NOVA: An Agentic Framework for Automated Histopathology Analysis and Discovery

Anurag J. Vaidya[1*] AJVAIDYA@BWH.HARVARD.EDU
Felix Meissen[2] T-FEMEISSEN@MICROSOFT.COM
Daniel C. Castro[2] DACOELH@MICROSOFT.COM
Shruthi Bannur[2] SHRUTHI.BANNUR@MICROSOFT.COM
Tristan Lazard[2] T-TLAZARD@MICROSOFT.COM
Drew F. K. Williamson[3] DREW.WILLIAMSON@EMORY.EDU
Faisal Mahmood[1] FAISALMAHMOOD@BWH.HARVARD.EDU
Javier Alvarez-Valle[2] JAALVARE@MICROSOFT.COM
Stephanie L. Hyland[2] STEPHANIE.HYLAND@MICROSOFT.COM
Kenza Bouzid[2†] KENZA.BOUZID@MICROSOFT.COM

[1] *Mass General Brigham, Boston, USA*
[2] *Microsoft Health Futures, Cambridge, UK*
[3] *Emory University, Atlanta, USA*

Abstract

Digitized histopathology analysis involves complex, time-intensive workflows and specialized expertise, limiting its accessibility. We introduce NOVA, an agentic framework that translates scientific queries into executable analysis pipelines by iteratively generating and running Python code. NOVA integrates 49 domain-specific tools (*e.g.*, nuclei segmentation, whole-slide encoding) built on open-source software, and can also create new tools ad hoc. To evaluate such systems, we present SLIDEQUEST, a 90-question benchmark—verified by pathologists and biomedical scientists—spanning data processing, quantitative analysis, and hypothesis testing. Unlike prior biomedical benchmarks focused on knowledge recall or diagnostic QA, SLIDEQUEST demands multi-step reasoning, iterative coding, and computational problem solving. Quantitative evaluation shows NOVA outperforms coding-agent baselines, and a pathologist-verified case study links morphology to prognostically relevant PAM50 subtypes, demonstrating its scalable discovery potential.

Keywords: Agentic Histopathology Analysis, Agent benchmarking, and Automated discovery

Data and Code Availability SLIDEQUEST is constructed from the TCGA Breast Invasive Carcinoma (BRCA), with whole-slide images (WSIs) and metadata obtained from the GDC portal (Heath et al., 2021). Additionally, data from PanopTILs (Liu et al., 2024), MoNuSeg (Kumar et al., 2019, 2017), Kumar (Kumar et al., 2017), and TCGA-Uniform Tumour (Komura, 2022) is used. Table C.1 provides links to public datasets. Agent framework and benchmark code is at https://github.com/microsoft/nova-agent.

Institutional Review Board (IRB) Proposed use of public datasets was reviewed by home institution. Under policy, use of de-identified public datasets is classified as Not Human Subjects Research [per 45§46.102(e)(1)(ii), 45§46.102(e)(5)]. Guidance and data reflection questions are provided to researchers including considerations to support representativeness, transparency and intended use.

1. Introduction

Histopathology is the gold standard for cancer diagnosis and treatment planning. The digitization

[*] Work done during internship at Microsoft.
[†] Corresponding author

of glass histology slides has allowed computational advances, such as predicting primary sites in cancers of unknown origin (Lu et al., 2021) and developing multimodal prognostic biomarkers (Jaume et al., 2024a). However, the sheer scale and complexity of these data present a significant barrier. Effective workflows often require multi-step processing and narrow, specialized tools, creating a gap between the questions researchers want to ask and their ability to answer them without extensive programming or bioinformatics expertise.

Recent advances in large language models (LLMs) and open-source histopathology tools (Zhang et al., 2025) create an opportunity to bridge this gap. Instead of designing bespoke workflows for each study, an LLM-based system equipped with domain-specific tools could autonomously generate and execute analysis pipelines in response to natural language queries. However, evaluating such flexible systems remains difficult: existing medical artificial intelligence (AI) benchmarks primarily assess text-based knowledge through multiple-choice questions (Singhal et al., 2023; Nori et al., 2023) or single-image visual question-answering (VQA) tasks on static, pre-processed images (He et al., 2020; Lau et al., 2018; Sun et al., 2024), which fail to capture the iterative reasoning, planning, and coding required for computational workflows.

To address this, we introduce Nova, an agentic framework that enables complex histopathology data analysis through a natural language interface. Nova uses a core LLM to interpret user query, generate Python code, and orchestrate a suite of modular, custom tools for multi-step analysis directly on whole-slide images (WSIs) and associated data. Unlike prior approaches that rely on fine-tuned models for narrow tasks such as diagnosis or VQA (Lyu et al., 2025; Ghezloo et al., 2025; Sun et al., 2025), Nova supports dynamic, interactive, and dataset-level scientific discovery without requiring instruction-fine-tuned models. It integrates 49 histopathology analysis tools (*e.g.,* nuclei segmentation and classification, tissue detection, supervised classification experiments) built on trusted open software packages, making it easily extensible.

We further introduce SlideQuest, a benchmark of 90 questions designed to evaluate computational agents in pathology. The tasks span four categories, including pyramidal data interrogation (**DataQA**), cellular analysis (**CellularQA**), histology region of interest (ROI) understanding (**PatchQA**), and gigapixel slide-level experimentation (**SlideQA**). Each question requires multi-step reasoning, iterative coding, and computational problem solving, in addition to image captioning and knowledge recall. All questions are independently verified by both a pathologist and a biomedical scientist, ensuring clinical and scientific validity. SlideQuest provides a rigorous testbed for evaluating agentic systems on scientifically relevant computational tasks.

In summary, our core contributions are:

1. Nova: a modular agentic framework that dynamically writes and executes Python code to build custom workflows from natural language queries, without requiring instruction-fine-tuned models;
2. A library of 49 custom histopathology analysis tools, built on open-source software, integrated into Nova to support diverse biomedical tasks;
3. SlideQuest: a computational benchmark of 90 pathologist- and scientist-verified questions for evaluating agentic workflows in pathology, released publicly for the community to extend further;
4. Comprehensive quantitative evaluation and a pathologist-verified interactive case study using Nova to link morphological properties to prognostically relevant PAM50 molecular subtypes (Parker et al., 2009);
5. Failure case analysis with examples to highlight practical issues encountered by agentic frameworks.

2. Related Works

Agent-based frameworks are increasingly applied to healthcare tasks. Prior work has explored multi-agent collaboration for sequential diagnosis (Tu et al., 2025; Nori et al., 2025), medical QA and VQA on benchmark datasets (Kim et al., 2024; Zhu et al., 2025; He et al., 2025), and orchestration of domain-specific tools for open-ended reasoning in oncology (Ferber et al., 2025) or radiology (Fallahpour et al., 2025). Other systems integrate with clinical infrastructure, such

as electronic health records, to automate workflow tasks (Jiang et al., 2025). While these approaches highlight the promise of agentic methods in medicine, they are typically text-focused and do not work directly with raw data modalities, like whole-slide images.

Several agentic systems have recently been proposed for computational pathology. Lyu et al. (2025) combined pretrained pathology-specific models in a pipelined ensemble for WSI classification and report generation. Ghezloo et al. (2025) and Sun et al. (2025) designed navigation-based agents that traverse WSIs with fine-tuned captioning or multimodal models for diagnosis-focused VQA. Similarly, Chen et al. (2025a) augmented a region-level model with navigation tools for open-ended diagnosis. These approaches, however, focus narrowly on diagnostic outputs, rely on fine-tuned models, and often operate on simplified ROIs or thumbnails of WSIs rather than directly engaging with full-resolution WSIs. By contrast, Nova works natively with WSIs and associated metadata, scales beyond single slides to dataset-level tasks, and leverages modular open-source tools without requiring instruction-tuned models for orchestration.

Evaluating dynamic agentic systems requires benchmarks that move beyond static question answering to capture the complexity of multi-step reasoning and dataset-scale analysis. Most medical LLM benchmarks evaluate text-based knowledge via multiple-choice exams or curated QA datasets (Singhal et al., 2023; Nori et al., 2023; Arora et al., 2025; Bedi et al., 2025). Multimodal benchmarks such as PathVQA (He et al., 2020), PathMMU (Sun et al., 2024), and SlideBench-VQA (Chen et al., 2025b) extend to pathology but rely on static captions, automatically generated questions, or compressed slide embeddings. These often produce unanswerable or trivial questions and remain limited to single-image reasoning. More importantly, they include questions for which an image is not necessary; an LLM only baseline in (Chen et al., 2025b), achieves 45% accuracy. While Chen et al. (2025a) evaluate an agent on whole-slide data, the focus is restricted to diagnosis and the dataset is not public. To date, no benchmark supports rigorous evaluation of computational agents performing iterative reasoning, coding, and dataset-level analysis in pathology. SlideQuest fills this gap by providing 90 pathologist- and scientist-verified tasks that demand multi-step workflows, tool orchestration, and hypothesis testing.

3. NOVA

Nova is a modular agentic framework—based on CodeAct (Wang et al., 2024) and developed using `smolagents` (Roucher et al., 2025)—that dynamically generates and executes Python code to orchestrate tool usage and answer user queries for scalable computational analysis (Figure 1).

3.1. NOVA Framework

Nova is organized around three main components: (i) a core LLM, (ii) a Python 3.11 interpreter (Appendix D) interacting with the user's file system, and (iii) a collection of modular tools (Section 3.2, Appendix E). User queries may include paths to data such as WSIs, ROIs, or associated metadata (Appendix F). To process a query, Nova first dynamically constructs a system prompt with three elements: (i) general instructions for code generation (default from `smolagents`[1]); (ii) tool descriptions including docstrings, inputs, and outputs; and (iii) any special instructions from the user. The combined prompt and query are passed to the core LLM, which produces structured JSON blocks containing both `thought` and `code` fields. The code is executed by the interpreter and results are fed back to the LLM for the next iteration, enabling a running memory of the reasoning process. The loop continues for up to 20 iterations, or until the LLM determines that the query has been fully answered. The core LLM thereby plays a vital role in Nova. These 20 iterations function as a per-task "scratchpad", and memory is cleared after each question (except during the conversational case study).

3.2. Tools

The custom tools form the operational backbone of Nova. Each tool is implemented as a Python function with a clearly defined capability and is intentionally designed to be atomic rather

1. GitHub: `structured_code_agent.yaml`

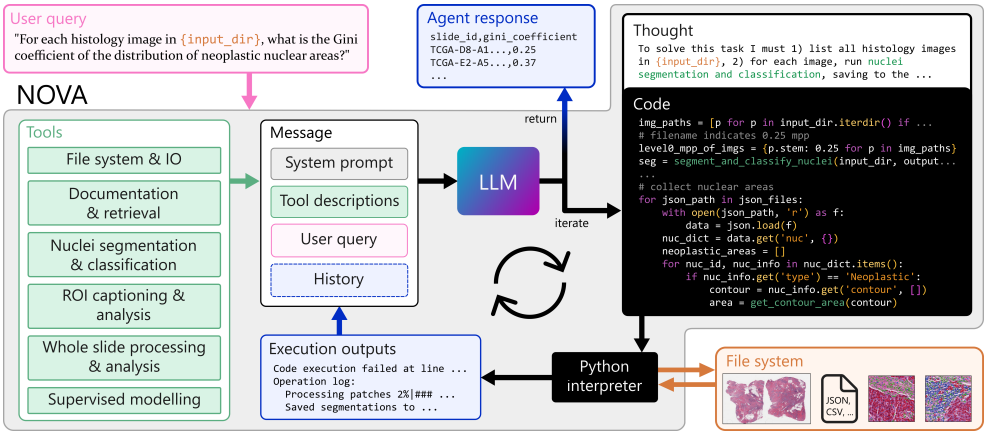

Figure 1: Nova framework. The system takes as input a user query about one or more histology images that are present on the file system. Using a collection of tools and in-built libraries, a core LLM generates Python code to conduct multi-step data processing and analysis towards answering the user query. Code is iteratively executed and fed back into the LLM context to enable dynamic and multi-stage action.

than a multi-step workflow. This modular design ensures reusability across diverse queries and allows the LLM to flexibly compose tools into larger, task-specific workflows. To ensure consistency and reliability, all tools follow a standardized docstring format (Figure E.1). All 49 tools in Nova are developed using open-source computational pathology packages (Zhang et al., 2025; Vaidya et al., 2025; Zheng et al., 2025), making them transparent and extensible. They are organized into seven categories (Tables E.1 to E.7), covering tasks from localized ROI analysis (e.g., nuclei segmentation and captioning) to whole-slide processing (e.g., tissue segmentation, patch extraction, and feature computation), as well as full supervised experiments such as training attention-based MIL models. Importantly, Nova uses LLMs that require no instruction fine-tuning to use the tools, lowering the barrier for adding new functionality. In addition to custom tools, Nova can access standard data science libraries (Table D.2) to autonomously generate additional tools when needed.

