Glioma Segmentation Using U-Net and Latent Feature Interpretation with BERT

Veda Kamaraju (vsk32), Antonio Garces (atg44)
Professor John Zimmerman
Cornell University

Abstract

Glioblastoma multiforme is among the most aggressive primary brain tumors and is associated with a poor prognosis despite advances in surgical and therapeutic interventions. Accurate characterization of tumor extent and morphology from magnetic resonance imaging (MRI) plays a critical role in diagnosis, treatment planning, and longitudinal monitoring. However, manual segmentation and interpretation are time intensive and subject to patient-specific variability, motivating the development of automated and interpretable computational tools. This project develops a machine learning pipeline that automatically segments glioblastoma tumor regions from MRI slices and extracts quantitative morphological descriptors for downstream interpretation. The approach integrates a U-Net convolutional neural network for 3-class pixelwise segmentation (background, edema, enhancing tumor), an improved convolutional autoencoder for learning latent morphological embeddings, and a retrieval-based natural language interpretation module based on BERT embeddings.

The dataset consists of multimodal 2D MRI slices from the BraTS 2020 collection, which are organized in HDF5 format and preprocessed to produce Z-score normalized image tensors and categorical segmentation masks. Slices that contain tumors were filtered using nonzero mask counts, resulting in a clean dataset suitable for supervised learning. The U-Net model was trained using cross-entropy loss and Adam optimization. An autoencoder was then trained on segmentation masks to produce 128-dimensional latent embeddings that capture tumor shape and structural complexity. These numerical features were converted into textual feature statements, encoded with a sentence-transformer model, and matched against a small curated library of clinical-style summaries using cosine similarity.

The trained U-Net generates qualitatively accurate segmentation maps with appropriate localization of edema and enhancing tumor regions. The autoencoder learns smooth latent representations that organize tumors by size and complexity in PCA space. The summary retrieval model successfully maps quantitative features to coherent descriptive statements. Collectively, this multimodal pipeline demonstrates the feasibility of integrating computer vision, representation learning, and natural language processing to analyze glioblastoma morphology. Future work may include full 3D modeling, improved class granularity, and expansion to larger clinical datasets.

Introduction

Glioblastoma multiforme represents one of the most invasive and clinically challenging brain cancers, with median survival near fifteen months and high recurrence rates despite aggressive therapy. MRI serves as the primary imaging modality for diagnosis and treatment planning, and radiologists routinely evaluate edema, contrast-enhancing tumor, and necrotic regions to characterize disease progression. Accurate segmentation of these tumor subregions is therefore critical for volumetric quantification, radiotherapy planning, and surgical decision-making. However, manual segmentation is laborious and varies significantly between patients, motivating the development of automated machine learning tools.

Deep learning approaches, especially U-Net-style convolutional architectures, have become the standard for medical image segmentation due to their ability to combine hierarchical feature extraction with precise pixel-level localization. Prior work on the BraTS dataset has demonstrated strong performance using variants of 2D and 3D U-Nets, attention modules, and hybrid ensemble approaches. Complementary methods have used autoencoders and latent embeddings to characterize tumor morphology, and recent research explores integrating imaging with natural language descriptions to support explainability.

This project deliberately pursues the more challenging of two possible language-integration strategies by incorporating an interpretable language model component for tumor characterization. Rather than attempting to directly generate text from imaging data, which carries risks of hallucination and reduced interpretability, the final design uses BERT-based sentence embeddings to map learned tumor features to pre-defined summary statements. This approach leverages the semantic structure of language embeddings while maintaining full control over clinical phrasing. By combining U-Net segmentation, autoencoder-based latent representations, interpretable geometric features, and retrieval-based language summarization, the project presents a robust and transparent framework for multimodal tumor analysis.

