AnnotAid: AI-Driven Data Annotation Tool for Histology Images

Peter Škríba*
Adam Bublavý†
Angelika Kissová‡
Supervised by: Vanda Benešová§ & Kristína Mikuš Kuracinová¶

Faculty of Informatics and Information Technologies
Slovak University of Technology
&
Faculty of Medicine, Comenius University
Bratislava / Slovakia

Abstract

Automatic processing of digital histology images can greatly benefit from the utilization of deep learning methods. The development of such methods requires large amounts of annotated histological images. However, currently available annotation tools often have very poor usability, resulting in ineffective annotation processes. We aim to address the urgent need for a simplified approach to annotating histopathology images, a task that is crucial for advances in automated diagnosis and analysis. By combining our expertise, we strive to develop a user-friendly annotation tool integrated with state-of-the-art deep learning techniques. This tool is designed to alleviate the burden on pathologists during the annotation process by leveraging artificial intelligence models adapted to the various challenges in the field, such as the Nottingham Grading System of breast cancer. Through a comprehensive analysis of breast cancer and existing annotation tools, we propose a solution in the form of a multiplatform annotation tool powered by AI, developed in close cooperation with medical domain experts. By combining our knowledge and resources, we aim to bridge the gap between manual annotation processes and the potential of AI-based solutions, which will ultimately improve patient outcomes and advance medical research in breast cancer diagnosis.

The annotation tool is available at: **annotaid.com**.

Keywords: Computer Vision, Annotation Tool, Deep Neural Networks, User Experience (UX), Nottingham Grading System, Histopathology

*xskriba@stuba.sk
†xbublavy@stuba.sk
‡kissova154@uniba.sk
§vanda_benesova@stuba.sk
¶kuracinova1@uniba.sk

1 Introduction

Artificial intelligence has affected many areas of human life; one of them is the field of medicine, where the authors of various studies are trying to help doctors in their work by creating intelligent tools to help doctors diagnose multiple diseases. Annotated data plays a crucial role in training deep neural networks, yet acquiring it can be particularly challenging, especially in domains like healthcare where data may be scarce or costly to procure [3]. This entire process is not only arduous and time-consuming but also prone to errors, significantly affecting patient outcomes. Moreover, engaging domain experts in the annotation process can incur substantial expenses. Thus, annotation tools with excellent usability represent invaluable assets that streamline the process and optimize the efficiency of domain experts.

QuPath [7], ASAP [9], Orbit [6], or Cytomine [5] are annotation tools commonly employed for annotating histological images. Feedback from our domain experts indicates that the majority of these tools, particularly QuPath, possess a low learning curve and need training for proficient use.

In this work, we introduce **AnnotAid, a user-friendly annotation tool that utilizes deep learning methods** to facilitate the annotation creation process and **to support the diagnosis of Nottingham Grading System (NGS) criteria in breast cancer, which are jointly developed along with our annotation tool.** It was developed in close collaboration with medical domain experts. To create AnnotAid, we introduced a novel communication concept involving domain experts, UX experts, and AI experts. This concept draws inspiration from the methods and principles of User-Centered Design (UCD), aiming to achieve an optimal User Experience (UX) while adhering to Human-Computer Interaction (HCI) principles consistently. The annotation tool serves **to streamline communication between domain experts and development teams**.

The paper is organized as follows: Section 2 presents an overview of existing annotation tools and approaches for

Nottingham Grading System (NGS) evaluation. In Section 3, we delve into the architecture, user interface, and functionality of the developed annotation tool. Preliminary results are provided in Section 4. Finally, Section 5 outlines the conclusions drawn from our work and discusses avenues for future research.

2 Related Work

In section 2.1 the approaches to solve each criterion of NGS are reviewed, and section 2.2 provides an overview of existing annotation tools.

2.1 Nottingham Grading System

Nottingham Grading System (NGS) [8] is a modified version of the Bloom & Richardson method used for grading breast cancer. This modification aims to introduce more objectivity into the criteria. The NGS involves a semiquantitative evaluation of three morphological criteria: **nuclear pleomorphism (NP)**, **mitotic count (MC)**, and **tubular formation (TF)**. Each criterion is assigned a score from 1 to 3, resulting in a final score ranging from 3 to 9.

