A Strategic Guide to Data Preprocessing for the RSNA 2025 Intracranial Aneurysm Detection AI Challenge

Section 1: A Strategic Framework for Pre-training Data Curation in Neuroradiology AI

1.1 Introduction: Beyond Supervised Learning for the RSNA 2025 Challenge

The Radiological Society of North America (RSNA) 2025 Intracranial Aneurysm Detection AI Challenge represents a significant undertaking in the field of medical AI, tasking participants with developing models to detect and localize intracranial aneurysms across a diverse array of imaging modalities.[1] The challenge dataset, curated by a global task force of neuroradiologists and neurointerventionalists, is unprecedented in its scale and variety, comprising over 6,500 imaging studies from 18 institutions across five continents.[1] It encompasses Computed Tomography Angiography (CTA), Magnetic Resonance Angiography (MRA), and conventional T1 post-contrast and T2-weighted MRI scans, reflecting the complex reality of clinical practice.[1]

Success in this competition is measured by a weighted multilabel area under the ROC curve (AUC), with a heavy emphasis on the "Aneurysm Present" class, alongside scores for 13 distinct anatomical locations where aneurysms commonly arise.[4] This evaluation metric underscores a critical requirement: the model must not only detect the presence of an aneurysm but also accurately pinpoint its location within the intricate cerebrovascular anatomy. The explicit inclusion of "real clinical variation, with data from different institutions, scanners and imaging protocols" is designed to test the model's ability to generalize beyond a clean, homogenous training set.[4]

Given these conditions, a conventional supervised learning approach, where a model is trained solely on the provided labeled data, is unlikely to achieve top-tier performance. Such models are prone to overfitting to the specific characteristics of the training set and often fail to generalize to unseen data from different scanners or patient populations. A more robust strategy is required—one that equips the model with a deep, foundational understanding of neurovascular anatomy before it ever sees a single labeled aneurysm.

1.2 The Pre-training Paradigm: Learning Robust Anatomical Priors

The most promising strategy for tackling the RSNA 2025 challenge is to adopt a two-stage learning paradigm: pre-training followed by fine-tuning. This approach involves first training a model on a massive corpus of unlabeled or weakly labeled medical images to learn a rich, invariant representation of the underlying anatomy. This pre-trained model is then fine-tuned on the smaller, task-specific, and expertly annotated RSNA challenge dataset. This methodology is heavily influenced by the success of Self-Supervised Learning (SSL) in fields like natural language processing and computer vision, a paradigm that is rapidly demonstrating its power in medical image analysis.[5]

The objective of the pre-training phase is not to teach the model to find aneurysms directly. Instead, the goal is to force the model to learn the fundamental patterns, textures, and spatial relationships of the brain and its vasculature. By training on a vast and diverse set of head and neck scans, the model learns what constitutes "normal" anatomy across a wide spectrum of individuals, scanners, and imaging protocols. This process effectively builds a foundational model for neuroradiology, equipping it with a strong anatomical prior that makes the subsequent task of identifying subtle abnormalities like aneurysms significantly more effective.

The strategic advantage of this approach is directly tied to the nature of the challenge itself. The RSNA dataset is intentionally heterogeneous to test generalization.[4] Therefore, pre-training on a dataset that mirrors or even exceeds this heterogeneity is paramount. Datasets like OpenMind, which aggregates 114,000 scans from 800 different OpenNeuro datasets, captured on over 30 different scanners, provide the ideal training ground.[9] A model exposed to such immense variability during pre-training is inherently more likely to develop representations that are robust to the nuisance variables of scanner manufacturer, field strength, and acquisition parameters. This creates a direct pathway to superior performance on the hidden test set of the competition. The preprocessing pipeline, therefore, bears a dual responsibility: it must harmonize the technical characteristics of the data (e.g., voxel spacing, orientation) to make them compatible for a single model, but it must do so without erasing the valuable anatomical and pathological variance that drives robust feature learning.

1.3 Overview of the Proposed Preprocessing and Training Workflow

The end-to-end workflow to implement this strategy can be conceptualized in four distinct stages, forming a comprehensive plan from raw data to a competitive submission.

1. **Stage 1: Multi-Source Data Acquisition**. The process begins with the collection of large-scale public datasets. This report focuses on a curated selection: the OpenMind dataset for its massive scale in MRI, the TCIA HEAD-NECK-PET-CT and HNSCC collections for relevant CT anatomy, and the DeepLesion dataset for its sheer volume of diverse CT scans.
2. **Stage 2: The Unified Harmonization Pipeline**. This is the technical core of this report. All acquired data, regardless of its source format (DICOM, NIfTI, PNG), modality, or original acquisition parameters, will be processed through a rigorous pipeline. The end product of this stage is a new, harmonized dataset where every image is a NIfTI file with a standard Right-Anterior-Superior (RAS) orientation and 1.0 mm isotropic voxel spacing. This unified format is essential for pre-training a single, coherent model.
3. **Stage 3: Self-Supervised Pre-training**. With the harmonized dataset, a large neural network (e.g., a 3D U-Net or a more modern transformer-based architecture like WaveFormer [10]) is pre-trained using a self-supervised task. A promising candidate is masked image modeling, as implemented in frameworks like SparK [6], where the model learns to reconstruct masked-out portions of the 3D images. This forces the network to learn contextual anatomical information.
4. **Stage 4: Supervised Fine-tuning**. The weights of the pre-trained model are used to initialize a new model for the final task. This model is then fine-tuned on the official, labeled RSNA 2025 training dataset. Because the model already possesses a strong understanding of neuroanatomy, it can learn to detect and localize aneurysms more efficiently and with better generalization capabilities than a model trained from scratch.

This report will provide an exhaustive, expert-level guide to executing Stage 2, the critical data harmonization pipeline that underpins the entire strategy.

Section 2: Critical Review of Public Datasets for Cerebrovascular Feature Learning

A successful pre-training strategy hinges on the selection of appropriate datasets. The chosen datasets must be large enough to facilitate the learning of generalizable features and anatomically relevant to the head and neck region. This section provides a critical analysis of four recommended public datasets, evaluating their suitability, accessibility, and the specific challenges they present for harmonization.

