TNM Staging of Hepatocellular Carcinoma Using Clinical Image Data (CT and MR)

Ujjwal Yadav
Data and Computer Science, Heidelberg University

March 4, 2025

Abstract

Medical image and cancer staging are increasingly important tasks in clinical practice, especially for personalized treatment planning. This work focuses on predicting the TNM stage (ranging from 0 to 4) of cancer patients. Over the course of this study, a dataset of approximately 50 GB—corresponding to 100 patients—was preprocessed, feature extractions, and stage classification experiments. Multiple machine learning and deep learning approaches were tested, including gradient boosting, random forests, a 3D CNN, and ultimately the Segment Anything Model (SAM) [4]. for robust feature extraction. The main contribution is demonstrating the feasibility and challenges of integrating a generic segmentation model (SAM) into a multi-class classification workflow for TNM stage prediction, as well as evaluating the performance using various metrics. The highest performance with SAM was approximately 49.7% accuracy and a mean AUC of 55.27% when merging the rarely occurring TNM stage 0 with stage 1. The results highlight the complexity of clinical data handling and the necessity of robust feature extraction and patient-level cross-validation.

1 Introduction

1.1 General Relevance of the Topic

Cancer staging plays a crucial role in determining the prognosis and guiding the treatment strategy for patients. The TNM classification system—where T refers to the size/extent of the primary tumor, N denotes lymph node involvement, and M indicates metastasis—provides a standardized way of describing cancer spread. Accurate staging is paramount for selecting appropriate therapies and evaluating outcomes.

Imaging modalities such as CT and MRI are widely used to gather detailed anatomical and pathological information.

1.2 Existing Challenges and Open Issues

Despite advances in imaging technology and machine learning:

- **Data Inconsistency and Quality**: Medical images frequently come from multiple centers or scanners with different acquisition protocols, leading to inconsistent image quality and metadata.
- **Feature Extraction and Computation**: Radiomics-based methods can be computationally expensive, especially if many high-dimensional features are extracted from large 3D volumes.
- **Class Imbalance**: Certain TNM stages (like stage 0 in this project) have

very few samples, making robust classification difficult. Moreover, some patients have as few as 0 scans and some have upto more than 50 scans.

1.3 Introduction on How to Overcome Those Challenges

The project addressed these challenges by:

- Converting DICOM data to a standardized NIfTI format [3]. and performing thorough data cleaning (removing incomplete patient records).
- Applying systematic preprocessing steps, including resizing, padding, and normalization, to ensure consistent input shapes for model training.
- Assessing scan quality and consistency using specialized metrics [1]. to reduce the risk of faulty inputs.
- Adopting patient-level cross-validation splits to avoid data leakage and better reflect real-world clinical scenarios.

1.4 Overview of the Proposed Approach

The core pipeline involved:

1. **Data Preparation**: Converting DICOM to NIfTI, extracting metadata, matching scans to TNM labels, and discarding incomplete cases.
2. **Feature Extraction**: Experimenting with various methods, including a 2D Gradient Boosting approach, classical machine learning models (Random Forest and Support Vector Classifier), a 3D CNN, and eventually the Segment Anything Model (SAM) for extracting deep features from MRI/CT volumes.
3. **Classification**: Training an XGBoost classifier on extracted features to predict TNM stages.
4. **Evaluation**: Measuring accuracy, balanced accuracy, and area under the ROC curve (AUC) across multiple cross-validation folds.

1.5 Hypothesis

Using the Segment Anything Model encoder to extract features from MR and CT images will facilitate more accurate classification of TNM stages as compared to standard 2D or 3D CNN approaches, especially when combined with an appropriate feature selection and patient-level cross-validation approach.

1.6 Results and Achievements

- Achieved approximately 49.7% accuracy and a mean AUC of 55.27% when merging stage 0 with stage 1 (due to low occurrence of stage 0).
- Demonstrated that simpler models (gradient boosting, random forest) performed around the low 60% accuracy range on the dataset without extensive feature engineering, while a 3D CNN yielded around 43.8% accuracy.
- Confirmed the complexity inherent in multi-class classification for TNM prediction using real-world clinical images.

1.7 Main Advantages of the Proposed Method

- **Generic Segmentation Backbone**: SAM does not require time-consuming manual annotations for initial segmentation.
- **Systematic Feature Selection**: An ensemble of mutual information and gradient-boosted feature importance helps select top features robustly.
- **Patient-level Splitting**: Ensures that the classification results are more realistic and generalizable.

