DOI: 10.1111/cgf.14812
Eurographics Conference on Visualization (EuroVis) 2023
D. Archambault, R. Bujack, and T. Schreck
(Guest Editors)

COMPUTER GRAPHICS *forum*
*Volume 42 (2023), Number 3*

# xOpat: eXplainable Open Pathology Analysis Tool

J. Horák[1], K. Furmanová[1], B. Kozlíková[1], T. Brázdil[1], P. Holub[2], M. Kačenga[1], M. Gallo[1], R. Nenutil[3], J. Byška[1,4], and V. Rusňák[2]

[1]Masaryk University, Faculty of Informatics, Brno, Czech Republic
[2]Masaryk University, Institute of Computer Science, Brno, Czech Republic
[3]Department of Pathology, Masaryk Memorial Cancer Institute, Brno, Czech Republic
[4]University of Bergen, Department of Informatics, Bergen, Norway

**Abstract**

*Histopathology research quickly evolves thanks to advances in whole slide imaging (WSI) and artificial intelligence (AI). However, existing WSI viewers are tailored either for clinical or research environments, but none suits both. This hinders the adoption of new methods and communication between the researchers and clinicians. The paper presents xOpat, an open-source, browser-based WSI viewer that addresses these problems. xOpat supports various data sources, such as tissue images, pathologists' annotations, or additional data produced by AI models. Furthermore, it provides efficient rendering of multiple data layers, their visual representations, and tools for annotating and presenting findings. Thanks to its modular, protocol-agnostic, and extensible architecture, xOpat can be easily integrated into different environments and thus helps to bridge the gap between research and clinical practice. To demonstrate the utility of xOpat, we present three case studies, one conducted with a developer of AI algorithms for image segmentation and two with a research pathologist.*

**CCS Concepts**
• ***Human-centered computing*** → ***Visualization systems and tools; Scientific visualization;***

## 1. Introduction

Whole slide imaging (WSI), also known as virtual microscopy, refers to a cost-effective method to digitize whole glass slides containing, for example, stained tissue samples. The technology emerged in the late 1990s and is considered a disruptive technology that enabled new diagnostic, educational, and research methods [PFP15] in digital pathology [GEMF13; HPS20]. WSI scanners produce gigapixel resolution scans (thereof referred to as *images*) that are typically stored in pyramidal formats [AAB*84]. The main tasks performed by pathologists on these images are their inspection and annotation, i.e., the pathologists search for and annotate anomalies and other regions of interest in a tissue (e.g., cancer cells) using specialized WSI viewers.

Recent advances in artificial intelligence (AI) methods for segmentation [HMP*22], feature detection, and classification [HMK*17] show promising ways to speed up and support pathologists in their work. However, the development of these techniques and their adoption into broader practice is hindered by limited options for collaboration between researchers and clinical practitioners. While this is a nontrivial problem, it is partially caused by a lack of easily configurable WSI viewers that could work with various data formats and would be suitable for both research and clinical environments. Developing new algorithms requires expert pathologists' feedback to validate the results. At the same time, clinical practitioners need opportunities to try new methods without complicated setups to integrate them into their workflows. The research-oriented viewers, such as QuPath [BLF*17] or Cytomine [MRS*16], provide many tools for analyzing the images and thus suit the exploratory use cases and prototyping of new workflows. However, their integration into existing infrastructure is often tedious or even impossible, as they typically require specific back-end services or databases. On the other hand, clinicians typically use viewers integrated into expensive commercial ecosystems such as Tribun [Hea] that are rigid with limited support for adding custom functionality (e.g., the latest AI algorithms). As a result, researchers and clinicians often use several WSI viewers, which complicates their collaboration when developing new algorithms or testing new methods in practice.

In this paper, we take the first steps to alleviate these issues. Our primary contribution is the design and open-source implementation (`muni.cz/go/xopat-repo`) of a modular and protocol-agnostic WSI viewer that we call **xOpat**. Due to its adaptability to various data inputs (including non-image data) provided by multiple servers, xOpat can be integrated into different environments without changing existing infrastructure. Additionally, custom plugins can extend xOpat's functionality, for example, by modifying the incoming data before displaying them in the viewer. This benefits both researchers, who can quickly test various workflows, and clinicians, who can utilize the new functionality as soon as it is ready and approved.

Furthermore, as xOpat runs in a web browser, it allows an easy exchange of information and findings between pathologists and researchers through a unified user interface. To further strengthen this exchange, we have implemented a plugin that supports creating narrated presentations of image analysis workflow and findings. Thus, the main contributions of xOpat are:

- Image server independence and protocol-agnostic architecture that enables easy integration into different environments (Section 4 and Section 6).
- Flexible handling and rendering of multilayer data (Section 5.2 and Section 6).
- Support for collaboration and communication via viewer session sharing and data-based storytelling (Section 5.5).

We demonstrate how xOpat can assist in developing and verifying AI algorithms and integrating them into practice in three case studies with an AI algorithm developer and a research pathologist.

## 2. Related Work

Existing WSI viewers offer similar core functionalities such as image rendering and support for manual annotations. Additionally, some offer features focusing on specific tasks or incorporate AI computational capabilities to support image analysis (e.g., segmentation and classification). Our goal is a general versatile viewer. Therefore, rather than focusing on specific feature sets of each viewer, we will structure this section according to key properties that set these viewers apart in terms of their general applicability and versatility. These include *compatibility and integrability* with existing environments, flexible *data handling* of multiple data sources (e.g., computational analysis results), and *collaborative capabilities* simplifying the information sharing among users. We provide a detailed comparison of individual viewers in Table 1 of the supplementary material.