2.1 The OpenMind Dataset

The OpenMind dataset stands out as a premier resource for self-supervised pre-training in 3D medical imaging.[9]

- **Description and Strategic Value**: OpenMind is a large-scale (114,000 scans) collection of 3D MRI images of the head and neck, explicitly aggregated for the purpose of advancing self-supervised learning research.[9] It is sourced from 800 distinct datasets on the OpenNeuro platform, encompassing 23 different MRI modalities and data from over 30 different scanner models.[9] This immense heterogeneity is its greatest strength. A model pre-trained on OpenMind is forced to learn anatomical representations that are invariant to scanner type, acquisition sequence, and patient population, directly preparing it for the multi-site, multi-modal nature of the RSNA challenge. Its sheer scale makes it the ideal cornerstone for the MRI component of the pre-training corpus.
- **Access and Format**: The dataset is readily accessible and downloadable from the Hugging Face Hub.[9] A significant advantage is that the data is already provided in the NIfTI format (
  .nii.gz), which eliminates the initial, often complex, DICOM-to-NIfTI conversion step. The data is organized in a BIDS (Brain Imaging Data Structure) like format, a common standard in neuroimaging research that facilitates programmatic

access.[9]

- **Quality and Harmonization Challenges**: While the NIfTI format is convenient, the dataset is intentionally "raw," with minimal preprocessing applied to preserve the original image characteristics.[14] This means that voxel spacing, image orientation, and intensity distributions are highly variable across the 114,000 scans. A robust harmonization pipeline is therefore essential. The dataset provides a crucial openneuro_metadata.csv file, which contains paths to the images and associated metadata, including scanner parameters, which is invaluable for managing the processing of this vast collection.[9] Furthermore, OpenMind includes anatomy_masks and deface_masks. The anatomy masks, which delineate brain tissue, are particularly useful. They can be used to guide brain extraction algorithms, focus SSL reconstruction tasks on anatomically meaningful regions, or ensure that sampling for contrastive learning methods does not draw from empty background space.[9]

2.2 The TCIA Head and Neck Squamous Cell Carcinoma (HNSCC) Collections

The Cancer Imaging Archive (TCIA) hosts several collections relevant to head and neck oncology, which are valuable for their anatomical focus.

- **Description and Strategic Value**: These collections contain multi-modal imaging data (CT, PET, MR) from patients with Head and Neck Squamous Cell Carcinoma (HNSCC).[15] The primary value for the aneurysm challenge lies in the contrast-enhanced CT scans, which provide detailed anatomical views of the head and neck, including the major vascular structures. If accessible, this data would be a high-quality source for pre-training on CT anatomy.
- **Access and Format**: The data is originally in DICOM format.[15] However, access presents a significant and potentially insurmountable barrier. The TCIA website for the main HNSCC collection explicitly states that downloads are "Unavailable" due to changes in the NIH Controlled Data Access Policy.[15] Gaining access may require a formal application through the Database of Genotypes and Phenotypes (dbGaP) [18], a process that is often lengthy and not guaranteed to succeed. This makes the HNSCC collection a high-risk, lower-priority target for data acquisition.
- **Quality and Harmonization Challenges**: As clinical datasets, they exhibit significant heterogeneity. One research paper that utilized the TCIA-HNSCC data reported resampling the scans to 1.0 mm isotropic voxels as a necessary

preprocessing step, confirming that the raw data is anisotropic and requires spatial harmonization.[19] The pipeline would need to handle standard DICOM to NIfTI conversion and subsequent spatial and intensity normalization.

2.3 The TCIA HEAD-NECK-PET-CT Collection

This specific TCIA collection offers a more accessible alternative for obtaining head and neck CT data.

- **Description and Strategic Value**: This dataset comprises FDG-PET/CT and radiotherapy planning CT scans for 298 H&N cancer patients, collected from four different institutions.[20] The multi-institutional nature ensures a degree of scanner and protocol variability, which is beneficial for pre-training. The CT component is the key asset for this project.
- **Access and Format**: The data is in DICOM format.[21] Unlike the broader HNSCC collection, this dataset appears to be more readily accessible. TCIA provides a direct download link for a 58.1 GB file containing the imaging data.[20] However, some documentation suggests that a "TCIA Restricted License Agreement" may need to be signed and submitted before access is granted.[21] This is a manageable administrative step compared to the dbGaP application process.
- **Quality and Harmonization Challenges**: The data requires the full preprocessing pipeline: DICOM to NIfTI conversion, orientation standardization, isotropic resampling, and intensity normalization. Research utilizing this dataset confirms the need for these steps, though specific details like original voxel spacing are not always published.[22]

2.4 The DeepLesion Dataset

The DeepLesion dataset is a massive, general-purpose medical imaging resource that presents both unique opportunities and significant technical challenges.

- **Description and Strategic Value**: DeepLesion is an enormous dataset containing over 32,000 annotated lesions found in 32,120 axial CT slices from over 10,000 studies.[23] While it is not specific to the head and neck, its sheer scale and diversity of findings make it an extremely valuable resource for teaching a model what constitutes "normal" and "abnormal" CT anatomy across the entire body. This can help the model learn to ignore pathologies that are not aneurysms.

- **Access and Format**: The dataset is publicly available for download from the NIH.[24] However, its format is a major hurdle. The images are provided not as DICOM or NIfTI volumes, but as individual 16-bit PNG slices.[24] This necessitates a complex and careful data reconstruction step before it can be used for 3D pre-training.
- **Quality and Harmonization Challenges**: The primary challenge is the reconstruction of 3D volumes from 2D slices. This is a critical, error-prone process that requires meticulous engineering. The workflow involves:
  1. Reading the 16-bit pixel data from the PNG files.
  2. Converting these pixel values back to Hounsfield Units (HU) by applying the formula: HU=pixel_value−32768.[24]
  3. Extracting the geometric metadata for each slice, particularly PixelSpacing (in-plane resolution) and SliceThickness or SliceInterval (through-plane resolution), from the master DL_info.csv file that accompanies the dataset.[24]
  4. Stacking the 2D slice arrays into a 3D NumPy array.
  5. Constructing a valid NIfTI affine transformation matrix using the extracted spacing information. This matrix correctly maps the voxel grid to physical space.

Any error in this process, such as assuming isotropic pixels or a default slice thickness, will result in geometrically distorted 3D volumes that would corrupt the pre-training dataset. The availability of community-developed scripts, such as DL_save_nifti.py mentioned in the LesionHarvester GitHub repository, provides a crucial starting point for this task.[28] Even with these tools, it is likely that not all scans in DeepLesion will have the complete metadata required for perfect 3D reconstruction, so some data loss is expected. The high variability in slice thickness across the dataset (one paper mentions interpolating slices at 2 mm intervals [29]) further complicates this process.

