

Hybrid Deep Learning Framework for Liver Tumor Detection

Anmol Rajesh Patil
Dept. of Computer Science
Global Academy of
Technology Bengaluru, India

Bhargav M
Dept. of Computer Science
Global Academy of Technology
Bengaluru, India

Bhuvan M
Dept. of Computer Science
Global Academy of
Technology Bengaluru, India

Hitesh H
Dept. of Computer Science
Global Academy of
Technology Bengaluru, India

Prof. Thejeswini A.M
Assistant Professor, CSE
Global Academy of Technology
Bengaluru, India

Dr. R Rajkumar
HOD, CSE
Global Academy of
Technology Bengaluru, India

Abstract—Hepatocellular carcinoma (HCC) remains a significant cause of cancer-related mortality, emphasizing the importance of reliable computational methods for early risk assessment and tumour identification. Existing computer-aided diagnostic approaches frequently analyse clinical information and medical images separately, limiting the integration of their complementary diagnostic contributions. This study proposes a sequential framework for liver tumour risk assessment and segmentation, combining clinical evaluation with CT-based image analysis through two independently developed learning modules. The clinical module employs Extreme Gradient Boosting (XGBoost) to analyse patient clinical parameters and estimate the risk of liver tumour pathology. Based on the resulting risk assessment, cases identified as high risk are directed to the subsequent imaging stage for further analysis. The imaging module adopts a two-and-a-half-dimensional (2.5D) segmentation architecture that integrates an ImageNet-pretrained ConvNeXt-Tiny encoder with a U-Net decoder to delineate tumour regions in CT images. By combining adjacent slices into a multi-channel input, the model incorporates inter-slice contextual information while retaining a two-dimensional processing strategy. To improve interpretability, Seg-Grad-CAM is employed to generate visual explanations of the segmentation process by identifying influential encoder features and incorporating the predicted tumour-probability mask into the activation maps. The clinical and imaging modules remain architecturally independent, allowing each model to operate according to its respective learning objective without joint optimisation. The framework is evaluated using the Liver Tumor Segmentation (LiTS) dataset and a synthetic clinical cohort to assess its segmentation and clinical risk-prediction performance, respectively. The proposed approach establishes a structured diagnostic workflow that combines clinical risk stratification, automated tumour delineation, and visual interpretability, offering a potential foundation for computer-assisted liver tumour assessment.

Index Terms—Hepatocellular carcinoma (HCC), Liver tumour segmentation, XGBoost, ConvNeXt-Tiny, 2.5D U-Net, Seg-Grad-CAM, Clinical risk assessment, Computed tomography (CT).

I. INTRODUCTION

LIVER cancer, of which hepatocellular carcinoma (HCC) is the principal primary form, remains a major chal-

lenge for diagnostic medicine. Artificial intelligence (AI) is increasingly examined as an aid across the diagnostic pathway, from image interpretation to risk estimation [1], [2]. The practical value of detecting a liver tumor early lies in the treatment options it preserves. A lesion found while it is small and confined to the liver can be managed in ways that are no longer available once the disease has progressed. For this reason, surveillance and diagnostic workflows for at-risk patients depend on tools that flag suspicious findings reliably and early, and that communicate those findings in a form a clinician can check.

In practice, two broad categories of evidence inform an HCC work-up. The first is serum laboratory testing. Alpha-fetoprotein (AFP), its lectin-reactive fraction (AFP-L3%), and des-gamma-carboxy prothrombin (DCP, also called PIVKA-II) are tumor-associated markers. Albumin, total bilirubin, and platelet count reflect the underlying condition of the liver, which is frequently compromised by chronic viral hepatitis or cirrhosis. The second category is cross-sectional imaging, most commonly computed tomography (CT), which allows a lesion to be located, delineated, and measured directly. The two categories answer different questions. Laboratory values indicate whether biochemical risk is elevated but cannot say where a lesion is. Imaging shows where a lesion is and how large it is, but CT review is time-consuming, and small or low-contrast lesions are easy to miss on a single slice.

Conventional practice interprets these evidence streams separately and leaves their synthesis to the treating clinician. Manual review of lab reports requires transcribing values from heterogeneous report layouts. Manual CT reading requires slice-by-slice inspection of volumes that can contain hundreds of slices. Both are labor-intensive, and both depend on the reader. Deep learning has substantially advanced automated analysis of liver CT: convolutional neural networks (CNNs) have been used to detect tumors [3], [4], and segmentation networks have been used to delineate the liver and its lesions [5], [6]. In parallel, machine-learning models trained on

structured clinical data have been proposed to estimate liver cancer risk [7], [8]. However, the image-only models tend to be computationally heavy and do not always fit into a clinician's everyday routine, while models built only on clinical data lack the spatial evidence that imaging provides.

These observations have motivated hybrid and multimodal approaches that draw on both clinical and imaging information [9]–[12]. Such systems are motivated by the complementarity described above. The design question they raise, however, is not only *whether* to use both modalities but *how* to combine them. Joint-embedding fusion architectures merge clinical and imaging representations into a single learned prediction. This can be effective, but it also makes it difficult to attribute a conclusion to one source of evidence or the other. That property matters for a decision-support tool, whose role is to help a clinician reason, not to replace that reasoning with an unexplained verdict. Explainability methods such as gradient-weighted class activation mapping (Grad-CAM) [13] and attention mechanisms [14] address part of this concern for imaging models, but they are typically demonstrated on classification networks and not on segmentation networks, which produce no single class score to explain.

This work addresses that gap with an implemented, staged framework, in which clinical and imaging evidence are each assessed by a dedicated and inspectable component and then presented together without being fused. The framework follows the order of a typical diagnostic encounter. First, a laboratory report is uploaded and its markers are extracted by optical character recognition (OCR). The user confirms the values, which are then assessed by a transparent, threshold-based rule engine and, independently, by a gradient-boosted decision-tree (XGBoost) classifier whose individual trees can be inspected. When the evidence indicates elevated risk, the workflow proceeds to CT analysis. There, a U-Net segmentation network with a ConvNeXt-Tiny encoder operates on 2.5D inputs of three adjacent slices and predicts a tumor mask. Tumor location, size, and intensity statistics are then derived from that mask, and a segmentation-adapted Grad-CAM (Seg-Grad-CAM) highlights the image regions that drove the prediction.

An earlier design of the project did fuse modalities: a neural clinical model produced an embedding that was combined with imaging features. That design was retired when the clinical model was replaced by XGBoost, because a gradient-boosted tree ensemble does not produce the differentiable embedding such a fusion network needs. The present framework therefore treats the stages as independent by design, and this paper reports each stage on its own terms.

The contributions of this paper are as follows:

- 1) A staged, non-fused clinical and imaging decision-support architecture. OCR-based lab-report parsing, a deterministic rule engine, a standalone clinical XGBoost model, and a CT segmentation model each produce their own verdict.
- 2) A 2.5D liver tumor segmentation pipeline built on an ImageNet-pretrained ConvNeXt-Tiny encoder and a U-Net decoder. It is trained on a volume-level, leakage-safe split of the full LiTS dataset and optimized with a combined binary cross-entropy and Tversky loss that deliberately weights missed tumor pixels more heavily.
- 3) A Seg-Grad-CAM explainability module for a segmentation network. It backpropagates from the aggregated tumor probability, hooks an intermediate encoder stage whose resolution suits small lesions, and gates the resulting map by the model's own predicted tumor probability.
- 4) A leakage-safe clinical modeling pipeline over eleven laboratory and demographic features: a stratified split, preprocessing fitted on the training data only, cross-validated hyperparameter search within the training split, early stopping on a validation split, and a single evaluation on the test split.
- 5) A complete implementation as a FastAPI service with a React front end, covering human-in-the-loop correction of OCR output, decision-tree inspection, derived tumor measurements.

The remainder of this paper is organized as follows. Section II reviews related work and identifies the research gap. Section III describes the imaging and clinical data. Section IV presents the methodology of each stage. Section V describes the system implementation. Section VI defines the evaluation protocol and metrics. Section VII reports the results. Section VIII discusses them. Section ?? states the limitations, and Section IX concludes the paper and outlines future work.

II. RELATED WORK

This section organizes prior work into five themes that correspond to the components of the proposed framework: deep learning for liver tumor detection in CT, liver and tumor segmentation, clinical-data-driven risk prediction, hybrid and multimodal integration, and explainability. It closes by stating the gap this work addresses.

A. Deep Learning for Liver Tumor Detection in CT

The application of deep learning to radiology for cancer detection has been reviewed broadly [15], and tumor detection techniques based on deep learning have been surveyed across organ sites [16]. In the liver specifically, CNN-based detection of tumors in CT images has been reported [3], and deep-learning-based medical image analysis for liver cancer detection has been studied as a general framework [4]. Comprehensive reviews of AI in liver cancer diagnosis place these imaging methods within the wider diagnostic workflow [1], and reviews of AI-based medical diagnosis systems more generally discuss the conditions under which such tools can support clinical decisions [2]. Architectural diversity has also grown. Attention-based CNNs have been proposed to direct the network toward lesion-relevant regions [14], and vision transformers have been applied to liver tumor detection in CT [17]. Across these works, the common input is the CT image alone, and the common output is a detection or classification decision.

B. Liver and Tumor Segmentation

Detection establishes whether a tumor is present. Segmentation establishes where it is and how large it is, which is the

information needed for measurement and localization. Automated liver segmentation with deep learning has been studied in its own right [6], and multi-scale deep-learning approaches have been proposed for liver tumor segmentation to handle lesions that vary widely in size [5]. A survey of deep-learning-based liver tumor segmentation and classification summarizes the progression of these methods and the evaluation practices used to compare them [18]. Two recurring difficulties motivate several design choices in this paper. First, liver tumors are small relative to the image, so tumor pixels form a small minority class. Second, lesion appearance on a single slice can be ambiguous. The first difficulty bears on the choice of loss function and on how evaluation metrics should be read. The second motivates providing context from adjacent slices to an otherwise two-dimensional network.

C. Clinical-Data-Driven Risk Prediction

A separate line of work estimates liver cancer risk from structured clinical data. Clinical data-driven risk prediction models for liver cancer detection have been developed [7]. Deep learning has been applied to liver disease classification from clinical data [8]. Hybrid machine-learning models have been used for cancer prediction from clinical datasets [19], and machine-learning techniques for clinical data analysis in healthcare have been reviewed [20]. These models operate on quantities such as serum markers, liver-function tests, and patient history. They are inexpensive to compute and are available early in the diagnostic pathway, but they cannot localize a lesion.