2.1.1 Nuclear Pleomorphism

Xu et al. [21] proposed a deep-learning framework for nuclear atypia scoring, consisting of two stages: epithelial and stromal segmentation model and nuclear atypia-scoring models (x10, x20, x40 magnification). In the first stage, the relevant areas are segmented from images (epithelial and stromal), from which smaller patches are extracted and classified into 1-3 classes. For each magnification, the nuclear atypia score is determined with majority voting and then the final score is determined with plurality voting among all magnifications. On the other hand, Mathew et al. [15] proposed a framework for the extraction and classification of individual cells into nuclear atypia scores classified with the DenseNet121 model. The main idea of the proposed framework is to redesign the three-class problem (score 1-3) of slide image classification as a six-class problem (score 1-3, lymphocytes, necrotic cells, stroma cells) on nuclei classification. The final atypia score is assigned after aggregation of the nuclei classification results via plurality voting. The authors argued that the problem reformulation as a six-class problem and no four-class problem helped to increase performance. Sreeraj M. et al. [13] and Mercan et al. [16] used YOLO and RetinaNet detection models respectively, to detect and classify individual cells or patches into nuclear atypia scores.

2.1.2 Mitotic Count

Wang et al. [20] proposed a deep learning solution named FMDet, designed for the detection of mitotic cells. The authors tackled the problem as a segmentation task, where SE-ResNeXt50 encoder and an SK-based decoder were used. To address the domain shift problem, the authors proposed Fourier-based data augmentation where the low-frequency spectrum of the source domain is replaced by the low-frequency spectrum of the target domain. Jahanifar et al. [10] and Venugopal et al. [19] adopted a two-stage approach where the initial model detected cell candidates and a subsequent model classified them as mitotic or non-mitotic cells.

2.1.3 Tubular Formation

During our in-depth analysis of tubular formation, we identified only one approach by using segmentation models. There is a big scarcity of relevant papers addressing the tubular formation problem. Tekin et al. [18] proposed a deep learning framework designed for tubule segmentation. The paper introduces a novel in-house dataset comprising 51 Whole-Slide images (WSI). The authors employed reflection padding to tackle the challenge of incomplete tubules within patches. EfficientNetB3 demonstrated superior segmentation results, achieving a dice score of 95.33%.

2.2 Annotation Tools for Histology Images

LindvaN et al. [12] presents a comparative study between manual annotation and TissueWand, revealing a significant increase in annotation speed with the tool. TissueWand, designed for histopathological sample annotation, garnered preference from pathologists for its improved user experience. **The research methodology comprised user observations, prototyping, and interviews** to achieve a balance of manual control and automatic support, enhancing feedback speed and annotation assistance.

Several tools have emerged in the field of bioimage analysis, of which **QuPath** [7] stands out. This open-source desktop software is widely used in digital pathology for its ability to retrieve and navigate large, high-resolution whole slide images (WSI), along with its versatile annotation tools complemented by a range of available plug-ins. Another tool is **ASAP** [9] (Automatic Slide Analysis Platform), which stands out for its speed of image analysis. It provides users with tools to calculate area, perimeter, and other morphological measurements of annotated structures; **Orbit** [6] is a multi-platform open-source tool that can perform various image analysis algorithms from Orbit or other platforms. It facilitates real-time collaborative annotation, allowing multiple users to work simultaneously on the same image; **Cytomine** [5], an open-source RESTful web platform, operates via Docker containers and emphasizes remote collaboration. It facilitates data model organization, semantic annotation of high-resolution images, and image quantification via machine learning algorithms.

MONAI [4] is an open-source framework tailored for healthcare deep learning, leveraging PyTorch. It comprises three main components: MONAI core, MONAI label, and MONAI deploy. This framework facilitates integration with other annotation tools, such as QuPath, through a plugin architecture. **Our goal is to provide users with instant visualization of the result using AI, and MONAI is a framework that supports such functionality.**

3 Our Proposed Annotation Tool

Section 3.1 outlines the communication concept design within our team, while Section 3.2 elaborates on the system architecture. The user interface is detailed in Section 3.3, and available annotation methods are described in Section 3.4.