2.5 Comparative Analysis of Pre-training Datasets

To guide the data acquisition and processing strategy, the following table provides a concise summary and comparison of the recommended datasets. It distills the detailed analysis into an actionable format, allowing for a rational prioritization of effort. For instance, the access barrier for TCIA HNSCC suggests it should be a lower-priority target, while the format of DeepLesion indicates a high-effort, high-reward task.

Dataset	Primary Modality	Source Format	Scale (Approx.)	Access Method	Key Harmonization Challenges	Strategic Value for Pre-training
OpenMind	MRI (T1w, T2w, etc.)	NIfTI (.nii.gz)	114,000 scans	Public (Hugging Face)	High variability in spacing, orientation, contrast. Requires robust intensity normalization (e.g., histogram matching).	**Excellent**. The cornerstone for MRI pre-training due to scale and heterogeneity.
TCIA HNSCC	CT, PET, MR	DICOM	627 subjects	**Restricted** (Unavailable)	DICOM to NIfTI conversion. High variability. Access is the primary barrier.	**High (if accessible)**. Provides relevant contrast-enhanced CT anatomy.
TCIA HEAD-NECK-PET-CT	CT, PET	DICOM	298 subjects	Public (TCIA Download, may require license agreement)	DICOM to NIfTI conversion. Multi-institutional variability.	**Good**. Accessible source of head & neck CT data. A solid starting point for CT pre-training.

DeepLesion	CT	16-bit PNG	32,000+ lesions (~10k studies)	Public (NIH Box)	**Major challenge**: Reconstructing 3D NIfTI from 2D PNGs and metadata. Requires careful handling of spacing info from CSV to build correct affine matrix.	**High (with high effort)**. Enormous scale. Provides diverse CT data but requires significant, careful engineering to be usable in 3D.

Section 3: The Unified Preprocessing Pipeline: A Step-by-Step Implementation Guide

This section provides the technical core of the report: a detailed, step-by-step guide to constructing a robust and unified preprocessing pipeline. Each step is justified and accompanied by implementation patterns using the recommended Python libraries: SimpleITK, Nibabel, and MONAI. The goal is to transform the heterogeneous source data into a standardized format ready for pre-training.

3.1 Foundational Conversion: From DICOM to NIfTI

The first step for any clinical dataset is to convert it from the DICOM (Digital Imaging and Communications in Medicine) standard to the NIfTI (Neuroimaging Informatics Technology Initiative) format. DICOM is optimized for clinical workflow and stores data on a per-slice basis, while NIfTI is a research-oriented format that consolidates an entire 3D volume into a single file with a clear spatial orientation header.

This conversion is not merely a format change; it is a critical geometric consolidation. Inconsistencies between different conversion tools can introduce subtle errors in the image's affine matrix, which defines its position and orientation in space. Such errors can corrupt the entire downstream analysis pipeline.[30] Therefore, it is imperative to standardize on a single, well-vetted conversion tool.

Recommendation: For command-line batch processing, dcm2niix is the undisputed community gold standard, known for its robustness and ability to correctly interpret complex DICOM headers from various vendors.[31] For a pure Python-based workflow, SimpleITK's

ImageSeriesReader provides an equally reliable and powerful alternative.

Implementation Outline (SimpleITK):
The ImageSeriesReader automatically detects all slices belonging to a single series within a directory and stitches them into a coherent 3D volume.

Python

```python
import SimpleITK as sitk
import os

def convert_dicom_to_nifti(dicom_dir: str, output_path: str):
    """
    Converts a DICOM series to a NIfTI file using SimpleITK.

    Args:
        dicom_dir: Path to the directory containing DICOM files for a single series.
        output_path: Path to save the output.nii.gz file.
    """
    if not os.path.exists(dicom_dir):
        print(f"Error: DICOM directory not found at {dicom_dir}")
        return

    reader = sitk.ImageSeriesReader()
    try:
        dicom_names = reader.GetGDCMSeriesFileNames(dicom_dir)
        if not dicom_names:
            print(f"Error: No DICOM series found in {dicom_dir}")
            return

        reader.SetFileNames(dicom_names)
        image = reader.Execute()

        # Ensure output directory exists
        os.makedirs(os.path.dirname(output_path), exist_ok=True)
        sitk.WriteImage(image, output_path, useCompression=True)
        print(f"Successfully converted {dicom_dir} to {output_path}")

    except RuntimeError as e:
        print(f"Failed to process {dicom_dir}: {e}")
```

Special Case (DeepLesion): As discussed, DeepLesion requires a custom reconstruction process. This involves using a library like pandas to read the metadata, Pillow or imageio to read the PNG data, numpy to stack the slices, and nibabel to construct and save the final NIfTI file with a custom-built affine matrix based on the pixel and slice spacing from the metadata CSV.

3.2 Spatial Harmonization I: Orientation Standardization to RAS

Medical scans can be acquired and stored with the patient oriented in various ways relative to the image grid. A deep learning model, particularly a Convolutional Neural Network (CNN), has no innate understanding of anatomical directions like "left," "right," "anterior," or "posterior." It simply processes a 3D array of numbers. If the orientation is inconsistent across the dataset—for example, if the first axis corresponds to left-to-right in one scan and inferior-to-superior in another—the model cannot learn meaningful spatial features, and training will fail.

To solve this, all images must be reoriented to a single, canonical anatomical coordinate system. The most common standard in neuroimaging research and software is **RAS**, where the axes of the data array correspond to:

- **x-axis**: Left to **R**ight
- **y-axis**: Posterior to **A**nterior

- **z-axis**: Inferior to **S**uperior

Recommendation: Use a library that can read the orientation information from the NIfTI header (the affine matrix) and programmatically reorder and flip the data axes to match the target orientation. MONAI's dictionary-based transforms are perfectly suited for this, as they can operate on image data and automatically update the associated metadata.

Implementation Outline (MONAI):
The Orientationd transform is the ideal tool. It takes a target axis code (e.g., "RAS") and ensures the output image data and its affine matrix conform to that standard.

Python

```python
from monai.transforms import LoadImaged, Orientationd, SaveImaged

# This transform loads a NIfTI file and reorients it to RAS.
# 'd' at the end of the transform name indicates it's a dictionary-based transform.
orientation_transform = Orientationd(keys=["image"], axcodes="RAS")

# Example usage on a single file dictionary
data_dict = {"image": "path/to/input.nii.gz"}
reoriented_dict = orientation_transform(data_dict)

# The reoriented image is now in reoriented_dict["image"]
# The affine matrix in its metadata has been updated automatically.
```

For workflows not using MONAI, the nibabel.as_closest_canonical() function provides similar functionality, reorienting an image to the closest orientation to RAS.[33]

3.3 Spatial Harmonization II: Isotropic Resampling to 1.0mm

Clinical CT and MRI scans are often acquired with **anisotropic** voxels, meaning the dimensions of a single voxel are not equal in all directions (e.g., 0.6mm x 0.6mm in-plane resolution, but a 5.0mm slice thickness). This is problematic for 3D CNNs, which typically use cubic kernels (e.g., 3x3x3). On anisotropic data, such a kernel covers a different physical distance along each axis, biasing the network's perception of spatial relationships.