*Compatibility and integrability:* Older desktop viewers usually worked with the data stored locally [Com]. With the availability of high-speed network connection, image viewers shifted to a client-server model with images stored on servers in the network. We identified three broad categories of server dependency. Commercial viewers such as Aperio ImageScope [Ape], HALO [Ind], or Sectra [Sec] are part of the vendor's ecosystem, also comprising the server. The open-source projects either have their own implementations (e.g., Minerva [HRM*20] or Cytomine [MRS*16]) or rely on OMERO [ABM*12], developed under the Open Microscopy Environment consortium [SGBP03]. Orbit [SSV20] offers an interface for integrating almost arbitrary image servers at the expense of implementing complex connectors and services such as authentication. We strive to offer a fully server-independent solution based on a protocol-agnostic approach that unifies data access via a single flexible interface and promotes communication scheme adjustments.

*Data handling:* The development and integration of AI methods for segmentation poses new demands on support for retrieving data from various sources (potentially in different formats) and the ability to map the data into layers having the appropriate visual representations. Most viewers that support automated analysis using AI algorithms only provide a fixed configuration of layers, possibly adjusting attributes such as color, hue, or thresholding [PAA*23; RL21]. A certain degree of flexibility is provided by, e.g., ImagePro [Med], QuPath [BLF*17], and Scope2Screen [JKW*22], which encode data into individual image channels and allow user customization. The flexibility of working with layers can also be implemented programmatically [SLN*22], but such an approach poses demands on the user. Support for truly arbitrary layering from multiple sources is, therefore, still minimal. In xOpat, we address this by promoting synchronous data access and versatile rendering options.

*Collaboration support:* In the context of collaborative image analysis, the key features are mainly the export and sharing of analysis sessions with other users. The analytical workflows are traditionally based on asynchronous collaboration, where only one user interacts with the system at a time. While case management systems allow concurrent access to the files, a session-sharing capability delivers more context. Typically, the session is shared as a JSON file or as a parametric URIs [RL21; HRM*20; Gle; PAA*23]. These contain specific tool configurations, view positions, or zoom levels, allowing users to work in a "you see what I see" manner. Synchronous collaboration mode where the content is shared and dynamically synchronized between multiple users is supported less often [RL21; MRS*16; JSH*13; PZD*19]. Some tools also support storytelling by storing annotated view positions with text descriptions [HRM*20; Med; JKW*22]. These capabilities improve sharing insights and efficient navigation compared to plain session sharing. The xOpat takes inspiration from existing approaches and addresses both session-sharing and storytelling features to facilitate user collaboration.

xOpat focuses on three key features that we consider to be crucial for a truly versatile viewer: a) independence from the chosen image server, data, and metadata format; b) efficient manipulation and mapping of heterogeneous data sources that are rendered as additional image layers; c) supporting collaborations between users. Although many existing viewers partially support these features, particularly in the area of collaboration, none of the existing tools address all of them comprehensively.

## 3. Requirement Analysis

xOpat was developed by an interdisciplinary research team composed of visualization researchers, clinical pathologists, and developers of AI algorithms for the analysis of histopathology data. The system was designed in an iterative process, during which we refined users' needs, shaped the design, and continuously collected feedback on the implemented features. Based on experiences and discussions with team members, we identified workflows and tasks of potential user groups of our system, which we subsequently analyzed to elicit a set of requirements for a flexible visual analysis system for digital pathology.

### 3.1. Users

We have identified three main user groups that could benefit from a system for the visual exploration of histopathology images. In the following, we discuss their typical objectives and workflows.

**Developers** design and implement new AI algorithms for image segmentation and classification. Their main objective when visually examining the histopathology data is the evaluation of the algorithm

performance with respect to ground truth provided by pathologists as annotations. They compare all the data to identify differences between algorithmic and manual classifications and subsequently analyze these to discover possible causes of the differences to improve the algorithm. This often involves switching between different images, which is cumbersome and slow (due to the data size) and requires high mental effort from the developers.

**Research pathologists** evaluate new and existing digital pathology algorithms and approaches in terms of their applicability and suitability for various clinical scenarios. Thus, in their workflows, they need to compare outputs of multiple AI algorithms or analyze the performance and outcomes of a given algorithm across different patients and patient cohorts (e.g., patients with different comorbidities). Furthermore, research pathologists provide the domain knowledge for the developers—they use WSI viewers for manual image annotation to provide ground truth and validate the correctness of the algorithm results. As such, they heavily rely on mutual communication with the developers.

**Clinical pathologists** use the WSI viewers in clinical practice. Their main objective is an efficient examination of samples and detection of tissue morbidity or other anomalies. They need to be sure to notice everything noteworthy. To this end, they can be assisted by automated analysis methods. In the case of long-term treatment, they often collect and compare information about various changes in the tissue. It is also common for them to share and consult their observations and diagnoses with other colleagues.

### 3.2. Requirements

In this paper, we are making the first step towards a generic tool supporting all three user types. We primarily focus on the developers and research pathologists, but we kept the needs of clinical pathologists in mind when designing our system. Based on the discussions within our interdisciplinary research team and the analysis of typical user workflows, we have identified the following requirements for a flexible visual analysis system for digital pathology.

**R1: Extensibility and data source independence.** The tight coupling of visual analysis systems with specific data processing pipelines and protocols limits the adaptation of many existing histopathological solutions to continuously evolving algorithms and data sources. Moreover, adapting their existing frameworks to novel analytical solutions is too costly for many laboratories and hospitals. A flexible analytical system should thus be modular, easily extensible, and independent of specific data processing pipelines.