3.1 Proposed Concept of Communication

In Figure 1, **we present our proposal for the design of communication concept within our team**, where the processes and procedures of the individual actors are also included. Our team comprises a **domain expert**, a **UX expert**, and an **AI expert**. The process involves two main stages: the first focuses on specifying and understanding the requirements, while the second evaluates the design and implementation against these requirements. In the subsequent paragraphs, we will outline the responsibilities of each team member.

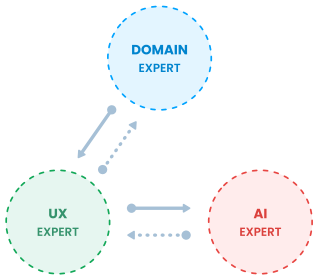

Figure 1: Concept of Communication

(1) Domain Expert: The domain experts **take the lead in specifying the requirements and functionalities** of the annotation tool under development. Throughout the development process, the UX expert collaborates with them to discuss each requirement and ensure their validation. Additionally, the domain expert provides their expertise and actively engages in data annotation, which is essential for training and enhancing the artificial intelligence methods employed by the annotation tool. In general, the domain expert serves as a potential user of the annotation tool.

(2) UX Expert: The UX Expert is **responsible for communicating with the domain expert to gain clarity on the specified requirements**. When necessary, they collaborate with the AI expert to define the requirements precisely and validate their fulfillment. Their primary objective is to incorporate the domain expert's requirements effectively into the annotation tool and guide the AI expert based on those needs.

(3) AI Expert: The AI expert is tasked with **analyzing and implementing the requirements** put forth by the UX expert. They focus on developing deep learning methods that support the annotation process.

3.2 System Architecture

The architecture diagram 2 delineates two main parts, where the implementation of the blue part is the responsibility of the UX expert and the implementation of the grey part is the responsibility of the AI expert. **The goal of the UX expert is to create the interface of the tool and integrate the AI API created by the AI expert**. This architectural view highlights the interaction and communication between the different components of the system. One of the main components of this architecture that we can highlight is the local server to support the reading and manipulation of WSI, which acts as an application module. The components of the system will be introduced in the paragraphs below.

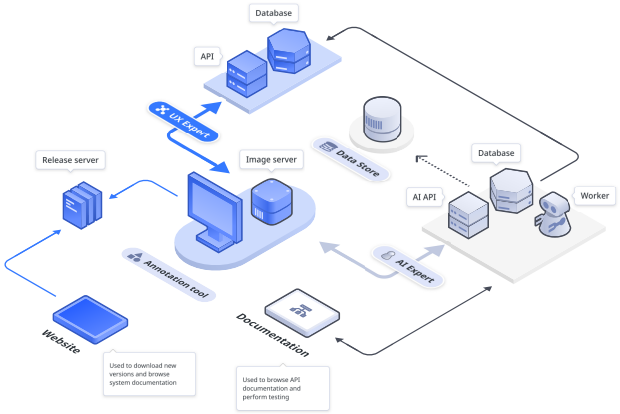

Figure 2: System Architecture

3.2.1 AI API

The AI API is developed using the FastAPI framework, facilitating communication between the annotation tool and the AI API through a REST API. The overall AI API architecture comprises distinct components, as illustrated in Figure 3: the **FastAPI backend**, **Redis message broker**, and **Celery worker**. Each component is encapsulated within a separate Docker container.

FastAPI[1] is a Python framework used to build modern and fast APIs. The FastAPI API utilizes an annotation

[1] fastapi.tiangolo.com

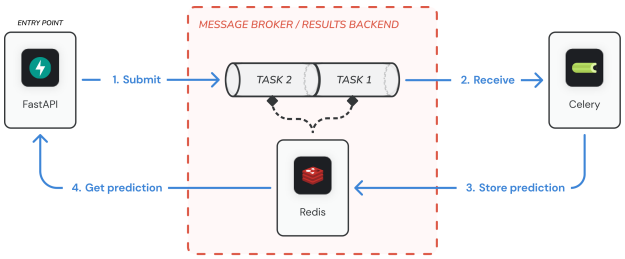

Figure 3: AI API Architecture

engine to make predictions using deep machine-learning models. **Upon receiving a request, the backend inserts the request into the message queue of the broker.** Subsequently, the client receives a unique task ID and uses the polling technique to query the task status and obtain the prediction result. The polling technique involves the client querying at regular intervals (e.g., every second) for the task result. If the task is completed, the client receives the prediction result; otherwise, they are notified to continue waiting. This architectural choice enables the asynchronous execution of a large number of prediction requests, treating them as background jobs to avoid blocking the server thread.