To ensure that spatial distance is consistent in all directions, all volumes must be resampled to have **isotropic** (cubic) voxels. A standard resolution for high-quality neuroimaging analysis is 1.0 x 1.0 x 1.0 mm.

Recommendation: Resample all images and any corresponding masks to a target spacing of (1.0, 1.0, 1.0) mm. The choice of interpolation algorithm during this process is critically important and depends on the data type.

- **For Continuous Intensity Data (CT/MRI Images)**: Use a smooth interpolation method to avoid introducing blocky artifacts. **Trilinear (or bilinear in MONAI's terminology for ND)** interpolation offers a good balance of speed and quality. **B-spline** (cubic) interpolation can produce even smoother results at a slightly higher computational cost and is often considered a superior choice in medical image processing.[34]
- **For Discrete Label Data (Segmentation Masks)**: Use **Nearest Neighbor** interpolation. This method assigns the value of the closest original voxel to the new voxel location, ensuring that no new, invalid label values are created (e.g., interpolating between a background voxel of 0 and a brain voxel of 1 will not result in a nonsensical label of 0.5).[34]

Implementation Outline (MONAI):
The Spacingd transform handles resampling. It is crucial to apply it with different mode parameters for the image and the mask.

Python

```python
from monai.transforms import Spacingd

# Define the target spacing
target_spacing = (1.0, 1.0, 1.0)

# Create a transform for the intensity image using smooth interpolation
# 'mode="bilinear"' is MONAI's name for n-dimensional linear interpolation.
image_resample_transform = Spacingd(
    keys=["image"],
    pixdim=target_spacing,
    mode="bilinear",
    align_corners=True # Recommended for better consistency
)

# Create a separate transform for the segmentation mask using nearest neighbor
mask_resample_transform = Spacingd(
    keys=["mask"],
    pixdim=target_spacing,
    mode="nearest"
)

# In a composed pipeline, these would be applied sequentially
# data_dict = {"image": "path/to/image.nii.gz", "mask": "path/to/mask.nii.gz"}
# data_dict = image_resample_transform(data_dict)
# data_dict = mask_resample_transform(data_dict)
```

The diagonal=False parameter of Spacingd (which is the default) is important, as it ensures that the orientation of the image is preserved during the resampling process.[36]

3.4 Modality-Specific Intensity Normalization

The raw intensity values in medical images are not directly comparable across different scans, scanners, or even patients. This variability can hinder a model's ability to learn consistent features. Therefore, intensity normalization is a mandatory step, and the optimal technique is modality-specific.

For CT Data (TCIA, DeepLesion):
CT images have a standardized intensity scale called Hounsfield Units (HU), where values are calibrated to the radiodensity of distilled water (0 HU) and air (-1000 HU).37 While the scale is standard, the full range of HU values (from -1024 to over 3000) is too wide to be meaningfully displayed or processed at once. Radiologists use a technique called
windowing to focus on the HU range of a specific tissue type.[38] The same principle can be applied to prepare data for a neural network.

Recommendation: Instead of a single normalization, create a multi-channel input image where each channel represents a different clinically relevant CT window. This provides the network with a richer set of features, highlighting different anatomical structures in each channel.

1. **Convert to HU**: For DICOM data, this is typically done using the RescaleSlope and RescaleIntercept tags from the metadata: HU=pixel_value×RescaleSlope+RescaleIntercept. For the DeepLesion PNGs, the conversion is HU=pixel_value−32768.[24]
2. **Apply Multi-Windowing**: Create three separate channels from the HU volume:
   ○ **Channel 1: Brain Window** (e.g., Width=80 HU, Level=40 HU). This window, ranging from 0 to 80 HU, is optimal for visualizing soft tissue contrast within the brain parenchyma.[38]
   ○ **Channel 2: Subdural/Blood Window** (e.g., Width=200 HU, Level=80 HU). This window (-20 to 180 HU) is better for detecting acute and subacute hemorrhages, which can have densities similar to those of contrast-filled aneurysms.[38]
   ○ **Channel 3: Bone Window** (e.g., Width=2800 HU, Level=600 HU). This wide window (-800 to 2200 HU) clearly delineates the skull and other bony structures, providing critical anatomical context for the surrounding vasculature.[38]
3. **Clip and Rescale**: For each channel, clip the HU values to the window's range and then rescale the resulting values to a standard range like or [-1, 1].
4. **Stack Channels**: Stack the three normalized arrays to form a single multi-channel 3D image.

For MRI Data (OpenMind):
Unlike CT, MRI intensity values have no absolute meaning and vary dramatically between scanners, sequences, and patients. Normalization is therefore even more critical.
Recommendation:

- **Basic Method**: Z-score normalization (subtracting the mean and dividing by the standard deviation of voxel intensities within a brain mask) is a simple and effective baseline.
- **Advanced Method (Recommended for OpenMind)**: For a highly heterogeneous dataset like OpenMind, more sophisticated techniques are warranted. **Nyúl & Udupa histogram normalization** is an excellent choice. This method learns a "standard" intensity histogram from a representative subset of the training data and then transforms the histogram of each new image to match this standard. This enforces a much higher degree of intensity consistency across the entire dataset. The Python package intensity-normalization provides a high-quality, easy-to-use implementation of this and other advanced methods like WhiteStripe.[40] Using this library is a key recommendation for maximizing the value of the OpenMind dataset.

3.5 The Brain Extraction Dilemma: A Risk-Benefit Analysis

Brain extraction, or skull stripping, is the process of creating a mask to remove all non-brain tissue (skull, scalp, muscle, fat) from the image. Whether to include this step is a critical decision with significant trade-offs.