Data Description and Pre-Processing

Initial Dataset

The dataset used in this project consists of multimodal MRI slices from the BraTS 2020 training collection, stored as HDF5 files. Each file contains a single 2D MRI slice of size 240 x 240 x 4, with the last dimension representing the four sequences used for brain tumor imaging: native (T1), post-contrast T1-weighted (T1Gd), T2-weighted (T2), and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR). These "sequences" represent combinations of radiofrequency pulses and magnetic gradients that are designed to highlight various tissue characteristics. The three segmentation classes used in this project are: Background (0), Edema (1), and Enhancing Tumor

(2). Each complete subject contains roughly 155 axial slices, but many of them do not contain tumor tissue, which is very important to account for since they can bias the model toward predicting background.

Preprocessing Steps

Preprocessing choices were guided by both the structure of the BraTS dataset and the clinical nature of MRI data. Z-score normalization was applied to each slice to reduce inter-scan intensity variability arising from different acquisition settings. Additionally, slices without tumor tissue were excluded to prevent the model from learning a trivial background solution, a common issue in medical segmentation tasks with strong class imbalance. This filtering step ensures that the network focuses on clinically relevant regions and improves stability during training.

The preprocessing pipeline implemented in the model's code performs:

1. Slice Extraction: All slices per subject are loaded from HDF5 files using a custom loader
2. Z - Score Normalization

$$x_{norm} = (x - \mu) \div (\sigma + 10^{-8})$$

3. One-Hot to Categorical Conversion

$$y(i,j) = arg\,max\,mask(i,j,k)$$

4. Dataset Merging: slices from multiple subjects are concatenated
5. Tumor Filtering: Slices with fewer than fifty tumor pixels are removed, producing clean final arrays. X_clean represents the MRI slices of shape (N,240,240,4) and Y_clean represents the segmentation masks of shape (N,240,240)

$$\sum_{i,j} 1[y(i,j) > 0] > 50$$

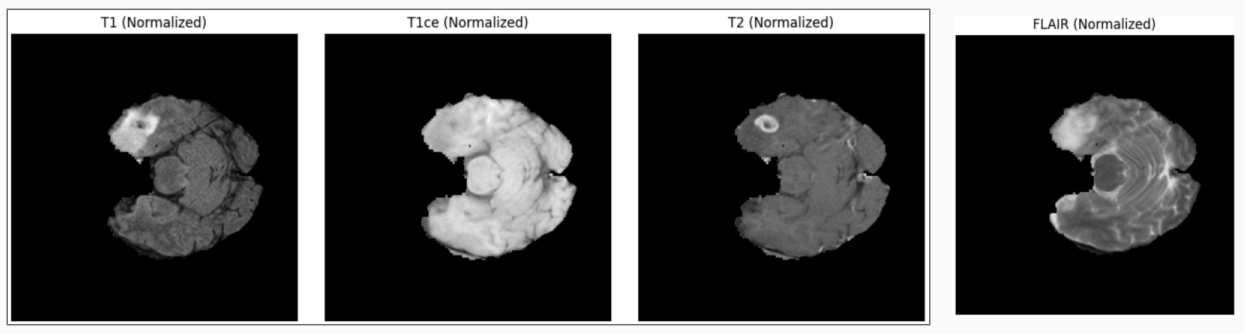


Figure 1: Visualization of normalized sequences in the dataset: T1, T1ce, T2, FLAIR normalized.

Data Representations to the Model

Each MRI slice is passed to the U-Net as a 4 x 240 x 240 tensor matrix. Each corresponding mask is a single channel 2D array with class labels {0,1,2}. This representation is compatible with PyTorch's CrossEntropyLoss, which expects integer-labeled masks.