4. SlideQuest

To evaluate the capabilities of Nova, we introduce a novel 90-question computational benchmark, SlideQuest. Every question is created from scratch, carefully formulated and verified to capture realistic challenges in computational pathology. SlideQuest is constructed entirely using publicly available data. Each question is reviewed by a computational scientist as well as a pathologist. See Appendix F for details on our unified question format and example instances. We release SlideQuest as an open benchmark, intended not only as a rigorous testbed for Nova but also as a template that the community can build upon and expand to other modalities. Details of datasets used to create SlideQuest are available in A.

4.1. SlideQuest Categories

SlideQuest is organized into categories spanning the major spatial and analytical scales of working with WSIs (Figure 2):

DataQA (25 Questions): Evaluates fundamental understanding of WSIs as a data type. Tasks include retrieving metadata from files (e.g., magnification, resolution, file format), switching between magnification levels, extracting tissue regions, and calculating basic tissue properties.

CellularQA (25 Questions): Nuclei-level tasks testing the ability to segment and classify

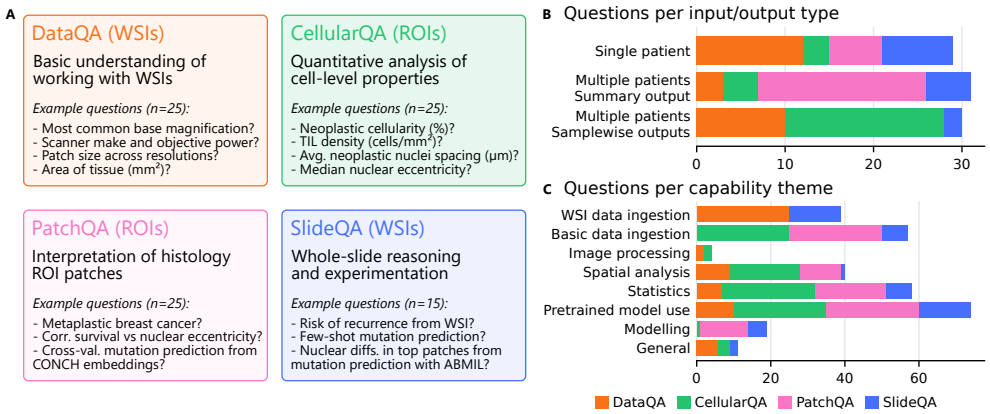

Figure 2: Overview of the SlideQuest benchmark. (A) The four benchmark categories. Listed examples are abridged for illustration only; see full exemplars in Appendix F. (B) Diversity of input and output types. Also note that DataQA and SlideQA contain whole-slide images (WSIs), whereas CellularQA and PatchQA operate on conventional flat images. (C) Themes of capabilities required to answer the questions (full break-down in Table G.1).

nuclei, and perform quantitative analyses such as computing cellular proportions or densities.

PatchQA (25 Questions): Assesses the ability to work with histology regions-of-interest (ROIs). Tasks include encoding ROIs using histology foundation models, classifying them, and comparing cellular properties.

SlideQA (15 Questions): Evaluates WSI-level reasoning and experimentation. Spans diagnosis from gigapixel slides, training and testing supervised models, and retrieving slides based on morphological or molecular similarity.

4.2. Task Diversity

SlideQuest has a mix of questions about single or multiple patients, and with histological data as conventional flat images or pyramidal WSIs. Answers are a structured combination of binary, categorical, and numeric values, and outputs may be sample-wise or summaries over an entire dataset. We further labelled our benchmark questions according to 33 capabilities needed to answer them, grouped into 8 major themes. Regarding *data ingestion*, SlideQuest largely demands handling pyramidal WSI files and metadata. Quantification questions involve *image processing* and precise *spatial analysis* (*e.g.* physical areas/distances, morphometry). A few of the questions also need *general* capabilities such as calculation and autonomous problem solving. Additionally, many tasks require *using pretrained models* (for embedding, segmentation, classification) and/or *training models* on the data provided or extracted embeddings. Lastly, most solutions need *statistical* capabilities for e.g. summarising outputs, cross-validating experiments, and testing hypotheses. A detailed break-down of specific capabilities is given in Table G.1.

4.3. Ground Truth and Evaluation

Depending on the question, the ground-truth answers are derived from (i) expert-provided annotations, (ii) clinical diagnoses associated with the patients, or (iii) human-written Python code. Multi-step questions require a combination of these approaches. Every code-derived answer is verified by a biomedical scientist to ensure correctness and reproducibility. Code to generate answers will be released with the benchmark. All baselines are instructed to produce answers as JSON files following a predefined schema in the question (Appendix F). The answers are compared against a corresponding ground truth JSON with task-specific tolerances (details in Appendix F.1).

5. Experimental Setup

We benchmark Nova using Azure OpenAI LLM endpoints. All experiments are conducted on machines with a single NVIDIA A100 GPU within Azure ML. To account for LLM stochasticity and to quantify variability, each experiment is repeated three times. Appendix H provides further details.

5.1. Baselines

All coding baselines have access to the same Python interpreter (Appendix D), with an identical set of libraries (Tables D.1 and D.2), and are constrained to the same maximum number of iterations (20). We compare Nova against:

LLM only: Answers queries in natural language and python code, but does not have access to a coding environment and tools.

LLM with PI: Has access to a Python 3.11 environment and can execute code only once, to evaluate whether single-shot code execution is sufficient to solve tasks.

LLM with PI and retries: Can additionally refine its code over multiple steps, correcting errors along the way. Corresponds to Nova without custom tools or custom system prompt.

6. Results

6.1. Performance on SlideQuest

Overall results. Figure 3 demonstrates the performance of Nova against baselines across the categories of SlideQuest. The LLM only baseline achieves an average score of 0 on SlideQuest, confirming that the benchmark evaluates computational rather than purely linguistic ability. Adding access to a Python interpreter (LLM with PI) improves performance score to 0.154, showing that even single-shot code execution can solve a subset of tasks. However, with LLM with PI and retries, this score reaches an average score of 0.269 by iteratively refining code and correcting errors. Nova achieves the highest performance on SlideQuest (0.477), outperforming all baselines. Averaged across all 90 benchmark questions, Nova surpasses LLM with PI by 0.323, and LLM with PI and retries by 0.208, establishing a clear margin of improvement (Figure 3 and Table I.1). Nova sequentially answered all 90 questions in 40 hours on a consumer grade GPU, which can be further sped up if running jobs in parallel asynchronously. Per category run-times are found in Appendix J.

Per category results. Stratifying Nova's performance on SlideQuest by category reveals substantial variation. Nova achieves its highest score on DataQA (0.777) and its lowest on CellularQA (0.323). Strong performance on DataQA is expected, as it primarily involves reading pyramidal metadata, a task well within the internal knowledge of LLMs. However, despite this understanding of WSIs, Nova exhibits a failure rate of 0.422 on SlideQA, highlighting the difficulty of computational WSI analysis. Scores below 1 were observed in CellularQA even when the correct nuclei segmentation and classification tools were used, reflecting the shortcomings of the current HoverNet (Graham et al., 2019) model used. We anticipate improved results as more robust tools are developed and integrated into Nova (Adjadj et al., 2025).

Different core LLMs. Stronger LLMs (GPT-5 vs. GPT-4.1) improve performance on harder categories such as SlideQA (0.472 vs. 0.551) but show no gain or even declines on easier ones like DataQA (0.777 vs. 0.708). GPT-5 also incurs substantially longer runtimes (e.g., averaging 47.4 vs. 31.2 hours for GPT-4.1 on SlideQA) (Figure J.2). GPT-5-mini provides a strong alternative to GPT-4.1, achieving the highest performance on SlideQA while matching or underperforming on the other categories (Table K.3).

Are custom tools needed? To understand the contribution of custom tools in our framework, we compared Nova (with custom tools) with two variants: (i) Nova (no custom tools), which measures the core LLM's ability to write tools using its internal knowledge (ii) Nova (with RAG), which does RAG on open-source computational pathology software packages to create tools based on user queries (details in Appendix K). Nova (with custom tools) shows clear gains over Nova (no custom tools), with use of custom tools outperforming across all categories (e.g., +0.240 on DataQA, +0.171 on CellularQA, +0.113 on PatchQA, +0.033 on SlideQA). Run

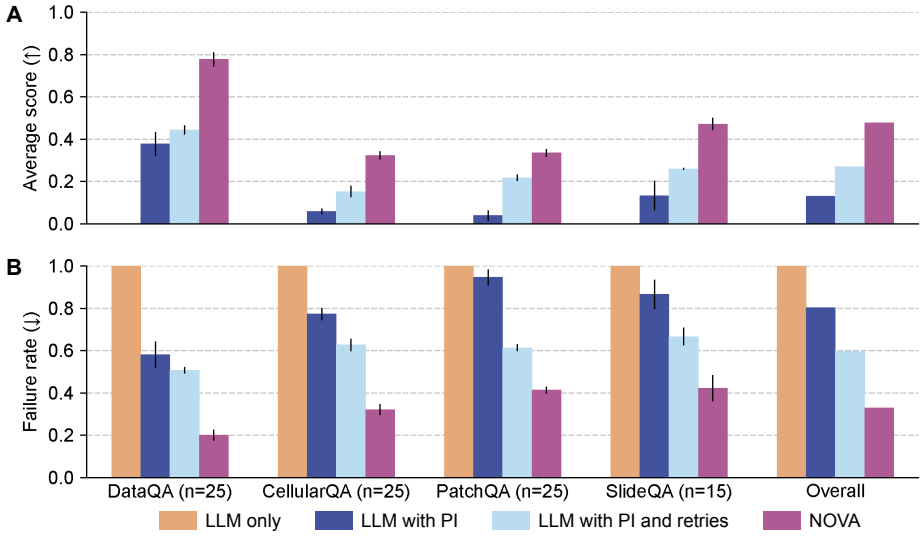

Figure 3: **A.** Average score (higher is better) on SlideQuest stratified by benchmark category. **B.** Failure rate (lower is better) showing the proportion of questions from SlideQuest on which the approach achieved a zero score. Overall is the average of each category weighted by number of questions in the category. Error bars are standard error of the mean from 3 trials. All results with GPT-4.1. "PI" stands for Python interpreter.

time increased substantially without tools but did not improve performance: on DataQA, Nova with tools required 2.76 h compared to 4.20 h without tools, while also achieving higher performance (0.777 vs. 0.537). Even against Nova (with RAG), the custom tool version performs better (overall performance of 0.337 with RAG and 0.477 with custom tools), indicating that RAG-based knowledge of documentation remains insufficient for effective tool creation (Table K.1). Overall, careful manual tool design remains essential.

