Medical Image Segmentation: Comparison Between U-Net and ResNet+FPN

Vaona Christian (VR495465)
Master's Degree in Artificial Intelligence
Department of Computer Science, University of Verona
vaonachristian@studenti.univr.it

Project report, Computer Vision & Deep Learning Course
Professor: Vittorio Murino

Contents

1 Introduction

1.1 Motivation and Rationale

Image segmentation is an important topic for image processing and computer vision, with applications in various fields such as scene understanding, medical image analysis, robotic automation, video surveillance, augmented reality, and more.

In particular, segmentation of medical images is a crucial in the analysis of medical images. This process, which consists of dividing an image into regions that correspond to different anatomical structures or areas of interest, is fundamental for many clinical applications, including diagnosis, disease monitoring, and treatment planning.

In recent years, deep learning has demonstrated great capabilities in extracting valuable data from medical images. Deep learning models, trained on large datasets, can recognize complex patterns and features that may not be easily recognizable to the human eye. These algorithms can even provide a new perspective about what image features should be valued to support decisions. One of the main advantages of using this AI system is its ability to improve the accuracy and efficiency to find disease. This can help healthcare professionals detect abnormalities, identify specific structures, and predict disease outcomes by analyzing medical images with speed and precision, helping to identify early-stage diseases that may be difficult to detect using traditional methods.

In this report, we perform a critical evaluation and comparison of two methodologies for medical image segmentation: U-Net and an approach combining ResNet and FPN (Feature Pyramid Network). In practice, UNet is a convolutional neural network architecture that has a U-shaped encoder-decoder network architecture, which consists of four encoder blocks and four decoder blocks that are connected via a bridge. ResNet+FPN instead integrates the strengths of ResNet that is famous for its deep architecture made trainable by residual connections and FPN that are intended to help detect objects in different scales, generating multiple feature map layers, combining low-level (high-res) and high-level (low-res) features to enhance the detection of tumors at various sizes, preserving both local details and global context.

2 State of the Art

Image segmentation is a fundamental operation in computer vision. It consists of dividing an image into several parts or regions that belong to the same class. This grouping is based on specific criteria, for example, color or texture.

2.1 Types of Image Segmentation

There are three main types of image segmentation, each addressing different aspects of scene understanding:

- **Semantic segmentation:** Classifies each pixel in an image into a specific category or class, without distinguishing between different instances of the same class.
- **Instance segmentation:** Similar to semantic segmentation, but also distinguishes between different instances of the same class, providing a unique label for each individual object in the image.

- **Panoptic segmentation:** Combines both semantic and instance segmentation. It provides a comprehensive understanding of the visual scene by assigning a category label to each pixel and a unique instance ID to each individual object, regardless of the class.

2.2 Traditional Methods vs Deep Learning

Traditional methods, which rely on classic image processing techniques such as thresholding and edge detection, have long been the foundation for segmenting medical images. These methods are straightforward and efficient, but they often struggle with the complexity and variability of medical images.

Deep learning techniques, especially convolutional neural networks (CNNs), have revolutionized medical image segmentation. Deep learning models can automatically learn patterns and features from data, improving segmentation accuracy and adaptability across different types of medical images. While deep learning methods require more computational resources and may be prone to overfitting, they offer flexibility and scalability that traditional methods cannot match.

Neural networks that perform segmentation typically use an encoder-decoder architecture. The encoder extracts features of an image through narrower and deeper filters, the decoder then selects the feature map from the latent space rich in semantic meaning and constructs the segmentation mask.

2.3 Deep Learning Architectures for Segmentation

Several deep learning models are developed to perform segmentation. Notable examples include:

- **U-Net:** A former SOTA but still used, it is a fully convolutional network primarily proposed for medical image segmentation (e.g., tumor detection in lungs and brain). Architecture details are in Section 3.1. Variants include: Attention U-Net, U-Net++, and Dense U-Net.

- **SegNet:** A deep fully convolutional network designed for semantic pixel-wise segmentation. Its decoder uses pooling indices from the encoder for non-linear upsampling, eliminating the need for learning to upsample. SegNet is mainly used for scene understanding applications.

- **DeepLab:** A CNN architecture that uses atrous (dilated) convolutions for upsampling, reducing computation cost while capturing more context. It concatenates features from multiple convolutional blocks before deconvolution.

2.4 Foundation Models for Segmentation

Foundation models have also been used for image segmentation. They divide an image into distinct regions or segments. Nowadays, Foundation models are typically based on transformer architectures, trained on massive datasets to perform a wide range of tasks:

- **Segment Anything Model (SAM):** One of the most important foundation model for image segmentation is SAM (Segment Anything Model), specifically designed for the task of promptable segmentation. Instead of processing text, SAM processes images and generates segmentation masks based on user-provided prompts, fig. 1.

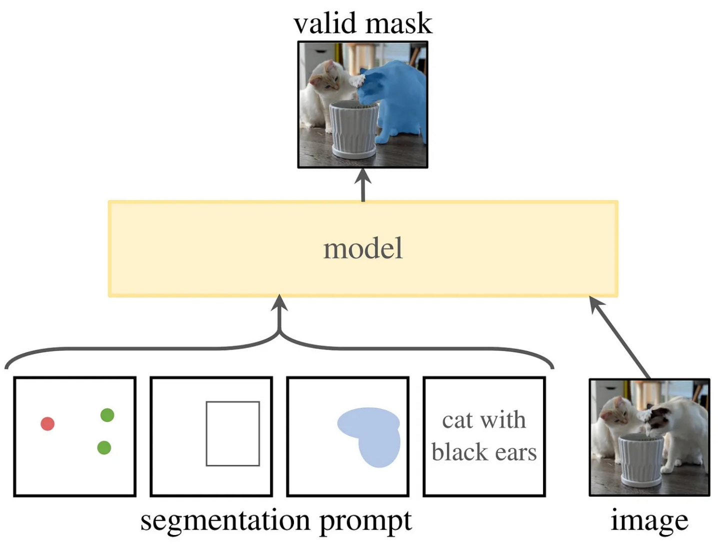


Figure 1: SAM Example

3 Limitations of Deep Learning Methods

While deep learning methods have significantly advanced medical image segmentation, they present several limitations:

- **Data requirements:** Deep models typically require large amounts of labeled data for training. In the medical field, obtaining such data can be challenging due to privacy concerns, labeling effort, and the need for expert annotations.
- **Computational resources:** Training deep models can be computationally intensive, demanding powerful hardware and extended training times.
- **Generalizability:** Models may not generalize well to new datasets or tasks beyond the training distribution (domain shift). That can lead to overfitting and can be resolved with data augmentation, dropout and batch normalization.
- **Segmentation accuracy:** Despite progress, challenges persist in achieving high accuracy, especially with small, low-resolution datasets.
- **Clinical requirements:** Inaccurate segmentation may not meet stringent clinical standards, limiting real-world applicability.