D. Hybrid and Multimodal Integration

Several recent works combine clinical and imaging data. Deep-learning-based liver tumor detection using hybrid clinical and imaging data has been reported [9]. Multimodal deep-learning frameworks for liver cancer diagnosis combine CT with clinical data [10]. Hybrid AI models integrate structured clinical features with imaging for liver tumor detection [11]. More broadly, hybrid AI systems for medical diagnosis using clinical and imaging data have been described as a general design pattern [12]. These works share the premise that the two modalities are complementary. They differ in where the combination happens, whether at the feature level, at the decision level, or through a learned joint representation. A learned joint representation requires both branches to produce compatible, differentiable embeddings, which constrains the choice of the clinical model.

E. Explainability in Liver Cancer AI

For clinical adoption, a model's output must be interpretable by the clinician who acts on it. Grad-CAM has been combined with CNNs for explainable liver cancer detection [13], producing heatmaps that indicate which image regions contributed to a classification. Attention-based networks offer a related, architecture-level route to interpretability [14]. Both routes are most naturally defined for networks that output a class score. A segmentation network outputs a dense per-pixel map instead,

and adapting gradient-based explanation to that output requires choosing a scalar target and a feature layer, choices that the classification setting leaves implicit. On the clinical side, tree-based models offer a different kind of transparency, because their individual decision paths can be read directly.

F. Research Gap

Three observations follow from this review. First, imaging-only and clinical-only systems each capture part of the diagnostic picture, which motivates the hybrid systems of [9]–[12]. Second, when integration is implemented as learned fusion, it becomes harder to attribute a conclusion to one evidence stream, and it restricts the clinical component to models that emit differentiable embeddings. This excludes tree ensembles, whose structure can otherwise be inspected. Third, gradient-based explainability has mainly been demonstrated for classification networks [13], whereas localization and size estimation require a segmentation network. What remains less explored is an end-to-end, implemented pipeline that (i) begins from the lab report a clinician actually receives, (ii) assesses clinical risk with components that are each inspectable, (iii) segments tumors with an imaging model that uses inter-slice context, (iv) explains the segmentation output itself, and (v) presents the separate verdicts together without collapsing them into a single fused score. The framework described in the following sections is designed to address this combination of requirements.

III. MATERIALS AND DATA

The framework uses two independent data sources: a public CT segmentation dataset for the imaging stage and a synthetically generated clinical cohort for the clinical stage. The two are not paired. No clinical record corresponds to any CT volume, which is consistent with the independent-stage design described in Section I.

A. CT Imaging Data: LiTS

The segmentation model is trained and evaluated on the Liver Tumor Segmentation (LiTS) dataset, distributed as pre-extracted PNG slices. The dataset comprises 131 CT volumes (identifiers 0–130) and approximately 58.6k axial slices in total. Each slice is stored as a 256×256 image, accompanied by two binary masks with pixel values in $\{0, 1\}$: a liver mask and a lesion (tumor) mask. The three file types reside in a single directory and are paired by volume and slice indices encoded in their filenames. The project's scope is tumor localization, not full liver segmentation. Only the tumor mask is used as a training target. The liver mask is not used as a training target.

A companion index file distributed with the PNG release includes per-slice "mask empty" flags and its own train/test partition. Neither is used. Spot checks during development found the emptiness flags to be inconsistent with the actual mask content, and the partition covered only 91 of the 131 volumes and did not match the required 70/15/15 ratio. Tumor presence is instead computed directly from mask pixels throughout the pipeline.

B. Volume-Level Data Partitioning

Adjacent CT slices within a volume are highly similar. If slices from the same volume were distributed across the training and test partitions, the model would be evaluated on near-duplicates of its training data, and the reported scores would be inflated. To prevent this leakage, the partitioning operates on whole volumes: every slice of a given volume is assigned to exactly one of the training, validation, or test splits. Volumes are first stratified by whether they contain any tumor-positive slice, again determined from mask pixels. Each stratum is then shuffled with a fixed seed (42) and divided in a 70/15/15 ratio, so that tumor-bearing volumes are represented proportionally in every split. The resulting assignment is persisted to disk. A verification routine confirms that no volume identifier appears in more than one split.

C. Full LiTS Training Dataset

The segmentation model is trained on the full LiTS dataset. All approximately 58.6k slices of the 131 volumes are used, and every slice remains in the split assigned to its volume in Section III-B: the 92 training volumes supply the training samples, the 20 validation volumes are used for model selection and early stopping, and the 19 test volumes are held out for the final evaluation. No slice is discarded, so the training data include tumor-bearing slices, liver slices without tumor, and slices outside the liver. Slices without tumor supply negative context, which teaches the model to return an empty mask when no lesion is present.

Using every slice of every volume also preserves slice-to-slice adjacency throughout each volume, which the 2.5D input depends on. For each center slice, the previous and next slices of the same volume are always available, except at the first and last slices of a volume, where the center slice is repeated in the missing neighbor channel. Because the split is defined at the volume level and no volume is reassigned, the full dataset inherits the leakage protection of the volume-level partition.

D. Clinical Data: Synthetic HCC Cohort

The clinical stage uses a synthetic tabular dataset of 5,000 patients, one row per patient, with balanced classes. The data are generated programmatically by the project's own generator. Its distributions are modeled on published diagnostic-accuracy characteristics of AFP, AFP-L3%, and DCP and on typical laboratory profiles in cirrhosis and chronic viral hepatitis. The data are not real patient data and are used for model prototyping and education only.

The HCC-negative cohort is built from overlapping clinical scenarios chosen so that it is not trivially separable from the positive cohort. The scenarios comprise healthy individuals; hepatitis B or C carriers without HCC; cirrhosis with or without viral etiology; abnormal liver-function tests without HCC; and deliberate marker false positives, in which AFP, DCP, AFP-L3%, or several markers are elevated without HCC. The HCC-positive cohort is generated per patient from conditional distributions. Cirrhosis is the principal axis (approximately 60% of positives are cirrhotic); etiology is split roughly evenly

TABLE I
CLINICAL FEATURE SET, REFERENCE RANGES, AND CLASS-WISE MEDIAN IN THE SYNTHETIC COHORT

Feature	Unit	Normal range	HCC=0	HCC=1
AFP	ng/mL	< 20	3.40	70.00
AFP-L3	%	< 10	4.30	14.25
DCP (PIVKA-II)	mAU/mL	< 40	23.60	100.00
Albumin	g/dL	3.5–5.0	3.90	3.20
Total bilirubin	mg/dL	0.1–1.2	0.88	1.92
Platelet count	$\times 10^9/L$	150–450	200	125
Age	years	18–90	48	60
Sex	M/F	–	(categorical)	
HBV status	0/1	–	(binary)	
HCV status	0/1	–	(binary)	
Cirrhosis status	0/1	–	0	1

between hepatitis B, hepatitis C, and non-viral causes; and a latent disease-stage variable (early, intermediate, or advanced) biases marker levels and liver-function values. A fraction of positives is generated as AFP-negative HCC, reflecting the known imperfect sensitivity of AFP.

Each record carries the eleven features listed in Table I, a patient identifier, and the binary HCC label. The table also reports the class-wise medians computed from the generated dataset, which illustrate the separation the generator encodes. The positive class has higher tumor-marker levels, lower albumin and platelet counts, higher bilirubin, and a higher prevalence of cirrhosis. The dataset contains no missing values. The clinical data are partitioned by a stratified 70/15/15 split with seed 42 into 3,500 training, 750 validation, and 750 test patients, each split balanced at 50% HCC-positive. Because each row is a distinct patient with no repeat visits, a row-wise stratified split is free of leakage. The patient-level split is persisted with the patient identifiers for auditability.

E. Synthetic Laboratory Reports

To exercise the OCR stage end to end, the project includes a generator of synthetic hepatology laboratory reports in PDF form. The reports contain a fictional letterhead, a patient and accession metadata block, and result tables for the tumor markers, the hepatic panel, and viral and cirrhosis status. Patient values are drawn from the same statistical distributions used for the clinical dataset. A candidate report is retained only if both the trained clinical XGBoost model and the deterministic rule engine confidently agree with its intended label. This construction ensures that the printed values, the rule-engine score, and the XGBoost verdict on each report are consistent. Ten such reports are included, five labeled HCC-positive and five HCC-negative. They are used as realistic test inputs for the complete workflow and not as a training or evaluation set.

IV. PROPOSED METHODOLOGY

A. Overall System Architecture

This model is organized as three sequential stages that mirror a diagnostic encounter. The overall architecture is shown in Fig. 1. Two input pathways run in parallel. In the

clinical pathway, an uploaded laboratory report passes through OCR and marker extraction and is then assessed by the rule engine and the clinical XGBoost classifier, which yield a clinical-risk verdict. In the imaging pathway, uploaded CT slices are assembled into a 2.5D stack and passed to the ConvNeXt-Tiny U-Net. Its output yields a tumor mask, the derived location and size, and a Seg-Grad-CAM explanation. Both pathways are served by a single FastAPI back end and presented side by side in a React front end.

The workflow is gated in the user interface. After the laboratory values have been confirmed and assessed, the CT stage is unlocked when the rule engine recommends imaging. The clinical XGBoost model's recommendation is displayed alongside the rule engine's verdict as independent supporting evidence. The remainder of this section describes each component in the order in which data flow through the system.

B. Laboratory Report OCR and Parsing

The user uploads a laboratory report as a PDF or an image. A PDF is detected by its declared content type, its file extension, or its leading %PDF signature, which makes the detection robust to mislabeled uploads. Each PDF page is rasterized at 200 dpi with PyMuPDF, and the resulting page images, or the uploaded image directly, are passed to the Tesseract OCR engine. The recognized text is then parsed line by line with pattern-matching rules.