**R2: Multi-layer data handling.** Histopathology images are often examined with additional layers of metadata (e.g., annotations or outputs from AI algorithms). The users need to be able to combine, reorder, and compare data presented in multiple layers over the background (e.g., tissue) images. Furthermore, they should be able to configure various settings for the layers (e.g., opacity or coloring).

**R3: Navigation within images.** Minimized image overviews are required to indicate which part of the high-resolution image is displayed in the main viewport. Furthermore, intuitive and efficient interaction with the image must ensure navigation to the potential points of interest (e.g., suspected tumor lesion).

**R4: Efficient data handling.** The size of the data produced by WSI also poses challenges for real-time exploratory analysis. Emphasis should therefore be put on rendering performance and low latency of the interactions within the system.

**R5: Annotation features.** Users often need to create many annotations to track their analysis and communicate their observations. The annotation process should thus be simple, without unnecessary repetitive settings, but at the same time, it should allow the creation of sophisticated annotations if needed.

**R6: Collaboration and data sharing.** To facilitate the exchange of information between users, e.g., algorithm developers and pathologists, the system should support easy exporting and sharing of meta-data related to the individual images, including annotations, points of interest, and their rendering configurations.

## 4. xOpat Architecture

Our primary goal is to provide the target users with a flexible tool that can be easily integrated into their routine workflows. As a result, xOpat is a modular viewer that anticipates existing connectors/components: data providers, and configuration management systems, as shown in Figure 1. We also provide an example implementation of a data provider and a management system to deliver an out-of-the-box working solution.

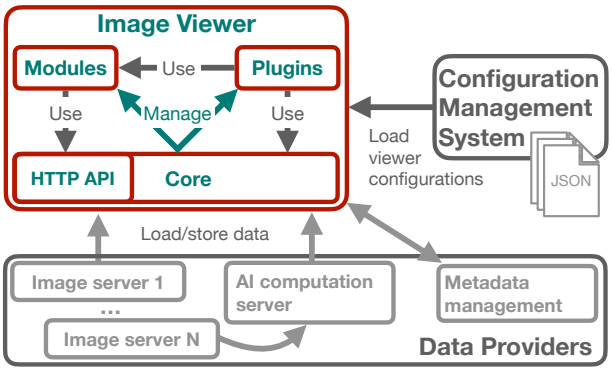


**Figure 1:** *xOpat environment architecture overview. The data providers (bottom) provide the xOpat viewer (top left) with the requested data and manage metadata. The configuration management system (top right) creates desired xOpat session configuration. The diagram omits the possibility of direct access via generated URLs and file exports.*

**xOpat Viewer** is a service-independent web application enabling easy integration into existing environments. xOpat viewer ensures the rendering of the images and other metadata, such as AI outputs, provides the primary user interface, and handles user interaction capabilities such as annotating. Its CORE contains essential features, such as the viewing logic, user interface subsystem, and API for managing plugins, modules, and data fetching. MODULES are similar to libraries in operating systems. They allow flexible inclusion of new features (e.g., WebGL rendering) and can form dependency trees. Modules should not have an implicit impact on the system. PLUGINS are similar to applications. They are instantiated and managed automatically, and their behavior is more restricted. They bring