Redis message broker, acting as an intermediary and implemented as a Redis database, facilitates communication between the triggering application and the worker processes. **FastAPI inserts prediction tasks into this queue, and the Celery worker picks up and executes these tasks based on its workload.** The results are stored in the Redis database, allowing FastAPI to retrieve and provide them to the client.

Celery is an open-source distributed task queue system for Python that allows you to run tasks asynchronously. It offers high availability and easy horizontal scaling. The Celery worker continuously checks the queue for pending tasks. When a task is identified, it is executed by the Celery worker, which houses all the deep learning models necessary for task execution. The outcome of the task is written to the Redis database for further retrieval by FastAPI.

3.2.2 Annotation App

The system is primarily centered around an annotation tool, which is presented as a **cross-platform desktop application**. This tool connects to both a backend and third-party services to provide additional functionality. The backend consists of an API and a database, which manage user data, tool settings, and annotations. This setup promotes portability and facilitates modifications.

Installation files for various platforms are available on the annotation tool's website. Additionally, there is a release server that offers automatic updates for the annotation tool.

At the core of the system lies a local image server, which is initiated as a child process when the annotation tool launches. It is compiled into an executable file for each supported operating system and operates without requiring additional support software to be installed. During the tool's build process, this executable is integrated into the resulting installation files, tailored to the selected target platform.

(1) Annotation Tool: The foundation of the system is constructed using the Electron framework, operating on the NodeJS runtime. This setup enables the development of cross-platform desktop applications, utilizing the Vite[2] tool for efficient building processes. Moreover, the complete implementation is crafted in TypeScript, employing the React library for user interface design. This architecture is further enhanced by the integration of various libraries including: **OpenSeadragon**[3]: Used as a high-resolution zoomable image viewer; **Annotorious**[4]: Functioning as an extension for the OpenSeadragon library, this tool facilitates image annotation through drawing, commenting, or labeling. It supports a wide range of plugins and offers a high degree of customizability.

(2) Local Image Server: The solution is developed in Java, primarily due to the requirement of the **Bioformats**[5] library, which is essential for reading and writing various life sciences image file formats. Additionally, an HTTP Server, crafted using the HttpServer library, facilitates communication with the application.

(3) Website: The website for the tool is developed using the Next.js framework, leveraging the React library and TypeScript for crafting a user interface. In addition to other libraries, the inclusion of the Stitches library simplifies the styling process.

(4) Release Server: The release server connects to GitHub to provide the latest versions and includes an interface for checking updates and downloading them. It's a modified version of the Hazel[6] update server for Electron apps.

The tool integrates with an AI API via HTTP requests, facilitating seamless communication. This architecture ensures modularity that makes it easy to modify or add additional methods, allowing independent development and offloading resource-intensive tasks to a robust server. **The API offers two types of methods: "instant" and "process".** Instant methods yield immediate results, ideal for operations requiring quick responses. On the other hand, process methods involve user-triggered actions, initiating

[2] electron-vite.org
[3] openseadragon.github.io
[4] annotorious.github.io
[5] openmicroscopy.org/bio-formats
[6] github.com/vercel/hazel

a sequence where necessary information is gathered, including image cropping. This data is then sent for analysis and queued for processing. Periodically, the tool checks for result availability, upon reception, the information undergoes parsing, storage, and display tailored to the specific process type. This communication style is selected to accommodate the longer processing time associated with the input of these methods.

3.3 User Interface

In the first step, wireframes were created to define the layout of various interface elements. These wireframes underwent consultation with domain experts, and after integrating their feedback, a high-fidelity prototype was created. Our design process embraced **rapid prototyping by directly implementing tool functionalities**, as our project timeline made it impractical to conduct separate prototype testing in Figma and subsequent implementation phases. The layout of this screen as well as other settings of the tool, which include changing the language, can be modified according to the user's preferences.