- **The Argument For Brain Extraction (Pros)**:
  - **Reduced Computational Load**: By masking out irrelevant voxels, the effective volume processed by the network is smaller, potentially reducing memory consumption and speeding up training.
  - **Removal of Distractors**: The skull has extremely high intensity values in CT and is a prominent feature. Removing it can prevent the model from focusing on these high-contrast but irrelevant structures.
  - **Improved Normalization**: Intensity normalization techniques like Z-scoring are more stable and meaningful when calculated only on brain tissue, avoiding contamination from the different intensity distributions of the skull and scalp.
- **The Argument Against Brain Extraction (Cons)**:
  - **CRITICAL RISK: Clipping of Peripheral Vasculature**: This is the most significant danger. Aneurysms, particularly those of the Circle of Willis, are located at the base of the brain, in close proximity to the skull. An imperfect or overly aggressive skull-stripping algorithm can easily and silently remove these critical vascular structures, effectively deleting the very pathology the model is supposed to find. This would catastrophically damage the dataset.
  - **Introduction of Artifacts**: A poor segmentation can leave behind fragments of skull or remove parts of the cortex, introducing false edges and artifacts that can confuse the model.

Recommendation and Mitigation Strategy:
The risk of inadvertently removing aneurysms is too high to recommend brain extraction as a default, mandatory step. The preferred initial approach is to avoid skull stripping and rely on a sufficiently deep and capable model to learn to ignore non-brain regions.
However, if initial experiments show that model performance is poor (potentially due to distraction by the skull) or if VRAM limitations are insurmountable even with patching, brain extraction can be considered as a targeted intervention. If this step is deemed necessary, the following protocol must be followed:

1. **Use a State-of-the-Art Tool**: Do not use older, atlas-based methods. The recommended tool is **SynthStrip**, a modern deep learning-based method that

has been specifically designed for robustness across a wide range of modalities (MRI, CT, PET) and populations.[41] It has demonstrated superior performance compared to other tools, especially on heterogeneous data, and its developers have actively fixed bugs related to NIfTI geometry issues.[41]

2. **Tune the Aggressiveness**: SynthStrip offers a --no-csf flag, which produces a "tighter" brain mask by excluding more of the surrounding cerebrospinal fluid. For aneurysm detection, it is safer to *not* use this flag initially, as the default, more generous mask is less likely to clip vessels at the brain's periphery.[41]
3. **Mandatory Visual Quality Assurance**: The pipeline must include a step to automatically generate and save 2D PNG "sanity checks" for a random subset of cases from each dataset. These images should show the contour of the generated brain mask overlaid on the corresponding axial, sagittal, and coronal slices of the original image. A human must review these images to ensure that major vascular territories, especially at the skull base, are not being erroneously excluded.

Section 4: VRAM-Aware Implementation and Performance Optimization

A significant practical constraint in 3D deep learning is the limited memory (VRAM) available on GPUs. Even high-end consumer or data center GPUs with 16-24 GB of VRAM are often insufficient to train a large 3D model on a full-resolution medical image.[43] This section details the nature of this bottleneck and provides concrete strategies for building a pipeline that is both effective and computationally feasible.

4.1 The VRAM Bottleneck: Quantifying the Memory Footprint

The memory required during training is a sum of several components: the model's weights, the gradients calculated during backpropagation, the optimizer's state variables, and, most significantly, the intermediate activation maps.[45] A simple calculation demonstrates why full-volume training is often impossible.

Consider a single 3D image preprocessed to a modest size of $256 \times 256 \times 192$ voxels, which is smaller than many full brain volumes after resampling.

- Input Image Size: With a batch size of 1 and using 32-bit floating-point precision (4 bytes/voxel), the input tensor alone requires:
  $1 \times 256 \times 256 \times 192 \times 4 \text{ bytes} \approx 50.3 \text{ MB}$
- Model Parameters: A standard 3D U-Net architecture can easily have 30 million parameters. Storing these weights in FP32 requires:
  $30{,}000{,}000 \times 4 \text{ bytes} = 120 \text{ MB}$
- **Gradients**: The gradients have the same size as the parameters, adding another 120 MB.
- Optimizer States: The commonly used Adam/AdamW optimizer stores two moments (mean and variance of gradients) for each parameter, requiring an additional:
  $30{,}000{,}000 \times 2 \times 4 \text{ bytes} = 240 \text{ MB}$
- **Activations**: This is the most demanding component. The intermediate feature maps generated by each convolutional layer can be enormous. For a 256×256×192 input, the initial layers of a U-Net will produce feature maps of a similar spatial size, but with many channels (e.g., 32 or 64). A single such layer could consume several gigabytes of VRAM.

Summing these components, it becomes clear that even with a batch size of 1, the total VRAM consumption for a full-volume 3D U-Net will rapidly exceed the 16-24 GB budget. This is an unavoidable hardware limitation that dictates the training strategy.

4.2 The Solution: Patch-Based Training

The only viable strategy to manage this VRAM constraint is **patch-based training**. Instead of feeding the entire 3D volume to the network at once, smaller 3D patches (or sub-volumes) are randomly extracted from the full image during each training iteration.

Recommendation: Train the model on randomly sampled 3D patches of a fixed size that fits comfortably within the VRAM budget, such as 128×128×128 or 96×96×96 voxels. This dramatically reduces the size of the activation maps, which are the primary consumer of VRAM. During inference on the validation or test set, a sliding window approach can be used to process the entire volume patch by patch and aggregate the results.[46]

Implementation (MONAI): MONAI's RandSpatialCropd transform is the standard tool for this task. It can be inserted directly into the data loading pipeline.

Python

```python
from monai.transforms import RandSpatialCropd

# This transform will randomly extract a 128x128x128 patch from the "image"
# and "mask" arrays in the data dictionary.
patch_transform = RandSpatialCropd(
    keys=["image", "mask"],
    roi_size=(128, 128, 128),
    random_size=False # Ensures all patches are the same size
)
```

4.3 Accelerating the Data Pipeline with MONAI

While patch-based training solves the VRAM issue, it can introduce a new bottleneck: data loading. If the full pipeline (loading from disk, resampling, reorienting, normalizing, cropping) is performed on-the-fly for every single patch, the CPU can become a bottleneck, leaving the expensive GPU idle while it waits for data. This is highly inefficient.

A more intelligent approach is to differentiate between deterministic, heavy preprocessing steps and stochastic, lightweight data augmentation.

- **Heavy Preprocessing**: Steps like DICOM conversion, resampling, and orientation are deterministic and computationally expensive. They only need to be performed *once* per image.
- **Lightweight Augmentation**: Steps like random cropping (patching), random flips, and minor intensity shifts are stochastic and should be performed on-the-fly during training to provide variety to the model.

Recommendation: Implement a two-stage data preparation workflow.

1. **Offline Harmonization**: Run the heavy preprocessing pipeline (as detailed in

Section 3) once for the entire collection of datasets. Save the resulting harmonized, full-resolution NIfTI files to a new directory. This becomes the "preprocessed dataset."