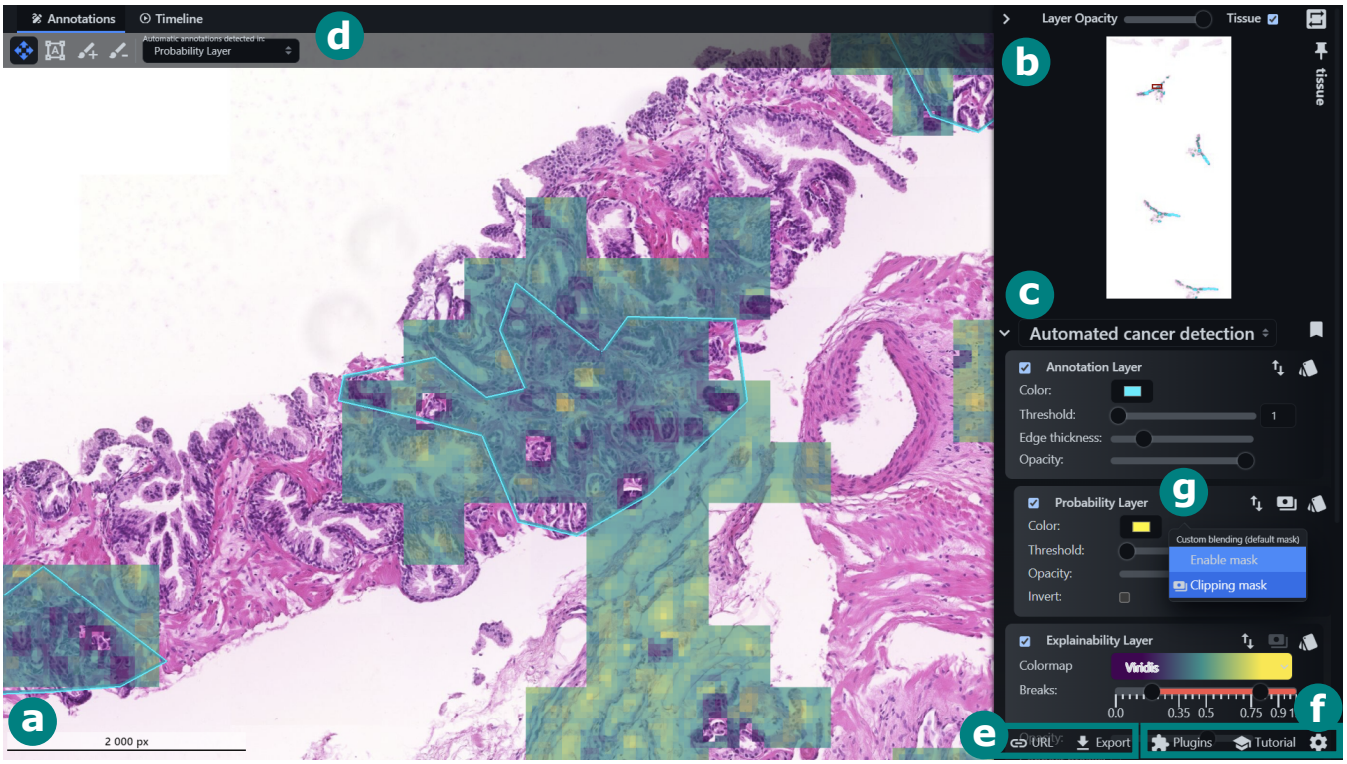


**Figure 2:** *The default configuration of xOpat viewer:* ***(a)*** *main canvas with a scale bar,* ***(b)*** *minimap with highlighted part of the image displayed in the main canvas,* ***(c)*** *visualization configuration options for individual layers,* ***(d)*** *toolbar with available interaction modes,* ***(e)*** *sharing options—URL and export to file* ***(f)*** *on-demand features—plugin manager, tutorials, and settings. The visualization shows a rasterized annotation of cancer provided by a pathologist (blue edges). Further, the explainability layer indicates the positive (yellow) or negative (purple) contribution to the prediction of AI algorithm results loaded as the probability layer. The probability layer is not visible since it is used as a clipping mask* ***(g)*** *on the explainability layer, so we render explainability only where the probability is higher than its current threshold.*

new functionality, such as annotation capabilities or support for neural network inspection.

**Data Providers** represent various services (e.g., image servers, WSI storages, computational servers, metadata providers) providing data to the image viewer. xOpat uses a protocol-agnostic approach that abstracts the communication with these third-party services without the need to re-organize data and metadata structures. In principle, the viewer sends HTTP requests conforming to existing data providers' APIs and processes their responses and vice versa. Our approach allows using arbitrary data fetching protocols (e.g., DeepZoom [Mic]) without requiring to change the image viewer implementation (**R1**) or limiting what data in what format is sent in the response as long as an appropriate handler is implemented. Data Providers also take care of arbitrary metadata handling (e.g., user data or storage of viewer configurations; see Section 5.4).

**Configuration Management System** is an optional component that provides UI-friendly access to the viewer. Whether implemented as a stand-alone service (our case) or as a part of the existing information system, it only needs to know how to construct xOpat configuration. It provides an entry point through which the users can initiate their work with the image viewer, and it can use Data Providers to open saved sessions or prepare more sophisticated session configurations.

## 5. xOpat Viewer

The xOpat Viewer consists of *Main Canvas* for rendering tissue images with various overlaid metadata (**R2**), minimap serving for navigation (**R3**), and controls allowing users to manage the appearance of rendered images and data (see Figure 2). The additional functionality, e.g., support for creating annotations (**R5**), is then provided via various plugins (Figure 2f) that can be loaded ad-hoc.

### 5.1. Main Canvas and Minimap

The *Main Canvas* (see Figure 2a) spans most of the viewer to provide users with enough resolution when exploring the whole slide images. Except for basic keyboard and mouse navigation (pan and zoom), the canvas inherits other controls from the OpenSeadragon [Cod] library, which we used for its implementation, e.g., canvas rotation. However, we enhanced the zooming with automatic speed adjustment for a better experience.

Following the "overview+detail" design concept [CKB09], the canvas is accompanied by a *Minimap* (Figure 2b) showing the position of the viewport and allowing quick zooming and navigation between distant areas within the large images (**R3**). The minimap can also be detached from the right panel, where it is positioned by default, and pinned on the canvas so that it always remains visible.

### 5.2. Visual Representation of Data Layers

When rendering the images, we distinguish between *image layers* and *data layers*. The image layers form the background (i.e., stained tissue slices) and allow only opacity adjustments. Multiple slices can be rendered simultaneously on top of each other or switched between one slice at a time. On the other hand, the data layers represent arbitrary data, either pre-computed or generated on-the-fly (e.g., the output of neural networks or manual annotations). Unlike image layers, data layers provide more rendering capabilities, such as configurable blending modes or customized visual data representations.

The data layers encode the values as overlays using color and opacity. However, histopathological images are already visually complex data, which limits the possible visual encodings of additional overlaid metadata. In xOpat, there are five built-in types (Figure 3a—e) of visual representations:

- *identity* showing the original data;
- *heat map* encoding values (e.g., network predictions) to pixel opacity of a selected color;
- *bi-polar heat map*, which is an extension of the heat map for diverging variables, encoding values also to pixel opacity but using a different color for each direction (suitable, for example, to depict occlusion-based AI explainability map describing positive or negative contributions to the network predictions);
- *color map* allowing to encode both discrete or continuous data with several colors using cyclic, sequential, or diverging color map sets (most generic option suitable for most data, but also the most occluding);
- *edges* rendering line borders at a given threshold (suitable for data containing large areas of close or constant values, such as rasterized annotations because, unlike other representations, edges do not occlude the tissue).