4 Objectives

The primary objective of this project is to evaluate and compare two methodologies for medical image segmentation: U-Net and ResNet+FPN(Feature Pyramid Network). By conducting an analysis, we aim to understand their relative strengths, weaknesses, and performance across the 2D brain tumor segmentation dataset. Our evaluation will be based on four key metrics:

1. **Dice Metric:** Dice coefficient to quantify overlap between predicted masks and ground truth.

2. **Precision:** Measure the false positive prediction value.

3. **Recall:** Measure the sensitivity of the model to false negative.

4. **Lesion Recall:** Measures detection rate of individual tumors.

5. **Time Inference:** The time it takes a trained deep learning model to process new input data and generate predictions.

By analyzing these metrics, we aim to determine which model performs better in terms of both computational efficiency and segmentation quality. These findings will guide practitioners in selecting the most suitable approach for specific medical image segmentation tasks.

5 Methodology

5.1 U-Net

The U-Net architecture is a widely used deep learning architecture for semantic segmentation, particularly in the field of biomedical image segmentation. It follows an encoder-decoder cascade structure, where the encoder gradually compresses information into a lower-dimensional representation, and the decoder decodes this information back to the original image dimension. The two paths are more or less symmetric, and so the architecture gets an overall U-shape, which leads to the name U-Net. The main properties of this architecture are:

- U-Net learns segmentation in an end-to-end setting.

- It requires very few annotated images (approx. 30 images for application in general) using augmented training data with deformations to use the available annotated samples more efficiently. This leads also to efficient training considering only a smaller amount of data while maintaining speed and accuracy, making UNet useful in the medical field where annotated data can be limited.

U-net architecture (example for 32x32 pixels in the lowest resolution). Considering fig. 2, each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The image shape is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations. The main parts are:

https://www.kaggle.com/datasets/nikhilroxtomar/brain-tumor-segmentation/data

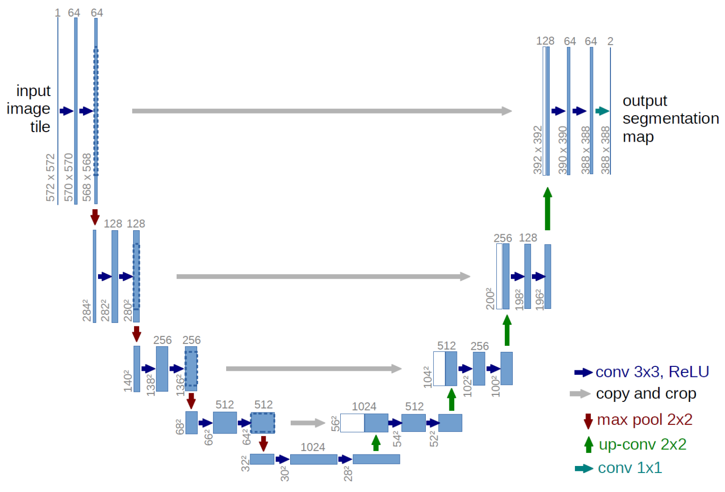


Figure 2: Unet architecture

- **Contracting path:** Repeated 3x3 convolutions followed by 2x2 max-pooling (with stride 2) to down-sample and capture global context.
- **Expansive path:** Upsampling via 2x2 up-convolutions, concatenation with corresponding encoder features (skip connections), and two 3x3 convolutions, enabling precise localization.
- **Final layer:** A 1x1 convolution maps feature vectors to class probabilities per pixel, yielding a binary segmentation map for foreground and background.

5.2 ResNet and FPN (Feature Pyramid Network)

5.3 Introduction

In general, a deep learning model for computer vision problems, like image classification, object detection, semantic segmentation, etc., consists of three components: backbone, neck, and heads as shown in fig. 3.

The backbone network extracts features from the input image and produces feature maps at different levels of resolution. The feature maps at the lower level contain more accurate spatial information due to their high resolution while the feature maps at the higher level have richer semantic information due to their large receptive field. The neck network fuses the features from all the levels to enhance features with both higher accuracy and richer semantic meaning. Based on the enhanced features fused by the neck network, various head networks are used to perform different tasks. For example, for object detection, classification head and regression head are normally used for classifying the object classes and regressing the object bounding boxes.

5.4 ResNet

The commonly used ResNet architectures include ResNet18, ResNet-34, ResNet-50, ResNet-101, and ResNet-152, each of which has different residual block structure and different number of residual blocks, as shown in Fig. 4.

As shown in Fig. 4, each of five ResNet architectures consists of five convolutional layers (conv1, conv2-x, conv5-x), an average pooling layer, a fully connected layer and

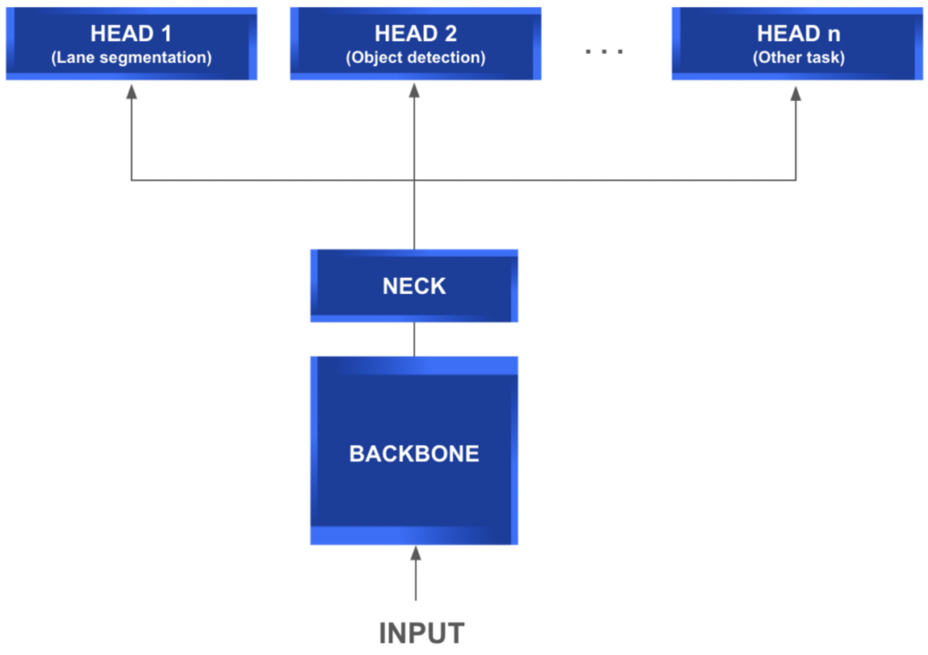
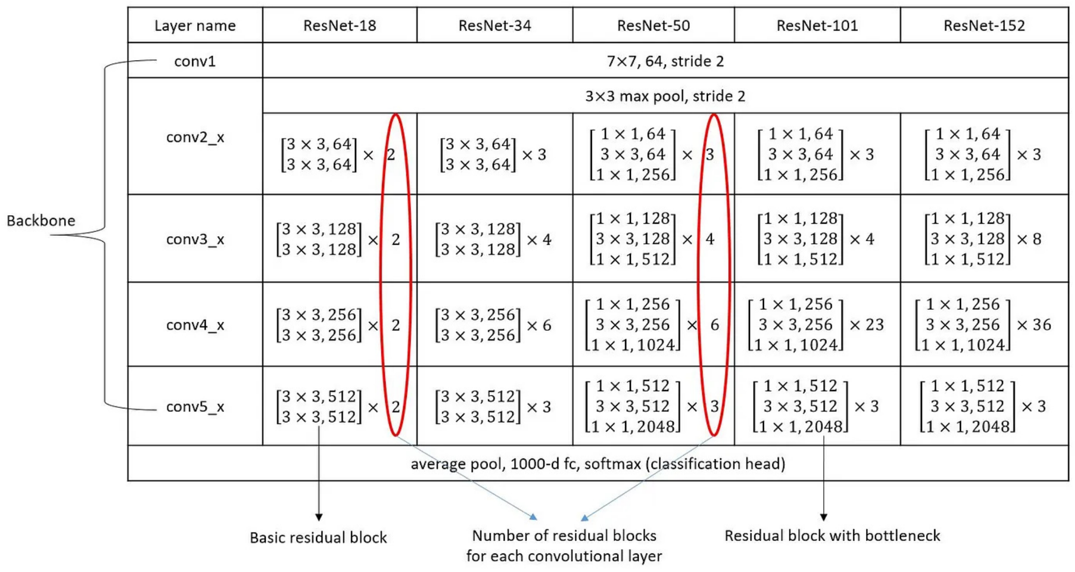