2. **Online Augmentation with Caching**: During training, the data loader should read directly from this preprocessed dataset. To further accelerate this, use MONAI's caching mechanisms. CacheDataset loads the entire preprocessed dataset into RAM at the beginning of the first epoch. Subsequent epochs read data directly from fast RAM instead of slower disk storage, dramatically increasing throughput.[46] If the dataset is too large for RAM, PersistentDataset can be used to create a disk-based cache of transformed data.[48]

Implementation (MONAI):
The training script would use CacheDataset to wrap a standard dataset class, and the transform chain would only contain the lightweight, on-the-fly augmentations.

Python

```python
from monai.data import CacheDataset, DataLoader, Dataset
from monai.transforms import Compose, LoadImaged, AddChanneld,
RandSpatialCropd, ToTensord

# Assume file_list is a list of dictionaries pointing to preprocessed images
# e.g., [{"image": "path/to/preprocessed_img1.nii.gz"},...]

# Define only the on-the-fly transforms
train_transforms = Compose([
    LoadImaged(keys=["image"]),
    AddChanneld(keys=["image"]),
    RandSpatialCropd(keys=["image"], roi_size=(128, 128, 128), random_size=False),
    ToTensord(keys=["image"]),
])

# Create the base dataset
base_dataset = Dataset(data=file_list, transform=train_transforms)

# Wrap it with CacheDataset to load all data into RAM for fast access
# For a large dataset, cache_rate can be set to < 1.0 to cache a subset.
# num_workers can be used to parallelize the initial caching process.
```

```
cached_dataset = CacheDataset(
    data=base_dataset,
    cache_rate=1.0,
    num_workers=8
)

# Use the cached dataset with a standard PyTorch DataLoader
train_loader = DataLoader(cached_dataset, batch_size=4, shuffle=True,
num_workers=0)
```

Additionally, for certain workloads where multiprocessing overhead is a concern, MONAI's ThreadDataLoader can be a more performant alternative to the standard torch.utils.data.DataLoader.[46]

Section 5: Integrated Python Pipeline Script Outline

This final section synthesizes all the preceding analysis and recommendations into a coherent, practical Python script outline. It demonstrates how to structure the project, combining offline preprocessing with an efficient, MONAI-based online data loading pipeline for model training. This outline serves as an actionable template for implementing the full strategy.

5.1 Overall Project Structure

A robust implementation will separate the one-time, heavy data harmonization from the iterative training process.

preprocess_data.py (Offline Script):
This script is run once to convert all source datasets into a unified, preprocessed format. It should contain separate functions to handle the unique characteristics of each source dataset.

Python

```python
# preprocess_data.py
import os
import pandas as pd
import numpy as np
import SimpleITK as sitk
import nibabel as nib
from monai.transforms import Compose, LoadImaged, Orientationd, Spacingd,
SaveImaged
from intensity_normalization.normalize import NyulNormalizer

# Define constants
TARGET_SPACING = (1.0, 1.0, 1.0)
OUTPUT_DIR = "./preprocessed_data"

def ct_windowing(image_array, width, level):
    """Applies CT windowing and normalizes to ."""
    lower = level - width / 2
    upper = level + width / 2
    windowed_array = np.clip(image_array, lower, upper)
    return (windowed_array - lower) / (upper - lower)

def process_tcia_ct_series(series_dir, output_path):
    """
   Full pipeline for a single TCIA CT series (DICOM input).
   1. DICOM to NIfTI (using SimpleITK for geometry)
   2. Reorient to RAS
   3. Resample to 1.0mm isotropic
   4. Apply multi-channel windowing
   5. Save final preprocessed file
   """
    #... implementation using SimpleITK for conversion...
    #... then MONAI transforms for reorient, resample...
    #... then custom windowing function and np.stack...
    #... save final multi-channel NIfTI...
    pass

def process_deeplesion_study(study_info, png_dir, output_path):
    """
   Full pipeline for a DeepLesion study (PNG input).
   1. Reconstruct 3D volume from PNGs and metadata CSV
```

```
    2. Reorient to RAS
    3. Resample to 1.0mm isotropic
    4. Apply multi-channel windowing
    5. Save final preprocessed file
    """
    #... implementation using pandas, Pillow, nibabel...
    #... carefully construct affine from pixel spacing and slice interval...
    #... then apply same steps as TCIA processing...
    pass

def process_openmind_mri(input_path, output_path, nyul_normalizer):
    """
    Full pipeline for an OpenMind MRI (NIfTI input).
    1. Load NIfTI
    2. Reorient to RAS
    3. Resample to 1.0mm isotropic
    4. Apply Nyul Histogram Normalization
    5. Save final preprocessed file
    """
    #... implementation using MONAI transforms...
    #... use intensity-normalization library for advanced normalization...
    pass

def main():
    # 1. Process TCIA HEAD-NECK-PET-CT dataset
    # Loop through patient/study directories and call process_tcia_ct_series

    # 2. Process DeepLesion dataset
    # Read DL_info.csv, group by study, and call process_deeplesion_study

    # 3. Process OpenMind dataset
    # First, fit a Nyul normalizer on a subset of the data
    # nyul_normalizer = NyulNormalizer()
    # nyul_normalizer.fit(subset_of_openmind_images)
    # Then, loop through all OpenMind files and call process_openmind_mri
    pass

if __name__ == "__main__":
    main()
```

train.py (Online Script):
This script performs the model training. It assumes the preprocessing has already been

completed and leverages MONAI's CacheDataset and on-the-fly transforms for maximum efficiency.

