Task 4

Task 4

Data Handling and Preprocessing (10 Points)

a. You can focus for now on loading the T1-weighted images and the matching labels.
b. Create a dataloader for the data using PyTorch's Dataloader (or Monai's Dataloader class)
c. Create suitable augmentations for the task to solve. Please note: If you apply transformations to the input data, you should think about if you need to apply any transformation to the label of the image as well.

We didnt used Monai's Dataloader class, because we wanted to use the data in the HPC. The dataset implementation is availabel in the `task4/brats_segmentations/dataloader.py`

`file name: ../task4/brats_segmentation/dataloader.py`

```python
class BraTSDataset(Dataset):
    """
    Custom PyTorch Dataset for BraTS data.
    """

    def __init__(self, base_path: str | os.PathLike, transform=None, limit=None):
        """
        Args:
            base_path (str): Path to the dataset directory.
            transform (callable, optional): Optional transform to be applied on a sample.
        """

        all_patients = list(glob.glob(os.path.join(base_path, "BraTS*")))
        if not all_patients:
            raise ValueError(f"No patients found in {base_path}")
```

```
        else:
            print(f"Found {len(all_patients)} patients in {base_path}")
        if limit:
            self.patient_dirs = all_patients[:limit]  # Limit to 20 patients for now
        self.patient_dirs = all_patients

        self.transform = transform

    def __len__(self):
        return len(self.patient_dirs)

    def __getitem__(
        self, idx
    ) -> dict[Literal["t1", "t1ce", "t2", "flair", "label"], np.ndarray]:
        """
        Load data and labels for a given patient.
        """
        patient_dir = self.patient_dirs[idx]
        sample = {}

        # Load the four MRI modalities
        modalities = ["t1", "t1ce", "t2", "flair"]
        modalities = ["t1"]
        print(patient_dir)
        for modality in modalities:
            file_path = os.path.join(
                patient_dir, f"{os.path.basename(patient_dir)}_{modality}.nii.gz"
            )
            # sample[modality] = nib.load(file_path).get_fdata()
            sample[modality] = file_path

        # Load the segmentation label
        label_path = os.path.join(
            patient_dir, f"{os.path.basename(patient_dir)}_seg.nii.gz"
        )
        # sample["label"] = nib.load(label_path).get_fdata()
        sample["label"] = label_path

        if self.transform:
            # for modality in modalities:
            #     sample[modality] = self.transform(sample[modality])
            sample_ = {}
```

```
            sample_["image"] = sample["t1"]
            sample_["label"] = sample["label"]
            print(sample_)
            sample = self.transform(sample_)

        return sample
```