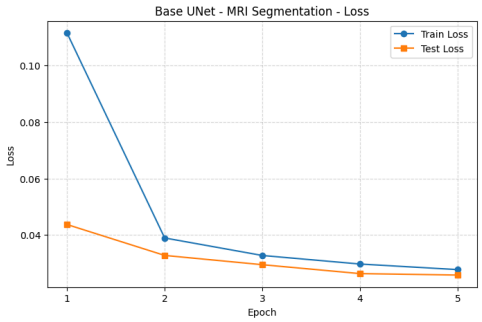
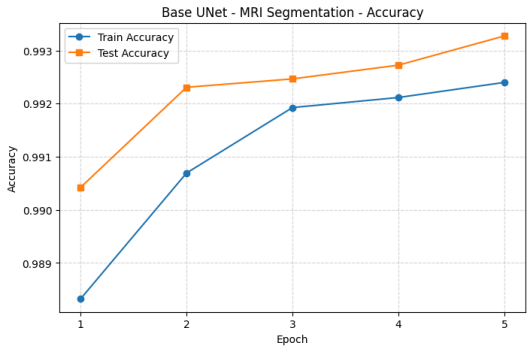
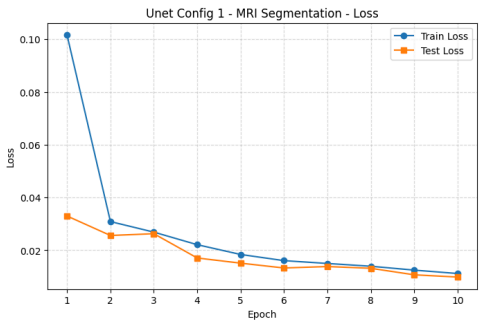
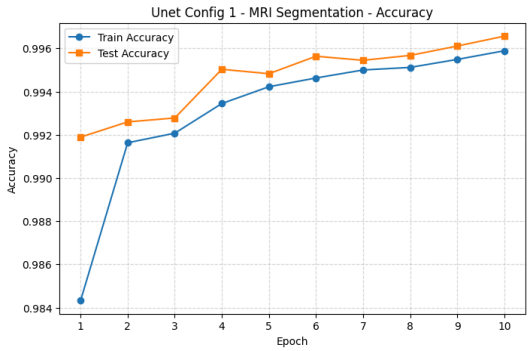
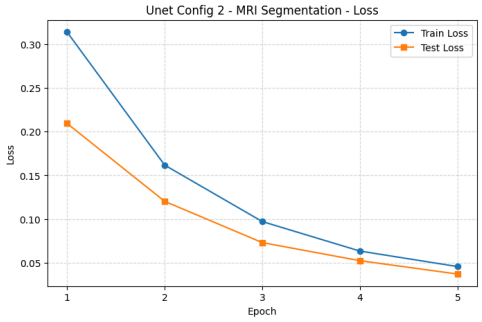
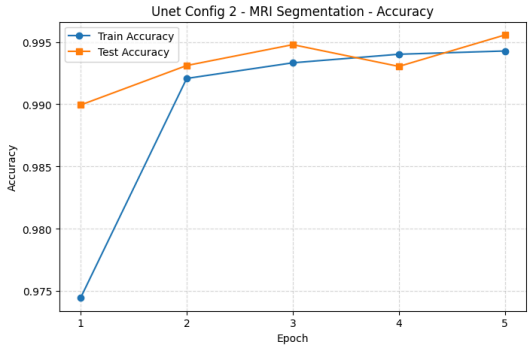
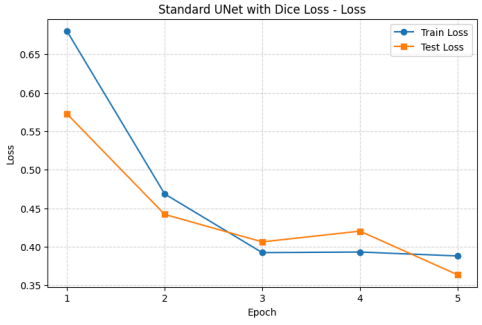
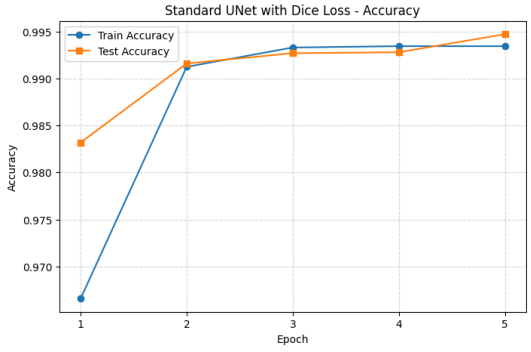
Hyperparameters