Python

```python
# train.py
import torch
from monai.data import CacheDataset, DataLoader, Dataset, list_data_collate
from monai.transforms import (
    Compose,
    LoadImaged,
    AddChanneld,
    RandSpatialCropd,
    RandFlipd,
    RandRotate90d,
    RandScaleIntensityd,
    RandShiftIntensityd,
    ToTensord,
)
# from your_model_file import My3DUnet  # Your model definition

def main():
    # --- 1. DATA PREPARATION ---
    # Create a list of file paths to your preprocessed data
    # This should combine files from OpenMind, TCIA, DeepLesion, etc.
    # data_files = [{"image": "path/to/preprocessed_file_1.nii.gz"},...]

    # Define the on-the-fly augmentations
    # These are lightweight and add variety to the training data
    train_transforms = Compose([
        LoadImaged(keys=["image"]),
        AddChanneld(keys=["image"]), # Ensure image has a channel dimension

        # Intensity augmentations
        RandScaleIntensityd(keys=["image"], factors=0.1, prob=0.5),
        RandShiftIntensityd(keys=["image"], offsets=0.1, prob=0.5),

        # Spatial augmentations
        # Patch-based training is essential for 3D models
        RandSpatialCropd(keys=["image"], roi_size=(128, 128, 128), random_size=False),
        RandFlipd(keys=["image"], prob=0.5, spatial_axis=0), # Flip along first spatial axis
        RandFlipd(keys=["image"], prob=0.5, spatial_axis=1),
        RandFlipd(keys=["image"], prob=0.5, spatial_axis=2),
        RandRotate90d(keys=["image"], prob=0.5, max_k=3),

        ToTensord(keys=["image"]),
    ])

    # Use CacheDataset to load all preprocessed data into RAM for fast access
    print("Caching dataset... this may take a while.")
    train_ds = CacheDataset(
        data=data_files,
        transform=train_transforms,
        cache_rate=1.0,
        num_workers=8
    )

    # Create the DataLoader
    # Use collate_fn=list_data_collate for MONAI dictionary-based data
    train_loader = DataLoader(
        train_ds,
        batch_size=4, # Adjust based on VRAM
        shuffle=True,
        num_workers=4,
        collate_fn=list_data_collate,
        pin_memory=torch.cuda.is_available()
    )

    # --- 2. MODEL, LOSS, OPTIMIZER SETUP ---
    device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
    # model = My3DUnet(in_channels=..., out_channels=...).to(device)
    # loss_function =...
    # optimizer =...
    # scaler = torch.cuda.amp.GradScaler() # For mixed-precision training

    # --- 3. TRAINING LOOP ---
    # Standard PyTorch training loop
```

```
    # for epoch in range(num_epochs):
    #     for batch_data in train_loader:
    #         inputs = batch_data["image"].to(device)
    #
    #         # with torch.cuda.amp.autocast():
    #         #     outputs = model(inputs)
    #         #     loss = loss_function(outputs, inputs) # For a self-supervised task like reconstruction
    #         #
    #         # scaler.scale(loss).backward()
    #         # scaler.step(optimizer)
    #         # scaler.update()

if __name__ == "__main__":
    main()
```

5.2 Conclusions and Final Recommendations

This report outlines a comprehensive, state-of-the-art strategy for preparing a large-scale, heterogeneous dataset for pre-training a model for the RSNA 2025 Intracranial Aneurysm Detection challenge. The core philosophy is that robust feature learning, driven by exposure to diverse anatomical and pathological data, is the key to building a generalizable and competitive model.

Key Actionable Recommendations:

1. **Prioritize Data Acquisition**: Begin by downloading the OpenMind and TCIA HEAD-NECK-PET-CT datasets, as they are the most accessible. Concurrently, start the engineering effort required to reconstruct 3D volumes from the DeepLesion dataset. De-prioritize the TCIA HNSCC collection unless a clear path to access is identified.
2. **Standardize Rigorously**: Implement the unified preprocessing pipeline exactly as described. Use a single, consistent tool for each step (e.g., SimpleITK for DICOM conversion, MONAI for spatial transforms) to avoid subtle geometric inconsistencies. The target format must be NIfTI, 1.0mm isotropic, and RAS orientation.
3. **Invest in Advanced Normalization**: For the highly heterogeneous OpenMind MRI dataset, go beyond simple Z-scoring. Use a library like intensity-normalization

to implement Nyúl-style histogram matching to achieve superior intensity consistency. For CT, adopt the proposed multi-channel windowing approach to provide the model with a richer feature set.

4. **Be Cautious with Brain Extraction**: Avoid skull stripping as the default strategy due to the high risk of clipping relevant peripheral vasculature. If it becomes necessary for performance or memory reasons, use the robust SynthStrip tool and implement a mandatory visual quality assurance protocol.
5. **Embrace a Two-Stage, VRAM-Aware Workflow**: Acknowledge the hardware constraints from the outset. Design the workflow around an offline, one-time harmonization step and an online, cached training step that uses patch-based sampling. This is the most efficient use of computational resources.

By meticulously following this guide, a participant in the RSNA 2025 challenge can create a high-quality, harmonized pre-training dataset that will serve as a powerful foundation for developing a winning model. The investment in a robust data preprocessing pipeline is not a preliminary chore but the most critical first step toward success.

Works cited

1. RSNA Launches Intracranial Aneurysm Detection AI Challenge - RSNA Press Releases, accessed August 11, 2025, https://press.rsna.org/timssnet/media/pressreleases/14_pr_target.cfm?ID=2596
2. AI challenges | RSNA, accessed August 11, 2025, https://www.rsna.org/rsnai/ai-image-challenge
3. RSNA launches 2025 Intracranial Aneurysm Detection AI Challenge | AuntMinnie, accessed August 11, 2025, https://www.auntminnie.com/imaging-informatics/artificial-intelligence/article/15751740/rsna-launches-2025-intracranial-aneurysm-detection-ai-challenge
4. RSNA Intracranial Aneurysm Detection - Kaggle, accessed August 11, 2025, https://www.kaggle.com/competitions/rsna-intracranial-aneurysm-detection/overview/evaluation
5. arXiv:2410.23132v3 [cs.CV] 4 Apr 2025, accessed August 11, 2025, https://arxiv.org/pdf/2410.23132?
6. HySparK: Hybrid Sparse Masking for Large Scale Medical Image Pre-Training - arXiv, accessed August 11, 2025, https://arxiv.org/html/2408.05815v1
7. Self-supervised learning for medical image classification: a systematic review and implementation guidelines - PubMed, accessed August 11, 2025, https://pubmed.ncbi.nlm.nih.gov/37100953/
8. Self-Supervised Learning | Papers With Code, accessed August 11, 2025, https://paperswithcode.com/task/self-supervised-learning
9. AnonRes/OpenMind · Datasets at Hugging Face, accessed August 11, 2025, https://huggingface.co/datasets/AnonRes/OpenMind