1.8 Structure of This Thesis

- **Two: Theoretical Background** – An overview of image segmentation and classification approaches, followed by a review of related state-of-the-art methods for cancer staging.
- **Three: Material and Method** – Details of the dataset, preprocessing steps, feature extraction workflows, and model training pipelines.
- **Four: Results and Evaluation** – Presentation of experimental results, system configuration, metrics, and comparisons with other approaches.
- **Five: Discussion** – Interpretation of the findings, their significance, and limitations.
- **Six: Conclusion and Outlook** – Summary of the work, conclusions drawn, and future research directions.

2 Theoretical Background

2.1 Section One: Theoretical Background

Feature extraction methods (both handcrafted radiomic features and deep learned features) aim to quantify image regions for classification or regression tasks.

Classification in a medical setting involves mapping these features to clinical labels, such as tumor stage or prognosis. Machine learning models (e.g., random forest, XGBoost) and deep neural networks (2D or 3D CNNs) are frequently used. However, robust training often requires balanced, high-quality data and strategies to prevent overfitting, especially when data is limited.

2.2 Section Two: State of the Arts

Recently, attention has shifted to advanced neural architectures and large-scale models for segmentation and feature extraction. Transfer learning from large, generic models can help address data scarcity. Models like the Segment Anything Model (SAM) leverage extensive training on diverse images, potentially reducing the need for domain-specific supervised segmentation data. Combined with ensemble or advanced classifiers, these approaches aim to improve multi-class label prediction accuracy.

2.3 References

Here are the major resources which aided me during the experiments:

- Dicom nifti conversion
- NiBabel Documentation
- PyRadiomics Features
- Segment Anything GitHub
- MRQy GitHub Repository

3 Material and Method

3.1 Section One: Material

3.1.1 Used Dataset

- **Size and Scope**: Approximately 50 GB of data from 100 patients provided to me by our mentor for the practical : Wenzhao Zhao
- **Modalities**: CT (658 Scans whose kVp ranges between 80 and 130) and MRI (1243 scans)(T1 which is ideal for visualizing anatomical structures like tumors) in DICOM format which I later converted to NIfTI (nii / nii.gz).

- **Labels**: Each patient had one or more scans on different dates, each possibly corresponding to a different TNM stage (0 through 4). An Excel file mapped patient IDs to their known TNM stages.

3.1.2 Used Toolboxes and Preprocessing Steps

- **NIfTI Conversion Tools**:
  - Tool Used: dcm2niix (v1.0 or later).
  - Purpose: Converts DICOM files to NIfTI(niigz) format while preserving essential metadata such as pixel dimensions, orientation, and acquisition parameters.
  - Reason: Ensures compatibility with downstream analysis pipelines and integration with Python-based tools for preprocessing and feature extraction.
- **Preprocessing Scripts**:
  - Steps Included:
    * Resizing and Padding: Resized scans to a uniform spatial resolution with padding applied to ensure consistent dimensions.
    * Normalization: Scaled intensities to the range $[0, 1]$ to standardize input for machine learning models.
    * Mask Generation: Generated masks using **Otsu thresholding** [5]. implemented in `scikit-image`.
  - Implementation: Python scripts utilizing `NumPy`, `SciPy`, and `scikit-image`.
  - Reason: Standardized input data ensures compatibility and improves model training consistency.
- **Quality Check Metrics**:
  - Metrics Used: Derived metadata fields, including spatial resolution, slice thickness, and acquisition parameters.
  - Tool: Custom Python script to validate metadata consistency across patients and flag outliers.
  - Reference: Metrics from MRQy [1]. formulas and tags available here.
- **Feature Extraction**:
  - Deep Learning: Feature extraction using pretrained models (e.g., Segment Anything Model (SAM)) implemented in PyTorch.
  - Machine Learning: Feature selection and classification implemented using `scikit-learn` and `XGBoost`.
  - Dimensionality Reduction: Applied PCA using `scikit-learn` to reduce feature dimensionality while retaining significant variance.
  - Reason: Pyradiomics [2]. was tested for feature extraction but was computationally expensive (6 hours for a single patient). SAM(vit-b) was used instead for automatic image feature extraction and out of them top 50 features were kept and fed to the classifier.
  - References: Libraries:
    * PyTorch: https://pytorch.org
    * scikit-learn: https://scikit-learn.org
    * XGBoost: https://xgboost.readthedocs.io
- **Patient-Level Cross-Validation (n=5)**:

- Implementation: Custom Python script grouped scans by patient IDs, divided patients into 5 folds using stratified sampling to maintain class balance, and ensured no data leakage by keeping all scans from a patient in the same fold.
- Reason: Provided reliable evaluation of model performance and ensured reproducibility of the pipeline.