Figure 3: The deep learning network consisting of backbone, neck, and prediction heads.

softmax. The five convolutional layers are the backbone network, as I discussed above. ResNet was first developed for image classification on the ImageNet dataset [2]. The average pooling layer, fully connected layer, and softmax together form the classification head for 1000 object classes of the ImageNet dataset. The backbone networks of ResNet, pretrained using ImageNet, are widely used to extract features at different levels of resolution (feature maps after conv2-x, conv3-x, conv4-x, and conv5-x). Those features are followed by particular neck and heads and fine-tuned for particular tasks, like object detection, semantic segmentation, etc. The five ResNet architectures use the residual block as the basic building block. Each of the convolutional layers from conv2-x to conv5-x uses different number of the residual blocks.


Backbone
Basic residual block
Number of residual blocks for each convolutional layer
Residual block with bottleneck

Layer name	ResNet-18	ResNet-34	ResNet-50	ResNet-101	ResNet-152
conv1	7×7, 64, stride 2				
conv2_x	3×3 max pool, stride 2				
	$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1,64\\3\times3,64\\1\times1,256\end{bmatrix}\times 3$
conv3_x	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3,128\\3\times3,128\end{bmatrix}\times 4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times 4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times 4$	$\begin{bmatrix}1\times1,128\\3\times3,128\\1\times1,512\end{bmatrix}\times 8$
conv4_x	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3,256\\3\times3,256\end{bmatrix}\times 6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times 6$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times 23$	$\begin{bmatrix}1\times1,256\\3\times3,256\\1\times1,1024\end{bmatrix}\times 36$
conv5_x	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times 2$	$\begin{bmatrix}3\times3,512\\3\times3,512\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times 3$	$\begin{bmatrix}1\times1,512\\3\times3,512\\1\times1,2048\end{bmatrix}\times 3$
	average pool, 1000-d fc, softmax (classification head)				

Figure 4: ResNet architectures.

The structure of the residual block used by ResNet-18 and ResNet-34 is different from the one used by ResNet-50, ResNet-101, and ResNet-152, as shown in Fig. 5. Starting from ResNet-50, the bottleneck structure is used to reduce the model parameters by alternating 1x1 and 3x3 convolutions. As shown in Fig. 5, compared to two convolutions of 3x3, 64 and 3x3, 256, the number of model parameters are reduced from 294912 to 69632. Another advantage of the bottleneck structure is that more activations are introduced due to more convolutions. Accordingly, the model capacity is increased.

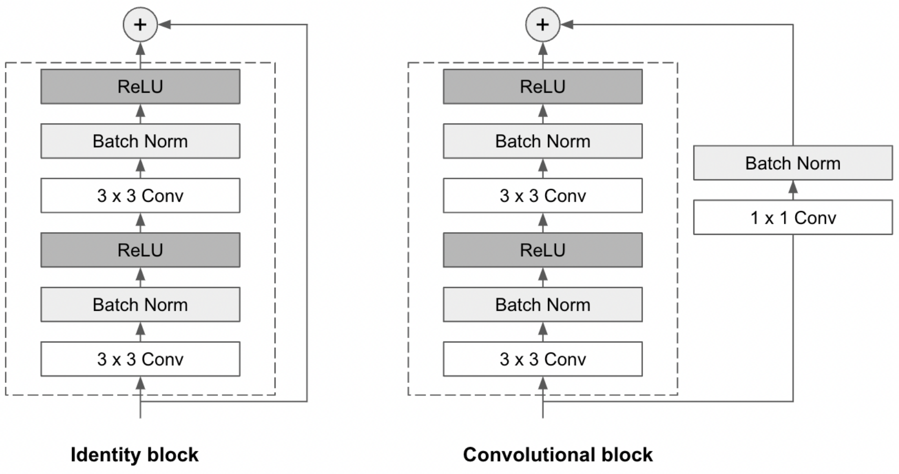


Figure 5: Residual blocks

ResNet extracts features at decreasing levels of resolution from conv2-x to conv5-x. In particular, s2 (stride=2) is used in the first 3x3 convolutions of the first residual blocks for conv3-x, conv4-x, and conv5-x to downsample feature maps to get larger receptive field. Therefore, special care is given to those residual blocks to match spatial dimension of feature maps and number of channels in order to perform the add operations for the skip connections. Figure 5 shows the skip connections go through 1x1 convolutions with s2 in the first residual blocks of conv3-x layers used in ResNet 18, 34, and ResNet 50, 101, 152, respectively, to match spatial dimension of feature maps and number of channels for the add operations of the skip connections.

