# practical

*Release 0.0.1*

**Ujjwal**

**Dec 27, 2024**

# CONTENTS:

CHAPTER

ONE

# 3D_CNN_MODEL MODULE

3D CNN Pipeline for TNM Stage Classification

This module loads 3D medical images from a specified directory, associates each patient scan with a TNM stage (from an Excel file), preprocesses and resizes the volumes, and then trains a 3D CNN model to classify patients into multiple TNM stages. Finally, the model is evaluated on a hold-out test set, and relevant metrics are printed.

3d_cnn_model.**build_3d_cnn**(*input_shape=(64, 64, 64, 1)*, *num_classes=5*)

Build a 3D CNN model for multiclass classification of TNM stages.

**Parameters**

- **input_shape** (*tuple, optional*) – Shape of the 3D input data, including channels. Defaults to (64, 64, 64, 1).
- **num_classes** (*int, optional*) – Number of output classes (e.g., TNM stages). Defaults to 5 (stages 0-4).

**Returns**

A compiled Keras Sequential model ready for training.

**Return type**

tensorflow.keras.models.Sequential

3d_cnn_model.**collect_image_data_and_labels**(*base_dir: str*, *tnm_df: DataFrame*, *output_size_3d: tuple = (64, 64, 64)*) → tuple

Traverse a directory of patient folders, load 3D medical images, resize them, normalize them, and collect their TNM labels.

This function expects each patient to have a corresponding row in the TNM DataFrame. Folders are skipped if they have no matching entry or if their images are not in NIfTI (.nii.gz) format with the expected 3D shape.

**Parameters**

- **base_dir** (*str*) – Path to the base directory containing patient subfolders.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing 'PatientID' and 'TNMStage' columns.
- **output_size_3d** (*tuple, optional*) – The desired 3D shape for resizing. Defaults to (64, 64, 64).

**Returns**

**A tuple of (image_data, labels, patient_ids).**

- image_data (np.ndarray): Preprocessed 3D volumes of shape (num_samples, 64, 64, 64).
- labels (np.ndarray): Array of TNM stage labels (one integer per sample).
- patient_ids (list): List of patient IDs corresponding to the volumes.

**Return type**

tuple

3d_cnn_model.**load_tnm_stage_labels**(*excel_path: str*) → DataFrame

Load TNM stage labels from an Excel file.

**Parameters**
**excel_path** (*str*) – Path to the Excel file containing TNM stage data. The file is expected to have a 'PatientID' column and a 'TNMStage' column.

**Returns**
DataFrame containing patient IDs and their corresponding TNM stages.

**Return type**
pd.DataFrame

3d_cnn_model.**main**()

Main function to orchestrate the data loading, preprocessing, model building, training, and evaluation pipeline.

**Steps:**

1. Load TNM labels from the provided Excel file.
2. Collect and preprocess image data from the specified directory.
3. Split data into training and testing sets, and one-hot encode labels.
4. Build and compile a 3D CNN model.
5. Train the model on the training set with a validation split (test set).
6. Evaluate the trained model using accuracy and classification metrics.

3d_cnn_model.**preprocess_data**(*image_data: ndarray*, *labels: ndarray*) → tuple

Remove samples with NaN labels, standardize each 3D volume, and split into train/test sets.

**Parameters**

- **image_data** (*np.ndarray*) – Array of shape (num_samples, 64, 64, 64) containing raw volume data.
- **labels** (*np.ndarray*) – Array of shape (num_samples,) containing stage labels.

**Returns**

**A 4-tuple of (X_train, X_test, y_train, y_test), where:**

- X_train (np.ndarray): Training images, shape (num_train_samples, 64, 64, 64, 1).
- X_test (np.ndarray): Testing images, shape (num_test_samples, 64, 64, 64, 1).
- y_train (np.ndarray): One-hot encoded training labels.
- y_test (np.ndarray): One-hot encoded testing labels.

**Return type**
tuple

CHAPTER

TWO

# MODEL MODULE

3D Volume Processing and Random Forest Classification

This module loads NIfTI medical images (using NiBabel), resamples them to a consistent shape, optionally applies PCA for dimensionality reduction, and trains a Random Forest classifier to predict TNM stages.

**Usage:**

- Configure the parameters in the "Configuration" section.
- Run this script to produce a classification report and accuracy score.

model.**collect_image_paths**(*base_dir: str*) → tuple

Collect all .nii.gz file paths from the base directory and track corresponding patient IDs.

**Parameters**
**base_dir** (*str*) – Path to the base directory containing subdirectories of patient data.

**Returns**
image_paths (list of str): List of complete file paths for each .nii.gz file. patient_ids (list of str): Corresponding patient IDs for each file path.

**Return type**
tuple

model.**determine_target_shape**(*image_paths: list*) → tuple

Determine the maximum shape across a collection of 3D images, which will serve as the target shape for resampling.