1) *Marker extraction*: For each marker, a set of label patterns identifies its result row. For example, AFP is matched by *alpha-fetoprotein* or by the token *AFP* when it is *not* followed by *L3*; a negative look-ahead prevents the AFP-L3 row from being read as AFP. The reported value is taken as the first numeric token that follows the matched label on the same line. On typical report layouts this token precedes the unit and reference-range columns. Qualitative results are parsed differently. Hepatitis B surface antigen and anti-HCV results are read as *reactive* or *non-reactive*, and cirrhosis status as *present* or *absent*. In each case the parser takes the leftmost match after the label, so that a reference-range column later on the same line (for example, "Non-reactive") cannot overwrite the actual result.

2) *Demographics extraction*: Patient name, identifier, age, sex, and study date are extracted by capturing the text that follows each label, bounded by the next known label on the line. This bound is needed because reports commonly place two label-value pairs on one line.

3) *Missing values and human review*: A field that cannot be located with confidence is returned as missing. It is never guessed at the OCR layer. The extracted values, the demographics, and the raw OCR text are presented in an editable review form, and the user must confirm or correct them before any assessment runs. This human-in-the-loop step is a deliberate safety measure, because OCR can misread digits.

C. Deterministic Rule Engine

The confirmed marker values are first scored by a rule engine that contains no learned parameters. The engine assigns

TABLE II
 RULE-ENGINE THRESHOLDS AND POINT ASSIGNMENTS

Marker	Condition	Points
AFP (ng/mL)	> 200 (markedly elevated)	+2
	> 10 and ≤ 200 (mildly elevated)	+1
AFP-L3 (%)	> 10	+1
DCP (mAU/mL)	> 150 (markedly elevated)	+2
	> 40 and ≤ 150 (elevated)	+1
Albumin (g/dL)	< 3.5	+1
Platelets (×10 ⁹ /L)	< 150	+1
Total bilirubin (mg/dL)	> 1.2	+1
CT recommended if total score ≥ 2 (at least one primary marker read).		

points for each marker that exceeds its clinical threshold, as listed in Table II, and sums them into a score

$$S_{\text{rule}} = \sum_{m \in \mathcal{M}} p_m(x_m), \quad (1)$$

where $\mathcal{M}$ is the set of available markers, x_m is the confirmed value of marker m , and $p_m(\cdot)$ is the point function in Table II. Markers that are missing contribute nothing. The three primary tumor markers (AFP, AFP-L3%, DCP) can each contribute up to two points. The three supporting liver-function markers contribute at most one point each, reflecting their role as indirect evidence. The engine recommends CT imaging when

$$S_{\text{rule}} \geq 2, \quad (2)$$

provided that at least one primary tumor marker is available. If none of AFP, AFP-L3%, or DCP could be read, the engine returns an *unknown* risk level and asks the user to enter them, instead of issuing a triage decision from supporting markers alone. Alongside the score, the engine returns a flag for each marker (normal, borderline, high, or low) and a list of plain-language reasons for every point awarded, so that its verdict can be traced to specific values.

D. Clinical XGBoost Risk Model

In parallel with the rule engine, the confirmed values are passed to a learned clinical-risk model. Its design follows the observation that clinical data alone support useful risk prediction [7], [19], and it adds the transparency of a tree ensemble.

1) *Feature contract and preprocessing*: The model consumes the eleven features of Table I. These are the three tumor markers (AFP, AFP-L3%, DCP), the three liver-status values (albumin, total bilirubin, platelet count), two demographic values (age and sex), and three etiology flags (HBV, HCV, and cirrhosis status). The patient identifier is excluded from the feature matrix. Preprocessing is implemented as a column transformer with three branches. Numeric features are median-imputed and standardized. Binary status flags are imputed with the most frequent value. Sex is imputed with the most frequent value and one-hot encoded into a single indicator. The dataset contains no missing values; the imputers are retained so that the deployed model can accept the partially filled forms that OCR may produce. The transformer is fitted on the training split only and then applied unchanged to the validation and

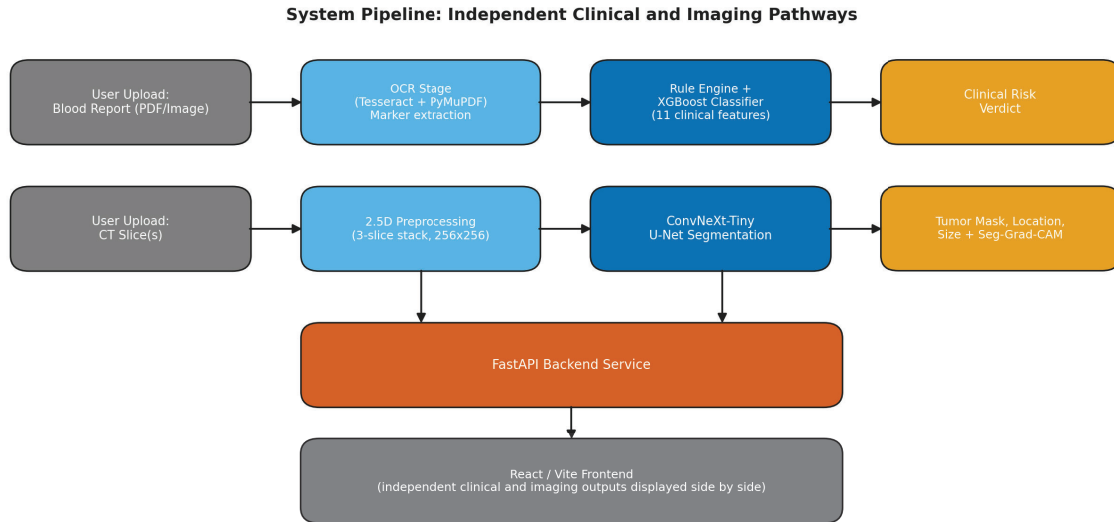


Fig. 1. Overall architecture of the model. The clinical pathway (top) extracts markers from an uploaded laboratory report by OCR and assesses them with a rule engine and an XGBoost classifier over eleven clinical features. The imaging pathway (middle) builds a 2.5D three-slice stack and segments tumors with a ConvNeXt-Tiny U-Net, producing the tumor mask, location, size, and a Seg-Grad-CAM map. Both pathways are served by a FastAPI back end and displayed side by side, without fusion, in a React/Vite front end.

test splits and at inference time. Standardization maps each numeric feature x_j to

$$\tilde{x}_j = \frac{x_j - \mu_j^{\text{train}}}{\sigma_j^{\text{train}}}, \quad (3)$$

where μ_j^{train} and σ_j^{train} are the mean and standard deviation estimated on the training split. Using training-split statistics alone keeps the validation and test data from influencing the preprocessing.

2) *Model*: The classifier is a gradient-boosted ensemble of K regression trees trained with the binary logistic objective. For a preprocessed feature vector $\tilde{\mathbf{x}}$, the predicted probability of HCC is

$$\hat{p} = \sigma \left(\sum_{k=1}^K f_k(\tilde{\mathbf{x}}) \right), \quad \sigma(z) = \frac{1}{1 + e^{-z}}, \quad (4)$$

where f_k is the output of the k -th tree. The base configuration uses row and column subsampling rates of 0.8, a minimum child weight of 1, L2 regularization $\lambda = 1$, and a fixed random seed of 42.

3) *Training protocol*: Hyperparameters are selected by grid search over tree depth $\{3, 4, 6\}$, learning rate $\{0.03, 0.05, 0.1\}$, and number of trees $\{200, 300, 500\}$. The search uses five-fold stratified cross-validation, is scored by the area under the ROC curve, and is confined to the training split. The selected configuration was a maximum depth of 6, a learning rate of 0.05, and 200 trees. The final model is refitted on the training split with these parameters, and the validation split serves as the evaluation set for early stopping with a patience of 20 boosting rounds. The test split is used once, after training is complete. The trained booster, the fitted preprocessor, and a metadata file recording the transformed feature order and

the selected hyperparameters are persisted together, so that inference reproduces training exactly.

4) *Decision rule and transparency*: At inference, the model returns $\hat{p}$ and recommends CT imaging when

$$\hat{p} > 0.5. \quad (5)$$

Any feature that was missing is imputed by the training-fitted preprocessor and reported back in a list of missing features, so the user can see which inputs were filled in by the model. Because the fitted booster names its features after the preprocessed columns, any individual tree of the ensemble can be exported as a nested JSON structure and rendered in the interface. This lets a user trace the split conditions along any decision path.

E. 2.5D CT Input Construction

Stage 2 analyzes CT imaging. A single axial slice gives a two-dimensional view of what is a three-dimensional structure. A lesion that is ambiguous on one slice may be clearer when its neighbors are also visible, because real tumors extend across several consecutive slices. Fully volumetric networks capture this context, but at a much higher computational cost. The framework instead adopts a 2.5D input. For a center slice at index s in a volume with maximum slice index $s_{\max}$, the network input is the three-channel tensor

$$\mathbf{X}_s = [I_{c(s-1)}, I_s, I_{c(s+1)}] \in \mathbb{R}^{3 \times 256 \times 256}, \quad (6)$$

where I_i denotes the grayscale slice i resized to 256×256 with bilinear interpolation, and $c(i) = \min(\max(i, 0), s_{\max})$ clamps the index at the ends of the volume. At the first and last slices, the missing neighbor is replaced by a copy of the center slice.

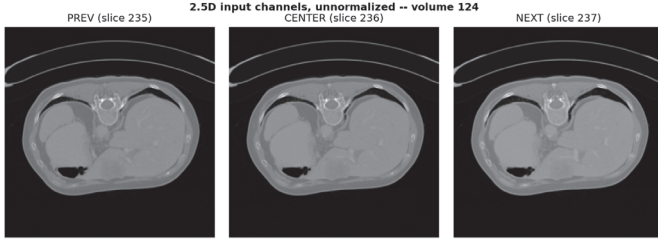


Fig. 2. 2.5D input construction illustrated on validation volume 124 before normalization. The previous (slice 235), center (slice 236), and next (slice 237) axial slices form the three input channels of the segmentation network.

The three channels are normalized with the ImageNet channel means (0.485, 0.456, 0.406) and standard deviations (0.229, 0.224, 0.225). This construction keeps the $(3, H, W)$ input shape that an ImageNet-pretrained encoder expects. The pretrained RGB filters are therefore reused without architectural change, while the channels now encode adjacent axial positions instead of color. It replaces the project's earlier scheme, in which a single slice was replicated into all three channels, which spent two of the three channels on redundant data. An example of the three input channels, before normalization, for a validation volume is shown in Fig. 2.