5.5 Feature Pyramid Network (FPN)

Feature Pyramid Network (FPN), is a neck network that fuses features with various levels of resolution obtained by a backbone network, like ResNet. As we know, a CNN (Convolution Neural Network) based backbone applies a series of convolution layers to an input image to obtain a set of feature maps with decreasing levels of resolution due to downsampling caused by pooling or convolution with a stride more than one. The low resolution feature maps capture more global information of the image and represent richer semantic meaning while the high resolution feature maps focus more on the local information and provide more accurate spatial information. The goal of FPN is to combine the high and low resolution feature maps to enhance the features with both accurate spatial information and rich semantic meaning. Figure 6 shows how FPN works. The FPN includes a bottom-up pathway and a top-down pathway. In the bottom-up pathway, a backbone network, like ResNet, is used to perform feature extraction to extract features with decreasing levels of spatial resolution. As the levels of resolution decreases, the semantic meaning of the feature maps increases, as indicated by the thickness of box boundaries in blue. In the top-down pathway, the feature maps are fused to have both rich semantic meaning and accurate spatial information. As shown in Fig. 6, the feature map with the highest resolution has the same level of semantic meaning as the feature map with the lowest resolution. To construct the feature map at the current level, the feature map at the previous level needs to first perform 2x upsampling (either using bilinear or nearest neighbor interpolation), and then perform element-wise addition with the corresponding feature map in the bottom-up pathway by a lateral connection. Since different levels of feature maps may have different number of channels, 1x1 convolution is used in the lateral connection to unify the number of channels of both feature maps that are used to perform the element-wise addition.

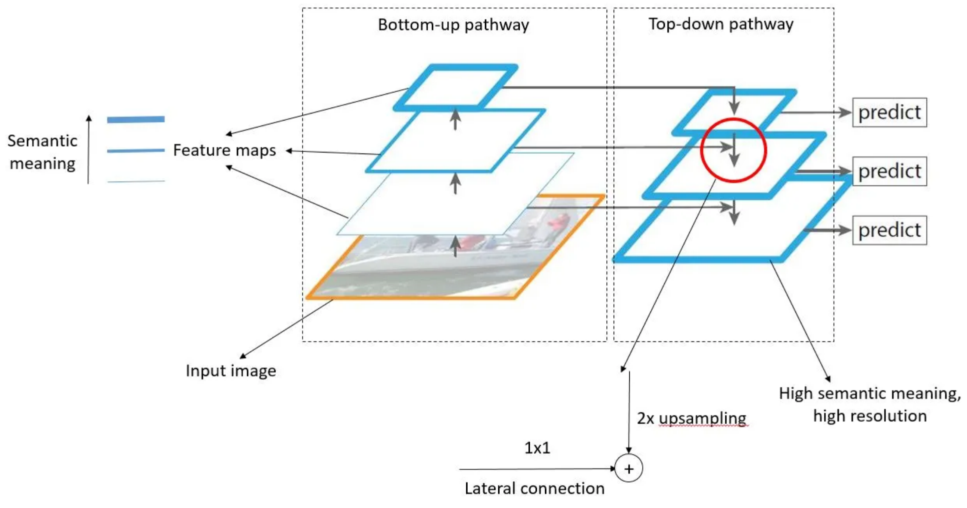


Figure 6: Feature Pyramid Network (FPN).

Due to its strength of constructing feature maps with both rich semantic meaning and high spatial accuracy, FPN is widely used in many computer vision problems: (a) it helps detect small objects for object detection problem, (b) it helps refine the segmentation boundaries for segmentation problem, etc.

5.6 Fusing Features from ResNet

Figure 7 shows an example of combining ResNet-50 and FPN to obtain enhanced features P2-P6. A 1x1 convolution is first applied to each feature map from C2-C5 output by ResNet-50, which unifies the number of channels of those feature maps to 256. Then the unified feature maps from C5 to C3 are upsampled by 2 and element-wise added to the unified feature maps at the next levels, i.e., C4-C2. Another 3x3 convolution is augmented to each of the resulting feature maps from C5 to C2 to provide the outputs from P5 to P2. The output P6 can be obtained by performing the max-pooling operation with kernel size 1 and stride 2 on P5. The outputs P2-P5 will be used in various prediction heads.

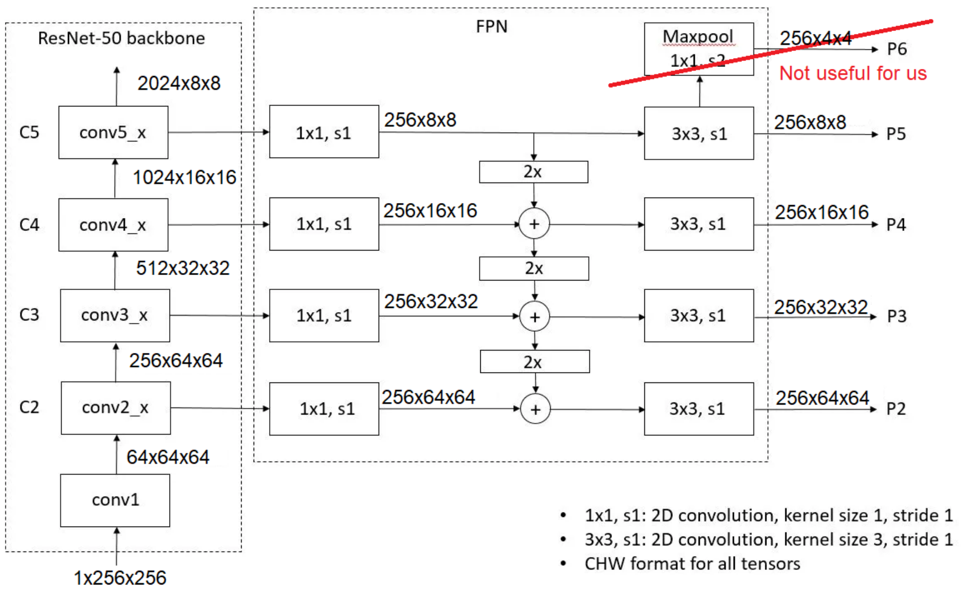


Figure 7: Example of combining ResNet-50 and FPN.

6 Turning Classic FPN into Semantic FPN

In object detection settings, ResNet+FPN has proven effective at combining localization and semantics. Here, we adapt this paradigm to semantic segmentation by:

- **Omitting coarse P6:** Unlike detection, segmentation prioritizes spatial detail over ultra-coarse context, so we drop the P6 level.
- **Using a shared segmentation head:** Replace classification/regression heads with a single decoder head for per-pixel labeling.
- **Concatenating multi-scale features:** Rather than per-level heads, we up-sample P3–P5 to P2 resolution and concatenate before decoding, ensuring unified global context and local detail.