**Parameters**
**image_paths** (*list of str*) – A list of file paths to .nii.gz images.

**Returns**
The maximum shape (depth, height, width) found among all images.

**Return type**
tuple

model.**load_and_process_images**(*image_paths: list*, *patient_ids: list*, *tnm_df: DataFrame*, *target_shape: tuple*) → tuple

Load, resample, and flatten a set of 3D images, associating each with its TNM stage.

**Parameters**

- **image_paths** (*list of str*) – A list of file paths to .nii.gz images.
- **patient_ids** (*list of str*) – Patient IDs corresponding to the image_paths list.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing columns 'PatientID' and 'TNM-Stage'.
- **target_shape** (*tuple*) – Desired (depth, height, width) for resampling each image.

**Returns**
images (np.ndarray): Array of flattened images, shape (num_samples, depth*height*width). labels (np.ndarray): Array of TNM stage labels, shape (num_samples,). valid_patient_ids (list of str): List of patient IDs for which valid data was found.

**Return type**
tuple

model.**load_tnm_stage**(*tnm_df: DataFrame*, *patient_id: str*) → float

Retrieve the TNM stage for a given patient ID from a TNM DataFrame.

**Parameters**

- **tnm_df** (*pd.DataFrame*) – DataFrame containing columns 'PatientID' and 'TNM-Stage'.
- **patient_id** (*str*) – Unique patient identifier used to look up the TNM stage.

**Returns**
The TNM stage for the specified patient. Returns None if not found.

**Return type**
float

model.**main**() → None

**Main function to orchestrate the pipeline:**

1. Load TNM stage data from an Excel file.
2. Collect all .nii.gz file paths from the specified base directory.
3. Determine a target shape from the largest volume found.
4. Load and resample all images.
5. Optionally apply PCA dimensionality reduction.
6. Train a Random Forest classifier.
7. Evaluate the classifier using a test split.

model.**resample_image**(*image: ndarray*, *target_shape: tuple*) → ndarray

Resample the input 3D image to the specified target shape using linear interpolation.

**Parameters**

- **image** (*np.ndarray*) – A 3D NumPy array representing the image volume.
- **target_shape** (*tuple*) – The desired output shape (depth, height, width).

**Returns**
The resampled 3D image volume as a NumPy array.

**Return type**
np.ndarray

CHAPTER

THREE

# MODEL_GRADIENT_BOOSTING MODULE

2D Slice Classification using XGBoost

This module loads medical image slices (taken from the middle slice of 3D volumes, or directly 2D if available), associates each patient with a TNM stage (from an Excel file), preprocesses the data, and trains an XGBoost classifier. The performance is evaluated using cross-validation and a final train/test split.

model_gradient_boosting.**gather_middle_slice_data**(*base_dir: str*, *tnm_df: DataFrame*, *output_size_2d: tuple = (128, 128)*) → tuple

Traverse patient directories and extract the middle slice of 3D images (or the entire 2D image if it's already 2D). Resizes to output_size_2d and normalizes each slice before flattening.

**Parameters**

- **base_dir** (*str*) – Path to the base directory containing patient subfolders.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing 'PatientID' and 'TNMStage' columns.
- **output_size_2d** (*tuple, optional*) – Desired (height, width) for the 2D images. Defaults to (128, 128).

**Returns**

**(image_data, labels, patient_ids)**
image_data (np.ndarray): Flattened 2D slices, shape (num_samples, height*width). labels (np.ndarray): Array of TNM stage labels, shape (num_samples,). patient_ids (list of str): Patient IDs corresponding to the slices.

**Return type**
tuple

model_gradient_boosting.**load_tnm_stage_labels**(*excel_path: str*) → DataFrame

Load TNM stage labels from an Excel file.

**Parameters**
**excel_path** (*str*) – Path to the Excel file containing TNM data. The file should have 'PatientID' and 'TNMStage' columns.

**Returns**
DataFrame with the columns 'PatientID' and 'TNMStage'.

**Return type**
pd.DataFrame

model_gradient_boosting.**main**()

Main function to orchestrate data loading, preprocessing, XGBoost training, and evaluation on 2D medical image slices for TNM stage classification.

Steps: 1. Load TNM labels from Excel. 2. Gather 2D slice data (middle slice if 3D). 3. Preprocess data: remove NaNs, standardize, reduce dimensions with PCA. 4. Train and evaluate XGBoost via cross-validation and final test split.

model_gradient_boosting.**preprocess_data**(*image_data: ndarray*, *labels: ndarray*, *n_components: int = 90*) → tuple

Remove samples with NaN labels, standardize the feature matrix, and optionally apply PCA for dimensionality reduction.

**Parameters**

- **image_data** (*np.ndarray*) – 2D array of flattened slices, shape (num_samples, features).
- **labels** (*np.ndarray*) – TNM stage labels, shape (num_samples,).
- **n_components** (*int*) – Number of principal components to retain (for PCA). Defaults to 90.