1) *Data augmentation*: During training only, geometric and photometric augmentations are applied identically to all three channels and to the target mask. These are a horizontal flip and a vertical flip, each with probability 0.5; a rotation by an angle drawn uniformly from $[-15^\circ, 15^\circ]$; and brightness and contrast scaling by factors drawn from $[0.9, 1.1]$. In addition, an elastic deformation with magnitude $\alpha = 20$ and smoothness $\sigma = 4$ is applied with probability 0.5. A single displacement field is drawn per sample and applied with bilinear interpolation to the image channels and with nearest-neighbor interpolation to the mask, followed by re-binarization, so the input and the target stay spatially consistent. The elastic deformation was added as a regularizer, because earlier training rounds showed a persistent gap between training and validation Dice, which pointed to overfitting.

F. ConvNeXt-Tiny U-Net Segmentation Network

The segmentation network follows the U-Net encoder-decoder design with skip connections, the design used throughout the liver segmentation literature [5], [6], [18]. Its encoder is ConvNeXt-Tiny, initialized with ImageNet-pretrained weights. The architecture is shown in Fig. 3.

1) *Encoder*: ConvNeXt-Tiny produces a four-stage feature hierarchy with 96, 192, 384, and 768 channels at $1/4$, $1/8$, $1/16$, and $1/32$ of the input resolution, respectively. For a 256×256 input, the deepest stage is therefore an 8×8 grid.

2) *Decoder*: Each decoder block upsamples its input by a factor of two, concatenates the encoder feature map of matching resolution through a skip connection, and applies two 3×3 convolution, batch-normalization, and ReLU layers. The skip connections carry high-resolution spatial detail from the encoder, which the decoder needs to delineate small lesion boundaries. Two further upsampling steps bring the $1/4$ -resolution decoder output back to the full input resolution.

A 1×1 convolution then produces one channel of tumor logits $\mathbf{Z} \in \mathbb{R}^{256 \times 256}$. The per-pixel tumor probability is $P = \sigma(\mathbf{Z})$. The primary implementation builds this network from a standard segmentation library that provides the U-Net with a ConvNeXt-Tiny encoder. A functionally equivalent hand-written decoder over the same encoder, with the channel widths shown in Fig. 3, is used automatically if that library is unavailable.

G. Loss Function

Tumor pixels are a small minority of the pixels in a CT slice. Measured on a sample of 300 training images from the tumor-bearing training pool, the ratio of non-tumor to tumor pixels was approximately 118:1. The loss therefore combines a pixel-wise classification term with a region-overlap term, and both terms are configured to favor recall:

$$\mathcal{L} = w_b \mathcal{L}_{\text{BCE}} + w_r \mathcal{L}_{\text{TV}}, \quad w_b = w_r = 0.5. \quad (7)$$

The first term is binary cross-entropy with a positive-class weight ω ,

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N} \sum_{i=1}^N [\omega y_i \log p_i + (1 - y_i) \log(1 - p_i)], \quad (8)$$

where $y_i \in \{0, 1\}$ is the ground-truth label of pixel i , $p_i = \sigma(z_i)$ is the predicted probability, and N is the number of pixels in the batch. The weight is set to $\omega = 4$. This is deliberately far below the raw imbalance ratio, because using the raw ratio would compound with the recall-favoring Tversky term and risk overshooting. The second term is the Tversky loss,

$$\mathcal{L}_{\text{TV}} = 1 - \frac{\text{TP}_s + \epsilon}{\text{TP}_s + \alpha \text{FP}_s + \beta \text{FN}_s + \epsilon}, \quad (9)$$

with soft counts computed from probabilities:

$$\text{TP}_s = \sum_i p_i y_i, \quad \text{FP}_s = \sum_i p_i (1 - y_i), \quad \text{FN}_s = \sum_i (1 - p_i) y_i. \quad (10)$$

Here $\alpha = 0.3$, $\beta = 0.7$, and $\epsilon = 1$ is a smoothing constant. The Dice loss is the special case $\alpha = \beta = 0.5$, which penalizes false positives and false negatives equally. Setting $\beta > \alpha$ makes missed tumor pixels cost more than false alarms. This loss replaced an earlier BCE plus Dice combination. Under that earlier loss, test precision was already high but recall was low. A sweep of the binarization threshold showed that lowering the threshold recovered little recall, which indicated that the deficit had to be addressed during training, not by post-hoc calibration. The code base also implements a focal-Tversky variant, $\mathcal{L}_{\text{TV}}^{1/\gamma}$ with $\gamma = 4/3$, but it is disabled in the configuration reported here.

H. Training Protocol

The training configuration is summarized in Table III. The network is optimized with AdamW at a learning rate of 10^{-4} and a weight decay of 2×10^{-2} . The weight decay was doubled from an earlier 10^{-2} as a further regularizer. Memory constraints on the training hardware ruled out a physical batch

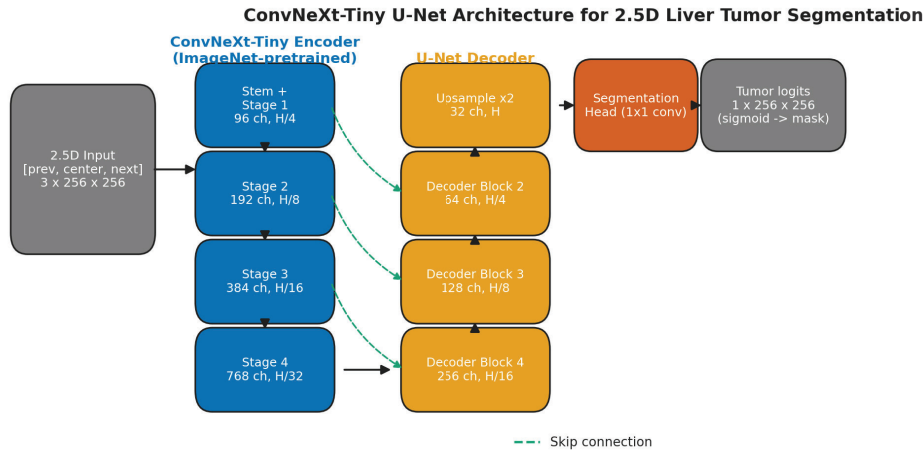


Fig. 3. Architecture of the ConvNeXt-Tiny U-Net for 2.5D liver tumor segmentation. The ImageNet-pretrained ConvNeXt-Tiny encoder produces features with 96, 192, 384, and 768 channels at strides of 4, 8, 16, and 32. The U-Net decoder upsamples progressively and concatenates encoder features through skip connections (dashed). A 1×1 convolutional head outputs single-channel tumor logits at 256×256 , which are converted to probabilities by a sigmoid.

TABLE III
 SEGMENTATION TRAINING CONFIGURATION

Setting	Value
Encoder	ConvNeXt-Tiny (ImageNet-pretrained)
Input	$3 \times 256 \times 256$ (2.5D: $[s - 1, s, s + 1]$)
Output	$1 \times 256 \times 256$ tumor logits
Loss	0.5 BCE ($\omega = 4$) + 0.5 Tversky ($\alpha = 0.3, \beta = 0.7$)
Optimizer	AdamW, lr 10^{-4} , weight decay 2×10^{-2}
Batch	8 per micro-batch $\times$ 4 accumulation = 32
LR schedule	ReduceLROnPlateau (validation loss), patience 5, factor 0.5
Epochs / early stop	Max 100 epochs, patience 15 on validation Dice
Augmentation	Flips, $\pm 15^\circ$ rotation, $\pm 10\%$ brightness/contrast; elastic ($\alpha = 20, \sigma = 4, p = 0.5$)
Binarization threshold	0.5
Seed	42 (Python, NumPy, PyTorch, data loaders)

of 32. The model therefore processes micro-batches of eight and accumulates gradients over four micro-batches before each optimizer step, giving an effective batch size of 32. Batch-normalization statistics are still computed per micro-batch of eight. A plateau scheduler halves the learning rate when the validation loss has not improved for five epochs. Training runs for at most 100 epochs, and early stopping is triggered when the validation Dice has not improved for 15 epochs. Early stopping and checkpoint selection monitor the validation Dice, not the validation loss, because Dice is the quantity of interest and the two do not always move together.

The validation Dice used for model selection is computed globally, by accumulating true positives, false positives, and false negatives over every validation pixel:

$$\text{Dice}_{\text{val}} = \frac{2 \sum \text{TP}}{2 \sum \text{TP} + \sum \text{FP} + \sum \text{FN}}. \quad (11)$$

An earlier version of the training script averaged Dice over batches. That average is biased downward by batches whose ground-truth tumor region is small or empty, where the smoothing term dominates. The global form is also the form used in the final test evaluation, so model selection and reporting are consistent. Two checkpoints are saved. A *latest* checkpoint is written after every epoch, together with the optimizer, scheduler, early-stopping state, and training history, so that interrupted training can resume exactly. A *best* checkpoint is written only when the validation Dice improves, and it is the checkpoint loaded by the deployed application. All random-number generators, including those of the data-loader workers, are seeded for reproducibility, within the determinism guarantees of the hardware back end.

I. Post-Processing and Tumor Characterization

At inference, the probability map P is binarized as

$$M(u, v) = \mathbb{I}[P(u, v) > 0.5], \quad (12)$$

and descriptors of the tumor are computed from the set $\Omega = \{(u, v) : M(u, v) = 1\}$ of predicted tumor pixels. If Ω is empty, the slice is reported as tumor-free. Otherwise, the system reports the pixel area $A = |\Omega|$ and its fraction of the 256×256 grid,

$$A\% = 100 \cdot \frac{|\Omega|}{256 \times 256}, \quad (13)$$

together with the centroid,

$$(\bar{u}, \bar{v}) = \frac{1}{|\Omega|} \sum_{(u, v) \in \Omega} (u, v), \quad (14)$$

and the axis-aligned bounding box of Ω . The centroid and the bounding box are rescaled to the coordinates of the original image and are also reported in normalized form. The minimum, mean, and maximum grayscale intensities (0–255) of the original image inside the mask are also reported. These are raw image intensities, not calibrated Hounsfield units, because the PNG input carries no DICOM rescale parameters.