Furthermore, the data layers can also be used as clipping masks for layers below them, so only areas where the clipping layer data is above a given threshold are rendered (e.g., in Figure 2g, probability layer is used as the clipping mask for explainability layer). Finally, xOpat integrates a GLSL code editor allowing users to script their own data layer shaders in real-time (Figure 3f).

The configuration options of individual layers are placed below the minimap in collapsible panels, stacked according to the order of layers (Figure 2c). The user can customize order, visibility, and other visualization-dependent parameters of individual superimposed layers (**R2**). While the superimposition of data layers allows easy in-place comparison of data (e.g., manual and automatic annotations), it is not always suitable due to occlusions and the high visual complexity of the data. To address this problem, it is possible to split the application into two synchronized windows. In such cases, both views move synchronously. However, the data can be visualized using different representations. Moreover, different plugins modifying the data can be activated in each window which enables flexible side-by-side comparisons of regions and available data. This is useful, for example, when comparing the outputs of two different AI models.

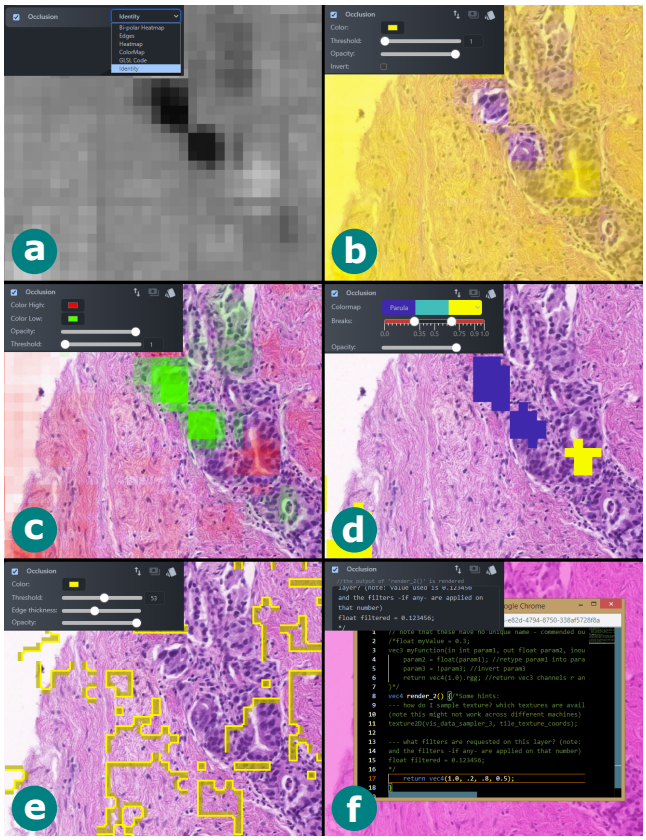


**Figure 3:** *Build-in data layer shaders:* ***(a)*** *identity,* ***(b)*** *heat map,* ***(c)*** *bi-polar heat map,* ***(d)*** *color map,* ***(e)*** *edges,* ***(f)*** *interactive console allowing to create custom shaders.*

### 5.3. Annotations and Presets

One of the main requirements was support for an effective way of image annotations (**R5**), avoiding cumbersome repetitive actions, such as selecting the annotation category. Since annotations are not always needed, we implemented the annotation subsystem as a collection of modules and a plugin. Currently, we support several annotation primitives: *rectangle* and *ellipse* for marking approximate regions of interest, *polygon* and *polyline* for the precise region selection, *ruler* for distance measurements, and *text* and *point* for other use cases (such as storytelling described in Section 5.5).

To speed up the annotation process, the users can create *Annotation presets*. They represent a binding between an annotation object and a user-defined collection of properties, i.e., visual features (color or shape) and configurable metadata (comments). As such, the presets can be quickly used to create specific annotation types. A unique feature implemented in our solution is that two annotation presets can be used simultaneously. Each mouse button can be assigned a different preset to annotate with, which is very useful for binary manual classification tasks, such as the decision of false- or true-positive outputs of the AI pipeline. The annotation presets can also be stored and shared with other users.

By default, annotations have three modes of operation. The users can switch between these modes (and adjust their respective settings) using keys or the toolbar menu (Figure 2d), which appears on demand when the annotation plugin is activated.

- **automatic mode** enables creating annotations by double-clicking on the canvas—the system will automatically outline the continuous area of a chosen layer spanning from the mouse cursor;
- **outline mode** (Alt key) enables to manually create selected primitives—for example, polygonal shapes by placing or drawing individual vertices using the mouse;
- **area mode** (Shift key) allows modifications of annotations by free-hand coverage of a filled area using a paint tool of a given radius. The filled area is then automatically converted to a polygon primitive. The user can adjust the tool radius using a mouse scroll. Shift+Alt modifier can be used to switch to area removal instead.

### 5.4. Configurability and Sharing

The appearance and functionality of the individual data layer controls (see Figure 4), as well as additional advanced features (e.g., gamma correction used when rendering), can be easily customized via a configuration file. Such flexibility allows configuration for various use cases depending on the user's needs and goals. The configuration (including plugins) is stored in JSON format and can be shared (**R6**), either as an exported HTML file or as a URL (Figure 2e). When the URL is loaded, the image viewer is initialized at the exact state. This includes the viewport settings (i.e., position and zoom level), so the users can easily share what they see with others.

