Project Documentation Summary

count.py

Purpose: Classifies each MRI slice (from FLAIR modality) as tumor or non-tumor based on the segmentation mask and saves it into respective folders.

Workflow:

- Loads `.nii` files (`flair` and `seg`).
- Normalizes the image slice.
- If any non-zero pixel exists in the segmentation mask, it's labeled as tumor.
- Saves the slice as `.png` in `tumor/` or `no_tumor/` folders.
- Prints count of each class at the end.

preprocess.py

Purpose: Extracts 2D slices from 3D `.nii` volumes and separates them into labeled (with tumor) and unlabeled datasets for semi-supervised learning.

Workflow:

- Extracts T1, T1CE, T2 images and segmentation mask for each patient.
- Resizes slices to 256×256.
- If the slice contains a tumor, saves to `labeled/images` and `labeled/masks`.
- If not, saves to `unlabeled/images`.

dataloader.py

Purpose: Loads the labeled and unlabeled datasets for training.

Classes:

- LabeledDataset: loads `image + mask`, converts mask into binary class label (`0` or `1`) for classification.
- UnlabeledDataset: loads unlabeled images, applies weak and strong augmentations for FixMatch

training.

Extras:

- get_augmentations() provides weak/strong transforms.

semi_supervised_unet_fixmatch.py

Purpose: Defines the segmentation model (U-Net with EfficientNet backbone), loss (Dice), and one training epoch for semi-supervised segmentation.

Key Components:

- Uses `segmentation_models_pytorch.Unet`.
- Weak & strong transforms are identity + noise.
- Implements FixMatch training with pseudo-labeling.
- Loss = supervised + lambda × unsupervised (Dice loss).

train.py

Purpose: Semi-supervised FixMatch training for classification using a `ResNet18` backbone.

Workflow:

- Loads labeled and unlabeled datasets with transforms.
- Uses cross-entropy for both supervised and pseudo-label-based unsupervised loss.
- Filters pseudo-labels with `confidence > 0.95`.
- Final loss is `loss_sup + loss_unsup`.

Final Notes:

- Your pipeline correctly reflects the FixMatch paradigm for classification.
- The count.py + preprocess.py steps ensure a realistic simulation of unlabeled/labeled split, crucial for semi-supervised learning.
- The paper you're comparing with does segmentation, but since you're doing classification, your use of these slices is valid. The classification-based labeling is what introduces deviation.