6.2. Failure Case Analysis

We manually reviewed all cases where Nova achieved a score of 0 in all runs. Failures fall into four main categories: (i) tool limitations, (ii) framework limitations, (iii) ignoring existing tools or data, and (iv) LLM fabrications. Examples of failure cases are shown in Appendix L.

Tool limitations. Tool issues occurred across CellularQA, SlideQA, and PatchQA, due to incorrect output from segmentation models or image–text models, causing the final agent output to be incorrect. Moreover, tools may present results in a slightly different manner than presented in the task (for example, class names differing between the tool and task) causing the model to occasionally omit relevant classes and fabricate irrelevant ones.

Framework limitations. The most common issue was exceeding the Python interpreter's operation limit, a safeguard against infinite loops. This caused premature termination even when the code was correct, especially in computation-heavy tasks like CellularQA and PatchQA. The agent often retried using subsets of the data, producing incomplete or incorrect answers.

Ignoring tools or data. In some cases, the agent recomputed values already provided by tools (e.g., convexity of tissue regions in DataQA) or rewrote code for existing functions (e.g., contour area). While not always incorrect, this behavior sometimes caused failures and often reduced efficiency. The agent also overused `try/except` blocks, skipping data that could have been recovered.

Fabrications. Failures included fabricating data when inputs could not be loaded, or relying on simplistic heuristics (e.g., "darker nuclei are cancerous") instead of using tools.

6.3. Case Study

We demonstrate Nova's ability to carry out a comprehensive computational pathology workflow exploring the morphological features of the four major PAM50 breast cancer subtypes: Luminal A, Luminal B, Basal-like, and HER2-enriched. Figure 4 shows similar to how scientists would approach this task, we first task Nova to gather relevant biological and clinical knowledge about the PAM50 subtypes from the literature. Next, we provide representative H&E WSIs for each subtype and ask Nova to analyse them. It defines a workflow involving tissue segmentation, patch-level feature extraction, text-prompt similarity analysis to localize subtype-specific features, and nuclei segmentation. From these analyses, Nova produces a comparative report that highlights both shared and distinct characteristics of the subtypes. The results closely match known histopathological findings (Heng et al., 2017). For instance, Luminal A tumours show limited necrosis and abundant connective tissue, whereas Basal-like tumours exhibit extensive necrosis, inflammation, and immune infiltration. This study demonstrates Nova's ability to orchestrate complex computational workflows and derive insights by integrating multiple tools and biological knowledge, illustrating practical utility in a real-world biomedical research scenario.

7. Discussion

Our findings show that strong performance on SlideQuest requires structured tool use and iterative coding, not natural language ability alone. Even with correct tool composition, many questions remain difficult due to current tool limitations. Modular agentic systems like Nova can readily incorporate stronger pretrained models as they emerge, and we anticipate rapid community progress on this benchmark. We also show Nova is compatible with various LLMs, where stronger models yield better results on challenging categories. Finally, we highlight that custom tools are essential; relying on LLM internal knowledge or API documentation is insufficient. Advances in automated tool creation and verification would further strengthen such systems (Wölflein et al., 2025).

Limitations. The evaluation mechanism in SlideQuest only checks the final outputs. Hence, incorrect intermediate reasoning—including fabricated data, random guessing, or baseless tool calls—is not penalized. The framework also does not distinguish between errors arising from tool implementations and errors from the agent's use of the tools. Second, while our 49 custom tools cover diverse histopathology tasks, it is not feasible to anticipate every edge case, and the tools themselves may contain mistakes, limiting scalability. Third, reproducibility of agentic behaviour is an open challenge. The relatively high variance across runs indicates that generated pipelines may vary between executions, making it difficult to guarantee consistent outcomes. Finally, TCGA includes known limitations such as demographic biases (Vaidya et al., 2024); we hope the community adds more diverse tasks to SlideQuest independent on TCGA.

Conclusion. We introduced Nova, a coding agent framework for histopathology data analysis equipped with 49 carefully engineered tools. To rigorously evaluate its capabilities, we developed SlideQuest, a 90-question benchmark spanning multiple analytical scales. Across all categories, Nova outperforms coding baselines, demonstrating the necessity of domain-specific tools. Additionally, in a pathologist-verified case study linking morphology to prognostically relevant PAM50 subtypes, we demonstrate Nova's scalable discovery potential.

Future Work. We hope to quantify the time savings with Nova for computational scientists. While we use histopathology as a proof of concept, the framework can be extended to other biomedical modalities. By releasing the framework, tools, and benchmark, we encourage the community to build on Nova by creating robust tools and SlideQuest by contributing new questions and tools to broaden its coverage.

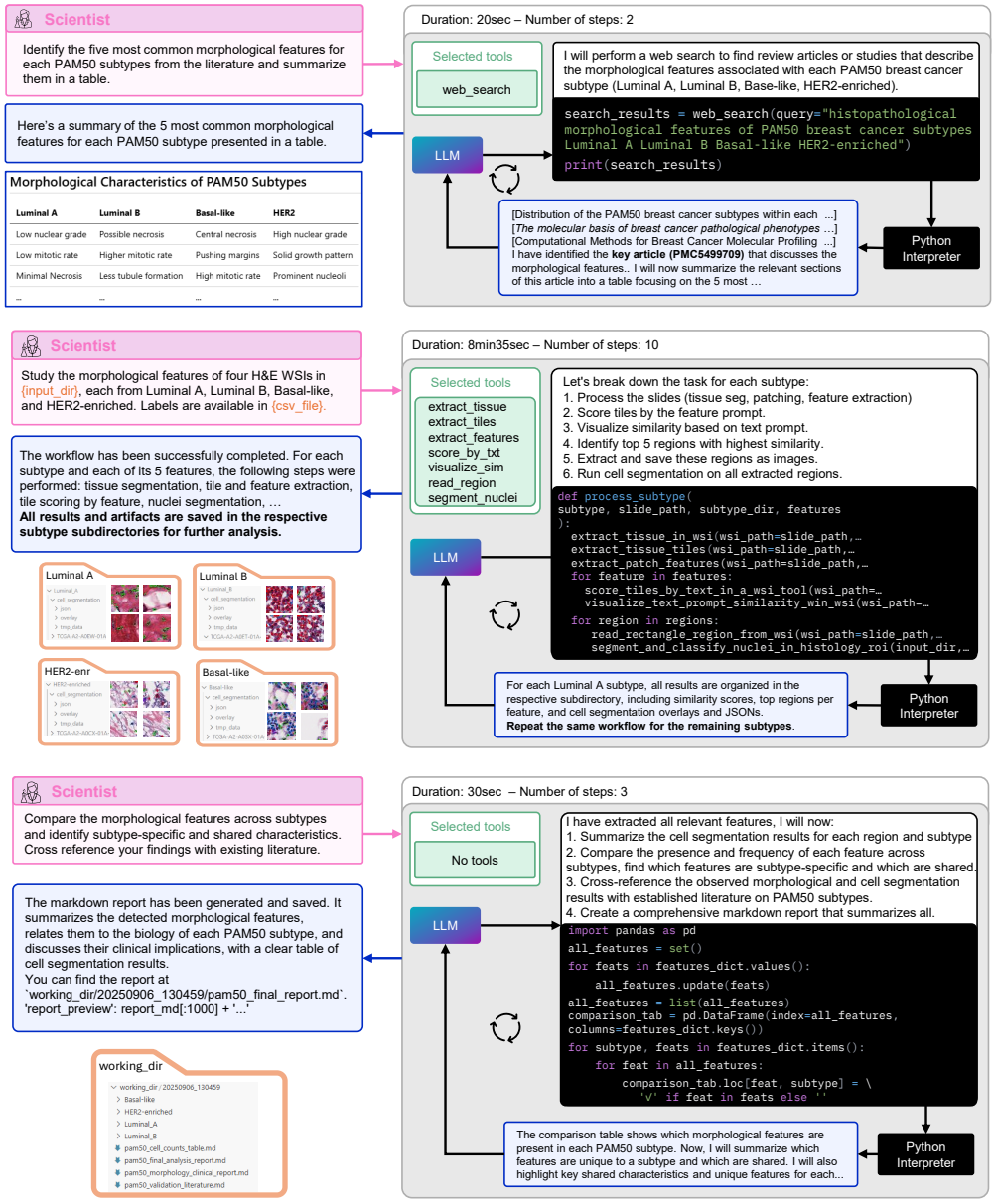

Figure 4: Case study showing the use of NOVA to explore the morphological features associated with PAM50 breast cancer subtypes (Luminal A, Luminal B, Basal-like, HER2-enriched) and assess their relationship with tumour characteristics. Only the main steps are shown for illustration purposes. The final report produced by NOVA is shown in Figures B.1 and B.2

.

References

Benjamin Adjadj, Pierre-Antoine Bannier, Guillaume Horent, Sebastien Mandela, Aurore Lyon, Kathryn Schutte, Ulysse Marteau, Valentin Gaury, Laura Dumont, Thomas Mathieu, et al. Towards comprehensive cellular

characterisation of h&e slides. *arXiv preprint arXiv:2508.09926*, 2025.

Rahul K Arora, Jason Wei, Rebecca Soskin Hicks, Preston Bowman, Joaquin Quiñonero-Candela, Foivos Tsimpourlas, Michael Sharman, Meghan Shah, Andrea Vallone, Alex Beutel, et al. Healthbench: Evaluating large language models towards improved human health. *arXiv preprint arXiv:2505.08775*, 2025.

Suhana Bedi, Hejie Cui, Miguel Fuentes, Alyssa Unell, Michael Wornow, Juan M Banda, Nikesh Kotecha, Timothy Keyes, Yifan Mai, Mert Oez, et al. Medhelm: Holistic evaluation of large language models for medical tasks. *arXiv preprint arXiv:2505.23802*, 2025.

Chengkuan Chen, Luca L Weishaupt, Drew FK Williamson, Richard J Chen, Tong Ding, Bowen Chen, Anurag Vaidya, Long Phi Le, Guillaume Jaume, Ming Y Lu, et al. Evidence-based diagnostic reasoning with multi-agent copilot for human pathology. *arXiv preprint arXiv:2506.20964*, 2025a.

Ying Chen, Guoan Wang, Yuanfeng Ji, Yanjun Li, Jin Ye, Tianbin Li, Ming Hu, Rongshan Yu, Yu Qiao, and Junjun He. Slidechat: A large vision-language assistant for whole-slide pathology image understanding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5134–5143, 2025b.

Tong Ding, Sophia J Wagner, Andrew H Song, Richard J Chen, Ming Y Lu, Andrew Zhang, Anurag J Vaidya, Guillaume Jaume, Muhammad Shaban, Ahrong Kim, et al. A multimodal whole-slide foundation model for pathology. *Nature Medicine*, pages 1–13, 2025.

Adibvafa Fallahpour, Jun Ma, Alif Munim, Hongwei Lyu, and Bo Wang. Medrax: Medical reasoning agent for chest x-ray. *arXiv preprint arXiv:2502.02673*, 2025.