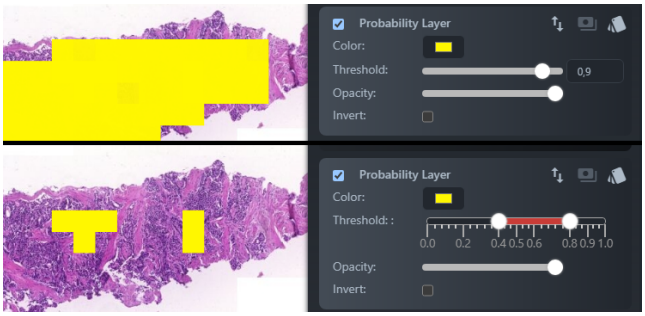


**Figure 4:** *Example of configurable controls used to specify filtering of the probability layer. Changing the JSON configuration allows quick switching from simple thresholding using one value (top) to range thresholding (bottom). The range slider is also further configurable (e.g., axis labels, number of ranges). This way, different visualization tasks can be supported.*

### 5.5. Storytelling

To increase the ability to share findings (**R6**), xOpat also features the *Presenter* plugin, which provides data storytelling support. It allows users to record the current state of the viewer, including all settings for the data layers as a keyframe. These keyframes are then automatically concatenated into data stories while allowing the user to parametrize the length and animation style of the transition between them. The UI for setting the parameters can be found in the toolbar menu (Figure 2d). The created stories can then be shared and replayed by others (see an exemplary story demonstrating Case Study 1 at `muni.cz/go/xopat-story`). The Presenter plugin can be used not only for presenting the case by clinicians, but also by researchers and developers for algorithm or performance evaluation; or by teachers in education.

### 5.6. Neural Network Inspection

*Neural network inspector* plugin further demonstrates the xOpat flexibility. This plugin with a python-based backend service provides visualization for explainability of AI algorithms (such as saliency-based image segmentation [Hru22]) used in the WSI analysis. The system allows non-trivial, interactive, and user-friendly inspection of the network architecture, see Figure 5. The plugin is used when identifying errors in the AI models during their development. In the future, it will also support adopting these models into clinical practice, as providing the users with a visual explanation of the network decision process will help to foster the trust of users in these AI systems and, more importantly, allow them to make informed decisions based on them.

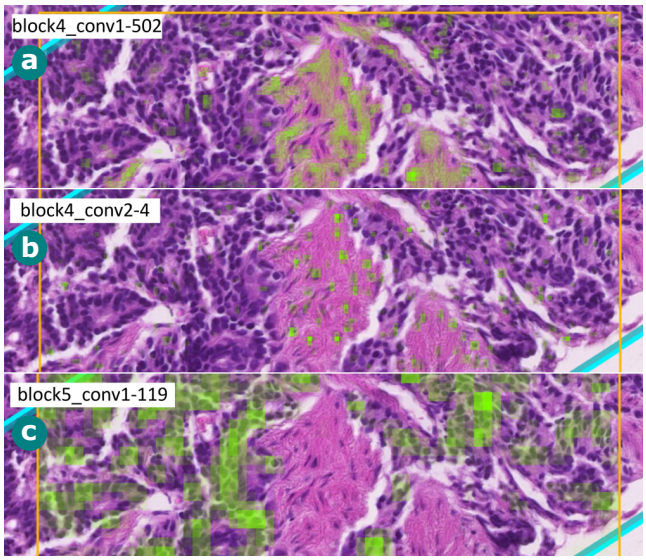


**Figure 5:** *Neural network inspector plugin. The image shows feature maps from different network convolution layers responding to:* ***(a)*** *stroma,* ***(b)*** *individual non-grouped nuclei, and* ***(c)*** *areas of grouped nuclei [Hru22].*

### 5.7. User Guidance

Our design decisions emphasize user guidance and the show-on-demand principle. For example, the system provides tutorials (Figure 2f) implemented using a modified version of EnjoyHint library [XB ]. An iterative example with all currently available tutorials can be found here: `muni.cz/go/xopat-tutorial`. To further improve user experience, pop-up dialogues are actionable. They not only provide hints or instructions but also offer interactive elements supporting users' actions. The dialogues may contain menus, highlight UI parts, or trigger actions.

## 6. Implementation Details

Our tool is open-source and available under MIT License at muni.cz/go/xopat-repo, together with documentation of provided features for developers. In addition to a service-agnostic viewer, we also supply exemplary services—*Data Providers*—that can be used alongside the viewer or replaced by custom implementations if needed. Ensuring efficient fetching and rendering of multiple image and data layers (**R4**) is necessary for a pleasing user experience and fast tissue inspection. Thus, we designed our rendering abilities and data exchange services to address performance issues.

### 6.1. Viewer

We are unaware of any existing WSI viewer that would provide a similar amount of visualization flexibility, i.e., a simple interface for configuring visualizations over arbitrary raster data, both file- and channel-wise. Existing WebGL libraries either implement viewing capabilities (handled by OpenSeadragon) or provide rendering interfaces that are neither suitable for requested visualization needs nor do they allow the amount of optimization we can do directly with WebGL as we use a lot of textures but have no explicit geometry or scene system. Our WebGL visualization module is designed both for performance and usability.

The viewer defines so-called *visualization goals* (see Figure 2c) representing a user's intention, i.e., a particular user-defined configuration of the individual data layers. Each visualization goal is defined by a set of so-called *layer shaders*, one for each data layer. The layer shaders specify which data to fetch from the server, the controls available in the UI, and how to render the data based on these controls.

Only GLSL code generating the visual output (pixel processing) must be provided to program a new layer shader. The module provides all low-level features (e.g., data access or WebGL version independence) for the final shader. The shader input values (i.e., user controls) can be syntactically described in JSON-like structure and used directly in the GLSL code. The module provides built-in support for blending modes, pre-filtering (e.g., gamma correction), attaching default controls (e.g., opacity setting), caching, and interchangeability of UI controls. Such a shader can then be instantly used and parametrized in the UI configurator.