**Returns**

image_data (np.ndarray): Transformed data after standardization and PCA. labels (np.ndarray): Cleaned labels (NaNs removed).

**Return type**

tuple

model_gradient_boosting.**train_and_evaluate_xgboost**(*image_data: ndarray*, *labels: ndarray*, *test_size: float = 0.2*) → None

**Train and evaluate an XGBoost classifier on the given data. Performs:**

1. 5-fold cross-validation to estimate general performance.
2. Final train/test split to report classification metrics.

**Parameters**

- **image_data** (*np.ndarray*) – Feature matrix of shape (num_samples, features).
- **labels** (*np.ndarray*) – TNM stage labels, shape (num_samples,).
- **test_size** (*float, optional*) – Proportion of the dataset used for test split. Defaults to 0.2 (i.e., 80% train, 20% test).

**Returns**

**Prints cross-validation accuracies, mean CV accuracy, test accuracy,**
and classification report.

**Return type**

None

CHAPTER

FOUR

# SEGMENT_ANYTHING_FINAL MODULE

Multi-task Feature Selection and Model Training with Segment Anything Model (SAM)

This script demonstrates a pipeline that: 1. Loads image metadata from a CSV file. 2. Uses a Segment Anything Model (SAM) to extract deep features from medical images. 3. Applies optional PCA for dimensionality reduction. 4. Employs a MultiTaskFeatureSelector to select features (using mutual information and XGBoost). 5. Performs patient-level stratified splitting for cross-validation. 6. Trains and evaluates an XGBoost classifier.

**class** segment_anything_final.**MultiTaskFeatureSelector**(*n_features_to_select=50*, *n_estimators=100*)

Bases: `object`

This class implements an ensemble-based feature selection approach that combines mutual information with XGBoost feature importances.

**n_features_to_select**

Number of top features to select.

**Type**
int

**n_estimators**

Number of estimators for the XGBoost classifier.

**Type**
int

**ensemble_selection**(*X*, *y*, *fold_seed=42*)

Combine mutual information and XGBoost importance for feature selection.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).
- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).
- **fold_seed** (*int, optional*) – Random seed used for consistent feature selection. Defaults to 42.

**Returns**

**Indices of the top *n_features_to_select* features**
chosen by the ensemble of mutual information and XGBoost.

**Return type**
np.ndarray

**mutual_information_selection**(*X*, *y*, *fold_seed=42*)

Select features based on mutual information, with a fold-specific random seed.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).

- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).
- **fold_seed** (*int, optional*) – Random seed for mutual_info_classif. Defaults to 42.

**Returns**

**Indices of the top *n_features_to_select* features**
based on mutual information.

**Return type**
np.ndarray

**xgboost_importance**(*X*, *y*)

Select features using XGBoost importance for multi-class data.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).
- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).

**Returns**

**Indices of the top *n_features_to_select* features**
based on XGBoost feature importances.

**Return type**
np.ndarray

segment_anything_final.**extract_sam_features**(*image_path*, *sam_model*, *device*)

Extract feature embeddings from a medical image using the Segment Anything Model (SAM).

**Parameters**

- **image_path** (*str*) – Path to a NIfTI (.nii, .nii.gz) image.
- **sam_model** (*nn.Module*) – Preloaded SAM model.
- **device** (*torch.device*) – Device to run computations on (e.g., 'cuda' or 'cpu').

**Returns**

- Mean-pooled SAM embeddings of shape (C,) if successful.
- None if an error occurs or no valid slices are processed.

**Return type**
np.ndarray or None

segment_anything_final.**main**()

Main entry point for loading data, extracting SAM features, applying PCA, and training/evaluating a multi-class XGBoost model.

**Steps:**

1. Load CSV metadata (image paths, labels, patient IDs).
2. Load and prepare the Segment Anything Model (SAM).
3. Extract features for each valid image slice.
4. Align features with labels and patient IDs.
5. Optionally apply PCA for dimensionality reduction.
6. Perform patient-level cross-validation with XGBoost.

segment_anything_final.**patient_stratified_split**(*X*, *y*, *patient_ids*, *test_size=0.4*, *seed=None*)

Perform a patient-level stratified train-test split.

**Ensures:**

1. All scans from a single patient are in either train or test set.
2. Class distribution is maintained across splits.
3. Representative sampling of each class.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features). (Not used in splitting, but kept for interface consistency.)
- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).
- **patient_ids** (*list[str] or np.ndarray*) – Identifiers (IDs) for each sample.
- **test_size** (*float, optional*) – Fraction of data to use for the test set. Defaults to 0.4 (i.e., 60% train, 40% test).
- **seed** (*int, optional*) – Random seed for reproducibility. Defaults to None.