10. WaveFormer: A 3D Transformer with Wavelet-Driven Feature Representation for Efficient Medical Image Segmentation - arXiv, accessed August 11, 2025, https://arxiv.org/html/2503.23764v1
11. WaveFormer: A 3D Transformer with Wavelet-Driven Feature Representation for Efficient Medical Image Segmentation - Powerdrill AI, accessed August 11, 2025, https://powerdrill.ai/discover/summary-waveformer-a-3d-transformer-with-wavelet-driven-cm90exfjxnl5x07pnv1woq4tm
12. An OpenMind for 3D medical vision self-supervised learning - ResearchGate, accessed August 11, 2025, https://www.researchgate.net/publication/387351388_An_OpenMind_for_3D_medical_vision_self-supervised_learning
13. An OpenMind for 3D medical vision self-supervised learning - arXiv, accessed August 11, 2025, https://arxiv.org/html/2412.17041v1
14. (PDF) A large-scale heterogeneous 3D magnetic resonance brain imaging dataset for self-supervised learning - ResearchGate, accessed August 11, 2025, https://www.researchgate.net/publication/392766424_A_large-scale_heterogeneous_3D_magnetic_resonance_brain_imaging_dataset_for_self-supervised_learning
15. HNSCC - The Cancer Imaging Archive (TCIA), accessed August 11, 2025, https://www.cancerimagingarchive.net/collection/hnscc/
16. HNSCC-3DCT-RT - The Cancer Imaging Archive (TCIA), accessed August 11, 2025, https://www.cancerimagingarchive.net/collection/hnscc-3dct-rt/
17. Data from Head and Neck Cancer CT Atlas(HNSCC) - Datasets - 國網中心資料集平台, accessed August 11, 2025, https://scidm.nchc.org.tw/en/dataset/hnscc
18. TCGA-HNSC - GDC Data Portal, accessed August 11, 2025, https://portal.gdc.cancer.gov/projects/TCGA-HNSC
19. Multimodal Deep Learning for Stage Classification of Head and ..., accessed August 11, 2025, https://www.mdpi.com/2072-6694/17/13/2115
20. HEAD-NECK-PET-CT - The Cancer Imaging Archive (TCIA), accessed August 11, 2025, https://www.cancerimagingarchive.net/collection/head-neck-pet-ct/
21. Cancer Imaging Archive Wiki: Confluence Mobile, accessed August 11, 2025, https://wiki.cancerimagingarchive.net/display/Public/Head-Neck-PET-CT
22. An Investigation on Radiomics Feature Handling for HNSCC ... - MDPI, accessed August 11, 2025, https://www.mdpi.com/2076-3417/12/15/7826
23. DeepLesion: a large-scale and diverse CT lesion dataset - Ke YAN, accessed August 11, 2025, http://yanke23.com/articles/research/2018/06/13/DeepLesion-dataset-CVPR-2018.html
24. NIH DeepLesion Subset - Kaggle, accessed August 11, 2025, https://www.kaggle.com/datasets/kmader/nih-deeplesion-subset
25. DeepLesion: automated mining of large-scale lesion annotations and universal lesion detection with deep learning - PMC, accessed August 11, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC6052252/
26. Open Access to Medical Imaging Dataset Could Advance Computer ..., accessed August 11, 2025,

https://www.photonics.com/Articles/Open_Access_to_Medical_Imaging_Dataset_Could/a63709
27. DeepLesion (10,594 CT scans with lesions) - Academic Torrents, accessed August 11, 2025, https://academictorrents.com/details/de50f4d4aa3d028944647a56199c07f5fa6030ff
28. JimmyCai91/LesionHarvester: DeepLesion - GitHub, accessed August 11, 2025, https://github.com/JimmyCai91/LesionHarvester
29. A universal lesion detection method based on partially supervised learning - PMC, accessed August 11, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC10427860/
30. How to make NIfTI files generated by nibabel/dicom2nifti compatible with SimpleITK/ITK (avoid orthonormal error)? - Stack Overflow, accessed August 11, 2025, https://stackoverflow.com/questions/79702301/how-to-make-nifti-files-generated-by-nibabel-dicom2nifti-compatible-with-simplei
31. dcm2niix on Biowulf - NIH HPC, accessed August 11, 2025, https://hpc.nih.gov/apps/dcm2niix.html
32. dcm2nii: Tool/Resource Info - NITRC, accessed August 11, 2025, https://www.nitrc.org/projects/dcm2nii/
33. Image voxel orientation - NiBabel - nipy.org, accessed August 11, 2025, https://nipy.org/nibabel/image_orientation.html
34. Survey: interpolation methods in medical image processing - Semantic Scholar, accessed August 11, 2025, https://www.semanticscholar.org/paper/Survey%3A-interpolation-methods-in-medical-image-Deserno-G%C3%B6nner/cbcf63a1ee9c1a9de1fc056a2e527f525be21487
35. A Comparative Analysis of Image Interpolation Algorithms - ResearchGate, accessed August 11, 2025, https://www.researchgate.net/publication/294738469_A_Comparative_Analysis_of_Image_Interpolation_Algorithms
36. monai.transforms.composables — MONAI 0.1.0 documentation, accessed August 11, 2025, https://docs.monai.io/en/0.1.0/_modules/monai/transforms/composables.html
37. Hounsfield Unit - StatPearls - NCBI Bookshelf, accessed August 11, 2025, https://www.ncbi.nlm.nih.gov/books/NBK547721/
38. Windowing in CT-Scans - Kaggle, accessed August 11, 2025, https://www.kaggle.com/code/sparkyjunior/windowing-in-ct-scans
39. Window Classification of Brain CT Images in Biomedical Articles - PMC, accessed August 11, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC3540547/
40. intensity-normalization - PyPI, accessed August 11, 2025, https://pypi.org/project/intensity-normalization/
41. SynthStrip: Skull-Stripping for Any Brain Image - FreeSurfer, accessed August 11, 2025, https://surfer.nmr.mgh.harvard.edu/docs/synthstrip/
42. [Literature Review] Boosting Skull-Stripping Performance for Pediatric Brain Images, accessed August 11, 2025,

https://www.themoonlight.io/en/review/boosting-skull-stripping-performance-for-pediatric-brain-images
43. nnU-Net Revisited: A Call for Rigorous Validation in 3D Medical Image Segmentation - arXiv, accessed August 11, 2025, https://arxiv.org/html/2404.09556v2
44. 3D Deep Learning on Medical Images: A Review - PMC, accessed August 11, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC7570704/
45. How To Calculate GPU VRAM Requirements for an Large-Language Model, accessed August 11, 2025, https://apxml.com/posts/how-to-calculate-vram-requirements-for-an-llm
46. Modules — MONAI 1.5.0 Documentation, accessed August 11, 2025, https://docs.monai.io/en/stable/modules.html
47. Applications — MONAI 1.5.0 Documentation, accessed August 11, 2025, https://docs.monai.io/en/stable/apps.html
48. Data — MONAI 1.5.0 Documentation, accessed August 11, 2025, https://docs.monai.io/en/latest/data.html