The front end converts pixel measurements to physical units with an assumed in-plane pixel spacing of $\delta = 0.7$ mm and an assumed slice thickness of $t = 2.5$ mm, because the true DICOM spacing is not available from PNG input. The derived quantities are width and height ($w\delta, h\delta$), area ($A\delta^2$), a single-slice volume estimate ($A\delta^2 t$), a bounding-box perimeter $2(w+h)\delta$, and a dimensionless circularity

$$C = \frac{4\pi A}{(2(w+h)\delta)^2}, \quad (15)$$

computed entirely in pixel units. The interface explicitly labels all millimeter values as approximate and based on assumed spacing.

J. Seg-Grad-CAM Explainability

Grad-CAM explains a classifier by backpropagating a class score to a convolutional feature map [13]. The segmentation network has no class score, so a scalar target must be defined from its dense output. The framework uses the aggregated tumor probability,

$$y = \sum_{u,v} \sigma(Z(u,v)), \quad (16)$$

which measures how much tumor the model sees in the slice. Summation is used instead of averaging so that the gradient remains appreciably non-zero when the predicted tumor region is small. Let $A^k \in \mathbb{R}^{h \times w}$ be the k -th channel of a chosen encoder feature map. The channel importance weights and the raw class activation map are

$$a_k = \frac{1}{hw} \sum_{i,j} \frac{\partial y}{\partial A_{ij}^k}, \quad L = \text{ReLU}\left(\sum_k a_k A^k\right). \quad (17)$$

1) *Choice of feature stage*: The deepest encoder stage is an 8×8 map at 256×256 input, so each cell covers a 32×32 -pixel region. LiTS tumors are frequently smaller than that; bounding boxes as small as 7×9 pixels were observed during development. A map at that resolution therefore cannot localize them. In practice it produced an all-zero map on one real tumor, border-located hotspots on another, and a strong false hotspot on a tumor-free slice. The framework instead hooks an intermediate stage whose map is 32×32 (8×8 pixels per cell), which is a closer match to small lesions, at the cost of somewhat less abstract features. The hooked tensor is obtained by running the encoder and decoder explicitly and retaining the gradient of the selected feature tensor. This is more reliable than registering hooks on an encoder module that returns a list of multi-scale features.

2) *Probability gating*: Because y aggregates over the whole image, the raw map L reflects every region that changes the aggregate score, not only the tumor. The raw map is therefore upsampled bilinearly to 256×256 and multiplied pixel-wise by the model's own predicted probability map:

$$\hat{L} = \text{Up}(L) \odot \sigma(\mathbf{Z}), \quad \bar{L} = \frac{\hat{L} - \min \hat{L}}{\max \hat{L} - \min \hat{L}}. \quad (18)$$

The gated map is min-max normalized. A constant map is returned as all zeros. The normalized map is rendered with a

jet colormap and alpha-blended with weight 0.5 over the original slice. The gating restricts the displayed attention to pixels the model actually predicts as tumor. The resulting overlay should therefore be read as an explanation of *the model's own segmentation*, not as an independent tumor detector.

V. SYSTEM IMPLEMENTATION

This section describes how the methodology of Section IV is realized as a working application. It covers the software and hardware environment, the back-end service and its API, how the trained models are integrated, the user-facing workflow, and the inference path of a request.

A. Software and Hardware Environment

The back end is written in Python. Model development uses PyTorch and torchvision, with the ConvNeXt-Tiny U-Net built through `segmentation_models_pytorch` over a `timm` encoder, and the clinical model uses XGBoost with scikit-learn preprocessing. Tabular I/O uses pandas and openpyxl. Image handling uses Pillow and OpenCV, OCR uses pytesseract over the Tesseract binary, and PDF rasterization uses PyMuPDF. Plots are produced with Matplotlib. The service layer uses FastAPI, the Uvicorn server, and Pydantic schemas. The front end is a React 19 single-page application built with Vite and styled with Tailwind CSS. It uses framer-motion and GSAP for interface transitions, three.js for the landing-page anatomical visualization, and lucide-react for icons.

The segmentation model was trained on a laptop with an Apple M4 processor and 16 GB of unified memory, using PyTorch's Metal Performance Shaders (MPS) back end. Two accommodations were required on this platform. First, operations lacking MPS kernels are allowed to fall back to the CPU through the corresponding PyTorch environment setting. Second, a batch-normalization back-propagation error on non-contiguous inputs, observed after the decoder's skip-connection concatenation, is avoided by making the batch-normalization input contiguous, a change with no numerical effect. The same memory limit motivated the gradient-accumulation scheme of Table III. At serving time, the segmentation engine loads onto a CUDA device when one is available and otherwise onto the CPU. The service has also been run from a Python 3.12 virtual environment on a Windows workstation.

B. Back-End Service and API

The FastAPI application loads both trained models once, in a start-up lifespan handler, so that no request pays the cost of loading a model. Loading is fault tolerant. If a model file is absent, the service logs a warning and starts anyway, and only the endpoints that depend on that model return HTTP 503 until the model has been trained. Cross-origin requests from local development origins are permitted through a pattern-based CORS policy that remains compatible with credentialed requests. Interactive API documentation (Swagger UI and ReDoc) is generated automatically. The endpoints are listed in Table IV.

TABLE IV
 REST API ENDPOINTS OF THE BACK END

Method	Endpoint	Function
GET	/	Health check and model-loaded flags
POST	/blood/extract	Stage 1a: OCR of a report; markers, demographics, 11 clinical features, raw text
POST	/blood/assess	Stage 1b: rule-engine score, flags, reasons, CT recommendation
POST	/clinical/predict	Stage 1c: XGBoost HCC-risk probability and CT recommendation
GET	/clinical/tree/{i}	JSON export of tree <i>i</i> of the XGBoost ensemble
POST	/segment	Single-slice segmentation, location, size, Seg-Grad-CAM
POST	/analyze	2.5D analysis from three required adjacent slices

Request and response bodies are validated against Pydantic schemas. Upload types are checked: PDF, PNG, or JPEG for laboratory reports, and PNG or JPEG for CT slices. Empty uploads are rejected with HTTP 400. A missing Tesseract binary is reported as HTTP 503, and other failures as HTTP 500 with a diagnostic message.

C. Model Integration

Each model is wrapped in a singleton inference engine that owns its weights and its preprocessing.

1) *Segmentation engine*: The segmentation engine restores the best checkpoint into a ConvNeXt-Tiny U-Net built without downloading pretrained weights, puts the model in evaluation mode, and attaches the Seg-Grad-CAM module to the same model instance. Its preprocessing mirrors the training dataset exactly: each slice is converted to grayscale and resized to 256×256 , the three slices are stacked in (previous, center, next) order, and the stack is normalized with ImageNet statistics. Inference and training therefore see identically constructed inputs.

2) *Clinical engine*: The clinical engine loads the XGBoost booster from its native JSON format, the fitted preprocessor, and the metadata file. It builds a one-row frame in the exact training column order, leaves absent fields as missing values for the fitted imputers, and returns the probability, the risk level, the CT recommendation, the list of imputed features, and a plain-language interpretation. For tree export, the engine counts trees directly from the booster's dump and not from a wrapper attribute, which is not restored when a model is loaded from disk. This guarantees that the valid index range reported to the interface matches the trees that can actually be exported.

D. User Interface and Application Workflow

The front end presents a three-step workflow tracked by a stepper component.

1) *Steps 1–2: laboratory report*: The user uploads a laboratory report. The extracted markers are shown in an editable review form with their units and reference ranges, together with a warning that OCR can misread digits and that the values

must be verified. On confirmation, the interface calls the rule-engine and clinical-model endpoints concurrently and displays both results. The rule engine's CT recommendation gates the "Proceed to CT" action. The XGBoost result is shown as an independent risk probability, and a decision-tree viewer lets the user browse any tree of the ensemble by index and follow its split conditions.

2) *Step 3: CT analysis*: The CT upload area requires the center slice together with its previous and next neighbors. Analysis starts only after all three have been supplied and the user explicitly requests it, and selecting files alone shows only a preview. A processing overlay reports progress during inference. The results view comprises panels for the original scan, the segmentation overlay, the Seg-Grad-CAM heatmap, the derived measurements with the spacing caveat.

E. Inference Pipeline

A complete analysis proceeds through the following sequence of requests. (i) The report is uploaded to `/blood/extract`, which returns markers, demographics, the full clinical feature set, and the raw OCR text. (ii) The user-confirmed values are sent to `/blood/assess` and `/clinical/predict`. (iii) If the rule engine recommends imaging, the three CT slices are sent to `/analyze`. That endpoint builds the 2.5D tensor, runs a gradient-free forward pass to obtain *P*, binarizes it with (12), and computes the descriptors of (13)–(14) and the intensity statistics. It also renders a red-tinted segmentation overlay and a binary mask image, and then runs a separate gradient-enabled pass to produce the Seg-Grad-CAM overlay of (18). Failure of the explanation step is logged and does not invalidate the segmentation result.

The `/analyze` endpoint requires all three slices by design. This guarantees that the model always receives the kind of input it was trained on, instead of a center slice duplicated into all channels. The single-image `/segment` endpoint remains available for convenience. It duplicates the center slice into the neighbor channels, which is the same edge case the model sees at the first and last slices of each training volume, and it should therefore be regarded as a degraded path.

VI. EVALUATION PROTOCOL AND METRICS

The two learned models are evaluated independently, each on its own held-out test split, and each test split is used only once, after training. No metric in this paper combines the two stages. The segmentation model is evaluated on the 19 test volumes of the volume-level LiTS split (Section III-C). The clinical model is evaluated on the 750 patients of the synthetic cohort. The rule engine has no trainable parameters and is not scored as a classifier; its thresholds are fixed design inputs (Table II).

A. Segmentation Metrics

With TP, FP, FN, and TN denoting true-positive, false-positive, false-negative, and true-negative counts for the tumor class, the segmentation model is characterized by the following quantities. The Dice coefficient, the primary overlap metric, is

$$\text{Dice} = \frac{2 \text{ TP}}{2 \text{ TP} + \text{FP} + \text{FN}}, \quad (19)$$

and the intersection-over-union (Jaccard index) is

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}}. \quad (20)$$

Both measure the agreement between the predicted and the reference tumor, and they are related monotonically by $\text{IoU} = \text{Dice} / (2 - \text{Dice})$. Precision and recall (sensitivity),

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad (21)$$

separate the two kinds of error: precision penalizes false alarms, and recall penalizes missed tumor. Accuracy,

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}, \quad (22)$$

counts correct decisions of both classes. It is strongly influenced by the size of the negative class.