**Returns**

train_indices (np.ndarray): Indices of training samples. test_indices (np.ndarray): Indices of test samples.

**Return type**

tuple

segment_anything_final.**train_and_evaluate**(*X*, *y*, *patient_ids*, *n_splits=5*)

Train and evaluate an XGBoost model using patient-level cross-validation.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).
- **y** (*np.ndarray*) – Integer-encoded class labels, shape (n_samples,).
- **patient_ids** (*np.ndarray or list[str]*) – Patient IDs (one per sample).
- **n_splits** (*int, optional*) – Number of cross-validation folds. Defaults to 5.

**Returns**

**A dictionary containing:**

'accuracies' (list[float]): Accuracy scores for each fold. 'balanced_accuracies' (list[float]): Balanced accuracy scores for each fold. 'aucs' (list[float]): Area under ROC curve (one-vs-rest) for each fold.

**Return type**

dict

CHAPTER

FIVE

# SUPPORT_VECTOR_CLASSIFIER MODULE

2D Medical Image Classification with SVM

This script performs the following steps: 1. Loads TNM stage labels from an Excel file. 2. Collects the middle slice (or entire 2D image if already 2D) from medical images in NIfTI format. 3. Resizes and normalizes each slice to a fixed size (128x128). 4. Flattens the slices into 1D feature vectors and performs standardization. 5. Optionally applies PCA for dimensionality reduction. 6. Trains and evaluates a Support Vector Machine (SVM) classifier on the preprocessed data.

support_vector_classifier.**collect_and_preprocess_images**(*base_dir: str*, *tnm_df: DataFrame*, *output_size_2d: tuple = (128, 128)*) → tuple

Collect medical images from a directory structure, extract middle slices for 3D volumes, resize them to output_size_2d, normalize, and flatten them into feature vectors.

**Parameters**

- **base_dir** (*str*) – Directory path containing patient subfolders with image data.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing columns 'PatientID' and 'TNM-Stage'.
- **output_size_2d** (*tuple, optional*) – The target (height, width) for 2D slices. Defaults to (128, 128).

**Returns**

**A 3-tuple:**

- image_data (np.ndarray): Array of shape (num_samples, height*width).
- labels (np.ndarray): TNM stage labels, shape (num_samples,).
- patient_ids (list): Patient IDs corresponding to each sample.

**Return type**
tuple

support_vector_classifier.**load_tnm_stages**(*excel_path: str*) → DataFrame

Load the TNM stage labels from an Excel file.

**Parameters**
**excel_path** (*str*) – Path to the Excel file containing TNM data with 'PatientID' and 'TNMStage' columns.

**Returns**
A DataFrame with the loaded TNM stage data.

**Return type**
pd.DataFrame

support_vector_classifier.**main**()

Main execution function: 1. Load TNM stage data. 2. Collect, resize, and preprocess 2D image slices. 3. Remove samples with NaN labels. 4. Standardize and apply PCA to reduce dimensionality. 5. Split data into training and testing sets. 6. Train and evaluate an SVM classifier.

support_vector_classifier.**remove_nan_labels**(*image_data: ndarray*, *labels: ndarray*) → tuple

Remove samples whose labels are NaN.

**Parameters**

- **image_data** (*np.ndarray*) – Feature matrix of shape (num_samples, features).
- **labels** (*np.ndarray*) – Label array of shape (num_samples,).

**Returns**
(cleaned_image_data, cleaned_labels)

**Return type**
tuple

support_vector_classifier.**standardize_and_reduce_dimension**(*image_data: ndarray*, *n_components: int = 40*) → ndarray

Standardize the features and optionally reduce dimensionality via PCA.

**Parameters**

- **image_data** (*np.ndarray*) – Feature matrix of shape (num_samples, features).
- **n_components** (*int, optional*) – Number of principal components. Defaults to 40.

**Returns**
Transformed feature matrix after standardization and PCA.

**Return type**
np.ndarray

support_vector_classifier.**train_svm_classifier**(*X_train: ndarray*, *y_train: ndarray*, *C: float = 1.0*, *gamma: str = 'auto'*, *kernel: str = 'rbf'*, *random_state: int = 42*) → SVC

Train an SVM classifier with specified parameters.

**Parameters**

- **X_train** (*np.ndarray*) – Training data, shape (n_samples, n_features).
- **y_train** (*np.ndarray*) – Training labels, shape (n_samples,).
- **C** (*float, optional*) – Regularization parameter. Defaults to 1.0.
- **gamma** (*str, optional*) – Kernel coefficient for certain kernels. Defaults to ‘auto’.
- **kernel** (*str, optional*) – Kernel type (‘linear’, ‘poly’, ‘rbf’, etc.). Defaults to ‘rbf’.
- **random_state** (*int, optional*) – Random seed. Defaults to 42.