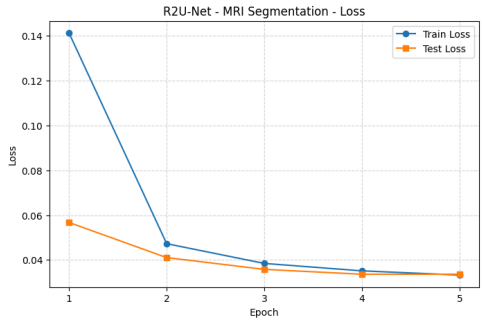
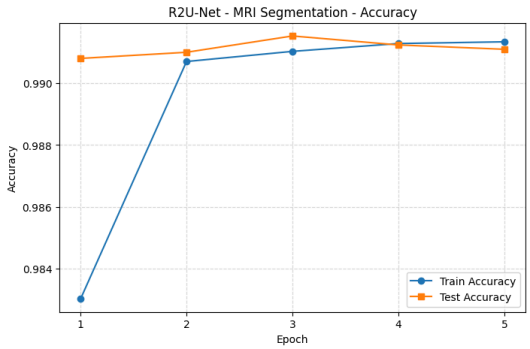
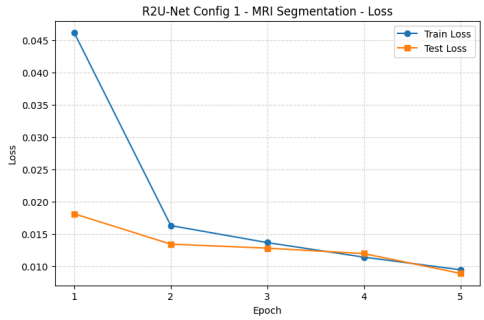
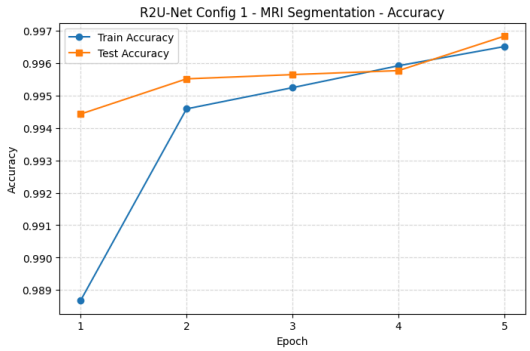
Model	Learning Rate	Epochs	Optimizer	Batch Size
Base U-Net Model	0.001	5	SGD	32
config_1 U-Net Model	0.001	10	Adam	32
config_2 U-Net Model	0.0001	5	Adam	32
Dice Loss U-Net Model	0.001	5	Adam	16
Base R2U-Net Model	0.0001	5	SGD	16
config_1 R2U-Net Model	0.0001	5	Adam	16

Training Statistics

Model	Train Loss	Train Accuracy	Test Loss	Test Accuracy	Training Time	Params
Base U-Net Model	0.0277	0.9924	0.0258	0.9933	202.56s	31,043,521
config_1 U-Net Model	0.0112	0.9959	0.0098	0.9966	324.00s	31,043,521
config_2 U-Net Model	0.0458	0.9943	0.0372	0.9956	160.77s	31,043,521
Dice Loss U-Net Model	0.3879	0.9934	0.3635	0.9947	154.99s	31,043,521
Base R2U-Net Model	0.0333	0.9913	0.0336	0.9911	202.65s	21,296,641
config_1 R2U-Net Model	0.0095	0.9965	0.0089	0.9968	201.57s	21,296,641

Accuracy & Loss Plots

Experiment Description	Loss Plot	Accuracy Plot
Base U-Net Model		
config_1 U-Net Model		
config_2 U-Net Model		
Dice Loss U-Net Model		