All predictions are binarized at the threshold of 0.5 used in deployment.

B. Clinical Model Metrics

For the clinical XGBoost model, HCC-positive is the positive class, and the predicted probability is thresholded at 0.5. Accuracy, precision, and recall are defined as in (21)–(22) at the patient level. Recall is also reported under its clinical name, sensitivity. Specificity,

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}, \quad (23)$$

measures the proportion of HCC-negative patients correctly identified as negative. The F1 score summarizes precision and recall. The threshold-independent area under the receiver operating characteristic curve (AUC) summarizes ranking quality across all operating points; the ROC curve plots the true-positive rate against the false-positive rate as the threshold varies. The test split is balanced (375 positive and 375 negative patients), so accuracy is not inflated by class imbalance for this model.

VII. RESULTS AND PERFORMANCE ANALYSIS

This section reports the documented outputs of the implemented models. Segmentation results are given first (Sections VII-A–VII-D), followed by the clinical model (Section VII-E) and the system-level outputs (Section VII-F). Segmentation and clinical results are kept separate throughout, because they come from different models evaluated on different data.

A. Segmentation Performance

The test-set metrics of the ConvNeXt-Tiny U-Net are listed in Table V and plotted in Fig. 4. The model reached a Dice coefficient of 0.8888 and an IoU of 0.7998. Precision was 0.882 and recall was 0.896, so on this test split the model recovered a slightly larger share of true tumor instances than it committed false alarms. This direction agrees with the recall-favoring Tversky weighting ($\beta > \alpha$) described in Section IV. No ablation was performed that isolates the effect of the loss,

TABLE V
TEST-SET RESULTS OF THE CONVNEXT-TINY 2.5D U-NET SEGMENTATION MODEL

Metric	Value
Dice coefficient (primary overlap metric)	0.8888
IoU (Jaccard)	0.7998
Precision	0.882
Recall / sensitivity	0.896
Accuracy	0.9799

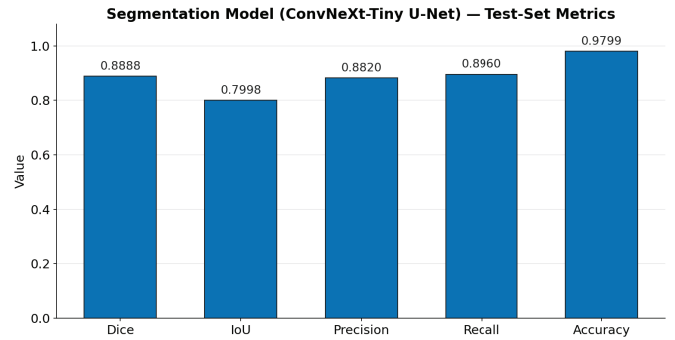


Fig. 4. Test-set metrics of the ConvNeXt-Tiny U-Net segmentation model: Dice coefficient, IoU, precision, recall, and accuracy.

so this agreement is an observation and not a demonstrated causal effect. Accuracy was 0.9799.

Accuracy (0.9799) is higher than the overlap metrics (0.8888 and 0.7998) because correct negative decisions dominate the accuracy figure.

B. Confusion Matrix Analysis

The confusion matrix of the segmentation model on the test data is shown in Fig. 5. It records 823 true positives, 96 false negatives, 110 false positives, and 9,195 true negatives, for a total of 10,224 decisions: 919 actual positives and 9,305 actual negatives. The 919 actual positives correspond to the tumor-bearing slices of the 19 test volumes, and the 9,305 actual negatives to their tumor-free slices. This indicates that the matrix tabulates tumor presence at the level of individual slices.

The metrics of Table V follow directly from these counts through (19)–(22):

$$\text{Dice} = \frac{2(823)}{2(823) + 110 + 96} = \frac{1646}{1852} = 0.8888,$$

$$\text{IoU} = \frac{823}{823 + 110 + 96} = \frac{823}{1029} = 0.7998,$$

$$\text{Precision} = \frac{823}{823 + 110} = 0.882, \quad \text{Recall} = \frac{823}{823 + 96} = 0.896,$$

$$\text{Accuracy} = \frac{823 + 9195}{10224} = 0.9799.$$

The reported metrics are therefore internally consistent with the confusion matrix. Several observations follow from the counts. First, the two error types are of similar size: 96 missed positives against 110 false alarms. Neither error mode dominates. Second, the positive class accounts for 919 of 10,224 decisions (about 9%). This is why accuracy exceeds the

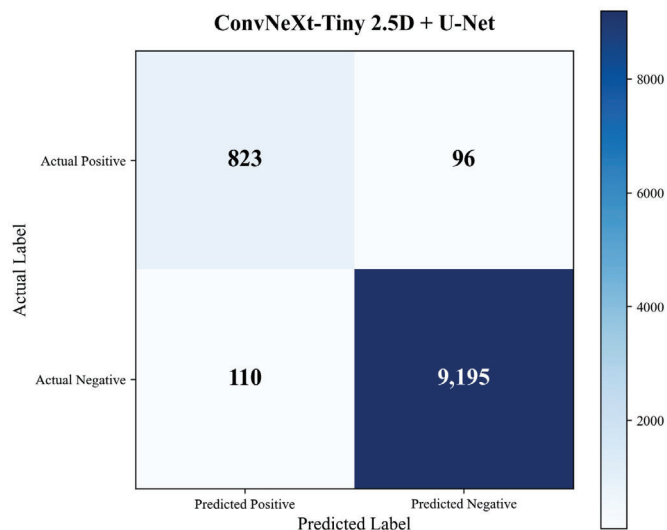


Fig. 5. Confusion matrix of the ConvNeXt-Tiny 2.5D + U-Net model on the test data: 823 true positives, 96 false negatives, 110 false positives, and 9,195 true negatives.

overlap metrics: the 9,195 true negatives contribute to accuracy but not to Dice or IoU. Third, of the 919 tumor-bearing cases, 823 were detected. The 96 misses are the more consequential error in a screening context, and Section VII-D examines a representative miss in detail. Because the matrix contains all four counts, specificity could also be computed from it. It is not reported here because it is not among the documented metrics of the segmentation model.

C. Representative 2.5D CT Inputs

The imaging stage analyzes CT in the 2.5D form defined in (6). The `/analyze` endpoint therefore receives three adjacent axial slices from the same scan for every analysis, not a single image. Fig. 6 shows three such input sets from different CT scans. Each row, (a) to (c), contains three adjacent 256×256 grayscale slices supplied together as one input.

The rows illustrate why neighboring slices are informative. In Fig. 6a, a bright, heterogeneous focal region is visible in all three slices, but its extent and shape change from slice to slice. The 2.5D input exposes exactly this inter-slice continuity to the network. In Fig. 6b and 6c, the adjacent slices differ only subtly. The neighbor channels therefore add local three-dimensional context without changing the overall anatomy the network sees. The scans also differ in field of view and in how large the body appears within the image, which is part of the input variation the model must handle. The figure documents the input format only; the model outputs for these particular inputs are not reported in this paper.

D. Failure Case Analysis

The worst-Dice validation volume, volume 116, contains a large lesion (2,597 px on slice 583) that the model missed entirely. This case was investigated by comparing it with two successfully segmented validation lesions. Fig. 7 shows each

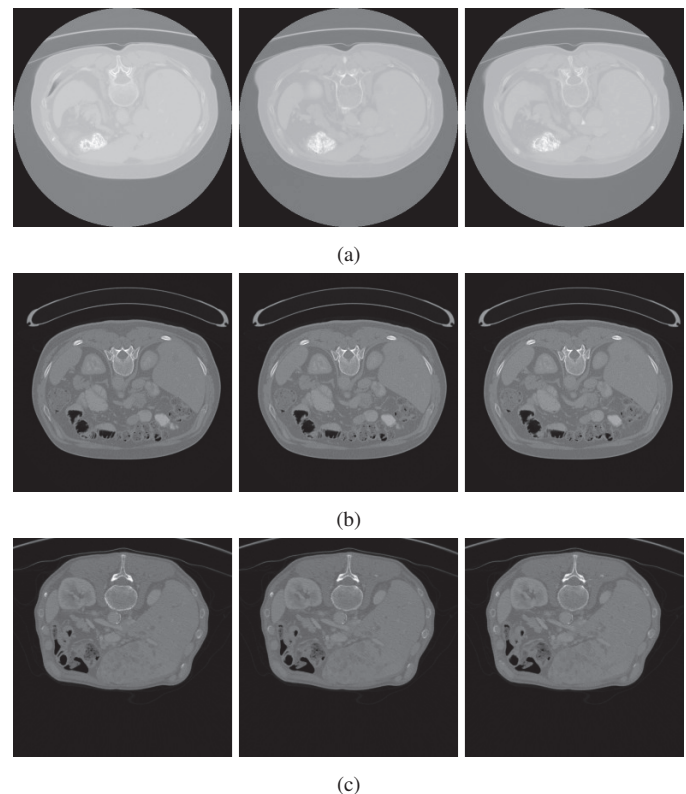


Fig. 6. Representative 2.5D CT inputs to the imaging stage. Each row shows three adjacent axial slices (256×256) from one CT scan, supplied together to the `/analyze` endpoint as a single 2.5D input. (a)–(c) Three different scans.

case with its ground truth, the raw (unthresholded) probability map, and the thresholded prediction.

For the missed lesion, the model's probability remains essentially zero everywhere in the image. The maximum probability anywhere in the image is 0.006, and inside the reference lesion the maximum is 0.0005 and the mean 0.0001. The miss is therefore not a thresholding artifact: no choice of threshold would recover this lesion. In each successful case, by contrast, the maximum probability inside the lesion reaches 1.0, and the mean probability inside the lesion is between 0.8378 and 0.8849. The grayscale intensity range of the missed lesion (123–148) overlaps that of successful volume 124 (130–148). A simple difference in lesion intensity therefore does not explain the failure, and the lesion is also large (2,597 px), so small size is not the explanation either. What the available diagnostics establish is that the model does not respond to this lesion at all. The underlying cause, such as an appearance or context specific to this volume that is under-represented in training, is not determined by the present analysis. The case illustrates that aggregate test metrics can coexist with complete per-lesion failures, which matters for how the system's output should be used (Section VIII).