Dyke Ferber, Omar SM El Nahhas, Georg Wölflein, Isabella C Wiest, Jan Clusmann, Marie-Elisabeth Leßmann, Sebastian Foersch, Jacqueline Lammert, Maximilian Tschochohei, Dirk Jäger, et al. Development and validation of an autonomous artificial intelligence agent for clinical decision-making in oncology. *Nature cancer*, pages 1–13, 2025.

Fatemeh Ghezloo, Mehmet Saygin Seyfioglu, Rustin Soraki, Wisdom O Ikezogwo, Beibin Li, Tejoram Vivekanandan, Joann G Elmore, Ranjay Krishna, and Linda Shapiro. Pathfinder: A multi-modal multi-agent system for medical diagnostic decision-making applied to histopathology. *arXiv preprint arXiv:2502.08916*, 2025.

Simon Graham, Quoc Dang Vu, Shan E Ahmed Raza, Ayesha Azam, Yee Wah Tsang, Jin Tae Kwak, and Nasir Rajpoot. Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical Image Analysis*, page 101563, 2019.

Xuehai He, Yichen Zhang, Luntian Mou, Eric Xing, and Pengtao Xie. Pathvqa: 30000+ questions for medical visual question answering. *arXiv preprint arXiv:2003.10286*, 2020.

Yexiao He, Ang Li, Boyi Liu, Zhewei Yao, and Yuxiong He. Medorch: Medical diagnosis with tool-augmented reasoning agents for flexible extensibility. *arXiv preprint arXiv:2506.00235*, 2025.

Allison P Heath, Vincent Ferretti, Stuti Agrawal, Maksim An, James C Angelakos, Renuka Arya, Rosita Bajari, Bilal Baqar, Justin HB Barnowski, Jeffrey Burt, et al. The nci genomic data commons. *Nature genetics*, 53(3): 257–262, 2021.

Yujing J Heng, Susan C Lester, Gary MK Tse, Rachel E Factor, Kimberly H Allison, Laura C Collins, Yunn-Yi Chen, Kristin C Jensen, Nicole B Johnson, Jong Cheol Jeong, et al. The molecular basis of breast cancer pathological phenotypes. *The Journal of pathology*, 241(3): 375–391, 2017.

Guillaume Jaume, Anurag Vaidya, Richard J Chen, Drew FK Williamson, Paul Pu Liang, and Faisal Mahmood. Modeling dense multimodal interactions between biological pathways and histology for survival prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11579–11590, 2024a.

Guillaume Jaume, Anurag Vaidya, Andrew Zhang, Andrew H. Song, Richard J. Chen, Sharifa Sahai, Dandan Mo, Emilio Madrigal, Long Phi Le, and Faisal Mahmood. Multistain pretraining for slide representation learning in pathology. In *European Conference on Computer Vision*, pages 19–37. Springer, 2024b.

Yixing Jiang, Kameron C. Black, Gloria Geng, Danny Park, James Zou, Andrew Y. Ng, and Jonathan H. Chen. Medagentbench: A virtual ehr environment to benchmark medical llm agents. *NEJM AI*, page AIdbp2500144, 2025. doi: 10.1056/AIdbp2500144. URL https://ai.nejm.org/doi/full/10.1056/AIdbp2500144.

Yubin Kim, Chanwoo Park, Hyewon Jeong, Yik S Chan, Xuhai Xu, Daniel McDuff, Hyeonhoon Lee, Marzyeh Ghassemi, Cynthia Breazeal, and Hae W Park. Mdagents: An adaptive collaboration of llms for medical decision-making. *Advances in Neural Information Processing Systems*, 37:79410–79452, 2024.

D. Komura. Universal encoding of pan-cancer histology by deep texture representations. *Cell Reports*, 38:110424, 2022. doi: 10.1016/j.celrep.2022.110424.

Neeraj Kumar, Ruchika Verma, Sanuj Sharma, Surabhi Bhargava, Abhishek Vahadane, and Amit Sethi. A dataset and a technique for generalized nuclear segmentation for computational pathology. *IEEE transactions on medical imaging*, 36(7):1550–1560, 2017.

Neeraj Kumar, Ruchika Verma, Deepak Anand, Yanning Zhou, Omer Fahri Onder, Efstratios Tsougenis, Hao Chen, Pheng-Ann Heng, Jiahui Li, Zhiqiang Hu, et al. A multi-organ nucleus segmentation challenge. *IEEE transactions on medical imaging*, 39(5):1380–1391, 2019.

Jason J Lau, Soumya Gayen, Asma Ben Abacha, and Dina Demner-Fushman. A dataset of clinically generated visual questions and answers about radiology images. *Scientific data*, 5(1): 1–10, 2018.

Shangke Liu, Mohamed Amgad, Deeptej More, Muhammad A Rathore, Roberto Salgado, and Lee AD Cooper. A panoptic segmentation dataset and deep-learning approach for explainable scoring of tumor-infiltrating lymphocytes. *NPJ Breast Cancer*, 10(1):52, 2024.

Ming Y. Lu, Tiffany Y. Chen, Drew F. K. Williamson, Ming Zhao, Mark Shady, Jana Lipkova, and Faisal Mahmood. Ai-based pathology predicts origins for cancers of unknown primary. *Nature*, 594(7861):106–110, 2021.

Xinheng Lyu, Yuci Liang, Wenting Chen, Meidan Ding, Jiaqi Yang, Guolin Huang, Daokun Zhang, Xiangjian He, and Linlin Shen. Wsi-agents: A collaborative multi-agent system for multi-modal whole slide image analysis. *arXiv preprint arXiv:2507.14680*, 2025.

Harsha Nori, Nicholas King, Scott Mayer McKinney, Dean Carignan, and Eric Horvitz. Capabilities of gpt-4 on medical challenge problems. *arXiv preprint arXiv:2303.13375*, 2023.

Harsha Nori, Mayank Daswani, Christopher Kelly, Scott Lundberg, Marco Tulio Ribeiro, Marc Wilson, Xiaoxuan Liu, Viknesh Sounderajah, Jonathan Carlson, Matthew P Lungren, et al. Sequential diagnosis with language models. *arXiv preprint arXiv:2506.22405*, 2025.

Joel S Parker, Michael Mullins, Maggie CU Cheang, Samuel Leung, David Voduc, Tammi Vickery, Sherri Davies, Christiane Fauron, Xiaping He, Zhiyuan Hu, et al. Supervised risk predictor of breast cancer based on intrinsic subtypes. *Journal of clinical oncology*, 27(8): 1160–1167, 2009.

Aymeric Roucher, Albert Villanova del Moral, Thomas Wolf, Leandro von Werra, and Erik Kaunismäki. 'smolagents': a smol library to build great agentic systems. https://github.com/huggingface/smolagents, 2025.

George Shaikovski, Adam Casson, Kristen Severson, Eric Zimmermann, Yi Kan Wang, Jeremy D Kunz, Juan A Retamero, Gerard Oakley, David Klimstra, Christopher Kanan, et al. Prism: A multi-modal generative foundation model for slide-level histopathology. *arXiv preprint arXiv:2405.10254*, 2024.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung,

Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620 (7972):172–180, 2023.

Yuxuan Sun, Hao Wu, Chenglu Zhu, Sunyi Zheng, Qizi Chen, Kai Zhang, Yunlong Zhang, Dan Wan, Xiaoxiao Lan, Mengyue Zheng, et al. Pathmmu: A massive multimodal expert-level benchmark for understanding and reasoning in pathology. In *European Conference on Computer Vision*, pages 56–73. Springer, 2024.

Yuxuan Sun, Yixuan Si, Chenglu Zhu, Kai Zhang, Zhongyi Shui, Bowen Ding, Tao Lin, and Lin Yang. Cpathagent: An agent-based foundation model for interpretable high-resolution pathology image analysis mimicking pathologists' diagnostic logic. *arXiv preprint arXiv:2505.20510*, 2025.

Tao Tu, Mike Schaekermann, Anil Palepu, Khaled Saab, Jan Freyberg, Ryutaro Tanno, Amy Wang, Brenna Li, Mohamed Amin, Yong Cheng, et al. Towards conversational diagnostic artificial intelligence. *Nature*, pages 1–9, 2025.

Anurag Vaidya, Richard J Chen, Drew FK Williamson, Andrew H Song, Guillaume Jaume, Yuzhe Yang, Thomas Hartvigsen, Emma C Dyer, Ming Y Lu, Jana Lipkova, et al. Demographic bias in misdiagnosis by computational pathology models. *Nature Medicine*, 30 (4):1174–1190, 2024.

Anurag Vaidya, Andrew Zhang, Guillaume Jaume, Andrew H Song, Tong Ding, Sophia J Wagner, Ming Y Lu, Paul Doucet, Harry Robertson, Cristina Almagro-Perez, et al. Molecular-driven foundation model for oncologic pathology. *arXiv preprint arXiv:2501.16652*, 2025.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*, 2024.

Georg Wölflein, Dyke Ferber, Daniel Truhn, Ognjen Arandjelović, and Jakob Nikolas Kather. Llm agents making agent tools. *arXiv preprint arXiv:2502.11705*, 2025.

Andrew Zhang, Guillaume Jaume, Anurag Vaidya, Tong Ding, and Faisal Mahmood. Accelerating data processing and benchmarking of ai models for pathology. *arXiv preprint arXiv:2502.06750*, 2025.

Yimin Zheng, Ernesto Abila, Eva Chrenková, Juliane Winkler, and André F Rendeiro. Lazyslide: accessible and interoperable whole slide image analysis. *BioRxiv*, pages 2025–05, 2025.

Yinghao Zhu, Ziyi He, Haoran Hu, Xiaochen Zheng, Xichen Zhang, Zixiang Wang, Junyi Gao, Liantao Ma, and Lequan Yu. Medagentboard: Benchmarking multi-agent collaboration with conventional methods for diverse medical tasks. *arXiv preprint arXiv:2505.12371*, 2025.

Supplementary Material

NOVA: An Agentic Framework for Automated Histopathology Analysis and Discovery

We provide supplementary information Nova framework, SlideQuest, additional results, run time analysis:

1. **Section** A: Details of datasets used for SlideQuest.
2. **Section** C: Links to public datasets used to construct SlideQuest.
3. **Section** D: Python 3.11 execution environment, allowed libraries, and operation limits.
4. **Section** E: Catalog of Nova tools (descriptions and categories).
5. **Section** F: Unified SlideQuest question schema with exemplars across categories.
6. **Section** G: Details on SlideQuest question types.
7. **Section** H: Experimental setup, including Nova configuration, LLM variants/parameters, and the experiment runner.
8. **Section** I: Results of baseliens and Nova on SlideQuest.
9. **Section** J: Runtime vs. accuracy analyses across baselines and core LLMs.
10. **Section** K: Ablations of Nova on SlideQuest.
11. **Section** L: Failure analysis with representative cases (tool limitations, recomputation, and other modes).

Appendix A. Datasets

Questions in SlideQuest are drawn from multiple publicly available datasets, all derived from TCGA WSIs but annotated at different levels of granularity. For **CellularQA**, we used three nuclei-level annotation datasets: MoNuSeg (Kumar et al., 2019, 2017) (51 images), Kumar (Kumar et al., 2017) (30 images), and PanopTILs (Liu et al., 2024). Each contains expert-verified nuclei boundaries and classifications. For PanopTILs, we restricted to the training set, which provides larger annotated regions of interest (ROIs) compared to the test set. We further limited PanopTILs to ROIs containing neoplastic and/or immune cells and only included patients with overall survival data. After filtering, we obtained 589 ROIs from PanopTILs (111 patients). For **DataQA** and **SlideQA**, we retrieved WSIs for the patients included above. When multiple WSIs existed for a patient, we randomly selected one, yielding 151 WSIs in total. For **PatchQA's** diagnostic capability, we used six BRCA patients (2 invasive ductal, lobular, and metaplastic) from TCGA Uniform tumour (UT) dataset (Komura, 2022), which provides tumour-level ROIs. We retained only three tumour ROIs per patient from the TCGA-UT dataset, randomly sampled when more than three were available. While SlideQuest is developed using breast cancer as an exemplar, we strongly encourage the community to extend this benchmark framework to other disease types.