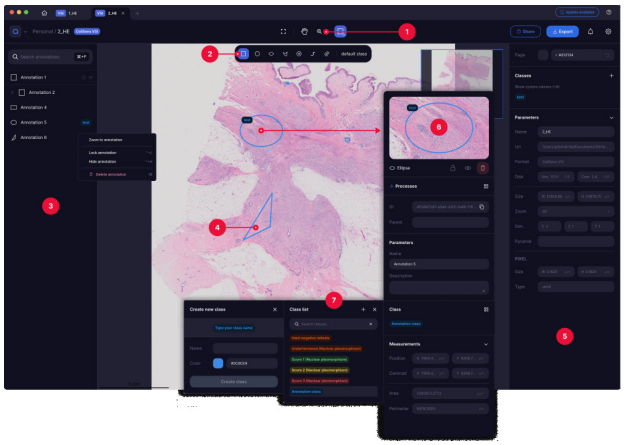

Figure 4: User Interface

Figure 4 shows **7 key parts and functionalities** of the annotation tool, the intent and focus of which we will discuss in the following sections:

(1) Toolbar: The Toolbar serves as the primary control hub for the tool, offering functions to manipulate images such as "Zoom to fit", a default "Hand" tool for navigation and annotation selection, and a "Zoom" tool for adjusting image magnification. The "Zoom" tool primarily utilizes the mouse wheel for control. Additionally, there's a tool to toggle annotation mode, facilitating seamless switching between browsing and annotation modes. Users can swiftly transition between annotation and navigation modes by holding down a key.

(2) Annotation Tools: When the annotation tool is selected from the toolbar, the interface switches to annotation mode, displaying all available tools along with the default class automatically assigned to newly created annotations. Depending on the selected tool, the cursor's appearance changes to reflect the tool icon. Keyboard shortcuts facilitate seamless tool switching.

(3) Annotation List: This panel primarily displays and allows searching of annotations, comprising a text input field for filtering annotations shown in a hierarchical tree structure. Each annotation includes basic information such as name, shape, and associated class. The tree structure also groups annotations based on visual hierarchy. Positioned on the left, it adheres to the principle of maintaining a familiar design for users.

(4) Annotation: Annotations appear as bounded shapes that become editable when clicked, enabling users to adjust the shape or position using handles. When hovered over, the border and shape colors reflect the selected class color for improved identification. The default border color is blue, chosen for its visibility in this image type.

(5) Image Properties: The right panel adapts dynamically based on selected functionality, annotation, or user state. Initially, it displays image information and workspace settings. Its placement on the right signifies it as an additional section linked to actions on the left side of the screen.

(6) Annotation Properties: Upon annotation creation or selection, the right bar transforms into an annotation detail panel, presenting information, parameters, previews, and additional functionalities. **Key parameters include the annotation name and description, facilitating communication between domain experts and the development team.** Automatic calculations such as position, size, or area are shown at the bottom of the panel. An annotation preview aids in identification and illustrates potential changes users can make within the panel.

(7) Annotation classes: The tool offers predefined classes for annotation categorization, with users able to create custom classes, each with a unique name and color. These user-defined classes are stored within the image settings for future reference and editing. Assigned classes are indicated on each annotation's top left corner, matched with corresponding colors for easy identification. When exporting annotations, class definitions are included, enabling the transfer of both annotations and their associated class information between projects. **This feature enhances utility, ensuring consistency and facilitating collaborative work across images.**

3.4 Annotation Methods

To streamline and enhance the annotation process of histological images, we have devised various approaches tailored to specific requirements and observations. These methods are categorized into two groups, delineated by user complexity and the degree of artificial intelligence assistance, to **manual** and **(semi)automated** which contains specialized subsection targeting the use case of **Nottingham Grading System**.

3.4.1 Manual Methods

Manual approaches can be defined as those that **do not use any form of AI assistance** in the process of creating the annotation. This means that the user is responsible for the overall result. In most cases, these forms of annotation must be performed before running the automated annotation method as a way of specifying the domain or input needed for the automated methods.