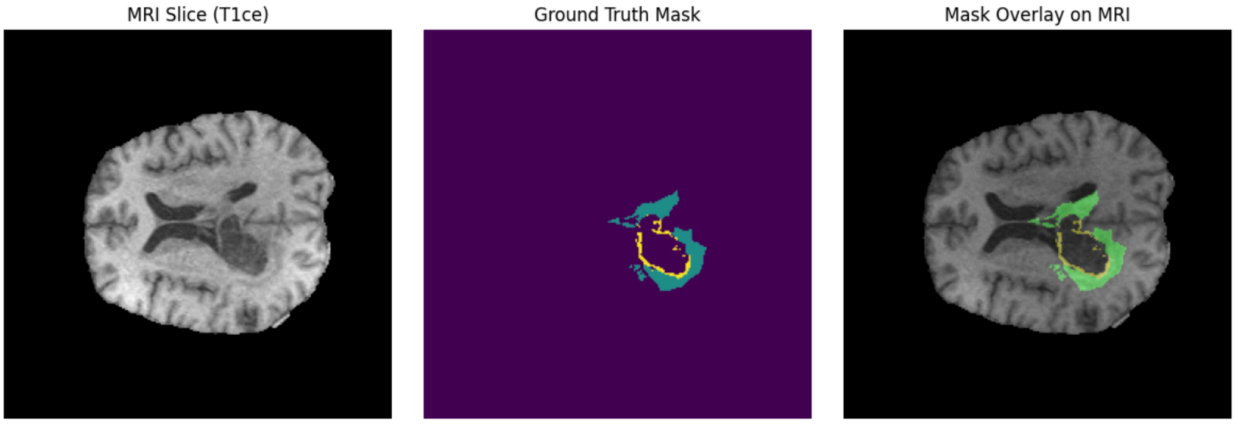


Figure 2: Visualization of MRI and ground truth mask overlay to understand pre-processed dataset slices.

Model Design and Implementation

Our pipeline includes three core pieces that contribute to segmentation and interpretation of the tumor: U-Net, a convolutional autoencoder, and a pretrained BERT model.

U-Net Architecture

The U-Net implements the classical encoder-decoder structure with skip connections. It accepts four MRI modalities and outputs a three-channel logit map corresponding to the three segmentation classes. The Encoder downsamples the data by going through sets of convolution and max pooling followed by a bottleneck.

Encoder:

- DoubleConv(4 → 16), MaxPool
- DoubleConv(16 → 32), MaxPool
- DoubleConv(32 → 64), MaxPool
- Bottleneck: DoubleConv(64 → 128)

This downsampled data goes through the latent space to generate key features. The point of maximum downsampling can be considered the latent space, or "bottleneck", where the most

abstract and information-dense features of the MRI images are represented. The decoder section then takes this compressed information and upsamples it back to the original input size.

Decoder:

- Transposed Conv 128 → 64, concatenate skip, DoubleConv
- Transposed Conv 64 → 32, concatenate skip, DoubleConv
- Transposed Conv 32 → 16, concatenate skip, DoubleConv

While the condensed information is getting upscaled, the model concurrently combines the upsampled feature maps with corresponding feature maps from the encoder via skip connections. This helps restore finer spatial details and produce a final, segmented output.

Output Layer:

- Conv2d(16 → 3, kernel = 1)

All convolutional layers use ReLu activation, and no dropout is included.

U-Net is well suited for biomedical segmentation because:

- Downsampling captures contextual tumor features.
- Upsampling restores spatial resolution for pixel-accurate boundaries.
- Skip connections preserve fine-grained structural details.
- The architecture performs well on small datasets.

The choice of using a 2D U-Net instead of 3D matches the dataset structure and avoids GPU memory limitations.

Training Parameters

- Loss: CrossEntropyLoss
- Optimizer: Adam
- Learning rate: 1×10^{-4}
- Batch size: 4
- Epochs: 10

Autoencoder for Latent Feature Extraction

After segmentation, we trained a convolutional autoencoder to learn a compact representation (latent vector) of each tumor mask. In order to output an embedding-based interpretation instead of mapping image to text generation, we established an autoencoder that compresses the tumor's shape, size, irregularity, and fragmentation into a low-dimensional vector. Due to the importance

of the latent embeddings being wholly representative of the tumor, we trained the autoencoder with 50 epochs.

The encoder features 3 convolutional 2D layers with a latent dimension of 64. The decoder reshapes the linear dimension with 3 ConvTranspose2D layers along with a sigmoid activation function. The training setup involves 50 epochs, MSE loss function, and the Adam optimizer. The autoencoder reconstructs the mask with the 64-D latent embedding that showcases the tumor's morphology.