3.2 Section Two: Method

3.2.1 An Overview of the Proposed Method (Flowchart or Pseudocode)

1. **Data Ingestion**
   (a) Read patient scans (NIfTI format).
   (b) Read metadata (patient ID, scan date, TNM labels).
   (c) Relate each scan to a TNM stage value based on matching the date of scan acquisition and the TNM stage of the patient that we have corresponding on that day.
   (d) Data is excluded in following criteria:
      i. TNM stage of the patient is unavailable.
      ii. The image is corrupted.
      iii. The image is not available for the patient.
2. **Preprocessing**
   (a) Perform basic normalization (zero-mean, unit variance).
   (b) Resize volumes to a standard dimension (128×128 for 2D where we use the middle slice, or 64×64×64 for 3D where all the slices are used and no data is lost) . In the particular case of SAM model which only uses 1024×1024 images, I used bilinear interpolation for resizing each slice to this desired standard .
   (c) Apply or generate masks using Otsu thresholding if there is no provided mask for the tumor.
3. **Feature Extraction**
   (a) SAM-based Embedding Extraction: For each slice in the volume, process the normalized image through the SAM encoder.
   (b) Collect slice-level features and compute an aggregate feature (e.g., mean across slices).
4. **Feature Selection**
   (a) Mutual information-based feature ranking.
   (b) XGBoost-based feature importance.
   (c) Combine and retain 50 top features.
5. **Classification**
   (a) Split data at the patient level to create training/validation folds.
   (b) Train XGBoost on selected features.
   (c) Evaluate predictions with accuracy, balanced accuracy, and AUC.

3.2.2 Describe Each Step of the Proposed Algorithm

Formulas and Parameters:

- **Normalization**:

$$\hat{x} = \frac{x - \mu}{\sigma + \varepsilon},$$

where μ and σ are slice-level mean and standard deviation, and ε is a small constant (e.g., 1×10^{-7}).

- **Mutual Information (MI) for Feature Selection**:

$$MI(X, y) = \sum_{x \in X} \sum_{y \in Y} p(x, y) \log \frac{p(x, y)}{p(x)\, p(y)}.$$

- **XGBoost Feature Importance**: Internally computed by tree splitting across multiple estimators.

Summing up all main steps covered above in a flowchart :

1. Load Data & Labels
2. Metadata Matching
3. Preprocessing (Normalization, Resizing)
4. Feature Extraction with SAM (slice-wise)
5. Aggregate Slices (mean or median pooling)
6. Feature Selection (MI + XGBoost)
7. Train Classifier (XGBoost)
8. Evaluate on Held-Out Folds

4 Results and Evaluation

4.1 System Configuration

All experiments were run on a workstation with:

- **GPU**: NVIDIA GeForce RTX 3090
- **RAM**: 24 GB.
- **Software**: Python 3.8 environment with PyTorch 1.8.1, scikit-learn, XGBoost, and supporting libraries.

4.2 The Main Objective to Prove or Disprove the Hypothesis

The main objective was to assess whether a large-scale pretrained segmentation model (SAM) could improve TNM stage prediction accuracy compared to baseline methods (2D gradient boosting, random forests, 3D CNN, etc.).

Evaluation Metrics:

- **Accuracy**:

$$\frac{\text{Number of correct predictions}}{\text{Total predictions}}.$$

- **Balanced Accuracy**: Average recall across all classes.
- **AUC (One-vs-Rest)**: Measures the area under the ROC curve for multi-class scenarios by handling each class separately.

4.3 Section One: Results for different Proposed Method

The following section presents the results obtained for different classifiers used during the experimentation. The classifiers and their corresponding accuracies are summarized in Table 1. Below, we provide further analysis and comments on the performance of each classifier.

4.3.1 Gradient Boosting (2D Slices, 128×128)

Gradient Boosting was applied to 2D slices of size 128×128. The model achieved an accuracy of approximately 63.8%. However, the slice-based approach inherently limits the model's ability to capture 3D contextual information, which may be critical for tasks involving volumetric data.

4.3.2 3D CNN (64×64×64 Volumes)

A 3D Convolutional Neural Network (3D CNN) was applied to $64 \times 64 \times 64$ volumes. The accuracy was relatively low, around 50.8%. This was likely due to the large memory requirements of 3D networks and the insufficient number of samples available for effective training.

