Home Assignment - Senior Data Scientist - Samueli Institute

Approach 1: Deep Learning-Based Slice Classification

Goal:

Train a convolutional neural network (CNN) to classify slices and identify the middle of L3.

Input:

A folder containing axial CT slices in DICOM format (folder_path).

Output:

The index or path of the DICOM file corresponding to the middle of L3.

Pre-Processing:

1. **Load DICOM files**:
   Read all DICOM files, extract slices as 2D images.
2. **Normalize intensity**:
   Scale Hounsfield Units (HU) to a standard range (e.g., [-1000, 1000]).
3. **Resample slices**:
   Ensure slices have consistent voxel spacing (e.g., 1mm × 1mm × 1mm).
4. **Crop ROI**:
   Crop images to focus on the lumbar spine region to reduce irrelevant data.

Data Preparation:

Label slices: Annotate a dataset with the slice index corresponding to the middle of L3 (based on radiologist input).

Augmentation: Apply random transformations (e.g., rotation, contrast adjustment) to increase robustness.

Model Design:

Use a CNN architecture (e.g., ResNet, EfficientNet) for slice classification.

Design output layer as a regressor to predict the slice index corresponding to L3.

Training:

Use a labeled dataset where each slice is associated with a label (e.g., middle of L3 etc.).

Loss function: Mean Squared Error (MSE) for regression.

Metric: Mean Absolute Error (MAE) between predicted and true slice index.

Inference:

Input all slices from the CT scan.

Pass each slice through the model to predict the likelihood of being the middle of L3.

Return the slice with the highest likelihood.

Pseudo-Code structure:

```
def find_middle_l3_ct(scan_folder):
    slices = load_dicom_slices(scan_folder)  # Load DICOM files
    preprocessed_slices = preprocess_slices(slices)  # Normalize and crop
    model = load_trained_model("l3_model.pth")  # Load pre-trained CNN model
    predictions = model(preprocessed_slices)  # Get predictions
    l3_index = np.argmax(predictions)  # Find slice with max likelihood
    return slices[l3_index]  # Return corresponding DICOM file
```

Approach 2: Spine Segmentation and Anatomical Analysis

Goal:

Use a segmentation model to identify, locate L3, and extract its middle slice.

Input:

A folder containing axial CT slices in DICOM format (folder_path).

Output:

The index or path of the DICOM file corresponding to the middle of L3.

Pre-Processing:

Load and normalize DICOM files:

Convert all DICOM slices to 2D images, resample to uniform spacing.

Stack slices: Create a 3D volume from 2D slices.

Segmentation:

Train a U-Net or equivalent model to segment vertebrae from CT scans.

Output: A binary mask where each vertebra is uniquely labeled (e.g., L1=1, L2=2, etc.).

Post-Processing:

Use connected components to isolate individual vertebrae.

Identify the region corresponding to L3 based on anatomical landmarks (e.g., L1-L5 sequence).

Calculate the centroid of L3 in 3D space.

Middle Slice Identification:

Find the slice containing the largest cross-sectional area of L3.

Alternatively, compute the 3D centroid of L3 and identify the slice closest to its z-coordinate.

Inference:

Input a full CT scan.

Segment the spine, extract L3, and calculate the middle slice.

Pseudo-Code:

```
def find_middle_l3_ct_with_segmentation(scan_folder):
    slices = load_dicom_slices(scan_folder)  # Load DICOM files
    volume = create_3d_volume(slices)  # Stack slices into a 3D volume
    model = load_trained_segmentation_model("spine_unet.pth") # Load segmentation model
    segmentation_mask = model(volume)  # Get vertebrae segmentation mask
    l3_mask = extract_labeled_region(segmentation_mask, label=3)  # Isolate L3
    l3_middle_slice = find_largest_cross_section(l3_mask)  # Find middle slice of L3
    return slices[l3_middle_slice]  # Return corresponding DICOM file
```

Approach 3: Template Matching for L3 Shape

Goal:

Use a predefined template of the L3 vertebra and perform template matching to identify the slice that best matches the L3 shape.

Input:

Folder containing axial CT slices in DICOM format.

A 2D template image of the L3 vertebra.

Output:

The DICOM file corresponding to the middle L3 slice.

Load DICOM Files:

Read all DICOM files and convert slices into 2D grayscale images.

Pre-Process Slices:

Normalize intensity and resize slices to a consistent resolution.

Template Matching:

Use a 2D image of an L3 vertebra as a template.

Perform template matching on each slice using OpenCV's matchTemplate function.

Identify Best Match:

Find the slice with the highest matching score.

Approach 3 - Implementation:

```
import cv2
import numpy as np
import pydicom
import os

def find_middle_l3_by_template(folder_path, template_path):
    # Load template image
    template = cv2.imread(template_path, cv2.IMREAD_GRAYSCALE)
    template = cv2.resize(template, (128, 128))  # Resize template if needed

    # Load and preprocess DICOM slices
    dicom_files = [pydicom.dcmread(os.path.join(folder_path, f)) for f in os.listdir(folder_path)]
    slices = [dcm.pixel_array for dcm in dicom_files]

    # Perform template matching
    match_scores[] =
    for slice_img in slices:
        slice_img_resized = cv2.resize(slice_img, (128, 128))  # Resize slice for template matching
        result = cv2.matchTemplate(slice_img_resized, template, cv2.TM_CCOEFF_NORMED)
        match_scores.append(result.max())

    # Find slice with the best match
    best_match_index = np.argmax(match_scores)
    return dicom_files[best_match_index]  # Return DICOM file of the best match
```

Comparison of Approaches:

	Approach 1	**Approach 2**	**Approach 3**
Type of Approach	**Slice Classification**	**Segmentation & 3D Volume Analysis**	**Template Matching**
Model Complexity	Simple - focuses only on L3 identification.	More complex - requires full spine segmentation.	Simple - uses OpenCV.
Accuracy	Dependent on training data for L3.	More robust uses anatomical structure.	More precise - depends on template quality only.
Pre-Processing Time	**Fast** - processes slices independently.	**Slow** - requires 3D reconstruction.	**Very Fast** – only needs to match template.
Training Data	Slice-level labels for L3.	Fully segmented vertebrae annotations.	Requires a predefined L3 template.
Generalization	Limited to L3 identification, simpler and faster.	Adaptable to other vertebrae, provides more anatomical insight.	Can generalize with good templates.

These approaches are simple, fast to implement, and work well in scenarios where high accuracy isn't critical.

Elnatan