**EcoViz API contract**

**Branch: seg3d\_joe**

**Server:** Run FastAPI server inside conda environment (generated by seg3Denv.yml) using “python app/main.py”

**Endpoints:**

* **/seg3dtest (POST):** Performs 3D segmentation on an uploaded file.

**Authentication:**

* Currently, no authentication is required for these endpoints. We should consider implementing authentication mechanisms (e.g., API keys) for production environments.

**Data Format:**

* **/seg3dtest (POST)**
  + **Request:**
    - Accepts a file upload using the multipart/form-data format.
    - The uploaded file should be 3D medical data in a format supported by the library used (e.g., Nifti).
    - An example of this request can be done by running “python app/ make\_request.py”.
    - Unable to do this in Postman or Swagger. See “Problems/Limitations” section.
  + **Response:**
    - This returns a raw byte stream containing the segmentation labels. The data type is uint8. This data type is sufficient for representing the range of return values (0-9).
    - The returned predictions are saved in app/Predictions/mask.np. The values of mask.np can be viewed using view\_mask.ipynb.

**Error Handling:**

* Both endpoints return JSON responses with an "error" field containing a message string in case of any exceptions.

**Problems/Limitations:**

* A **/seg3dtest (POST) request** will tile the uploaded file and process each tile one at a time. This has a long runtime, Find ways to use GPU/parallelize
* A **/seg3dtest (POST) request** needs file uploads (\*.nii files) that are ~70 Mb. These are too large for Postman, which has a limit of 5Mb upload. This file size seems to also be a problem for Swagger.

**Functionality of /seg3dtest (POST):**

* This endpoint performs 3D segmentation on the uploaded medical data using a pre-trained model (app/model/ls\_seg3d\_model/ m20230623-163203wh500epochs).
* For each request, the segmentation process involves:
  + Loading the model and history.
  + Pre-processing the uploaded data (resampling, normalization).
    - The original size of the input tensor is (200,200,160). This is padded to become (256,256,240).
  + Dividing the padded data into patches for prediction.
    - Each patch is (128,128,96)
  + Performing predictions on each patch individually using the model.
  + Combining patch predictions and normalizing the results.
  + Converting probabilities to final segmentation labels.
  + (optional) Saving the segmentation labels as a Nifti file.
* The response contains the raw segmentation labels in a byte stream format.