**Returns**
Trained SVM model.

**Return type**
SVC

CHAPTER

SIX

# INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

# SEVEN

# MODULES

3D CNN Pipeline for TNM Stage Classification

This module loads 3D medical images from a specified directory, associates each patient scan with a TNM stage (from an Excel file), preprocesses and resizes the volumes, and then trains a 3D CNN model to classify patients into multiple TNM stages. Finally, the model is evaluated on a hold-out test set, and relevant metrics are printed.

3d_cnn_model.**build_3d_cnn**(*input_shape=(64, 64, 64, 1)*, *num_classes=5*)

Build a 3D CNN model for multiclass classification of TNM stages.

**Parameters**

- **input_shape** (*tuple, optional*) – Shape of the 3D input data, including channels. Defaults to (64, 64, 64, 1).
- **num_classes** (*int, optional*) – Number of output classes (e.g., TNM stages). Defaults to 5 (stages 0-4).

**Returns**

A compiled Keras Sequential model ready for training.

**Return type**

tensorflow.keras.models.Sequential

3d_cnn_model.**collect_image_data_and_labels**(*base_dir: str*, *tnm_df: DataFrame*, *output_size_3d: tuple = (64, 64, 64)*) → tuple

Traverse a directory of patient folders, load 3D medical images, resize them, normalize them, and collect their TNM labels.

This function expects each patient to have a corresponding row in the TNM DataFrame. Folders are skipped if they have no matching entry or if their images are not in NIfTI (.nii.gz) format with the expected 3D shape.

**Parameters**

- **base_dir** (*str*) – Path to the base directory containing patient subfolders.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing 'PatientID' and 'TNMStage' columns.
- **output_size_3d** (*tuple, optional*) – The desired 3D shape for resizing. Defaults to (64, 64, 64).

**Returns**

**A tuple of (image_data, labels, patient_ids).**

- image_data (np.ndarray): Preprocessed 3D volumes of shape (num_samples, 64, 64, 64).
- labels (np.ndarray): Array of TNM stage labels (one integer per sample).
- patient_ids (list): List of patient IDs corresponding to the volumes.

**Return type**

tuple

3d_cnn_model.**load_tnm_stage_labels**(*excel_path: str*) → DataFrame

Load TNM stage labels from an Excel file.

**Parameters**
**excel_path** (*str*) – Path to the Excel file containing TNM stage data. The file is expected to have a 'PatientID' column and a 'TNMStage' column.

**Returns**
DataFrame containing patient IDs and their corresponding TNM stages.

**Return type**
pd.DataFrame

3d_cnn_model.**main**()

Main function to orchestrate the data loading, preprocessing, model building, training, and evaluation pipeline.

**Steps:**

1. Load TNM labels from the provided Excel file.
2. Collect and preprocess image data from the specified directory.
3. Split data into training and testing sets, and one-hot encode labels.
4. Build and compile a 3D CNN model.
5. Train the model on the training set with a validation split (test set).
6. Evaluate the trained model using accuracy and classification metrics.

3d_cnn_model.**preprocess_data**(*image_data: ndarray*, *labels: ndarray*) → tuple

Remove samples with NaN labels, standardize each 3D volume, and split into train/test sets.

**Parameters**

- **image_data** (*np.ndarray*) – Array of shape (num_samples, 64, 64, 64) containing raw volume data.
- **labels** (*np.ndarray*) – Array of shape (num_samples,) containing stage labels.

**Returns**

**A 4-tuple of (X_train, X_test, y_train, y_test), where:**

- X_train (np.ndarray): Training images, shape (num_train_samples, 64, 64, 64, 1).
- X_test (np.ndarray): Testing images, shape (num_test_samples, 64, 64, 64, 1).
- y_train (np.ndarray): One-hot encoded training labels.
- y_test (np.ndarray): One-hot encoded testing labels.

**Return type**
tuple

3D Volume Processing and Random Forest Classification

This module loads NIfTI medical images (using NiBabel), resamples them to a consistent shape, optionally applies PCA for dimensionality reduction, and trains a Random Forest classifier to predict TNM stages.

**Usage:**

- Configure the parameters in the "Configuration" section.
- Run this script to produce a classification report and accuracy score.

model.**collect_image_paths**(*base_dir: str*) → tuple

Collect all .nii.gz file paths from the base directory and track corresponding patient IDs.

**Parameters**
**base_dir** (*str*) – Path to the base directory containing subdirectories of patient data.

**Returns**
image_paths (list of str): List of complete file paths for each .nii.gz file. patient_ids (list of str): Corresponding patient IDs for each file path.

**Return type**
tuple

model.**determine_target_shape**(*image_paths: list*) → tuple

Determine the maximum shape across a collection of 3D images, which will serve as the target shape for resampling.