To convert a ResNet+FPN we need to retain the backbone and FPN, remove all RPN/ROI heads, and attach a per-pixel decoder on top of the pyramid. The minimal recipe is:

1. **Backbone + FPN remains unchanged.**
   We still produce pyramid features
   $$P_2, P_3, P_4, P_5 \in \mathbb{R}^{256\times H/2^i \times W/2^i} \quad (i = 2, 3, 4, 5),$$
   ignoring P_6 for per-pixel tasks.
2. **Attach a small conv-head to each pyramid level.**
   For each S_i, apply a 3×3 convolution:
   $$P_i = \text{Conv}_{3\times 3}(S_i), \quad i = 2, 3, 4, 5.$$
3. **Interpolate all P_i to a common high resolution.**
   Use the finest level P_2 grid ($H/4 \times W/4$) and upsample P_3, P_4, P_5 so that
   $$\tilde{P}_i = \text{Upsample}(P_i;\ H/4,\ W/4), \quad i = 3, 4, 5.$$
   or
   $$\tilde{P}_i = \text{Interpolate}(P_i;\ H/4,\ W/4), \quad i = 3, 4, 5.$$
4. **Concatenate and decode.**
   concatenate them:
   $$C = \text{Cat}(P_2, \tilde{P}_3, \tilde{P}_4, \tilde{P}_5).$$
   Then apply two 3×3 convolutions (1024→256) and a final 1×1 conv to get 1 channel:
   $$\text{logits} = \text{Conv}_{1\times 1}^{256\to 1}(F).$$
5. **Upsample to full image size.**
   Bilinearly upsample the logits from $H/4 \times W/4$ back to $H \times W$, then apply sigmoid.

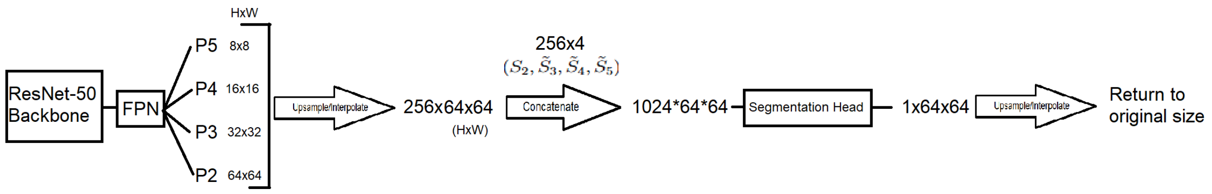


Figure 8: FPN pipeline with shapes explanation

6.1 ResNet-50 Backbone (C2–C5)

Building upon ResNet-50 pretrained on ImageNet, we have:

- Conv1 + BN + ReLU + MaxPool: Reduces resolution to 1/4 of the input size, forming the first stage.

- Layer1 (C2): 256 channels at stride 2.
- Layer2 (C3): 512 channels at stride 2.
- Layer3 (C4): 1024 channels at stride 2.
- Layer4 (C5): 2048 channels at stride 2.

Each stage captures progressively coarser spatial detail but richer semantics, laying the foundation for our FPN neck.

6.2 FPN Neck: Multi-Scale Feature Concatenation

The Feature Pyramid Network (FPN) constructs a top-down pathway with lateral connections to merge C2–C5 into a unified pyramid. In many object detection variants, an additional P6 level is produced by max-pooling P5 to facilitate region proposal at very coarse scales. However, for semantic segmentation where fine-grained detail is critical and extremely coarse features may lose spatial resolution we omit P6 and focus on P2–P5:

1. **Lateral 1×1 Convs:** Reduce each Ci to 256 channels.
2. **Top-Down Upsampling & Addition:** For i=5 to 2, upsample and add lateral features, then smooth with 3×3 conv.
3. **Pyramid Outputs:** P2–P5 have a resolution that is halved from the previous pyramid output, for example the output tensor P3 (32x32) have a resolution that is the half of P2 (64x64).

This fusion strategy preserves spatial accuracy from high-resolution maps while injecting semantic richness from deeper layers.

6.3 Semantic Segmentation Head

To generate dense, per-pixel class predictions, we attach a single segmentation head to the fused features:

1. Upsample P3, P4, P5 to P2 resolution via nearest-neighbor.
2. Concatenate P2, P3↑, P4↑, P5↑.
3. The fusion_conv_layer, 1x1 conv + BN + ReLU map from 1024 to 256 channels.
4. Bilinear upsampling to original resolution.

7 Experiment

7.1 Dataset

Our models are evaluated on a medical imaging dataset that contains 3064 pairs of MRI brain images and their respective binary mask indicating tumor in .png format. The original dataset is Figshare Brain Tumor dataset a brain tumor dataset containing 3064 T1-weighted contrast-inhanced images from 233 patients with three kinds of brain tumor: meningioma (708 slices), glioma (1426 slices), and pituitary tumor (930 slices).

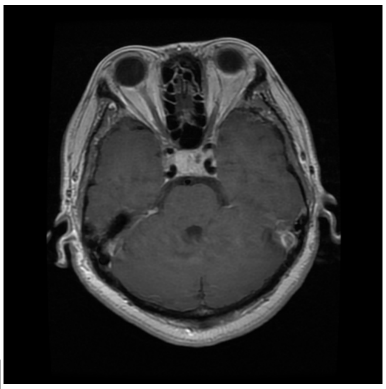
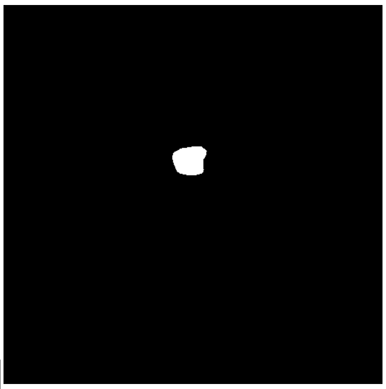
[Image]

[Mask]


Figure 9: Img and Mask side by side

7.2 Data Preprocessing

First, we load all the images from our dataset. If any images is not already in RGB format, we convert it to RGB to ensure consistency in color channels. Next, we do the normalization that ensures that pixel values are scaled appropriately. After normalization, we resize all images to a uniform size of 256x256 pixels. These resized images are then converted to `PyTorch` tensors.
For masks, we follow a simpler process: resizing the mask images to match the desired dimensions (256x256) and converting them to tensors.
Finally, we split our dataset into three subsets: the **training set** (70% of the data), the **validation set** (15% for hyperparameter tuning), and the **test set** (15% for evaluating model performance on unseen examples).

7.3 Loss Function

7.3.1 Cross-Entropy Loss

Cross-entropy Loss is a commonly used loss function in deep learning. It quantifies the dissimilarity between the predicted probability distribution and the actual probability distribution (ground truth) of a set of events or classes.

If there are K classes, and y_i represents the true distribution (a one-hot encoded vector with 1 at the true class index and 0 elsewhere), and $\hat{y}_i$ represents the predicted distribution, the cross-entropy loss formula is defined as follows:

$$CE(y, \hat{y}) = -\sum_{i=1}^{K} y_i log(\hat{y}_i)$$

The goal during training is to minimise this loss, so the model is learning to make its predicted probability distribution ($\hat{y}$) as close as possible to the true distribution (y).

There are two primary types of cross-entropy loss functions:

- *Binary cross-entropy loss*: used with only two possible classes or labels: positive and negative or true and false.
- *Categorical cross-entropy loss*: Used in multi-class classification tasks.