These manual methods include: **Rectangle annotation:** A basic shape to define boundaries or regions of interest (ROI), is most commonly used in defining regions for automated methods; **Circle annotation:** Circular annotation with the possibility to change the radius of the circle; **Ellipse annotation:** Similar to circular annotation with the possibility of changing two radii; **Polygon annotation:** Annotation with the possibility of adding more points, which results in a closed shape. It is used as a result of several automated annotation methods; **Free-hand annotation:** Freehand annotation that creates open shapes without points; **Point annotation:** An annotation point that indicates coordinates without the possibility of defining shape or size. Also used in defining the click position for automated methods.

3.4.2 (Semi)Automated Methods

Automated or semi-automated annotation methods (Fig. 5) can be defined as annotations where **only minimal user input is required** to specify the domain or interaction that is used as input for the AI models. Once the image and the specified input parameters have been analyzed, the result is returned in the form of a specified class (after classification), a modified annotation (as a polygon, for segmentation), or multiple annotations created (in the form of bounding boxes or polygons) with possible classification into multiple classes. These methods include:

(1) Nuclei Segmentation: NuClick [1] model is used for single-click cell segmentation. We acquired the model weights from the nuclick_torch[7] repository. The prediction outcome is a segmentation map which is subsequently adjusted using various postprocessing techniques aimed at enhancing the segmentation mask's quality. Postprocessing methods include removing objects smaller than a specified threshold, filling empty holes, and reconstruction. The refined segmentation mask is then converted into a polygon of points and transmitted to the annotation tool.

(2) Bbox Nuclei Segmentation: Nuclei segmentation from Bbox is a similar method to Nuclei Segmentation. It is a modified method where the user can get a more accurate annotation from the nuclei boundary by segmenting the nuclei from the selected Bbox.

(3) Segment Anything: SAM [11] is employed in the annotation tool for interactive segmentation of structures beyond the scope of the previously specified models. This model facilitates segmentation using bounding boxes and foreground/background clicks. Foreground clicks mark the desired segmentation area, while background clicks exclude it. The vit_b variant of the model is used for prediction, obtained from the segment-anything[8] repository. The predicted segmentation mask is subsequently adjusted using various postprocessing techniques identical to the NuClick model.

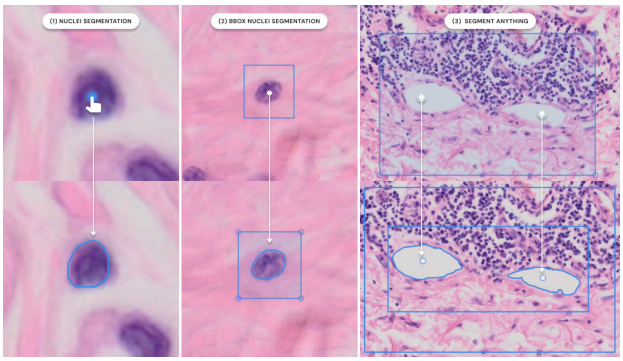

Figure 5: (Semi)Automated annotation methods

3.4.3 Use Case: Nottingham Grading System

Our objective was to develop methods that facilitate the scoring of the Nottingham grading system (Fig. 6). These methods have been incorporated into the automated annotation processes. These methods include:

(1) Mitosis Detection: Mitotic detection is employed to identify mitoses and hard-negative mitoses. The AI API utilizes a one-stage version comprising a YOLOv8 detector trained on the MIDOG++ dataset [2], see Table 1. The focus is solely on detecting mitoses and hard-negative mitoses, without evaluating the Nottingham Grading System's mitosis count criterion. The main reason behind this approach is rooted in the limitations of the developed annotation tool, which currently **cannot accurately reflect the**

[7] github.com/mostafajahanifar/nuclick_torch

[8] github.com/facebookresearch/segment-anything

real state of the score. This is due to the necessity of evaluating the score from an area equivalent to **10 High Power Fields (HPFs)** or approximately **2mm^2** [14]. We trained the model to detect hard-negative mitoses, aiming to provide domain experts with a definitive decision on whether a given instance is a mitosis or not. Its input is an ROI that delimits the area from which the analysis can give the user an annotated mitosis with an assigned class.

The most favorable outcomes were obtained with the medium variant of the pre-trained YOLOv8 model, developed by ultralytics[9], and employing FFTStain-Augmentation [20]. These results yielded an F1 score of 0.643, precision of 0.592, recall of 0.703, mAP50 of 0.664, and mAP of 0.476 on the test set.