**Parameters**
**image_paths** (*list of str*) – A list of file paths to .nii.gz images.

**Returns**
The maximum shape (depth, height, width) found among all images.

**Return type**
tuple

model.**load_and_process_images**(*image_paths: list*, *patient_ids: list*, *tnm_df: DataFrame*, *target_shape: tuple*) → tuple

Load, resample, and flatten a set of 3D images, associating each with its TNM stage.

**Parameters**

- **image_paths** (*list of str*) – A list of file paths to .nii.gz images.
- **patient_ids** (*list of str*) – Patient IDs corresponding to the image_paths list.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing columns 'PatientID' and 'TNM-Stage'.
- **target_shape** (*tuple*) – Desired (depth, height, width) for resampling each image.

**Returns**
images (np.ndarray): Array of flattened images, shape (num_samples, depth*height*width). labels (np.ndarray): Array of TNM stage labels, shape (num_samples,). valid_patient_ids (list of str): List of patient IDs for which valid data was found.

**Return type**
tuple

model.**load_tnm_stage**(*tnm_df: DataFrame*, *patient_id: str*) → float

Retrieve the TNM stage for a given patient ID from a TNM DataFrame.

**Parameters**

- **tnm_df** (*pd.DataFrame*) – DataFrame containing columns 'PatientID' and 'TNM-Stage'.
- **patient_id** (*str*) – Unique patient identifier used to look up the TNM stage.

**Returns**
The TNM stage for the specified patient. Returns None if not found.

**Return type**
float

model.**main**() → None

**Main function to orchestrate the pipeline:**

1. Load TNM stage data from an Excel file.
2. Collect all .nii.gz file paths from the specified base directory.
3. Determine a target shape from the largest volume found.
4. Load and resample all images.

5. Optionally apply PCA dimensionality reduction.
6. Train a Random Forest classifier.
7. Evaluate the classifier using a test split.

model.**resample_image**(*image: ndarray*, *target_shape: tuple*) → ndarray

Resample the input 3D image to the specified target shape using linear interpolation.

**Parameters**

- **image** (*np.ndarray*) – A 3D NumPy array representing the image volume.
- **target_shape** (*tuple*) – The desired output shape (depth, height, width).

**Returns**
The resampled 3D image volume as a NumPy array.

**Return type**
np.ndarray

2D Slice Classification using XGBoost

This module loads medical image slices (taken from the middle slice of 3D volumes, or directly 2D if available), associates each patient with a TNM stage (from an Excel file), preprocesses the data, and trains an XGBoost classifier. The performance is evaluated using cross-validation and a final train/test split.

model_gradient_boosting.**gather_middle_slice_data**(*base_dir: str*, *tnm_df: DataFrame*, *output_size_2d: tuple = (128, 128)*) → tuple

Traverse patient directories and extract the middle slice of 3D images (or the entire 2D image if it's already 2D). Resizes to output_size_2d and normalizes each slice before flattening.

**Parameters**

- **base_dir** (*str*) – Path to the base directory containing patient subfolders.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing 'PatientID' and 'TNMStage' columns.
- **output_size_2d** (*tuple, optional*) – Desired (height, width) for the 2D images. Defaults to (128, 128).

**Returns**

**(image_data, labels, patient_ids)**
image_data (np.ndarray): Flattened 2D slices, shape (num_samples, height*width). labels (np.ndarray): Array of TNM stage labels, shape (num_samples,). patient_ids (list of str): Patient IDs corresponding to the slices.

**Return type**
tuple

model_gradient_boosting.**load_tnm_stage_labels**(*excel_path: str*) → DataFrame

Load TNM stage labels from an Excel file.

**Parameters**
**excel_path** (*str*) – Path to the Excel file containing TNM data. The file should have 'PatientID' and 'TNMStage' columns.

**Returns**
DataFrame with the columns 'PatientID' and 'TNMStage'.

**Return type**
pd.DataFrame

model_gradient_boosting.**main**()

Main function to orchestrate data loading, preprocessing, XGBoost training, and evaluation on 2D medical image slices for TNM stage classification.

Steps: 1. Load TNM labels from Excel. 2. Gather 2D slice data (middle slice if 3D). 3. Preprocess data: remove NaNs, standardize, reduce dimensions with PCA. 4. Train and evaluate XGBoost via cross-validation and final test split.

model_gradient_boosting.**preprocess_data**(*image_data: ndarray*, *labels: ndarray*, *n_components: int = 90*) → tuple

Remove samples with NaN labels, standardize the feature matrix, and optionally apply PCA for dimensionality reduction.

**Parameters**

- **image_data** (*np.ndarray*) – 2D array of flattened slices, shape (num_samples, features).
- **labels** (*np.ndarray*) – TNM stage labels, shape (num_samples,).
- **n_components** (*int*) – Number of principal components to retain (for PCA). Defaults to 90.

**Returns**

image_data (np.ndarray): Transformed data after standardization and PCA. labels (np.ndarray): Cleaned labels (NaNs removed).