Appendix B. Case study final report

final_report.md 2025-09-08

PAM50 Subtype Morphological Features Exploration

Overview

This report summarizes the key morphological features detected in H&E WSIs for each PAM50 breast cancer subtype (Luminal A, Luminal B, Basal-like, HER2-enriched), relates them to the underlying tumor biology, and discusses their clinical implications. Cell segmentation results are summarized in the table below:

Cell Type	Luminal A	Luminal B	Basal-like	HER2-enriched
Background	0	4	1	5
Connective	32	3	32	13
Inflammatory	2	11	86	6
Necrosis	1	7	1	6
Neoplastic	3	180	10	8
Non-Neoplastic Epithelial	0	0	3	3

Subtypes characterics

Luminal A

Morphological Features: Low nuclear grade, well-formed tubules/glandular structures, low mitotic rate, minimal necrosis, low pleomorphism.

Biology: Luminal A tumors are typically ER-positive, have low proliferation rates, and retain glandular differentiation. The predominance of connective tissue and low neoplastic/necrotic cell counts reflect their indolent nature.

Clinical Implications: Luminal A cancers have the best prognosis, respond well to hormone therapy, and rarely require chemotherapy. Their morphology supports a less aggressive clinical course.

Luminal B

Morphological Features: Intermediate to high nuclear grade, less tubule formation, higher mitotic rate, moderate pleomorphism, possible necrosis.

Biology: Luminal B tumors are also ER-positive but have higher proliferation and more genomic instability than Luminal A. The high neoplastic cell count and increased necrosis reflect their more aggressive biology.

Clinical Implications: Luminal B cancers have a worse prognosis than Luminal A, may require chemotherapy in addition to hormone therapy, and are more likely to recur.

1 / 1

Figure B.1: Final analysis markdown report (Part 1 of 2) produced by Nova for the exploration of the morphological features associated with molecular PAM50 breast cancer subtypes.

Basal-like

Morphological Features: High nuclear grade, high mitotic rate, pushing margins, central necrosis, syncytial growth pattern.

Biology: Basal-like tumors are often triple-negative (ER-, PR-, HER2-), highly proliferative, and genomically unstable. The high inflammatory and connective cell counts, along with necrosis, reflect their aggressive and immune-infiltrated nature.

Clinical Implications: Basal-like cancers have the poorest prognosis, limited targeted therapy options, and are more likely to respond to chemotherapy. Their morphology is associated with rapid growth and early metastasis.

HER2-enriched

Morphological Features: High nuclear grade, frequent comedo-type necrosis, solid growth pattern, high mitotic rate, prominent nucleoli.

Biology: HER2-enriched tumors overexpress the HER2 protein, leading to increased proliferation and genomic instability. The mix of neoplastic, necrotic, and connective cells reflects their high grade and aggressive behavior.

Clinical Implications: HER2-enriched cancers have an intermediate to poor prognosis but benefit from HER2-targeted therapies (e.g., trastuzumab). Their morphology supports the need for aggressive treatment.

Shared and Unique Features

- **Shared:** High mitotic rate and high nuclear grade are present in both Basal-like and HER2-enriched subtypes, reflecting their aggressive biology.
- **Unique:** Luminal A is distinguished by low grade and well-formed structures; Basal-like by necrosis and inflammation; HER2-enriched by comedo-type necrosis and solid growth.

Conclusion

The detected morphological features and cell segmentation results are coherent with the known biology and clinical behavior of each PAM50 subtyp found in 'The molecular basis of breast cancer pathological phenotypes' (PMC5499709). These findings can inform prognosis and guide treatment decisions in breast cancer management.

Figure B.2: Final analysis markdown report (Part 2 of 2) produced by Nova for the exploration of the morphological features associated with molecular PAM50 breast cancer subtypes.

Appendix C. Links to datasets

Table C.1: Public datasets used to construct SlideQuest.

Dataset	Link
TCGA BRCA (WSI + metadata)	https://portal.gdc.cancer.gov/
PanopTILs	https://sites.google.com/view/panoptils/
MoNuSeg	https://monuseg.grand-challenge.org/Data/
Kumar	Google Drive link
TCGA-Uniform Tumour (TCGA-UT)	https://huggingface.co/datasets/dakomura/tcga-ut

Appendix D. Python environment

A local Python 3.11 environment with basic modules and data science libraries is used for code execution and file system operations. The interpreter automatically raises an error if the code generated by the LLM imports packages outside the allowed set. Security-sensitive packages such as `os`, which enable unrestricted access to the file system, are disallowed by default. A maximum number of 10^7 code operations is allowed to avoid infinite loops. Packages and libraries available in the interpreter are listed in Tables D.1 and D.2.

Table D.1: Default Python packages available to all baselines

Category	Modules
Data structures and iteration	collections, itertools
Date and time handling	datetime, time
Mathematical functions	math
Random number generation	random
Text and regular expressions	re
Statistical utilities	stat, statistics
Data structures (queues)	queue
Unicode utilities	unicodedata
Data serialization and file paths	json, pathlib

Table D.2: Additional Python data science libraries available to all baselines

Category	Libraries
Numerical computing	numpy, scipy
Data analysis and statistics	pandas, statsmodels
Machine learning and survival analysis	sklearn, sksurv, lifelines
Visualization	matplotlib, seaborn
Image processing and computer vision	cv2, skimage, PIL, openslide
Deep learning	torch, torchvision
Graphs and single-cell analysis	networkx, scanpy
Data storage and serialization	zarr, h5py, pickle
Geospatial and spatial data	geopandas, spatialdata, shapely
Utilities (I/O, file ops, progress bars)	pathlib, shutil, tqdm

Appendix E. NOVA tools

The core tools within NOVA for tasks like cell segmentation, feature extraction, metadata parsing, and region-of-interest analysis are applicable to histopathology images from any tissue.

Table E.1: Histology ROI captioning and analysis tools available in Nova.

Tool Name	Description
caption_single_histology_image_tool	Generate a descriptive caption for a single histology image.
caption_and_summarize_set_of_histology_images_tool	Caption multiple histology images and provide a summary.
score_single_histology_image_using_text_tool	Score a histology image based on text-based criteria.
encode_histology_roi_tool	Encode histology region of interest into vector representation.

Table E.2: Dataset processing check tools available in Nova.

Tool Name	Description
dataset_of_wsi_get_valid_slide_paths_tool	Get valid WSI file paths from a directory with optional extension filtering.
dataset_of_wsi_check_tissue_segmentation_exists_tool	Check if tissue segmentation files exist for a dataset of WSIs.
dataset_of_wsi_check_patch_coordinates_exist_and_schema_tool	Check if patch coordinate files exist and validate their schema for a dataset of WSIs.
dataset_of_wsi_check_patch_features_exist_and_schema_tool	Check if patch feature files exist and validate their schema for a dataset of WSIs.
dataset_of_wsi_check_slide_features_exist_and_schema_tool	Check if slide-level feature files exist and validate their schema for a dataset of WSIs.

Table E.3: Dataset processing pipeline tools available in Nova.

Tool Name	Description
dataset_of_wsi_tissue_segmentation_tool	Perform tissue segmentation on a dataset of WSI files.
dataset_of_wsi_patch_coordinate_extraction_tool	Extract patch coordinates from tissue regions in a dataset of WSIs.
dataset_of_wsi_patch_features_extraction_tool	Extract patch-level features from a dataset of WSIs using foundation models.
dataset_of_wsi_slide_features_extraction_tool	Extract slide-level features from patch features for a dataset of WSIs using slide encoders like TITAN (Ding et al., 2025), MADELEINE (Jaume et al., 2024b), and PRISM (Shaikovski et al., 2024).
dataset_of_wsi_create_score_heatmap_tool	Create score heatmaps overlaid on WSIs for visualization and analysis.

Table E.4: Documentation retriever tools available in Nova.

Tool Name	Description
trident_docs_retriever	Search and retrieve information from Trident documentation for WSI processing.
lazyslide_docs_retriever	Search and retrieve information from LazySlide documentation for WSI analysis.
hovernet_docs_retriever	Search and retrieve information from HoverNet documentation for nuclei segmentation and classification.

Table E.5: Nuclei segmentation and contour analysis tools available in Nova.

Tool Name	Description
segment_and_classify_nuclei_in_histology_roi_tool	Segment and classify nuclei in histology ROIs into six classes using HoVer-Net.
get_contour_area	Calculate the area of a contour.
get_contour_perimeter	Calculate the perimeter of a contour.
get_contour_convex_hull	Calculate the convex hull of a contour.

Table E.6: Single WSI-level processing and analysis tools available in Nova.

Tool Name	Description
visualize_text_prompt_similarity_on_wsi_tool	Visualize text prompt similarity scores overlaid on WSI tissue regions.
predict_wsi_label_tool	Predict WSI-level class labels using text-based zero-shot classification.
generate_wsi_report_with_prism_tool	Generate a pathology report for a WSI using the PRISM model.
caption_single_wsi_tool	Generate descriptive captions for a single WSI by clustering and summarizing.
score_tiles_by_text_in_a_wsi_tool	Score individual tiles in a WSI based on text-based similarity criteria.
retrieve_properties_from_wsi_tool	Retrieve metadata and properties from a single WSI file.
extract_tissue_in_wsi_tool	Perform tissue segmentation on a single WSI file.
extract_tissue_tiles_in_wsi_tool	Extract tissue tiles/patches from a single WSI file.
extract_patch_features_in_wsi_tool	Extract patch-level features from a single WSI using foundation models.
encode_wsi_tool	Encode a single WSI with slide-level features using LazySlide backend.
check_tissue_segmentation_key_in_wsi_tool	Check if tissue segmentation results exist for a specific key in WSI.
check_tile_key_in_wsi_tool	Check if tile extraction results exist for a specific key in WSI.
check_patch_features_key_in_wsi_tool	Check if patch features exist for a specific key in WSI.
check_slide_features_key_in_wsi_tool	Check if slide-level features exist for a specific key in WSI.
check_clustering_key_in_wsi_tool	Check if clustering results exist for a specific key in WSI.
check_reduction_key_in_wsi_tool	Check if dimensionality reduction results exist for a specific key in WSI.
access_zarr_hierarchy	Access and explore the hierarchical structure of WSI Zarr files.
read_zarr_data_tool	Read data from a Zarr file.
visualize_wsi_tool	Create visualizations of WSI with optional tissue contours and tile overlays.
reduce_single_wsi_patch_feature_space_tool	Reduce dimensionality of patch features using PCA, UMAP, or t-SNE.
run_leiden_clustering_tool	Perform Leiden clustering on patch features for morphological analysis.
visualize_morphological_clusters_on_wsi_tool	Visualize morphological clusters overlaid on WSI tissue regions.
get_topk_close_patch_coords_to_embedding_space_clusters_tool	Get coordinates of top-k patches closest to embedding space cluster centers.
read_rectangle_region_from_wsi_tool	Extract rectangular regions from WSI at specified coordinates and magnification.

Table E.7: WSI classification tools available in Nova.