The multiple-layer data can be fetched within a single image request by concatenating them into different color channels. While this approach is the most straightforward, it requires specific storage for the source images, severely limiting flexibility. OpenSeadragon [Cod], by default, allows fetching the individual images separately via multiple asynchronous requests. To further optimize, we decided to promote synchronous data fetching. We enhanced OpenSeadragon protocol API so that protocols can have full control over the data flow, allowing the use of any image server (even those unable to process image array queries) and any data. We have also extended tile caching control support for the protocols. We decided to leverage synchronous requests due to the following reasons:

- **Reduced memory:** Since all the data for a single tile is available at once, we can combine them and cache the resulting tile instead of storing multiple tiles.
- **Faster rendering:** By combining the tiles from multiple sources into one, we can also improve rendering performance by setting up the OpenGL pipeline once and using it repeatedly for all tiles without the context switching.
- **Fewer HTTP requests and better scaling**: Each tile *position* is requested, converted, decompressed, and processed only once without solving the synchronization on the client side. The number of HTTP requests is lower since their amount does not grow with the number of data layers. We include the performance evaluation of this aspect in the supplementary material.
- **More control:** Thanks to synchronous data access, advanced features such as custom blending and clipping of data layers are possible. Users can also define and change the parameters for this blending in real-time.

By default, the rendering engine can receive the data as a single concatenated image or an array of images. Optionally, one can add support for other formats by extending proper interfaces.

The downside of synchronous requests is that an *Image Data Provider* (i.e., an image server) must either support image array requests or store the data in a single image via channel concatenation. Therefore, when such a provider is integrated, one has to put more effort into the process to achieve optimal performance.

**Portable Request/Response Schemes**. The viewer also aims to eliminate the dependency on particular metadata and service providers by defining a scheme translation for metadata within the system. Such metadata is then propagated through requests. Plugins are responsible for offering adjustable API schemes (which is the case for the annotation plugin). This is a somewhat naïve approach that should be refined in the future, but it allows modifying the service communication through dedicated translations.

### 6.2. Exemplary Data Providers

Since no suitable high-resolution image server could work with image arrays, we implemented our own. We were able to design a solution that scales well for up to tens of independent image sources and allows us to employ viewer-efficient, multi-layered visualization, maintaining a highly responsive user interface. To facilitate synchronous data fetching, we extended the DeepZoom [Mic] protocol. The protocol can fetch data both synchronously and asynchronously. The synchronous requests return data concatenated in a single image or tiles as a *zip* file. The protocol is part of the viewer's default protocol set. Integration of a custom image server can happen at different levels:

I. Use our IIPImage Server extension. It is a fast server, but image data must be stored as a pyramidal JPEG2000 or TIFF.
II. Use a custom image server implementation that would handle image array requests capable of fetching images from multiple files via our Extended DeepZoom protocol (which requires modifications of the existing image server).
III. Use an image server capable of sending the data in an arbitrary format synchronously (better performance) or asynchronously. Extend xOpat with the protocol of choice defining how the data is stored and loaded to the GPU—if a loader is unavailable for the given data type. For example, it is possible to reuse a single-

tile protocol and extend a *multiplexing* capability for it; this is provided for all available protocols.

IV. Do not support image arrays and fall back to either asynchronous requests or channel concatenation. The downside of asynchronous access is the performance impact, which can be solved partially by data channel concatenation—not very flexible since the data must be stored in one file across channels instead of reading from multiple files.

## 7. Case Studies

In this section, we present three case studies demonstrating the utility of xOpat for two user groups—developers of AI-powered image analysis algorithms and research pathologists. The datasets were provided by the Department of Pathology of Masaryk Memorial Cancer Institute. All patients provided written informed consent to use their leftover material for research. The datasets are not publicly available outside of our live tool demo.

### 7.1. Case 1: Validation of the U-net Classification Model

In the first case study, we present an investigation of the results of the U-net model [RFB15] for detecting cancer lesions and identifying possible problems within the model by its developer. The study was performed by a junior developer who uses xOpat daily to investigate the results of his algorithms and presents his typical workflow. In addition to the developer, the manager of the AI development team was also present during the case study demonstration and provided further feedback.

The dataset with the network results contained: the tissue image, manual annotations of cancerous regions by a pathologist, and the output of the evaluated algorithm represented by the probability and explainability data. At this point, the researcher utilized the configurability of xOpath and mapped the original heterogeneous data to individual data layers without the need to adjust the data itself. The data is all displayed in Figure 6 in separate layers with different visual representations: annotations as region outlines, probability as a yellow-toned heat map, and explainability as a bi-polar heat map.

Note that neural networks split the images into smaller chunks that are different from the tiles displayed by the viewer. To differentiate these chunks from tiles, we use the term *patches*. In the probability layer, each rectangular patch indicates the probability that the region within this patch contains cancer as identified by the network. The explainability layer is obtained via occlusion sensitivity analysis [ZF14] and indicates which parts of the image contribute to AI decision and how (i.e., positively, indicating that the region contains cancer, or negatively).

After selecting the dataset, the developer checked the layers with respect to the position of the original image. He did this first for the annotation layer to check if the manual annotations were correctly aligned with the image. Then he switched off the annotation layer and inspected the probability layer. The developer noted that this procedure has previously led to the discovery of cases where the network generated high probability output for regions outside of imaged tissue which was caused by an error in the initial network setup with misaligned data. Besides the positioning, the developer also quickly checked the general shape of the network output and looked for patterns since results exhibiting regular patterns or blurred regions also indicate errors in the model setup, such as overlapping input patches. A live example of network output with an erroneous setup can be found at muni.cz/go/xopat-ex1.