Dataset	Mitotic figures	Hard-negative figures
train	17216	20944
val	862	961
test	2467	2916

Table 1: Number of mitotic and non-mitotic figures per set in FFT augmented MIDOG++ dataset

(2) Nuclear Pleomorphism: Another criterion for the Nottingham Grading System is Nuclear Pleomorphism. The EfficientNetB4 model aims to predict the nuclear atypia score based on images received by the AI API from the annotation tool. During image processing, the input is divided into patches, and a nuclear atypia score (1, 2, or 3) prediction is generated for each patch. The ultimate score is determined through a majority voting mechanism. This approach draws inspiration from the methodology outlined in the article [21]. MITOS-ATYPIA-2014 dataset [17] was used to train the model, see Table 2.

Dataset	Score 1	Score 2	Score 3
train	1941	13017	2150
val	88	2472	394
test	3078	4582	1747

Table 2: Number of patches per nuclear atypia score

The most favorable outcomes were obtained with the EfficientNetB4 model, pre-trained on the ImageNet dataset. These results yielded an F1 score of 0.349, accuracy of 0.511, precision of 0.480, recall of 0.382, and AUC of 0.547 on the test set. The results achieved are unsatisfactory, which forces us to look for improvements through annotations of our own data.

(3) Tubular Formation: We did not address this criterion because we could not find any public dataset.

[9]https://github.com/ultralytics/ultralytics

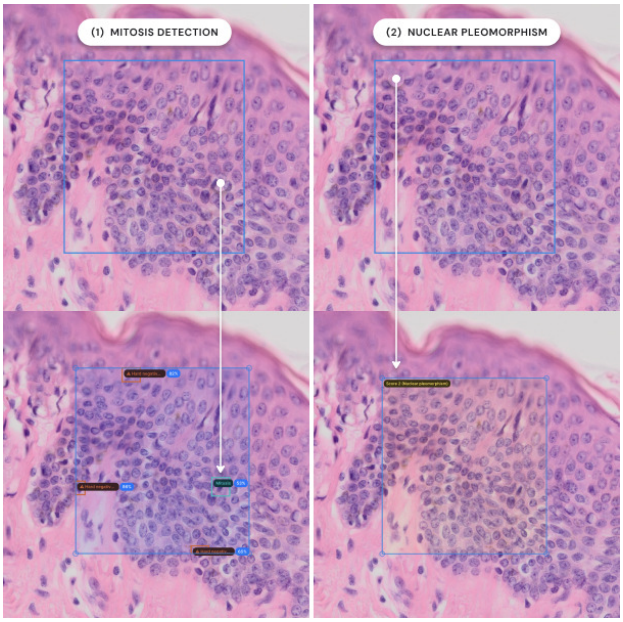

Figure 6: NGS Annotation methods

4 Results

Our preliminary user-testing with domain expert resulted in UI changes related to behaviour of the annotation tool based on user expectations and their feedback. Additionally, in near future we plan to conduct A/B testing of the AnnotAid and QuPath. The main goal of this user testing is to simulate real-word scenario of evaluation mitotic count criterion of Nottingham Grading System. We plan to compare efficiency of manual (in both tools) and semi-automated methods (in AnnotAid).

Based on the results obtained from developing models for evaluation individual criteria of the Nottingham Grading System, it is evident that there is a requirement for additional data and improvement of these models. **This can be accomplished through the utilization of the annotation tool we have developed.**

5 Conclusion and Future work

In our work, **our primary focus centered on developing an annotation tool** in close cooperation with domain experts. We conducted multiple user testing sessions where valuable feedback was incorporated into the functionality of the annotation tool. **Our research primarily revolves around the creation of deep learning methods to support the diagnosis of Nottingham Grading System criteria.**

Moving forward, we aim to **integrate active learning** into our annotation tool, prioritizing methods to improve model performance and user experience, alongside **enhancing model explainability**. Additionally, we plan to introduce **new annotation methods** and collaboration fea-

tures, informed by regular user tests to ensure ongoing refinement of the tool's functionality and usability, aligning with user expectations and preferences.