Integration of BERT

To convert these numerical features into interpretable descriptions, we created a small library of clinically meaningful summary sentences. These summaries ranged from "No tumor detected in this slice" to descriptions of large, irregular, or fragmented tumors.

```
summary_library = [
    "No tumor in this slice.",
    "A small tumor is visible.",
    "A small tumor with uneven edges is visible.",
    "A medium-sized tumor with mostly smooth boundaries is visible.",
    "A medium-sized tumor with some uneven or irregular edges is visible.",
    "A large-sized tumor with mostly smooth boundaries is visible.",
    "A large tumor with some uneven or irregular edges is visible.",
    "A tumor that appears spread out or diffuse is visible.",
]
```

Since BERT is not a generation model, we used a pretrained Sentence-BERT to embed sentences into the vector space. For each tumor slice, we computed cosine similarity between the slice's feature embedding and each summary sentence embedding. The sentence with the highest similarity was selected as the interpreted result.

$$\text{cosine\_similarity}(u, v) = \frac{u \cdot v}{\|u\|\|v\|}$$

Results and Analysis

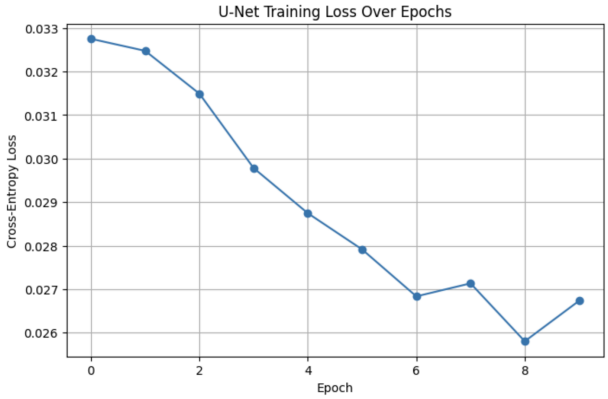


Figure 4: UNet Training Loss over Epochs. Via CrossEntropyLoss, the loss decreased consistently.

The U-Net segmentation model trained successfully over 10 epochs, with training loss decreasing steadily to a final value of 0.0216. This indicates that the model learned to distinguish tumor versus non-tumor regions. Since we are dealing with tumor segmentation patterns and predictions, and then later a reconstruction, we present a visual overlay of the ground truth and prediction.

While the prediction outlined the general tumor location and size, the model struggled with the thin boundaries of the edema. Due to our extended pipeline and limited dataset size, we expected these errors. We plan to expand the dataset pool and attempt the UNet training in 3D to better capture the fine details of the tumor.