Cross-entropy loss has several advantages, including its ability to penalize confidently wrong predictions more heavily, making it suitable for training models in classification tasks. Additionally, it provides a smooth and differentiable objective function, which is crucial for gradient-based optimization algorithms used in training neural networks.

7.3.2 Focal Loss

In semantic segmentation, where each pixel in an image must be classified, we frequently encounter extreme class imbalance at the pixel level (like 1% tumor vs. 99% healthy tissue) disturbing training. Standard cross-entropy loss exacerbates this issue: even when models correctly classify background pixels with high confidence (e.g., $\mathrm{p_t}$=0.9), these "easy" pixels still generate substantial loss values. When summed across thousands of such pixels, they drown out signals from rare but critical foreground regions.

The focal loss rebalances this dynamic through an adaptive modulating factor. Using binary cross-entropy where loss depends on $\mathrm{p_t}$ (the predicted probability of the correct class) it introduces $(1\text{-}\mathrm{p_t})^{\mathrm{y}}$ to the loss equation. For misclassified pixels ($\mathrm{p_t}$ near 0), this factor remains near 1, preserving full loss impact to ensure the model focuses on hard examples. Conversely, for confident correct predictions ($\mathrm{p_t}\rightarrow 1$), it approaches zero. When γ=0, FL is equivalent to CE, and as γ is increased the effect of the modulating factor is likewise increased. This automatically shifts focus toward ambiguous boundaries and rare classes.

The focal loss is defined as:

$$\mathrm{FL}(p_t) = -(1 - p_t)^{\gamma} \log(p_t)$$

7.3.3 Dice Loss

The Dice Loss, is a loss function commonly used in image segmentation tasks, suitable for tasks where the class of interest occupies a small portion of the image, like in medical image segmentation. The Dice Loss is derived from the Dice coefficient, which is a similarity measure between two sets. In binary image segmentation, the Dice coefficient is given by:

$$Dice(A, B) = \frac{2 \cdot |A \bigcap B|}{|A| + |B|}$$

where A is the set of pixels in the predicted segmentation mask, and B is the set of pixels in the ground truth mask. The Dice coefficient ranges from 0 to 1, where 0 indicates no overlap between the predicted and true masks, and 1 indicates a perfect match.

The Dice loss is then defined as the complement of the Dice coefficient to ensure it behaves as a loss function:

$$DiceLoss(A, B) = 1 - Dice(A, B)$$

This loss function is designed to be minimized during the training of a segmentation model. The goal is to encourage the model to produce segmentation masks that have a high overlap with the ground truth.

In the context of deep learning, the Dice loss is often applied to each class independently in multi-class segmentation problems. The overall loss for a model predicting K classes is the sum of the Dice losses for each class.

Dice loss is popular in medical image segmentation tasks where imbalances between foreground and background classes are common. It helps address issues associated with class imbalance by focusing on the relative overlap between predicted and true masks rather than absolute pixel-wise differences.

7.3.4 Combined Loss

The DiceFocalLoss in my implementation is a hybrid loss that compute both Dice loss and Focal Loss and return the weighted sum of these two losses. It can be expressed as:

$$DiceFocalLoss(A, B) = \alpha \cdot Dice(A, B) + \beta \cdot FocalLoss(A, B)$$

where $Dice(A, B)$ is the Dice loss, $FocalLoss(A, B)$ is the Focal variant of Binary Cross-Entropy, and α, β are the weighting parameters that control the influence of each loss term.

The Dice loss component encourages accurate pixel-wise predictions and is effective in handling class imbalance. On the other hand, the Focal loss component (derived from Binary Cross-Entropy) introduces a focusing mechanism that down-weights well-classified examples, intensifies learning on hard misclassified pixels and handles extreme class imbalance than standard Cross-Entropy.

The choice of α and β depends on the segmentation task's specific requirements and the dataset's characteristics.By adjusting these parameters, one can prioritize mask overlap accuracy (via α-weighted Dice loss) or emphasize correction of hard misclassified pixels (via β-weighted Focal loss).

This loss is particularly useful when dealing with imbalanced datasets or when there is a need to strike a balance between capturing fine details in the segmentation masks and ensuring a high level of overall segmentation accuracy. It combines the strengths of both Dice and Focal losses to provide a more flexible and adaptive loss function for training segmentation models.

7.4 Training

Both UNet and ResNet+FPN were trained for a total of 5 times with a randomly generated seed to gather valid statistic results. For every run we set 80 epochs with the values of Scheduler LR, and LR are respectively set to 1e-5 and 1e-4.

7.4.1 Dropout

For a better training a dropout regularization is employed within the models architecture to mitigate overfitting and enhance model generalization. By selectively dropping units during training, dropout encourages the network to learn robust features and prevent reliance on specific nodes, thereby enhancing its capacity to generalize across unseen data instances.

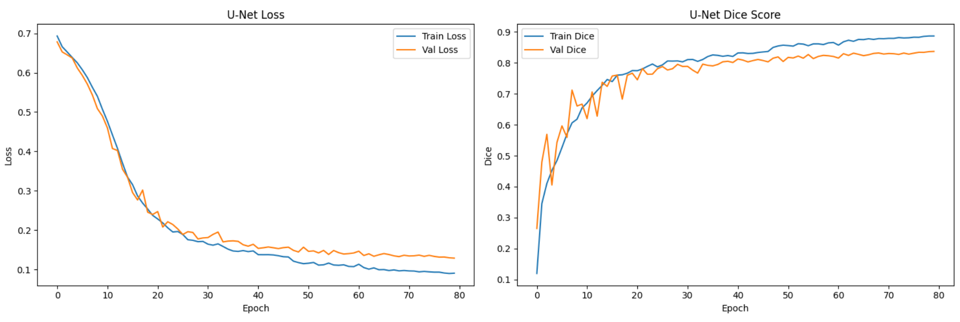


Figure 10: Unet curves (seed 49913)

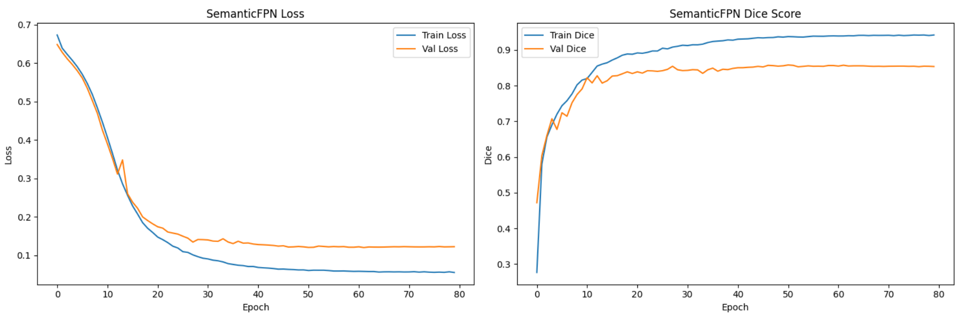


Figure 11: SemanticFPN curves (seed 49913)

From pervious figure it can be deduced that all the losses manage to converge with the number of established epochs.