Tool Name	Description
`train_test_wsi_classification_mil_model`	Train and test multiple instance learning models for WSI classification.
`create_wsi_classification_splits`	Create train/validation/test splits for WSI classification datasets.
`prepare_wsi_classification_metadata`	Prepare metadata files for WSI classification experiments.

```python
def dataset_of_wsi_tissue_segmentation_tool(
    job_dir: str,
    wsi_source: str,
    skip_errors: bool = False,
    search_nested: bool = False,
    holes_are_tissue: bool = True,
    batch_size: int = 64,
    segmentation_model_name: str = 'grandqc',
    tissue_seg_confidence_thresh: float = 0.5,
    overwrite: bool = False,
    skip_specific_wsi: list[str] | None = None,
    keep_only_these_wsi: list[str] | None = None,
    max_workers: int = 16,
) -> dict:
    """
    Run tissue segmentation on multiple WSIs and return the locations of output files.
    Optimized to process multiple WSIs, but can be used with selected WSIs as well.

    Tissue segmentation is the first step for WSI pipelines (patching, feature extraction, etc.).
    Options control whether holes are treated as tissue, thresholding for tissue prediction,
    and artifact removal.

    Notes:
      - If overwrite=True, run segmentation on all valid slides in `wsi_source`.
      - Creates:
          {job_dir}/contours_geojson/{wsi_name}.geojson
          {job_dir}/contours/{wsi_name}.jpg
          {job_dir}/thumbnails/{wsi_name}.jpg
          {job_dir}/_config_segmentation.json
          {job_dir}/_logs_segmentation.txt
      - If overwrite=False, check for existing results and skip processing if found.
      - 'grandqc' performs artifact filtering; 'hest' does not.
      - GeoJSON outputs are GeoPandas GeoDataFrames with `tissue_id` and `geometry`.

    Prerequisites:
      - `job_dir` exists and is writable.
      - `wsi_source` contains valid WSI files.

    Returns (dict):
      - 'dir_with_geojson_contours'
      - 'dir_with_tissue_contours_jpg'
      - 'dir_with_slide_thumbnails'
      - 'tissue_segmentation_log_file'
      - 'tissue_segmentation_config_file'
      - 'number_of_processed_segmentations'
      - 'operation_log'

    Args:
      job_dir: Path to job directory.
      wsi_source: Path to input WSI directory.
      skip_errors: Skip WSIs with errors (default=False).
      search_nested: Recursively search `wsi_source` (default=False).
      holes_are_tissue: Treat holes as tissue (default=True).
      batch_size: Batch size for tile processing (default=64).
      segmentation_model_name: ['grandqc','hest'] (default='grandqc').
      tissue_seg_confidence_thresh: Confidence threshold (default=0.5).
      overwrite: Rerun segmentation if True (default=False).
      skip_specific_wsi: List of WSIs to skip (default=None).
      keep_only_these_wsi: List of WSIs to keep (default=None).
      max_workers: Number of workers (default=16).
    """
```

Figure E.1: Example Python tool function: `dataset_of_wsi_tissue_segmentation_tool`

Appendix F. SlideQuest question format

Each question in SlideQuest follows a unified schema that specifies the task, inputs, and evaluation criteria. The schema enforces consistency across queries while remaining flexible to different modalities and levels of analysis. Core fields include metadata (`id`, `data_type`, `dataset_relative_path`), the main `question`, and any `additional_instructions` or `output_instructions` required for reproducibility. Evaluation is standardized through the `id_column` and `columns_to_compare_and_tolerance`, which define how agent outputs are matched to reference answers. Tolerances may be expressed as numeric thresholds (e.g., allowable percentage error for quantitative tasks) or as sets of acceptable responses for text-based answers. Each query also records a biomedical `rationale` and verification flags (`is_pathologist_verified`, `is_biomedical_scientist_verified`) to ensure clinical and scientific validity. Table F.1 summarizes the schema fields, and Figures F.1 to F.4 shows example instances from each benchmark category. Whenever relevant, question schema's `additional_instructions` key specifies to set the seed to 42 for non-deterministic packages (numpy, random, torch, *etc.*) to ensure reproduciliblity between ground truth answers as well as between different runs of the Nova and baselines.

Table F.1: User query schema used in Nova.

Field	Description
id	Unique identifier for the query.
data_type	Type of data involved in the task (e.g., single WSI, multiple WSIs, summary of multiple WSIs).
dataset_relative_path	Relative path to the dataset.
question	The actual biomedical or computational question to be answered.
additional_instructions	Supplementary instructions to provide clarifications.
output_instructions	Explicit requirements on how outputs must be formatted and stored.
id_column	Column name used to compare outputs of agent with ground truth answers.
columns_to_compare_and_tolerance	Fields that are compared and their respective tolerances.
rationale	Biomedical motivation or reasoning for why the task is important.
is_pathologist_verified	Boolean indicating whether the question has been verified by a pathologist.
is_biomedical_scientist_verified	Boolean indicating whether the question has been verified by a biomedical scientist.


```
{
   "id": "21",
   "data_type": "single_wsi",
   "slide_relative_path": "tcga_brca_to_use/WSI_flat/TCGA-EW-A1P8-01Z-00-DX1.E9852193-8CDD-49EF-B49B-
        DA6931198F0D.svs",
   "question": "For the histology slide at {path_to_slide}, what percentage of the tissue pixels are more
        strongly stained with hematoxylin than with eosin?",
   "additional_instructions": "Your working directory is: {working_dir}, which you can use to save
        intermediate outputs and results. Work with the lowest resolution level of the whole slide image.
        You can set non-tissue regions to white before computing stains. Apply 0-1 scaling to each stain
        matrix before computing percentages. Consider a pixel hematoxylin-dominant if its hematoxylin value
         exceeds the eosin value by more than 0.02. Report the hematoxylin-dominant tissue percentage to
        two decimal places.",
   "output_instructions": "You **must** save your outputs as a JSON file in your working directory. Create
        a file named `answer.json` containing your results as a **list of dictionaries** (JSON array). Use
        4-space indentation for readability. The slide name should not include the file extension.
        hematoxylin_percent **must** be a float rounded to two decimal places. For example, save the
        following format: [{\"slide_id\": \"slide_id1\", \"hematoxylin_percent\": 44.23}] to the answer.
        json file with proper indentation.",
   "id_column": "slide_id",
   "columns_to_compare_and_tolerance": {
      "hematoxylin_percent": 0.1
   },
```

```
    "rationale": "This calculation quantifies the proportion of tissue dominated by nuclear staining (
        hematoxylin) versus cytoplasmic/protein staining (eosin), which can indicate cellularity and tissue
         composition.",
    "is_pathologist_verified": true,
    "is_biomedical_scientist_verified": true
}
```


Figure F.1: Example DataQA question schema


```
{
    "id": "1",
    "data_type": "multiple_wsi",
    "dataset_relative_path": "panoptils_idc_mini/rgbs/",
    "question": "For each breast histology image in {path_to_dataset}, what is the neoplastic cellularity
        percentage, defined as (number of neoplastic cell nuclei / total number of nuclei) * 100?",
    "additional_instructions": "Your working directory is: {working_dir}, which you can use to save
        intermediate outputs and results. Don`t consider any background nuclei.",
    "output_instructions": "You **must** save your outputs as a JSON file in your working directory. Create
        a file named `answer.json` containing your results as a **list of dictionaries** (JSON array). Use
        4-space indentation for readability. Percentages should be float values between 0 and 100 with
        exactly 2 decimal places. For example, if the image name is `sample1.tif` and the neoplastic
        cellularity is 54.32%, you should save: [{\"slide_id\": \"sample1\", \"
        neoplastic_cellularity_percentage\": 54.32}] to the answer.json file with proper indentation.",
    "id_column": "slide_id",
    "columns_to_compare_and_tolerance": {
      "neoplastic_cellularity_percentage": 0.15
    },
    "rationale": "Accurate quantification of neoplastic cellularity is essential for assessing neoplastic
        burden, eligibility for molecular testing, and making informed treatment decisions in oncology.",
    "is_pathologist_verified": true,
    "is_biomedical_scientist_verified": true
}
```


Figure F.2: Example CellularQA question schema


```
{
    "id": "25",
    "data_type": "multiple_wsi",
    "dataset_relative_path": "panoptils_idc_mini/rgbs/",
    "path_to_metadata": "panoptils_idc_mini/metadata/OS_days/splits.csv",
    "question": "Does adding the average nuclear solidity of neoplastic cells improve Cox proportional
        hazards model prediction of survival compared to using only clinical covariates (age, stage) in the
         dataset at {path_to_dataset}? Report the mean C-index IPCW across 5 folds for both conditions",
    "rationale": "Nuclear solidity is a proxy for morphological abnormality; testing its added prognostic
        value over clinical variables assesses integration of imaging biomarkers with standard clinical
        data",
    "additional_instructions": "Your working directory is: {working_dir}. Metadata with survival time, event
         status, clinical covariates (age, stage), patient image paths, and premade splits are in {
        path_to_metadata} (column `OS` indicates event status, 1 being event observed). To determine the
        absolute path to the actual image files, join {path_to_dataset} (the dataset root) with the value
        from the `image_name` column. For each image, compute mean neoplastic nuclear solidity, then
        average per patient. Using 5-fold CV, train two Cox models: (1) clinical covariates only, (2)
        clinical covariates and neoplastic nuclear solidity. Compute test-set C-index IPCW per fold, then
        report mean across folds. Fix random seeds (NumPy, PyTorch, scikit-learn, torch, random, etc.) to
        42. A higher risk score indicated by higher values!",
    "output_instructions": "You **must** save your results to a JSON file named `answer.json` in your
        working directory. The file should contain a list of dictionaries with your outputs. Use json.dump
        () with indent=4 for readability. Each object must contain `model` (clinical_only/
        clinical_plus_solidity) and `mean_c_index` (3 decimals). For example: [{\"model\": \"clinical_only
        \", \"mean_c_index\": 0.652}, {\"model\": \"clinical_plus_solidity\", \"mean_c_index\": 0.687}]",
    "id_column": "model",
    "columns_to_compare_and_tolerance": {
       "mean_c_index": 0.15
```

```
    },
    "is_pathologist_verified": true,
    "is_biomedical_scientist_verified": true
}
```

Figure F.3: Example PatchQA question schema

```
{

    "id": "11",
    "data_type": "multiple_wsi",
    "dataset_relative_path": "tcga_brca_to_use/WSI_flat",
    "path_to_metadata": "tcga_brca_to_use/tcga_brca_metadata/OS_days/metadata.csv",
    "question": "Classify all WSIs at {path_to_dataset} into high and low likelihood of TP53 mutation and
        report the number of cases in each category. For patients who also have survival data, compare mean
         overall survival days between the predicted high- and low-likelihood groups and test whether the
        difference is statistically significant. Consider both censored and uncensored patients.",
    "rationale": "TP53 alterations are linked to prognosis; relating predicted mutation propensity to
        survival helps validate risk stratification from histology-derived features.",
    "additional_instructions": "Your working directory is: {working_dir}, which you can use to save
        intermediate outputs and results. Use {path_to_metadata} for patient ids, slide ids, and survival
        days. Use a two-sided Mann-Whitney U test to compare survival days distributions between groups.
        Patients missing survival data can be ignored for the survival comparison.",
    "output_instructions": "You must save your results to a JSON file named `answer.json` in your working
        directory. The file must contain a list of dictionaries (JSON array). Use json.dump() with indent=4
         for readability. The dictionary must include numeric keys: `number_high_likelihood` (int), `
        number_low_likelihood` (int), `avg_survival_days_in_high_likelihood` (float, round to 1 decimal
        place), `avg_survival_days_in_low_likelihood` (float, round to 1 decimal place), and `p-value` (
        float, round to 3 decimal places). For example: [{\"number_high_likelihood\": 38, \"
        number_low_likelihood\": 74, \"avg_survival_days_in_high_likelihood\": 820.5, \"
        avg_survival_days_in_low_likelihood\": 1160.2, \"p-value\": 0.010}].",
    "id_column": null,
    "columns_to_compare_and_tolerance": {
        "number_high_likelihood": 0.15,
        "number_low_likelihood": 0.15,
        "avg_survival_days_in_high_likelihood": 1.0,
        "avg_survival_days_in_low_likelihood": 1.0,
        "p-value": 0.15
    },
    "is_pathologist_verified": true,
    "is_biomedical_scientist_verified": true
}
```

Figure F.4: Example SlideQA question schema

F.1. SlideQuest evaluation

To align keys between output and ground truth JSON files, we use Hungarian matching, which minimizes penalties from alternative but valid formatting choices. Values are then compared using task-specific tolerances: percentage thresholds for quantitative outputs (15% in this study) and sets of acceptable responses for textual outputs. A value is scored as 1 if it falls within tolerance, and 0 otherwise. If no JSON is produced, the question is given a score of 0. Overall benchmark performance is reported as the average score across all questions, including the ones where no output JSON was produced. To define failure rate, we find the percentage of questions where the score is 0 or no valid JSON file is created.