Network Architecture

The 3D CNN used in this implementation consists of the following architecture:

- **Input Layer:** A 3D input tensor of shape $(64, 64, 64, 1)$ representing the preprocessed medical imaging data.
- **Convolutional Layers:** Three 3D convolutional layers with the following configurations:
  - Layer 1: 32 filters with a kernel size of $(3, 3, 3)$ and ReLU activation, followed by batch normalization and max-pooling of size $(2, 2, 2)$.
  - Layer 2: 64 filters with a kernel size of $(3, 3, 3)$ and ReLU activation, followed by batch normalization and max-pooling of size $(2, 2, 2)$.
  - Layer 3: 128 filters with a kernel size of $(3, 3, 3)$ and ReLU activation, followed by batch normalization and max-pooling of size $(2, 2, 2)$.
- **Dense Layers:** A fully connected dense layer with 256 units and ReLU activation, followed by a dropout layer with a rate of 0.4. This is connected to the output layer with 5 units (corresponding to the number of classes) and a softmax activation function.
- **Output Layer:** Produces a probability distribution over the 5 TNM stages.

Training Procedure

The training process involved the following steps:

- **Data Preprocessing:** The medical imaging data was normalized and resized to ensure consistent dimensions $(64, 64, 64)$, and labels were one-hot encoded for multiclass classification.
- **Optimization:** The network was trained using the Adam optimizer with a learning rate of 0.001.
- **Loss Function:** Categorical cross-entropy was used as the loss function since this is a multiclass classification problem.
- **Epochs and Batch Size:** The model was trained for 10 epochs with a batch size of 32.
- **Validation:** Validation accuracy and loss were monitored during training to ensure the model's performance was improving on unseen data.

Hyperparameter Optimization Process

The hyperparameter optimization was conducted using the following steps:

- **Search Space:** The search space for hyperparameters included:
  - Filters in convolutional layers: Tuned over discrete values (16 to 256 with appropriate steps).
  - Dense units: Tuned over discrete values (64 to 256 with steps of 64).
  - Dropout rate: Tuned over continuous values in $[0.3, 0.7]$.
  - Learning rate: Tuned over discrete values in $[0.001, 0.0001]$.
- **Tuning Method:** Keras Tuner's `Hyperband` method was used to efficiently explore the hyperparameter

search space by running trials with increasing resource allocation.

- **Evaluation Metric:** Validation accuracy was used as the evaluation metric for hyperparameter selection.
- **Best Hyperparameters:** The best hyperparameters obtained from the tuning process were then used to build and train the final model.

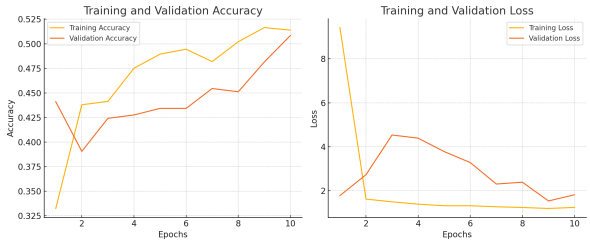

Figure 1: Training and validation accuracy and loss curves for the 3D CNN model.

4.3.3 Random Forest

The Random Forest classifier achieved an accuracy of approximately 62%. While less computationally demanding than Gradient Boosting, it still suffered from limitations in feature representation, highlighting the need for better feature extraction methods.

Classifier	**Accuracy (%)**
Gradient Boosting (2D slices, 128×128)	63.8 ± 0.03723
3D CNN ($64 \times 64 \times 64$ volumes)	50.8 ± 0.04077
Random Forest	62.0 ± 0.35812

Table 1: Accuracy of Different Classifiers

4.4 Section Two: Results and Evaluation of the SAM based approach

The evaluation and measurements were conducted using SAM-based deep features combined with the XGBoost classifier. The results are summarized in Table 2. Further explanations of the results and observations are provided below.

Table 2: Results of SAM-Based Deep Features with XGBoost Classifier

Evaluation Scenario	**Accuracy (%)**
Classification across five TNM stages (0–4)	43.6 ± 0.0282
After merging stage 0 with stage 1	49.7 ± 0.0119

4.4.1 Explanations and Observations

1. **Classification Across Five TNM Stages (0–4)**: The classification achieved an accuracy of approximately 43.6%. This relatively low accuracy can be attributed to the very small number of samples in stage 0, which significantly impacts the classifier's ability to generalize across all stages.

2. **Merging Stage 0 with Stage 1**: By merging stage 0 with stage 1, the accuracy improved to approximately 49.7%, with a mean AUC of around 55.27%. This improvement suggests that combining stages with very small sample sizes can mitigate the class imbalance issue, leading to better classification performance.