8 Results & Conclusions

8.1 Results

To ensure robustness and account the training stochasticity, we conducted each experiment 5 times, with a fixed randomly generated seed and subsequently averaging the results for every run composed by 80 epochs. This approach not only provides a more reliable estimate of model performance but also mitigates the effects of random initialization and other sources of variability inherent in neural network training. By repeating the experiments multiple times and aggregating the outcomes, we aimed to attain a comprehensive understanding of the comparative performance between the U-Net and ResNet+FPN architectures.

The results of the comparative evaluation between the U-Net and ResNet+FPN architectures are summarized in Table 1. The evaluation metric used to assess model performance is the Dice coefficient, a common measure of segmentation accuracy.

Metric	**U-Net (mean $\pm$ std)**	**SemanticFPN (mean $\pm$ std)**
Dice	0.8427 ± 0.0031	$\mathbf{0.8613 \pm 0.0025}$
Precision	0.8623 ± 0.0075	$\mathbf{0.8836 \pm 0.0069}$
Recall	0.8240 ± 0.0091	$\mathbf{0.8401 \pm 0.0045}$
Lesion Recall	0.9552 ± 0.0070	$\mathbf{0.9717 \pm 0.0022}$
Inference Time (ms)	$\mathbf{6.5036 \pm 0.2663}$	12.419 ± 1.300

Table 1: Performance comparison of U-Net and SemanticFPN: mean and standard deviation calculated on 5 experiments.

Performance Comparison SemanticFPN demonstrates a clear advantage over U-Net in segmentation quality across all evaluated metrics, while U-Net retains its edge in computational efficiency. On average, SemanticFPN achieves a Dice coefficient of 0.8613(±0.0025) compared to U-Net's mean Dice of 0.8427(±0.0031), indicating that SemanticFPN produces masks that more closely match ground-truth tumor delineations and does so with greater consistency across runs. This superiority in overlap accuracy is complemented by a higher precision of 0.8836(±0.0069) for SemanticFPN versus 0.8623(±0.0075) for U-Net, suggesting that SemanticFPN is better at avoiding false positive segmentations of healthy tissue.

When it comes to sensitivity, SemanticFPN again outperforms U-Net: its recall averages 0.8401(±0.0045), while U-Net records 0.8240(±0.0091). The lesion recall further amplifies this trend, with SemanticFPN reaching an average of 0.9717(±0.0022) against U-Net's 0.9552(±0.0070). SemanticFPN thus not only segments tumors more accurately overall but also exhibits a superior capacity to identify individual tumor lesions, with less risk of missing small or faintly defined regions.

On the other hand, U-Net maintains a significant advantage in inference speed. Its average time per image is only 6.50ms(±0.27ms), compared to 12.42ms(±1.30ms) for SemanticFPN. The smaller standard deviation in U-Net's inference time underlines its more predictable performance in real time applications. In practical terms, the choice between these architectures comes down to the balance between segmentation fidelity and computational efficiency.

Random examples from results Below are some examples for each model, considering both good performance and some predicted masks that don't reflect the ground truth. In the qualitative example, U-Net sometimes misses fine protrusions or under segments structures (fig. 12 second example) compared to SemanticFPN, it may also "exceed" beyond object boundaries like fig. 13 last example but that appen also for SemanticFPN. Sometime U-Net miss entirely like in the fig. 12 last example, but SemanticFPN is able to catch a small portion of it and also map some more that are not in the ground-truth mask.