**Return type**

tuple

model_gradient_boosting.**train_and_evaluate_xgboost**(*image_data: ndarray*, *labels: ndarray*, *test_size: float = 0.2*) → None

**Train and evaluate an XGBoost classifier on the given data. Performs:**

1. 5-fold cross-validation to estimate general performance.
2. Final train/test split to report classification metrics.

**Parameters**

- **image_data** (*np.ndarray*) – Feature matrix of shape (num_samples, features).
- **labels** (*np.ndarray*) – TNM stage labels, shape (num_samples,).
- **test_size** (*float, optional*) – Proportion of the dataset used for test split. Defaults to 0.2 (i.e., 80% train, 20% test).

**Returns**

**Prints cross-validation accuracies, mean CV accuracy, test accuracy,**
and classification report.

**Return type**

None

Multi-task Feature Selection and Model Training with Segment Anything Model (SAM)

This script demonstrates a pipeline that: 1. Loads image metadata from a CSV file. 2. Uses a Segment Anything Model (SAM) to extract deep features from medical images. 3. Applies optional PCA for dimensionality reduction. 4. Employs a MultiTaskFeatureSelector to select features (using mutual information and XGBoost). 5. Performs patient-level stratified splitting for cross-validation. 6. Trains and evaluates an XGBoost classifier.

**class** segment_anything_final.**MultiTaskFeatureSelector**(*n_features_to_select=50*, *n_estimators=100*)

Bases: object

This class implements an ensemble-based feature selection approach that combines mutual information with XGBoost feature importances.

**n_features_to_select**

Number of top features to select.

**Type**
int

**n_estimators**

Number of estimators for the XGBoost classifier.

**Type**
int

**ensemble_selection**(*X*, *y*, *fold_seed=42*)

Combine mutual information and XGBoost importance for feature selection.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).
- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).
- **fold_seed** (*int, optional*) – Random seed used for consistent feature selection. Defaults to 42.

**Returns**

**Indices of the top *n_features_to_select* features**
chosen by the ensemble of mutual information and XGBoost.

**Return type**
np.ndarray

**mutual_information_selection**(*X*, *y*, *fold_seed=42*)

Select features based on mutual information, with a fold-specific random seed.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).
- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).
- **fold_seed** (*int, optional*) – Random seed for mutual_info_classif. Defaults to 42.

**Returns**

**Indices of the top *n_features_to_select* features**
based on mutual information.

**Return type**
np.ndarray

**xgboost_importance**(*X*, *y*)

Select features using XGBoost importance for multi-class data.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).
- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).

**Returns**

**Indices of the top *n_features_to_select* features**
based on XGBoost feature importances.

**Return type**
np.ndarray

segment_anything_final.**extract_sam_features**(*image_path*, *sam_model*, *device*)

Extract feature embeddings from a medical image using the Segment Anything Model (SAM).

**Parameters**

- **image_path** (*str*) – Path to a NIfTI (.nii, .nii.gz) image.
- **sam_model** (*nn.Module*) – Preloaded SAM model.
- **device** (*torch.device*) – Device to run computations on (e.g., 'cuda' or 'cpu').

**Returns**

- Mean-pooled SAM embeddings of shape (C,) if successful.
- None if an error occurs or no valid slices are processed.

**Return type**
np.ndarray or None

segment_anything_final.**main**()

Main entry point for loading data, extracting SAM features, applying PCA, and training/evaluating a multi-class XGBoost model.

**Steps:**

1. Load CSV metadata (image paths, labels, patient IDs).
2. Load and prepare the Segment Anything Model (SAM).
3. Extract features for each valid image slice.
4. Align features with labels and patient IDs.
5. Optionally apply PCA for dimensionality reduction.
6. Perform patient-level cross-validation with XGBoost.

segment_anything_final.**patient_stratified_split**(*X*, *y*, *patient_ids*, *test_size=0.4*, *seed=None*)

Perform a patient-level stratified train-test split.

**Ensures:**

1. All scans from a single patient are in either train or test set.
2. Class distribution is maintained across splits.
3. Representative sampling of each class.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features). (Not used in splitting, but kept for interface consistency.)
- **y** (*np.ndarray*) – Integer-encoded labels, shape (n_samples,).
- **patient_ids** (*list[str] or np.ndarray*) – Identifiers (IDs) for each sample.
- **test_size** (*float, optional*) – Fraction of data to use for the test set. Defaults to 0.4 (i.e., 60% train, 40% test).
- **seed** (*int, optional*) – Random seed for reproducibility. Defaults to None.

**Returns**
train_indices (np.ndarray): Indices of training samples. test_indices (np.ndarray): Indices of test samples.

**Return type**
tuple

segment_anything_final.**train_and_evaluate**(*X*, *y*, *patient_ids*, *n_splits=5*)

Train and evaluate an XGBoost model using patient-level cross-validation.