3. **Key Insights**: While the accuracy remains modest, these results demonstrate the feasibility of using SAM-based embeddings for TNM staging. With more balanced data and potentially improved feature representations, significant performance improvements could be achieved in future work.

5 Discussion of the Results

The results suggest that while large, generic segmentation models (like SAM) can offer a convenient pipeline for extracting features from clinical MR and CT scans, the classification outcome is still constrained by factors such as class imbalance (rare stage 0) and the variability inherent in multi-institutional imaging data , for this , changing TNM of the single patient with 0 TNM to 1 , the accuracy improved by 6 percent. The use of a patient-level cross-validation

strategy provided a realistic estimate of generalizability, emphasizing that achieving higher accuracy requires a sufficiently large and balanced dataset.

Moreover, the feasibility of applying deep embeddings from pretrained segmentation models indicates potential for future improvements if more domain-specific finetuning is performed. The discrepancy in performance between simpler 2D approaches and resource-intensive 3D CNNs reinforces the importance of carefully managing the computational load versus the added benefit of 3D context. Balancing the dataset or generating synthetic examples for rare TNM stages might also lead to higher classification accuracy. Overall, integrating SAM into the classification workflow moves the field forward by streamlining segmentations and feature extractions, but additional refinements—such as advanced domain adaptations—remain necessary. Limitations included the unavailability of certain patients' data and the computational cost of some radiomics pipelines, which were ultimately abandoned.

6 Conclusion and Outlook and Learning

This work set out to build a pipeline for MR and CT image segmentation and TNM stage prediction for cancer patients. The primary achievement was successfully experimenting with various segmentation and classification techniques, culminating in the use of the Segment Anything Model (SAM) for deep feature extraction and an XGBoost classifier for stage prediction.

A Short Summary of the Work

- **Objectives**: Streamline data preprocessing (including DICOM-to-NIfTI conversion, metadata extraction, and quality checks) and evaluate multiple classification methods.
- **Implementation**: Demonstrated a robust pipeline capable of handling complex, multi-institutional imaging data.
- **Results**: A maximum of $\sim$49.7% accuracy and $\sim$55.27% mean AUC was attained after merging rare TNM stages (0 with 1). Gradient boosting, random forest, and a 3D CNN showed varying degrees of success, all significantly close to the SAM-based approach.

Extent to Which Objectives Were Accomplished

The modest performance indicates that while SAM offers a promising route for universal feature extraction, further data augmentation, domain-specific finetuning, or targeted balancing of rare classes could potentially boost accuracy. All statements are grounded in the experimental metrics and cross-validation results.

Practical Applications and Future Steps

- **Domain-Specific Fine-Tuning**: Adapting SAM or a similar large model specifically to cancer segmentation tasks.
- **Augmentation of Underrepresented Stages**: Synthetic data generation for rare TNM stages to address class imbalance.
- **Hybrid Models**: Combining SAM embeddings with classical radiomic features or more advanced transformers.
- **Scalability**: Extending this pipeline to larger, multi-center datasets to validate robustness and improve generalizability.

By addressing these points, future iterations of this project can aim for higher accuracy and better clinical applicability, ul-

timately assisting in more reliable and automated cancer staging workflows.

Learning Outcomes

Through this practical, I learned to apply the data science pipeline to a real-world problem, from data preprocessing (e.g., DICOM-to-NIfTI conversion, metadata extraction, and quality checks) to implementing and evaluating segmentation and classification models. Experimenting with various methods, such as Gradient Boosting, Random Forest, 3D CNNs, and SAM-based feature extraction combined with XGBoost, provided me with insights into model selection and performance comparison. I also understood the importance of addressing challenges like class imbalance, as seen in merging rare TNM stages to improve accuracy. Additionally, I gained experience in analyzing results, identifying limitations, and proposing improvements, such as domain-specific fine-tuning, synthetic data generation, and hybrid modeling approaches. Overall, this project enhanced my ability to design scalable and robust solutions for complex medical imaging problems.

References

[1]. *MRQy for quality check metrics* https://github.com/viswanath-lab/MRQy

[2]. *Pyradiomics* https://pyradiomics.readthedocs.io/en/latest/features.html

[3]. *Dicom-nifti conversion* https://gist.github.com/sarthakpati/45d94c408e377ff168f8e9f816f780f2

[4]. *Segment Anything Model* https://github.com/facebookresearch/segment-anything?tab=readme-ov-file

[5]. *Otsu thresholding* https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_thresholding.html