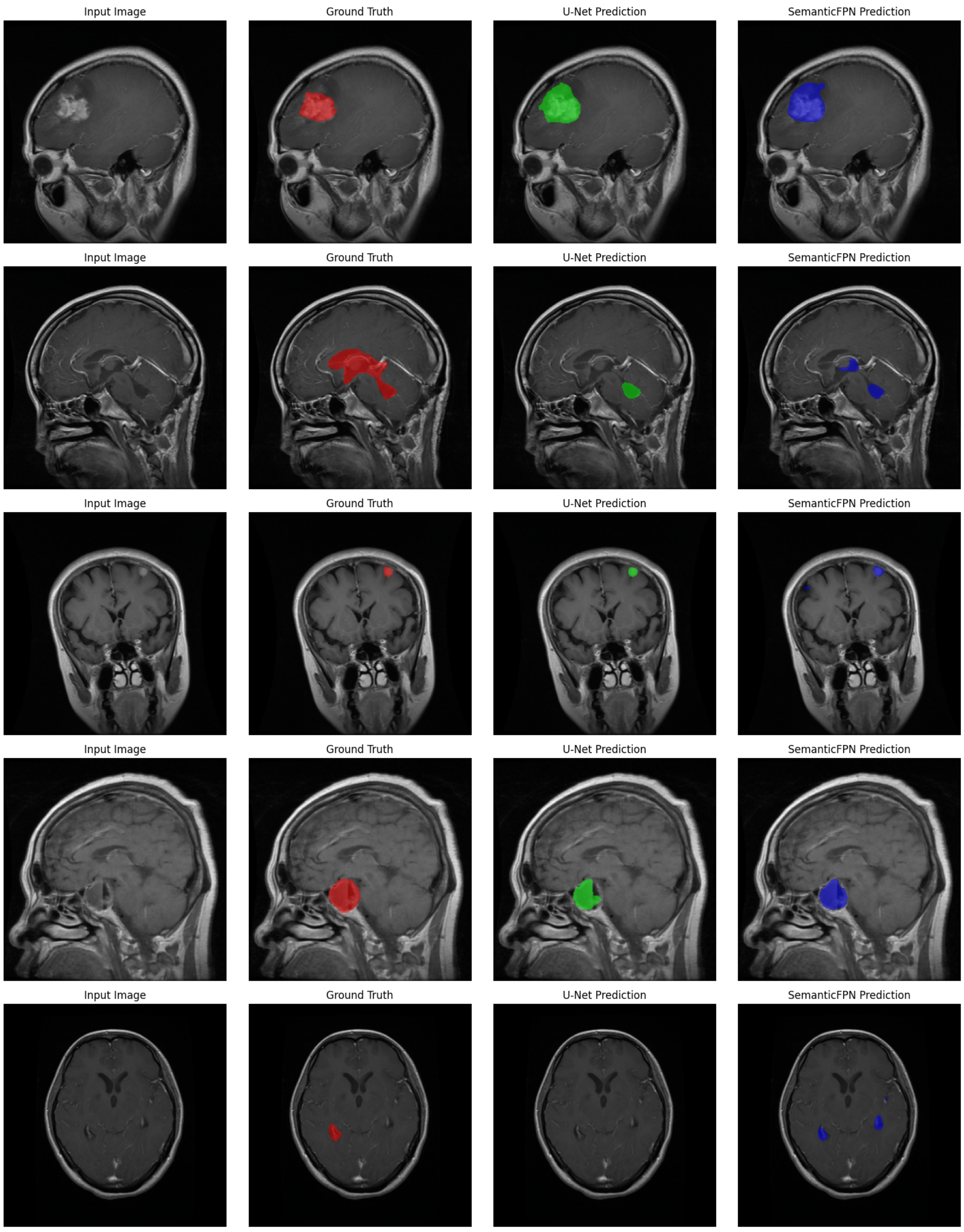


Figure 12: Example for seed 49913

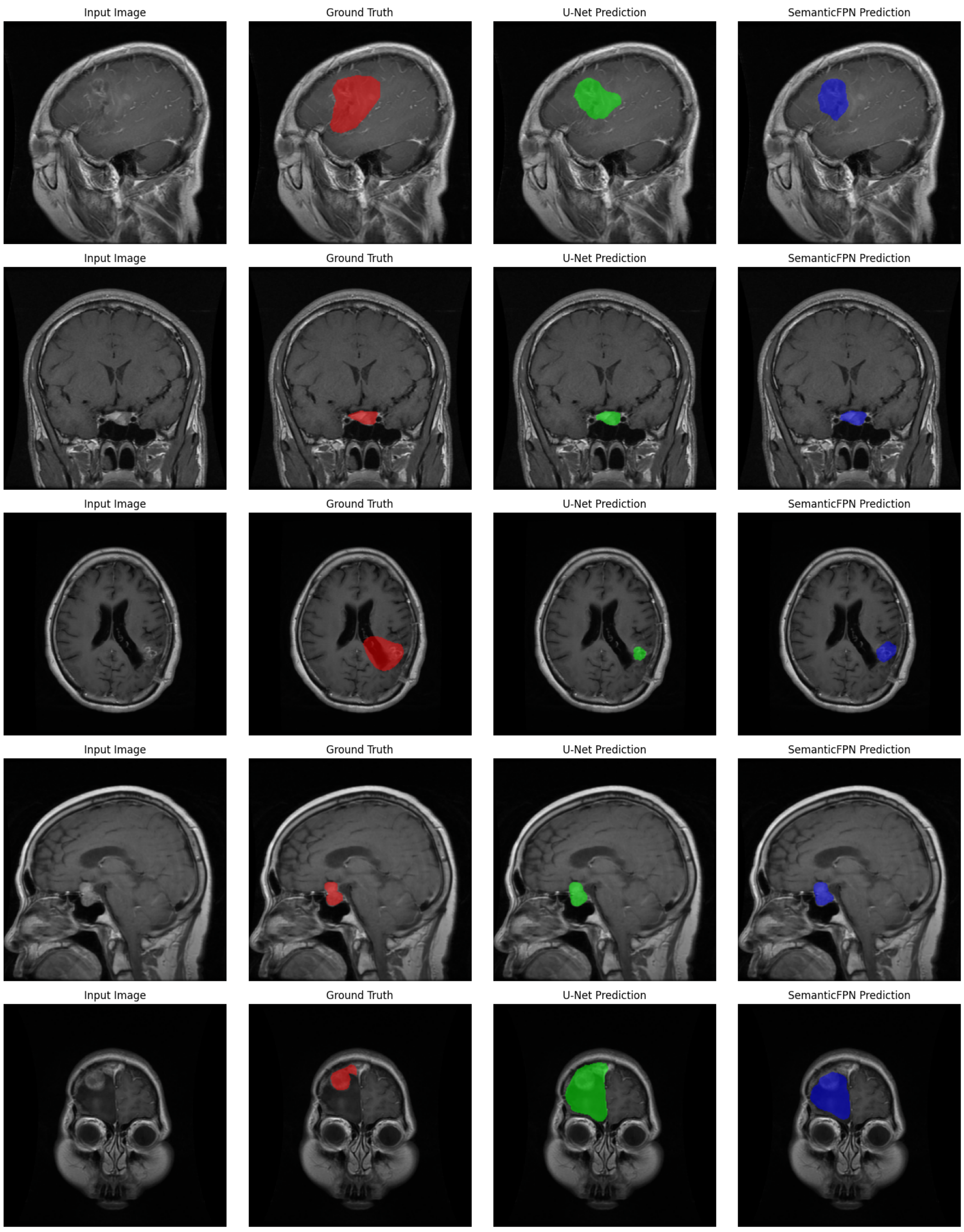


Figure 13: Example for seed 80541

8.2 Conclusion

The analysis highlight that SemanticFPN offers better segmentation performance, with higher metrics on all accuracy indicators, but at the cost of longer inference time. U-Net, while having slightly lower performance, excels on the computational efficiency side. The key of this design is its simplicity: information flows directly from the encoder to the decoder at matching resolutions, preserving spatial detail while still allowing the network to learn increasingly abstract representations. On the other hand in SemanticFPN, starting with a ResNet backbone and maintain 256 channels at every stage, the model can learn very rich, multi-scale context, which leads to higher accuracy on metrics like Dice and precision. However, it also carries more parameters and incurs extra cost from multiple interpolations. Basically, the choice of model will therefore depend on the desired trade-off between segmentation quality and processing speed.

References

[1] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 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*, pages 234–241. Springer, 2015.

[2] Alexander Kirillov, Ross B. Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. *CoRR*, abs/1901.02446, 2019.

[3] Jun Cheng. brain tumor dataset. 4 2017.

[4] Reza Azad. Loss Functions in the Era of Semantic Segmentation: A Survey and Outlook.

[5] Mohd Halim Mohd Noor and Ayokunle Olalekan Ige. A survey on state-of-the-art deep learning applications and challenges, 2025.

[6] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018.

[7] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, pages 3–11. Springer, 2018.

[8] Sijing Cai, Yunxian Tian, Harvey Lui, Haishan Zeng, Yi Wu, and Guannan Chen. Dense-unet: a novel multiphoton in vivo cellular image segmentation model based on a convolutional neural network. *Quantitative imaging in medicine and surgery*, 10(6):1275, 2020.

[9] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023.

[10] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[11] Michael Yeung, Evis Sala, Carola-Bibiane Schönlieb, and Leonardo Rundo. Unified focal loss: Generalising dice and cross entropy-based losses to handle class imbalanced medical image segmentation. *Computerized Medical Imaging and Graphics*, 95:102026, 2022.

[12] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[13] Saeid Asgari Taghanaki, Yefeng Zheng, S. Kevin Zhou, Bogdan Georgescu, Puneet Sharma, Daguang Xu, Dorin Comaniciu, and Ghassan Hamarneh. Combo loss: Handling input and output imbalance in multi-organ segmentation, 2021.