E. Clinical XGBoost Model Performance

The test-set results of the clinical model on the 750 synthetic patients are listed in Table VI and plotted in Fig. 8. The model reached an accuracy of 0.9520, a precision of 0.9619, a recall

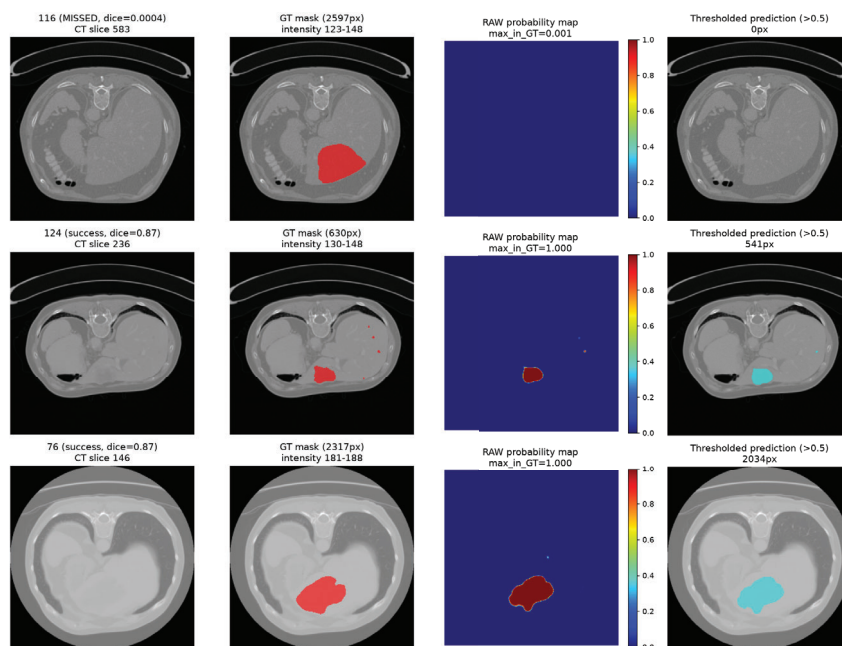


Fig. 7. Investigation of the missed validation lesion (volume 116, slice 583; volume-level Dice 0.0004) compared with two successful validation cases (volumes 124 and 76; volume-level Dice 0.87 and 0.87). Columns: CT slice, ground-truth mask with its grayscale intensity range, raw probability map with the maximum probability inside the ground truth, and the thresholded prediction.

TABLE VI
 TEST-SET RESULTS OF THE CLINICAL XGBOOST MODEL ($n = 750$ SYNTHETIC PATIENTS)

Metric	Value
Accuracy	0.9520
Precision	0.9619
Recall	0.9413
Sensitivity	0.9413
Specificity	0.9627
F1 score	0.9515
AUC	0.9931

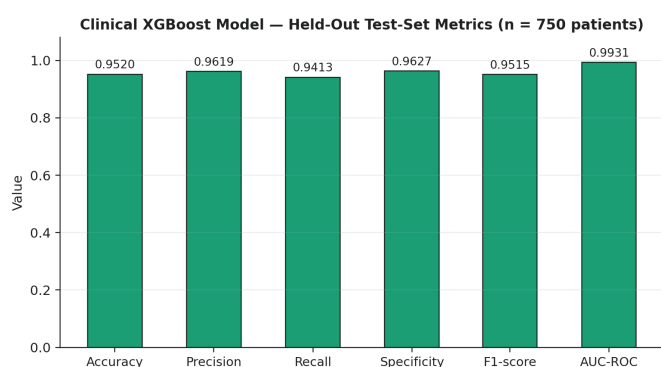


Fig. 8. Test-set metrics of the clinical XGBoost model on 750 synthetic patients.

(sensitivity) of 0.9413, a specificity of 0.9627, an F1 score of 0.9515, and an AUC of 0.9931.

The corresponding confusion matrix is shown in Fig. 9. Of the 375 HCC-negative patients, 361 were correctly classified and 14 were false positives. Of the 375 HCC-positive patients,

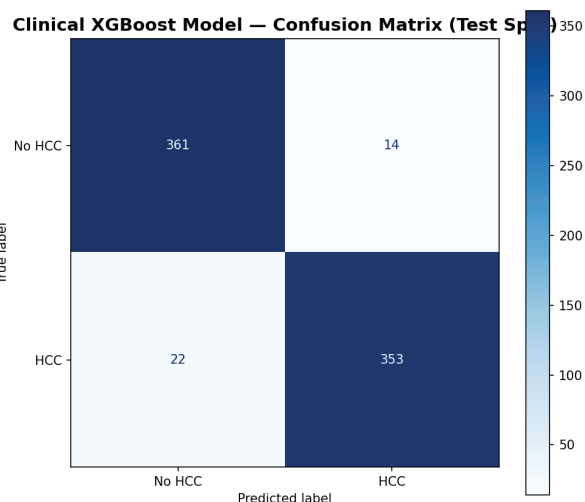


Fig. 9. Confusion matrix of the clinical XGBoost model on the test split (threshold 0.5): 361 true negatives, 14 false positives, 22 false negatives, and 353 true positives.

353 were correctly classified and 22 were false negatives. The model thus errs somewhat more often by missing HCC than by raising false alarms, which agrees with its recall (0.9413) being lower than its specificity (0.9627). The HCC-negative cohort deliberately includes marker false-positive scenarios, and the HCC-positive cohort includes AFP-negative cases (Section III). These are the kinds of records for which classification from laboratory values is inherently ambiguous. The present analysis does not attribute the individual errors to specific scenarios.

The ROC curve in Fig. 10 rises steeply near the origin and

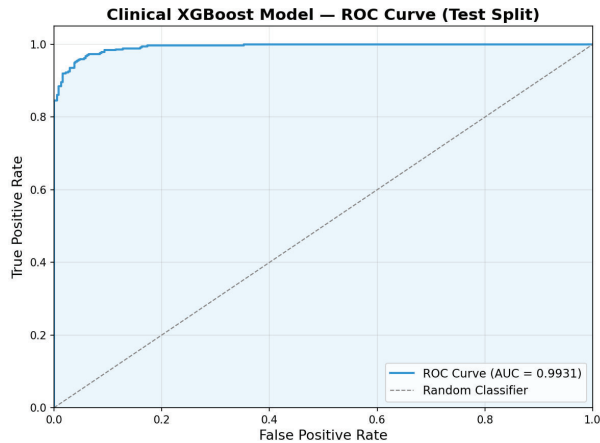


Fig. 10. ROC curve of the clinical XGBoost model on the test split (AUC = 0.9931). The dashed diagonal is the random-classifier reference.

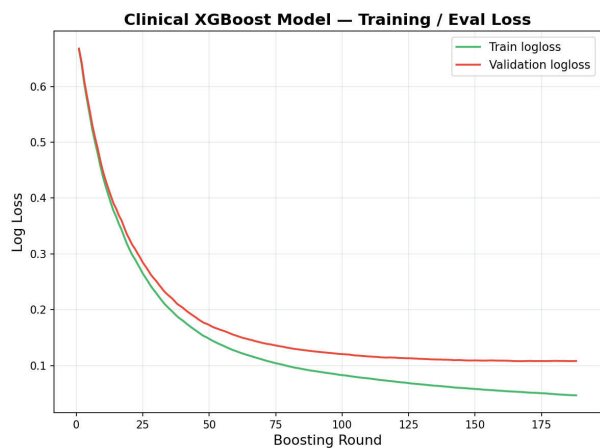


Fig. 11. Training and validation log-loss of the clinical XGBoost model across boosting rounds.

stays close to the top-left corner, which corresponds to the AUC of 0.9931. The curve shows that the model separates the two classes well over a wide range of thresholds, not only at the deployed threshold of 0.5.

The training dynamics are shown in Fig. 11. Training and validation log-loss decrease together over the early boosting rounds. The validation loss then levels off while the training loss continues to decline slowly, producing a gap that widens gradually without the validation loss turning upward. This pattern is consistent with the ensemble continuing to fit the training data while generalization remains stable, and with the early-stopping safeguard, which monitors the validation split.

These results must be read against the nature of the data. The cohort is synthetic, and its class-conditional distributions were specified by the generator (Table I). The metrics therefore show how well the model recovers the structure encoded in that generator. They do not show diagnostic performance on real patients.

F. System-Level Outputs

The deployed system produces, for each analysis, the structured outputs defined in Section IV. From the laboratory

report, it produces the extracted markers and demographics, the rule-engine score with per-marker flags and reasons, and the XGBoost probability with its CT recommendation and any imputed features. From the CT slices, it produces the binary tumor mask, the segmentation overlay, the pixel area and its percentage of the image, the centroid, the bounding box, the in-mask intensity statistics, the millimeter estimates derived under the stated spacing assumptions, and the Seg-Grad-CAM overlay. Finally, it produces a sectioned, template-based clinical report. The synthetic laboratory reports described in Section III serve as realistic inputs for exercising this complete path. By construction, their rule-engine and XGBoost verdicts agree with their intended labels, so they are not used as an evaluation set. No quantitative evaluation of the OCR extraction accuracy, or of the Seg-Grad-CAM maps is included in this study.

VIII. DISCUSSION

A. What the Segmentation Results Indicate

The segmentation model's documented test metrics (Dice 0.8888, IoU 0.7998, precision 0.882, recall 0.896) and its confusion matrix support one main conclusion: on the LiTS volumes that were never seen in training, the model identifies most tumor-bearing cases while keeping false alarms at a similar, modest level. Because the split is made at the volume level, this measures generalization to unseen patients' scans, not memorization of neighboring slices, which is the failure mode that slice-level splitting would conceal. The near-balance between false negatives (96) and false positives (110), together with recall slightly above precision, suggests that the operating point at the deployed threshold of 0.5 does not strongly favor either error type.

Two qualifications apply. First, the confusion-matrix analysis indicates that these metrics tabulate tumor presence per slice. They therefore describe how reliably the system flags tumor-bearing slices. They do not by themselves quantify how precisely each lesion's boundary is traced. The qualitative results address the boundary question. In the successful cases of Fig. 7, the thresholded predictions contain 541 pixels against 630 in the ground truth (volume 124) and 2,034 against 2,317 (volume 76). Detected lesions therefore tend to be matched closely or slightly under-segmented in extent, so the pixel areas and millimeter estimates derived from the mask should be treated as indicative and not as precise volumetry. Second, the failure analysis (Fig. 7) shows that high aggregate scores can coexist with individual lesions that the model does not respond to at all. For a screening-oriented tool, this is the most important caveat on the imaging results. A negative segmentation result on a single slice cannot be taken as evidence that the patient has no tumor.