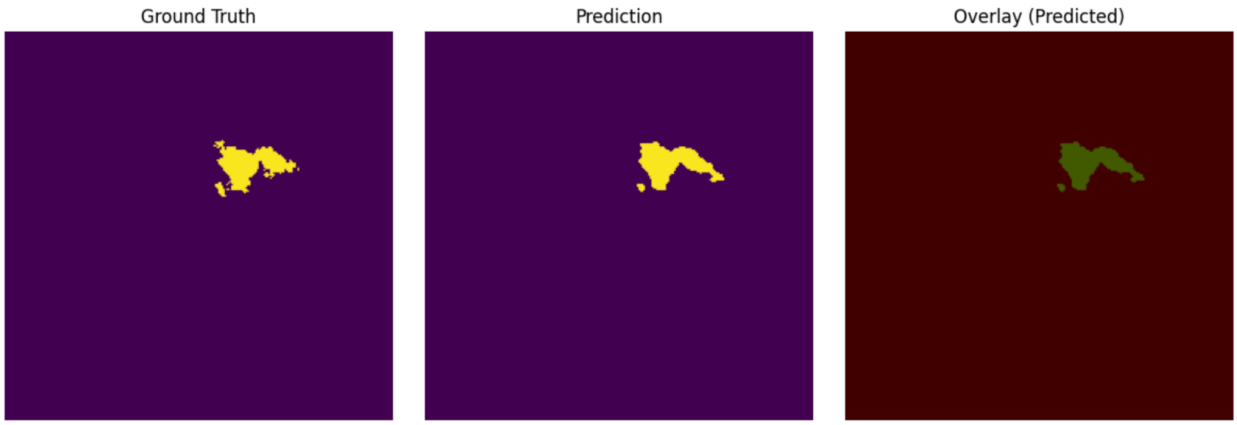


Figure 5: Qualitative comparison of U-Net segmentation performance on a representative MRI slice containing tumor. The left panel shows the ground truth segmentation mask, the middle panel shows the U-Net predicted segmentation, and the right panel shows the predicted mask overlaid on the MRI slice. Edema and enhancing tumor regions are highlighted, demonstrating that the model captures the overall tumor location and extent, while finer edema boundaries remain challenging

We trained the autoencoder with 50 epochs, reducing the loss from Epoch 1/50 | Loss = 0.01770 to Epoch 50/50 | Loss = 0.00212. To visualize the latent space, we created a PCA plot to show clusters that correspond to tumor size and irregularity. The slices with the larger tumors clustered away from the slices with small regions. The clustered PCA showed that we are able to use the latent representations for interpretations.

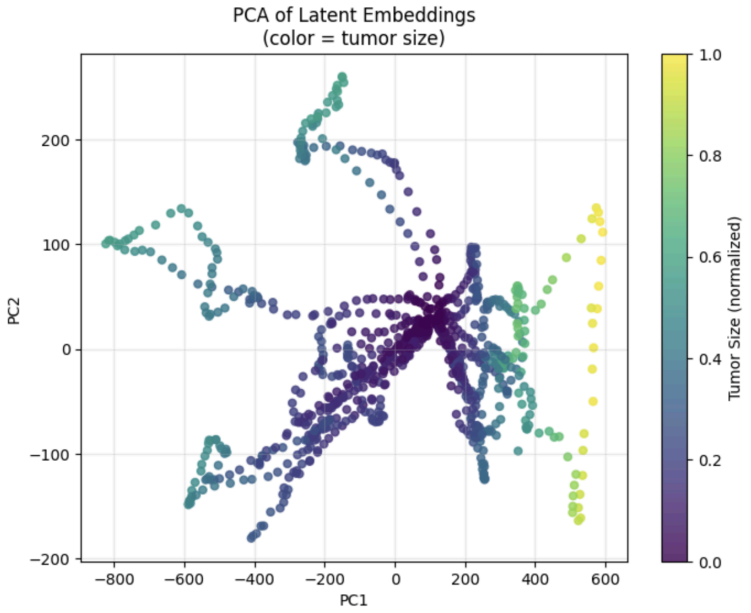


Figure 6: Principal Component Analysis (PCA) projection of 64-dimensional autoencoder latent embeddings learned from tumor segmentation masks. Each point corresponds to a tumor-containing slice, colored by normalized tumor size. Slices with larger tumors cluster farther from those with smaller regions, indicating that the latent space encodes meaningful information related to tumor size and intricate morphology.

To convert these embeddings into interpretable descriptions, we used a BERT-based embedding retrieval system. For each slice, we generated a short description containing tumor size, compactness, and irregularity scores to then encode into Sentence-BERT. There were consistent results in the sentence descriptions. For slices with no tumor, the model selected "No tumor in this slice" across all subjects. For slices with large tumors but irregularity in boundaries, the model selected summaries such as "A large tumor with some uneven or irregular edges is visible." The descriptions aligned with the visual we produced earlier in the pipeline.

```
==============================
Tumor Slice 12 Summary
==============================

Raw Features:
 • Tumor size: 201
 • Complexity: 0.504
 • Compactness: 0.020
 • Latent mean: 0.016

Selected Summary:
No tumor in this slice.
==============================

==============================
Tumor Slice 13 Summary
==============================

Raw Features:
 • Tumor size: 262
 • Complexity: 0.397
 • Compactness: 0.024
 • Latent mean: 0.016

Selected Summary:
A large-sized tumor with mostly smooth boundaries is visible.
==============================
```

Figure 7: Snippet of tumor slice summaries from subject 10. Each summary highlights the features from the latent space in an interpretable written text. The selected summary, based on the similarity scores and the text fed into the model, is written as well.