Appendix G. SlideQuest details

Table G.1: Numbers of questions requiring each type of capability, per benchmark category.

Theme	Capabilities	CellularQA	DataQA	PatchQA	SlideQA	All
WSI data ingestion	WSI reading	0	12	0	14	26
	WSI metadata retrieval	0	20	0	0	20
	WSI patching	0	3	0	2	5
Basic data ingestion	Image reading	25	0	25	0	50
	CSV reading	0	0	19	7	26
	Embeddings reading	0	0	0	1	1
Image processing	Texture analysis	2	0	0	0	2
	Image filtering	0	1	0	0	1
	Stain estimation	0	1	0	0	1
Spatial analysis	Morphometry	10	4	10	1	25
	Spatial calibration	3	8	2	0	13
	Distance computation	6	0	1	0	7
Statistics	Summary statistics	23	7	8	4	42
	Statistical analysis	2	0	8	5	15
	Cross-validation	0	0	11	3	14
	Model evaluation	0	0	10	3	13
	Attention-based patch retrieval	0	0	0	1	1
Pretrained model use	Nuclei cell type classification	13	0	12	1	26
	Nuclei segmentation	14	0	10	1	25
	Nuclei detection	11	0	2	0	13
	Tissue segmentation	2	10	0	0	12
	Zero-shot WSI interpretation	0	0	0	10	10
	Patch embedding	0	0	7	2	9
	Zero-shot multi-patch interpretation	0	0	6	0	6
	WSI embedding	0	0	0	3	3
	Artefact segmentation	0	1	0	0	1
Modelling	Supervised learning	0	0	10	1	11
	Survival modelling	0	0	3	0	3
	Few-shot learning	0	0	0	2	2
	Multiple-instance learning	0	0	0	2	2
	Clustering	1	0	0	0	1
General	Scalar calculation	3	6	0	0	9
	Problem solving	0	0	0	2	2

Appendix H. Experimental setup

H.1. NOVA configuration

Nova is built based on `smolagents` library. We use the CodeAgent class with the parameters detailed in Table H.1. We provide a dynamic set of tools depending on the tools category chosen by the user. Nova with tools, *i.e.,* the default configuration, uses all 49 tools. When evaluating on SlideQuest, no baseline and Nova has access to web search. The web search tool is added when evaluating the qualitative case study. Additionally, we increase the maximum number of steps to 200 and memory reset after query to False for the case study to emulate a real interactive conversation with memory.

Table H.1: Default Nova configuration parameters

Parameter	Description
tools	all tools in Nova
model	SmolAgentsLLM configured with GPT-4.1 by default
additional_authorized_imports	list of authorized libraries
executor_type	local
planning_interval	null
use_structured_outputs_internally	True
verbosity_level	1
provide_run_summary	False
max_steps	20
name	codeagent_with_tools
description	The code agent has access to many tools for whole slide image data processing and analysis. Additionally, it can use the following libraries (list of libraries).
special_instructions	## Security Restrictions - **Strict restrictions:** You are absolutely not allowed to use these modules in your code: ['os'] ## Core Objectives & Approach - Your primary goal is to help the user fully achieve their objective. - Always address the user's underlying question or need, not just the surface request. Ensure your answer is complete and fully covers the question and any related aspects. - **Task Decomposition:** Break down complex tasks into smaller, manageable subtasks and address them sequentially. Execute each subtask one by one, using the appropriate tools and libraries. ## Library & Tool Usage - **Task Resolution:** Only generate tools and functions from scratch if provided tools and libraries are not able to solve the task. - **Computer Vision:** When working with image processing, contours, segmentation, or spatial analysis tasks, make use of existing computer vision libraries (cv2, skimage, scipy.spatial and others) before writing custom implementations from scratch. - The machine you're running in has a gpu. Make sure to always use cuda:0 when a device is required to run a tool. ## Output & Communication - Display or share outputs---such as figures, files, or results---directly with the user whenever possible - Exactly follow user instructions on output format, file names, and other details - Communicate in a sincere, helpful, and user-focused way. Be clear, honest, and avoid unnecessary jargon.

H.2. LLMs variants and parameters

We use Azure OpenAI endpoints to access the LLMs and benchmark 4 variants of GPT models: GPT-4.1, GPT-4.1-mini, GPT-5-mini, and GPT-5. The experiments were run within Azure ML, which provides streamlined monitoring and experiment management. For the GPT-4.1 variants, we used a fixed `temperature=0` to reduce stochasticity and `max_retries=20` to overcome any internal LLM errors. GPT-5 series were run with `temperature=1` as it is currently the only permitted value by the OpenAI API for GPT-5 series models. We use the default reasoning configuration for GPT-5. All the baselines were run with the same LLM parameters for fair comparison.

H.3. Experiments Runner

We provide an experimental runner to streamline the execution of SlideQuest experiments. Built on `hydra`, it allows dynamic configuration of baselines and system arguments. The runner manages the agent's lifecycle—including stepwise execution, tool invocation, and result aggregation—while handling evaluation, logging, and saving of intermediate model outputs in separate folders. By resetting the agent's state and providing a fresh working directory for each query, it prevents leakage between benchmark tasks. This setup also facilitates parallelization, which is particularly useful since each experiment is repeated three times. The runner will be open-sourced as part of the Nova framework.

Appendix I. SlideQuest Results

Table I.1: SlideQuest average score (higher is better) over 3 runs with standard error. Average score is weighted by number of questions in each category. All results with GPT-4.1.

Baseline	DataQA (n=25)	CellularQA (n=25)	PatchQA (n=25)	SlideQA (n=15)	Average
LLM only	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000
LLM with PI	0.377 ± 0.053	0.058 ± 0.011	0.039 ± 0.022	0.133 ± 0.067	0.154
LLM with PI and retries	0.443 ± 0.019	0.152 ± 0.025	0.217 ± 0.012	0.259 ± 0.002	0.269
Nova	0.777 ± 0.030	0.323 ± 0.017	0.335 ± 0.016	0.472 ± 0.027	0.477

Table I.2: Failure rate on SlideQuest (lower is better) over 3 runs with standard error. Average failure is weighted by number of questions in each category. All results with GPT-4.1.

Baseline	DataQA (n=25)	CellularQA (n=25)	PatchQA (n=25)	SlideQA (n=15)	Average
LLM only	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000
LLM with PI	0.580 ± 0.060	0.773 ± 0.027	0.947 ± 0.035	0.867 ± 0.067	0.783
LLM with PI and retries	0.507 ± 0.013	0.627 ± 0.027	0.613 ± 0.013	0.667 ± 0.039	0.596
Nova	0.200 ± 0.023	0.320 ± 0.023	0.413 ± 0.013	0.422 ± 0.059	0.330

Appendix J. Run time analysis

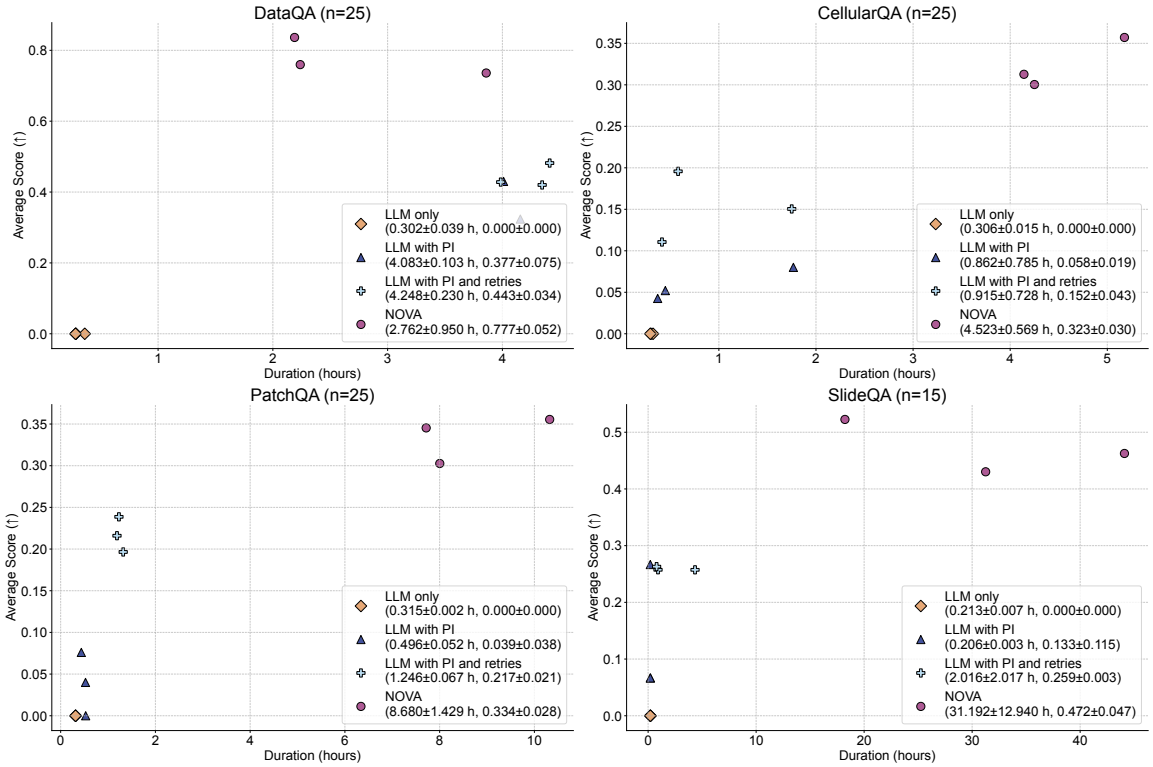

Figure J.1: Run time duration (h) vs. average score on SlideQuest for LLM only, LLM with PI, LLM with PI and retries, and Nova. All results with GPT-4.1. Legend shows average run time and score with baseline name.