B. Relationship Between Segmentation and Detection

The framework contains no separate CT classifier. Detection is derived from segmentation: a slice is reported as tumor-bearing if and only if the binarized mask is non-empty. This design has a practical advantage. Every positive imaging verdict comes with a spatial support (a mask, a centroid, and

a bounding box) that a reader can check against the image, whereas a classifier's positive verdict carries no location. The same coupling means that detection inherits every lesion the segmentation network fails to respond to, as in the missed case of Fig. 7. Detection is also tied directly to the binarization threshold. The documented failure case, however, shows that some misses occur at probabilities so low that no threshold would recover them. This observation agrees with the project's earlier finding that recall had to be improved through the training objective, not through threshold tuning (Section IV).

C. Role of Staged Hybrid Evidence

Earlier hybrid systems motivate the use of both clinical and imaging evidence [9]–[12]. The model uses both, but it deliberately does not fuse them into a single learned score. The documented results show why keeping the stages separate is informative. The two models are evaluated on different populations (synthetic patients and public CT volumes), have different error profiles, and answer different questions. A fused score would hide these differences. In the staged design, a clinician sees an elevated laboratory risk and the imaging finding, or its absence, side by side, and can recognize discordance: for example, a high clinical-risk probability together with a negative segmentation on the uploaded slices. Given the failure modes described above, such discordance is precisely the situation that calls for closer human review. The retirement of the earlier fusion design was driven by a technical constraint, namely that tree ensembles produce no differentiable embedding. The resulting architecture nonetheless keeps each stage's contribution attributable, which a fused design does not.

Within the clinical stage, the rule engine and the XGBoost model play complementary roles. The rule engine is fully deterministic, and its thresholds and point assignments can be audited line by line (Table II). The XGBoost model captures interactions among eleven features that a fixed scoring rule cannot, and it still allows inspection of individual trees. Presenting both lets a user see when a learned model and a transparent heuristic agree, and when they do not.

D. Explainability

Explainability is provided in a form suited to each model. For the rule engine, the explanation is the list of reasons attached to each point. For the XGBoost model, it is the inspectable tree structure together with the list of imputed features. For the segmentation model, it is the gated Seg-Grad-CAM overlay. The design decisions in Seg-Grad-CAM respond to the conditions of the task, not to a generic recipe. The aggregated-probability target replaces the missing class score. The intermediate feature stage matches the scale of small LiTS lesions. The probability gating suppresses the diffuse attention that an image-wide target otherwise produces. The gating also has an important consequence. Because the map is multiplied by the model's own prediction, it cannot highlight a lesion that the model failed to segment. For the missed case of Fig. 7, a gated map would be essentially empty. Seg-Grad-CAM in this framework therefore explains *why the*

model segmented what it segmented. It does not provide an independent check for missed lesions. This distinction should be conveyed to users so that an empty heatmap is not mistaken for reassurance.

E. Observed Strengths

Several properties of the implementation are supported directly by the project artifacts. The first is leakage control. The imaging split is made at the volume level and verified so that no volume appears in more than one split. The clinical split is stratified and persisted, preprocessing is fitted on training data only, hyperparameter search is confined to the training split, and each test split is used once. The second is transparency. Each stage produces its own verdict with reasons. The third is safety-oriented engineering: human review of OCR output, refusal to triage when no primary marker is available, required neighboring slices for 2.5D analysis, explicit caveats on intensity units and assumed spacing. The fourth is reproducibility, supported by fixed seeds, persisted splits, stored hyperparameters, and resumable checkpoints.

F. Practical Implications

In its current form, it is best understood as a research and educational prototype of a staged decision-support workflow. Taken together, the results suggest a usage pattern. Laboratory assessment can prioritize which patients' imaging deserves attention. Segmentation can direct the reader to specific slices and regions and provide approximate size information. The explanation layers let each output be checked against its evidence. The documented failure modes indicate that the imaging output should serve as an aid to reading, not as a replacement for it, and that negative results in particular require corroboration.

IX. CONCLUSION

This work presented an end-to-end decision-support framework for hepatocellular carcinoma that follows the order in which a clinician typically approaches a suspected case: serum biomarkers first, imaging next, and a consolidated report last. Laboratory reports are digitised through OCR and pattern-based field extraction. A verification step then lets the user correct any misread values before they influence a decision. The confirmed markers are evaluated in two complementary ways: a transparent rule engine built on accepted clinical thresholds, which acts as the gate for imaging, and a gradient-boosted tree model that estimates HCC risk from tumour markers, liver function indices, viral status and demographic factors. For the imaging stage, a U-Net with an ImageNet-pretrained ConvNeXt-Tiny encoder segments lesions from CT. Its input stacks three adjacent axial slices, which supplies interslice context at the cost of a 2D network. A recall-oriented Tversky loss addresses the scarcity of tumour pixels, and a segmentation-adapted Grad-CAM, gated by the predicted lesion probability, shows clinicians which regions drove each prediction. The stages are served through a FastAPI backend and a React interface, and the outputs are summarised in

a generated report. Together, these elements show that a modular, interpretable pipeline running on modest hardware can bring biochemical and radiological evidence into a single workflow without hiding the reasoning behind its outputs.

Several directions remain open. The clinical risk model should be retrained and validated on real, multi-centre patient records so that the relationships it has learned reflect actual clinical populations, and the segmentation network should be tested prospectively on scans from different scanners and acquisition protocols. Extending the slice-stacking approach towards full volumetric modelling, adding MRI and contrast-phase information, and fusing clinical and imaging evidence into one jointly calibrated estimate are likely to improve the detection of small or atypical lesions. Integration with hospital information systems, structured feedback from radiologists, and formal evaluation of the explanation maps by domain experts would also be needed before the system could support routine screening. With these extensions, the framework could become a practical aid for earlier HCC detection, particularly in settings where specialist radiological expertise is limited.

REFERENCES

- [1] V. Rao, P. Kulkarni, and S. Iyer, "Artificial intelligence in liver cancer diagnosis: A comprehensive review," *IEEE Rev. Biomed. Eng.*, vol. 15, pp. 210–225, 2021.
- [2] R. Singh and M. Kumar, "Artificial intelligence-based medical diagnosis systems: A review," *Comput. Biol. Med.*, vol. 123, 2020, Art. no. 103842.
- [3] A. Khan, M. Hassan, and S. Ahmed, "CNN-based liver tumor detection in CT images," *Multimedia Tools Appl.*, vol. 81, no. 5, pp. 6573–6588, 2022.
- [4] T. Nguyen, H. Pham, and J. Lee, "Deep learning-based medical image analysis for liver cancer detection," *IEEE Access*, vol. 9, pp. 134567–134580, 2021.
- [5] J. Li, K. Wang, and Y. Zhang, "Multi-scale deep learning approaches for liver tumor segmentation," *IEEE Trans. Med. Imaging*, vol. 42, no. 6, pp. 1550–1562, Jun. 2023.
- [6] R. Mehta and V. Shah, "Automated liver segmentation techniques using deep learning," *IEEE Access*, vol. 10, pp. 45678–45690, 2022.
- [7] S. Patel, D. Shah, and V. Rao, "Clinical data-driven risk prediction models for liver cancer detection," *Comput. Biol. Med.*, vol. 162, 2023, Art. no. 106945.
- [8] S. Das, A. Roy, and P. Ghosh, "Deep learning for liver disease classification using clinical data," *Comput. Methods Programs Biomed.*, vol. 214, 2022, Art. no. 106563.
- [9] Y. Li, X. Chen, and H. Wang, "Deep learning-based liver tumor detection using hybrid clinical and imaging data," *IEEE Trans. Med. Imaging*, vol. 44, no. 1, pp. 102–115, Jan. 2025.
- [10] R. Kumar, S. Gupta, and A. Verma, "Multimodal deep learning framework for liver cancer diagnosis using CT and clinical data," *IEEE Access*, vol. 13, pp. 45012–45025, 2025.
- [11] L. Chen, M. Zhang, and Y. Zhao, "Hybrid AI models for liver tumor detection integrating structured clinical features and imaging," *IEEE J. Biomed. Health Informat.*, vol. 29, no. 2, pp. 890–902, Feb. 2024.
- [12] D. Patel and A. Shah, "Hybrid AI systems for medical diagnosis using clinical and imaging data," *IEEE Access*, vol. 9, pp. 99876–99890, 2021.
- [13] T. Nguyen and K. Park, "Explainable AI for liver cancer detection using Grad-CAM and CNN," *IEEE Trans. Med. Imaging*, vol. 42, no. 3, pp. 654–666, Mar. 2023.
- [14] H. Wang, J. Li, and T. Zhou, "Attention-based convolutional neural networks for liver lesion detection in CT images," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 1789–1801, Apr. 2024.
- [15] A. Verma, S. Jain, and P. Mishra, "Deep learning applications in radiology for cancer detection," *IEEE Access*, vol. 8, pp. 145678–145690, 2020.
- [16] P. Sharma and R. Gupta, "Tumor detection techniques using deep learning: A review," *Multimedia Tools Appl.*, vol. 79, no. 29, pp. 21045–21060, 2020.
- [17] Y. Zhang, H. Chen, and X. Li, "Vision transformer-based liver tumor detection in CT images," in *Proc. IEEE Int. Symp. Biomed. Imaging (ISBI)*, 2022, pp. 345–349.
- [18] P. Singh, R. Mehta, and K. Sharma, "Deep learning-based liver tumor segmentation and classification: A survey," *IEEE Access*, vol. 12, pp. 112233–112250, 2024.
- [19] M. Gupta, R. Singh, and A. Sharma, "Hybrid machine learning models for cancer prediction using clinical datasets," *IEEE Access*, vol. 11, pp. 90321–90335, 2023.
- [20] S. Iyer and R. Nair, "Machine learning techniques for clinical data analysis in healthcare," *Comput. Biol. Med.*, vol. 130, 2021, Art. no. 104245.