**Parameters**

- **X** (*np.ndarray*) – Feature matrix of shape (n_samples, n_features).
- **y** (*np.ndarray*) – Integer-encoded class labels, shape (n_samples,).
- **patient_ids** (*np.ndarray or list[str]*) – Patient IDs (one per sample).
- **n_splits** (*int, optional*) – Number of cross-validation folds. Defaults to 5.

**Returns**

**A dictionary containing:**
'accuracies' (list[float]): Accuracy scores for each fold. 'balanced_accuracies' (list[float]): Balanced accuracy scores for each fold. 'aucs' (list[float]): Area under ROC curve (one-vs-rest) for each fold.

**Return type**
dict

2D Medical Image Classification with SVM

This script performs the following steps: 1. Loads TNM stage labels from an Excel file. 2. Collects the middle slice (or entire 2D image if already 2D) from medical images in NIfTI format. 3. Resizes and normalizes each slice to a fixed size (128x128). 4. Flattens the slices into 1D feature vectors and performs standardization. 5. Optionally applies PCA for dimensionality reduction. 6. Trains and evaluates a Support Vector Machine (SVM) classifier on the preprocessed data.

support_vector_classifier.**collect_and_preprocess_images**(*base_dir: str*, *tnm_df: DataFrame*, *output_size_2d: tuple = (128, 128)*) → tuple

Collect medical images from a directory structure, extract middle slices for 3D volumes, resize them to output_size_2d, normalize, and flatten them into feature vectors.

**Parameters**

- **base_dir** (*str*) – Directory path containing patient subfolders with image data.
- **tnm_df** (*pd.DataFrame*) – DataFrame containing columns 'PatientID' and 'TNM-Stage'.
- **output_size_2d** (*tuple, optional*) – The target (height, width) for 2D slices. Defaults to (128, 128).

**Returns**

**A 3-tuple:**

- image_data (np.ndarray): Array of shape (num_samples, height*width).
- labels (np.ndarray): TNM stage labels, shape (num_samples,).
- patient_ids (list): Patient IDs corresponding to each sample.

**Return type**
tuple

support_vector_classifier.**load_tnm_stages**(*excel_path: str*) → DataFrame

Load the TNM stage labels from an Excel file.

**Parameters**
**excel_path** (*str*) – Path to the Excel file containing TNM data with 'PatientID' and 'TNM-Stage' columns.

**Returns**
A DataFrame with the loaded TNM stage data.

**Return type**
pd.DataFrame

support_vector_classifier.**main**()

Main execution function: 1. Load TNM stage data. 2. Collect, resize, and preprocess 2D image slices. 3. Remove samples with NaN labels. 4. Standardize and apply PCA to reduce dimensionality. 5. Split data into training and testing sets. 6. Train and evaluate an SVM classifier.

support_vector_classifier.**remove_nan_labels**(*image_data: ndarray*, *labels: ndarray*) → tuple

Remove samples whose labels are NaN.

**Parameters**

- **image_data** (*np.ndarray*) – Feature matrix of shape (num_samples, features).
- **labels** (*np.ndarray*) – Label array of shape (num_samples,).

**Returns**
(cleaned_image_data, cleaned_labels)

**Return type**
tuple

support_vector_classifier.**standardize_and_reduce_dimension**(*image_data: ndarray*, *n_components: int = 40*) → ndarray

Standardize the features and optionally reduce dimensionality via PCA.

**Parameters**

- **image_data** (*np.ndarray*) – Feature matrix of shape (num_samples, features).
- **n_components** (*int, optional*) – Number of principal components. Defaults to 40.

**Returns**
Transformed feature matrix after standardization and PCA.

**Return type**
np.ndarray

support_vector_classifier.**train_svm_classifier**(*X_train: ndarray*, *y_train: ndarray*, *C: float = 1.0*, *gamma: str = 'auto'*, *kernel: str = 'rbf'*, *random_state: int = 42*) → SVC

Train an SVM classifier with specified parameters.

**Parameters**

- **X_train** (*np.ndarray*) – Training data, shape (n_samples, n_features).
- **y_train** (*np.ndarray*) – Training labels, shape (n_samples,).
- **C** (*float, optional*) – Regularization parameter. Defaults to 1.0.
- **gamma** (*str, optional*) – Kernel coefficient for certain kernels. Defaults to ‘auto’.
- **kernel** (*str, optional*) – Kernel type (‘linear’, ‘poly’, ‘rbf’, etc.). Defaults to ‘rbf’.
- **random_state** (*int, optional*) – Random seed. Defaults to 42.

**Returns**
Trained SVM model.

**Return type**
SVC

CHAPTER

EIGHT

# INDICES AND TABLES

- genindex
- modindex
- search

# PYTHON MODULE INDEX

## 3



## m



## s

# INDEX

## Symbols



## B



## C



## D



## E



## G



## L



## M



## N



## P



## R



## S



## T

## X