References

[1] Navid Alemi Koohbanani, Mostafa Jahanifar, Neda Zamani Tajadin, et al. Nuclick: A deep learning framework for interactive segmentation of microscopic images. *Medical Image Analysis*, 65:101771, 2020.

[2] Marc Aubreville, Frauke Wilm, Nikolas Stathonikos, et al. A comprehensive multi-domain dataset for mitotic figure detection. *Scientific Data*, 10(1):484, 7 2023.

[3] Sugata Banerji and Sushmita Mitra. Deep learning in histopathology: A review. *WIREs Data Mining and Knowledge Discovery*, 12(1):e1439, 2022.

[4] M. Jorge Cardoso, Wenqi Li, Richard Brown, et al. Monai: An open-source framework for deep learning in healthcare, 2022.

[5] Cytomine community. Introduction. `https://doc.cytomine.org/admin-guide/`, 2022. Accessed: 2022-12-09.

[6] Orbit docs authors. Tissue quantification. `https://www.orbit.bio/tissue-quantification/`, 2023. Accessed: 2023-05-22.

[7] QuPath docs authors. What is qupath? `https://qupath.readthedocs.io/en/stable/docs/intro/about.html`, 2022. Accessed: 2022-12-09.

[8] Christopher W. Elston and I. O. Ellis. pathological prognostic factors in breast cancer. i. the value of histological grade in breast cancer: experience from a large study with long-term follow-up. *Histopathology*, 19, 1991.

[9] Computational Pathology Group. Asap. `https://computationalpathologygroup.github.io/ASAP/`, 2018. Accessed: 2022-12-09.

[10] Mostafa Jahanifar, Adam Shephard, Neda Zamani Tajeddin, et al. Stain-robust mitotic figure detection for the mitosis domain generalization challenge, 2021.

[11] Alexander Kirillov, Eric Mintun, Nikhila Ravi, et al. Segment anything, 2023.

[12] Martin LindvaN, Alexander Sanner, Fredrik Petre, et al. Tissuewand, a rapid histopathology annotation tool. *Journal of Pathology Informatics*, 11(1):27, 2020.

[13] Sreeraj M. and Jestin Joy. A machine learning based framework for assisting pathologists in grading and counting of breast cancer cells. *ICT Express*, 7(4):440–444, 2021.

[14] Siddhartha Mantrala, Paula S Ginter, Aditya Mitkari, et al. Concordance in breast cancer grading by artificial intelligence on whole slide images compares with a multi-institutional cohort of breast pathologists. *Archives of pathology & laboratory medicine*, 146(11):1369–1377, 2022.

[15] Tojo Mathew, C.I. Johnpaul, B. Ajith, et al. A deep learning based classifier framework for automated nuclear atypia scoring of breast carcinoma. *Engineering Applications of Artificial Intelligence*, 120:105949, 2023.

[16] Caner Mercan, Maschenka Balkenhol, Roberto Salgado, et al. Deep learning for fully-automated nuclear pleomorphism scoring in breast cancer. *npj Breast Cancer*, 8(1):120, 11 2022.

[17] Daniel Racoceanu, Jessica Calvo, Elham Attieh, et al. Detection of mitosis and evaluation of nuclear atypia score in breast cancer histological images. 2014.

[18] Eren Tekin, Çisem Yazıcı, Huseyin Kusetogullari, et al. Tubule-u-net: a novel dataset and deep learning-based tubule segmentation framework in whole slide images of breast cancer. *Scientific Reports*, 13(1):128, 1 2023.

[19] Anjali Venugopal and Lekha S. Nair. Two-phase mitotic detection using deep learning techniques. In Milan Tuba, Shyam Akashe, and Amit Joshi, editors, *ICT Infrastructure and Computing*, pages 479–489, Singapore, 2023. Springer Nature Singapore.

[20] Xiyue Wang, Jun Zhang, Sen Yang, et al. A generalizable and robust deep learning algorithm for mitosis detection in multicenter breast histopathological images. *Medical Image Analysis*, 84:102703, 2023.

[21] Jun Xu, Chao Zhou, Bing Lang, and Qingshan Liu. *Deep Learning for Histopathological Image Analysis: Towards Computerized Diagnosis on Cancers*, pages 73–95. Springer International Publishing, Cham, 2017.