Discussion and Future Work

An important contribution of this project is the principled integration of language modeling through a retrieval-based framework rather than free-form generation. The project intentionally pursues the more difficult option of incorporating a language model while maintaining interpretability and clinical safety. By using BERT sentence embeddings and cosine similarity to retrieve pre-written summaries, the system avoids common pitfalls associated with generative models, including hallucination and unverifiable medical claims.

The results indicate that the learned autoencoder latent space captures meaningful tumor morphology, as reflected by the alignment between numerical features and selected text summaries. This suggests that the latent embeddings encode semantic structure that is compatible with language embedding spaces, supporting the feasibility of multimodal representation learning in this context.

Several extensions could meaningfully improve both the performance and clinical relevance of this framework. First, segmentation accuracy could be improved by expanding training to full 3D MRI volumes using a 3D U-Net or an attention-based architecture, allowing the model to exploit inter-slice spatial continuity that is lost in a 2D formulation. Additional class granularity, such as separating non-enhancing and necrotic tumor regions, would enable richer morphological characterization and closer alignment with clinical tumor subtyping. From a training perspective, incorporating alternative loss functions such as Dice loss or focal loss could better address class imbalance, particularly for small enhancing tumor regions. The language interpretation module could be further validated by comparing retrieved summaries against expert radiologist annotations or by expanding the curated summary library to capture a broader range of tumor morphologies. Finally, joint representation learning approaches that explicitly align latent visual embeddings with language embeddings may improve semantic consistency between learned tumor features and textual descriptions while preserving interpretability and safety.

AI Supplement:

Due to the complexity of the code, we analyzed the primary parts of the pipeline that benefited our work the most.

Prompt: I'm developing a UNET for the segmentation of tumorous MRI scans. I want 3conv layers per encoder and decoder, and to run 10 epochs. Write the template.

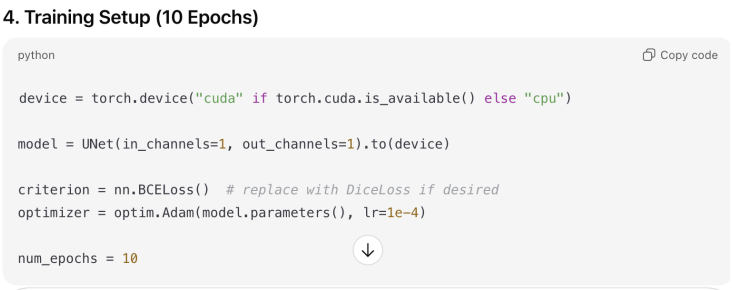
```
        d4 = self.dec4(torch.cat([d4, e4], dim=1))

        d3 = self.up3(d4)
        d3 = self.dec3(torch.cat([d3, e3], dim=1))

        d2 = self.up2(d3)
        d2 = self.dec2(torch.cat([d2, e2], dim=1))

        d1 = self.up1(d2)
        d1 = self.dec1(torch.cat([d1, e1], dim=1))

        return torch.sigmoid(self.out(d1))
```



4. Training Setup (10 Epochs)

```
python

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

model = UNet(in_channels=1, out_channels=1).to(device)

criterion = nn.BCELoss()  # replace with DiceLoss if desired
optimizer = optim.Adam(model.parameters(), lr=1e-4)

num_epochs = 10
```

Analysis: The LLM provided the full template with our requirements. We chose these requirements based on the initial setup we did prior to the presentation. We stuck with 10 epochs due to our focus on the extended pipeline and the computational time for more. However, it still produced Dice Loss, and we are more familiar with CrossEntropyLoss and figured we could interpret it better this way. So, we switched to CrossEntropyLoss.