Experiment Description	Loss Plot	Accuracy Plot
Base R2U-Net Model		
config_1 R2U-Net Model		

Analysis

Base U-Net Performance

1. Impact of Optimizer: SGD vs. Adam The initial experiments focused on establishing a baseline using the standard U-Net architecture. Comparing the **Base U-Net** (SGD) with the **U-Net (config_1)** (Adam) reveals a significant performance advantage for the Adam optimizer. With the same learning rate (0.001), the model trained with Adam achieved a substantially lower test loss (0.0098 vs. 0.0258) and higher accuracy (99.66% vs. 99.33%). This aligns with Adam's ability to adapt learning rates for individual parameters, allowing it to navigate the loss landscape more efficiently than standard SGD, especially in the early stages of training.

2. Convergence Speed and Epochs The results from **U-Net (config_1)** demonstrate that the model converges remarkably quickly. Even within the first few epochs, accuracy reached over 99%, and loss dropped significantly. Doubling the epochs from 5 to 10 (comparing Base vs. config_1) caused some improvements, but the marginal gain suggests that the U-Net architecture is highly efficient for this segmentation task and does not require extensive training durations to achieve high accuracy rates.

3. Learning Rate Sensitivity Comparing **U-Net (config_1)** (LR=0.001) with **U-Net (config_2)** (LR=0.0001) highlights the importance of learning rate selection. The lower learning rate in config_2 resulted in a higher final loss (0.0372 vs. 0.0098), indicating that the model was converging too slowly and likely got stuck in a suboptimal local minimum within the fixed 5-

epoch window. A learning rate of 0.001 with the Adam optimizer appears to be a more optimal configuration for this dataset.

4. Overfitting Analysis A common concern with deep learning models on smaller medical datasets is overfitting, where the model memorizes the training data but fails to generalize. However, our results indicate minimal signs of overfitting. Across all successful experiments, the **Test Loss** closely tracks or is even slightly lower than the **Train Loss** (e.g., config_1 U-Net: Train 0.0112 vs. Test 0.0098). This consistency suggests the model is generalizing well to unseen data, likely because of the robust feature extraction capabilities of the U-Net's encoder-decoder structure.

5. Advanced Architectures and Loss Functions The **Dice Loss U-Net** experiment, while achieving high accuracy (99.47%), reported a significantly different loss scale (~0.36) compared to cross-entropy runs. This is expected as Dice Loss operates on a different mathematical principle (overlap maximization).

The best results came from the **R2U-Net (config_1)**. By using recurrent residual blocks, this model achieved the lowest test loss (**0.0089**) and highest accuracy (**99.68%**). This confirms that these connections help the network gather detailed information more effectively, allowing it to draw much sharper and more precise outlines around the tumor.

The choice of optimizer was also key. While the version using SGD did well, switching to Adam (config_1) gave the best results of all. What makes this even more impressive is how efficient the model is. The R2U-Net actually uses about 30% fewer parameters than the standard U-Net (21 million vs. 31 million). Despite this, it trained in 201 seconds; basically the same amount of time as the simpler Base U-Net.

Conclusion

This study confirms that the U-Net architecture is highly effective for brain MRI segmentation, demonstrating rapid convergence and short training times across all tested configurations. While three distinct frameworks with varying hyperparameters were evaluated, the results indicate that optimal performance can be achieved with a relatively small number of epochs using the Adam optimizer. Most notably, the Recurrent Residual U-Net (R2U-Net) emerged as the superior model. It not only outperformed the standard U-Net and Dice Loss variants but did so with fewer trainable parameters and at a training speed comparable to the base models, highlighting its efficiency and suitability for medical image segmentation tasks.

References & Acknowledgements

References

Alom, M. Z., Hasan, M., Yakopcic, C., Taha, T. M., & Asari, V. K. (2018). Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. *arXiv preprint arXiv:1802.06955*.

Buda, M., Saha, A., & Mazurowski, M. A. (2019). Association of genomic subtypes of lower-grade gliomas with shape features automatically extracted by a deep learning algorithm. *Computers in Biology and Medicine*, 109, 218-225. Dataset available at Kaggle.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18* (pp. 234-241). Springer International Publishing.

Acknowledgements

I would like to thank ai.paratera.com for providing the cloud infrastructure and computing resources used to train and validate the models in this experiment.