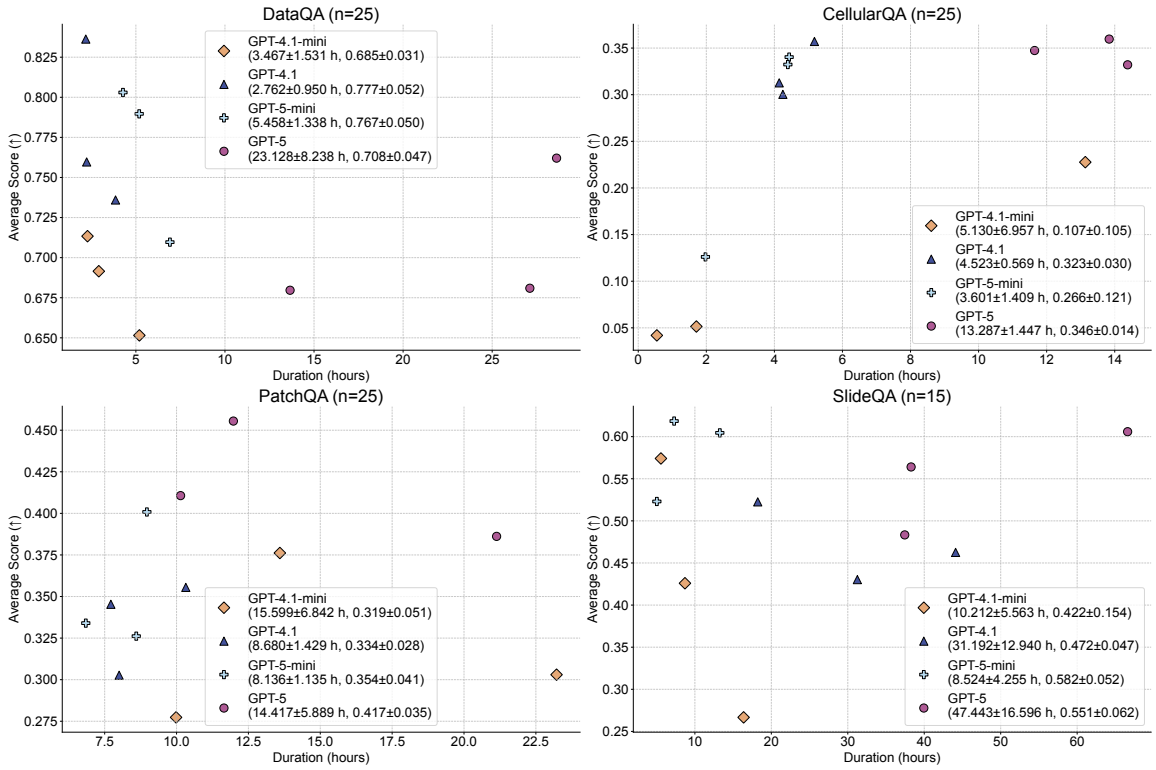

Figure J.2: Run time duration (h) vs. average score on SlideQuest for Nova with GPT-4.1-mini, GPT-4.1, GPT-5-mini, and GPT-5. Legend shows average run time and score with baseline name.

Appendix K. NOVA ablations

K.1. Are custom tools needed?

Nova includes 49 custom tools, each designed and validated by experienced biomedical scientists. To assess whether such handcrafted tools are necessary, we compare Nova against variants where the LLM must generate tools on its own. In the first setting, Nova (no custom tools), the LLM relies solely on its base knowledge to create ad hoc functions.

In the second, Nova (with RAG), the LLM is given access to vector databases built from the GitHub repositories of Trident[2], LazySlide[3], and HoVerNet[4] (the open-source libraries used for tool development), allowing it to retrieve domain-specific knowledge when creating new tools. The agent can query these repositories to dynamically generate and compose tools on the fly. This baseline assesses whether access to raw domain-specific resources is sufficient to replace carefully designed tools.

Table K.1: Average score over 3 runs with standard error for Nova (no custom tools), Nova (with RAG), and Nova (with custom tools) on SlideQuest (higher is better). Average score is weighted by number of questions in each category. All results with GPT-4.1.

Baseline	DataQA (n=25)	CellularQA (n=25)	PatchQA (n=25)	SlideQA (n=15)	Average
Nova (no custom tools)	0.537 ± 0.017	0.152 ± 0.018	0.222 ± 0.026	0.439 ± 0.010	0.326
Nova (with RAG only)	0.556 ± 0.006	0.165 ± 0.018	0.213 ± 0.015	0.464 ± 0.013	0.337
Nova (with custom tools)	0.777 ± 0.030	0.323 ± 0.017	0.335 ± 0.016	0.472 ± 0.027	0.477

Table K.2: Failure percentage over 3 runs with standard error for Nova (no custom tools), Nova (with RAG), and Nova (with custom tools) on SlideQuest (lower is better). Average failure is weighted by number of questions in each category. All results with GPT-4.1.

Baseline	DataQA (n=25)	CellularQA (n=25)	PatchQA (n=25)	SlideQA (n=15)	Average
Nova (no custom tools)	0.413 ± 0.013	0.387 ± 0.035	0.560 ± 0.061	0.378 ± 0.022	0.441
Nova (with RAG only)	0.400 ± 0.000	0.573 ± 0.013	0.573 ± 0.013	0.400 ± 0.039	0.496
Nova (with custom tools)	0.200 ± 0.023	0.320 ± 0.023	0.413 ± 0.013	0.422 ± 0.059	0.330

2. https://github.com/mahmoodlab/TRIDENT
3. https://github.com/rendeirolab/LazySlide
4. https://github.com/vqdang/hover_net

K.2. Choice of core LLM

Table K.3: Performance of different core LLMs on SlideQuest. Average score (higher is better) over 3 runs with standard error. Average score is weighted by number of questions in each category. Prices are taken from https://platform.openai.com/docs/pricing and are the sum of input and output prices per 1M tokens.

Baseline	Price per 1M tokens	DataQA (n=25)	CellularQA (n=25)	PatchQA (n=25)	SlideQA (n=15)	Average
GPT-4.1-mini	$2.00	0.686 ± 0.018	0.107 ± 0.060	0.319 ± 0.030	0.422 ± 0.089	0.379
GPT-4.1	$10.00	0.777 ± 0.030	0.323 ± 0.017	0.335 ± 0.016	0.472 ± 0.027	0.477
GPT-5-mini	$2.25	0.767 ± 0.029	0.266 ± 0.070	0.354 ± 0.024	0.582 ± 0.030	0.482
GPT-5	$11.15	0.708 ± 0.047	0.346 ± 0.008	0.417 ± 0.020	0.551 ± 0.036	0.498

Table K.4: Failure percentage of different core LLMs on SlideQuest. Failure percentage (lower is better) over 3 runs with standard error. Average failure percentage is weighted by number of questions in each category. Prices are taken from https://platform.openai.com/docs/pricing and are the sum of input and output prices per 1M tokens.

Baseline	Price per 1M tokens	DataQA (n=25)	CellularQA (n=25)	PatchQA (n=25)	SlideQA (n=15)	Average
GPT-4.1-mini	$2.00	0.280 ± 0.023	0.653 ± 0.109	0.547 ± 0.048	0.511 ± 0.135	0.496
GPT-4.1	$10.00	0.200 ± 0.023	0.320 ± 0.023	0.413 ± 0.013	0.422 ± 0.059	0.330
GPT-5-mini	$2.25	0.213 ± 0.035	0.373 ± 0.096	0.373 ± 0.035	0.289 ± 0.022	0.315
GPT-5	$11.15	0.270 ± 0.025	0.280 ± 0.023	0.373 ± 0.027	0.244 ± 0.022	0.297

Appendix L. NOVA Failure Analysis

We show specific failure cases from Nova highlighting different modes of failure. Question 2 from DataQA (Figure L.1) shows failure due to tool issues. In solving this task, Nova uses tool `score_single_histology_image_using_text_tool`, which incorrectly predicts the image class. As an example of the agent ignoring existing tools or data, we look at question 24 from DataQA (Figure L.2), where Nova fails as it incorrectly recomputes the convexity of tissue regions. Specfically, Nova uses the full tissue area when recomputing convexity, instead of using the tissue area without including holes. The convexity measure is already provided by tool `extract_tissue_in_wsi_tool` (the docstring also mentioning this), which Nova has already referenced to obtain the tissue segmentation.

(a) Question

```
I have three images of a tumour from a breast cancer patient at /.../tcga_ut_brca_zeroshot/TCGA-AR-A2LR.
When considering all the images, does the patient have invasive lobular, invasive ductal, or metaplastic
breast cancer? Additional instructions: ...
```

(b) Code (partial)

```
----------------------------------- Step 2 -----------------------------------
  classes = ['invasive lobular', 'invasive ductal', 'metaplastic']
  results = []
  for img_path in image_paths:
      result = score_single_histology_image_using_text_tool(
          image_path=img_path,
          classes=classes,
          device='cuda:0',
          apply_softmax=True,
          prompts=None
      )
      print(f"Image: {img_path}", result)
      results.append(result)

----------------------------------- Step 3 -----------------------------------
  import re

  # Parse similarity scores and sum probabilities for each class
  class_probs = {cls: 0.0 for cls in classes}
  for result in results:
      for score_str in result['similarity_scores']:
          cls, prob = score_str.split(':')
          cls = cls.strip()
          prob = float(prob.strip())
          class_probs[cls] += prob
  print(class_probs)

  # Select the class with the highest summed probability
  diagnosis = max(class_probs, key=class_probs.get)
  patient_id = Path(patient_folder).name
  output = [{"patient_id": patient_id, "diagnosis": diagnosis}]
```

(c) Results and ground truth

```
Tool output:
   invasive lobular: 0.007
   invasive ductal: 1.981
   metaplastic: 1.012

Agent result: invasive ductal
Ground Truth: metaplastic
```

Figure L.1: Question 2 from PatchQA: A failure case as Nova relies on the output of tool `score_single_histology_image_using_text_tool`, which incorrectly predicts that a majority of images are of type invasive ductal breast cancer and only correctly predicts metaplastic breast cancer for one out of three images.

(a) Question

```
For histology WSI at /.../tcga_brca_to_use/WSI_flat/TCGA-A2-A0CM-01Z-00-DX1.AC4901DE-4B6D-4185-BB9F
-156033839828.svs, compute the convexity of each tissue region, then return the average convexity per slide.
Work in pixel space! Additional instructions: ...
```

(b) Code (partial)

```
----------------------------------- Step 5 -----------------------------------
# Extract exterior coordinates as a list of [x, y] points
polygon = tissues_gdf.geometry.iloc[0]
contour_points = list(polygon.exterior.coords)
print(contour_points)

# Compute area
area_result = get_contour_area(contour=contour_points)
print('Area:', area_result)
# Compute convex hull
hull_result = get_contour_convex_hull(contour=contour_points)
convex_hull = hull_result['contour_convex_hull']
print('Convex hull:', convex_hull)

# Compute convex hull area
hull_area_result = get_contour_area(contour=convex_hull)
print('Convex hull area:', hull_area_result)
```

(c) Execution logs

```
[...]
Area: {'contour_area': 3084211442.668519}
Convex hull: [...]
Convex hull area: {'contour_area': 3603253238.057195}
```

(d) Code (partial)

```
----------------------------------- Step 6 -----------------------------------
import json
slide_id = 'TCGA-A2-A0CM-01Z-00-DX1.AC4901DE-4B6D-4185-BB9F-156033839828'
area = 3084211442.668519
hull_area = 3603253238.057195
convexity = area / hull_area
avg_convexity = round(convexity, 3)
result = [{"slide_id": slide_id, "avg_convexity": avg_convexity}]
```

(e) Results and ground truth

```
Agent result: 0.856
Ground Truth: 1.196
```

Figure L.2: Question 24 from DataQA: Failure case as the agent incorrectly recomputes convexity, a value already available through one of the used tools.