Prompt: What's the template for an autoencoder with 3 conv layers encoder and decoder

Copy code

```
class Encoder(nn.Module):
    def __init__(self, in_channels=1, latent_channels=256):
        super().__init__()

        self.encoder = nn.Sequential(
            # Conv 1
            nn.Conv2d(in_channels, 64, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(inplace=True),

            # Conv 2
            nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(inplace=True),

            # Conv 3
            nn.Conv2d(128, latent_channels, kernel_size=3, stride=2, padding=1),
            nn.BatchNorm2d(latent_channels),
            nn.ReLU(inplace=True),
        )

    def forward(self, x):
        return self.encoder(x)
```

python Copy code

```
for epoch in range(num_epochs):
    model.train()
    epoch_loss = 0

    for images, _ in train_loader:  # masks not needed
        images = images.to(device)

        optimizer.zero_grad()
        recon = model(images)
        loss = criterion(recon, images)
        loss.backward()
        optimizer.step()

        epoch_loss += loss.item()

    print(f"Epoch [{epoch+1}/{num_epochs}], Loss: {epoch_loss/len(train_loader):.4f}")
```

Analysis: We used the structure and training loop but decided to add an additional layer for more depth. The decoder code used the Sigmoid function, which we didn't change. But we actually prompted the code to add 50 epochs into the code. We ended up using the 50-epochs training loop.

Prompt: Help me build a summary library.

```python
SUMMARY_LIBRARY = [
    {
        "id": "S5",
        "tags": ["large", "focal", "irregular", "heterogeneous"],
        "text": "MRI reveals a large focal lesion with irregular margins and heterogeneou
    },
    {
        "id": "S8",
        "tags": ["diffuse", "poorly defined", "heterogeneous"],
        "text": "MRI reveals extensive pathological changes with poorly circumscribed mar
    }
]
```

Analysis: We just wanted the text, and in order to effectively interpret, we wrote many lines of text for the summary rather than the IDs and tags.

Prompt: How do we extract the tumor features into a definition from the latent space

```python
def tumor_features_to_semantics(f):
    semantics = []

    if f["relative_area"] > 0.15:
        semantics.append("large tumor burden")
    elif f["relative_area"] > 0.05:
        semantics.append("moderate tumor burden")
    else:
        semantics.append("small lesion")

    if f["compactness"] < 0.4:
        semantics.append("irregular boundaries")
    else:
        semantics.append("well-circumscribed margins")

    if f["num_components"] > 1:
        semantics.append("multifocal lesions")
    else:
        semantics.append("focal lesion")

    return semantics
```

Analysis: After prompting, we ended with a simpler version that we understood. We selected just size, complexity, and compactness.

Prompt: describe size, complexity, and compactness into code

```python
def compute_compactness(mask):
    """
    Computes compactness (circularity) of tumor shape.
    """
    labeled = label(mask)
    regions = regionprops(labeled)

    if len(regions) == 0:
        return {"compactness": 0.0}

    total_area = sum(r.area for r in regions)
    total_perimeter = sum(perimeter(r.image) for r in regions)

    if total_perimeter == 0:
        compactness = 0.0
    else:
        compactness = (4 * np.pi * total_area) / (total_perimeter ** 2)

    return {"compactness": float(compactness)}
```

Prompt: How do we take these representations and pass it onto BERT

Use a **Sentence-BERT** model (not raw BERT).

```python
from sentence_transformers import SentenceTransformer, util

bert_model = SentenceTransformer("all-MiniLM-L6-v2")
```

Encode the query:

```python
query_text = tags_to_text(tags)
query_embedding = bert_model.encode(query_text, convert_to_tensor=True)
```

Your summary library should already exist:

```python
SUMMARY_LIBRARY = [
    "MRI shows a small focal lesion with well-circumscribed margins.",
    "MRI reveals a moderate tumor burden with irregular boundaries.",
    "MRI demonstrates a large lesion with high internal heterogeneity."
]
```

Encode once and cache:

```python
summary_embeddings = bert_model.encode(
    SUMMARY_LIBRARY, convert_to_tensor=True
)
```

Analysis: Since we understood the approach at a high level, we prompted the LLM to write the code to take the representations we already created, the summary text we had, and create a definition that prints the tumor's summary for a subject.

Transcript: https://chatgpt.com/share/694325c0-92ec-8012-b416-4882a9598025