In this case, the initial inspection of the probability layer did not reveal anything unexpected, so the developer proceeded with a more detailed examination. He re-added the annotation layer and checked if the network's output corresponded with the pathologist's annotations. In most annotated regions, the network also predicted a high probability of cancer. However, there is one region where the network failed to detect cancer (Figure 6, right).

To further inspect this result, the developer added the explainability layer (Figure 6). Here it was evident that some parts of the cancerous region were incorrectly identified as negative (red) while others were actually marked correctly as positive (green). This demonstrates that additional information from the explainability layer is useful in resolving why and on what parts of the tissue the prediction failed.

The developer concluded that with this, the first iteration of the model result examination was finished. At this point, he would utilize xOpat's URL or file sharing capabilities and communicate his findings with the development team manager to decide whether the findings were caused by some technical issue (e.g., misaligned data) or whether the network model should be further inspected and adjusted. The manager confirmed that further refinement would be necessary, followed by another iteration of the result inspection.

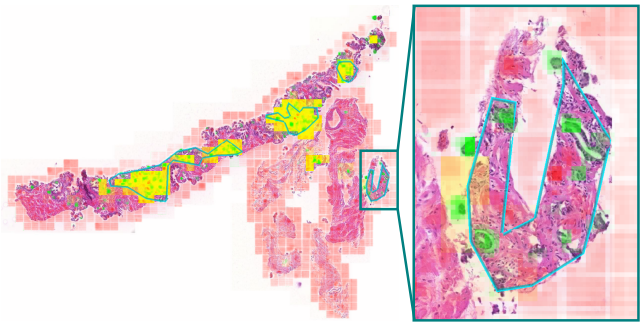

**Figure 6:** *Results of neural network model composed of three layers: pathologist's annotations (blue outline), probability (yellow), and explainability (red-green). Image on the right shows the portion of the tissue where network failed to identify cancer. The interactive example is at muni.cz/go/xopat-ex2).*

### 7.2. Case 2: Evaluation of U-net Classification Results

The second use case was conducted by a pathologist collaborating on developing novel AI algorithms. In this case, the task was to evaluate the performance of a previously developed U-net model trained to identify epithelium in hematoxylin-eosin-stained tissue sections. This network was trained on breast and colorectal cancer tissue. The investigation aimed to identify whether and to which extent it can be applied to prostate tissue. Therefore, we set up an image server that streamed the original data and configured xOpat to overlay the prostate core needle images with the neural network's

output represented as a separate segmentation layer, classifying image pixels as epithelial. The pathologist investigated the epithelial segmentation and marked them as either true or false positives (Figure 7). He used two annotation presets (one for true and one for false segmentations) mapped to the left and right mouse buttons. Using the xOpat annotation plugin, he could quickly and easily annotate the network results without switching between different annotation categories. In this case, there was a large number of false positives. According to the pathologist, the reason was that the prostatic stroma contained plump fibromuscular cells, which were not represented in the training dataset. The pathologist saved the annotations to share them directly with the developers, who could use them to further refine or retrain the model. Based on his observations, the developers concluded that the model must be retrained for the prostate tissue.

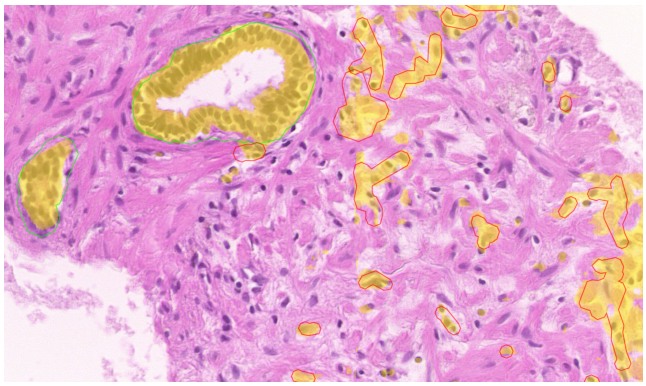

**Figure 7:** *Presentation of U-net segmentations over hematoxylin-eosin stained prostate tissue sample. Pathologist annotations further classified the epithelium segmentations (yellow) as true positive (green) and false positive (red). The false positivity appears in plump fibromuscular stromal cells.*

### 7.3. Case 3: Evaluation of Tissue Patterns

The third use case was conducted by the same pathologist. The dataset consisted of a similar set of layers as in Case 1. The task was to evaluate *i)* whether the regions contributing to the classification of image patches (either cancerous or noncancerous) can be assigned to some previously estimated tissue patterns, *ii)* and the relative frequency of such patterns in different cancer grades. Both tasks usually require error-prone and time-consuming pixel-level data exploration. However, the pathologist used xOpat's ability to load custom data and added a layer that contained information about unambiguous regions that positively or negatively contributed to the classification. The custom layer was precomputed from the explainability layer, which was regularly sampled. The sampling points were placed on a square grid of side length 280px. Then, if the absolute difference between the mean positive and negative explainability scores in the $15\times15$px square neighborhood of the sampled point was greater than 70, the point and its neighborhood were marked as unambiguous (Figure 8). Thanks to this additional sampling layer, the pathologist could quickly locate regions with unambiguous classification and evaluate the whole image with reasonable effort in a reasonable time.

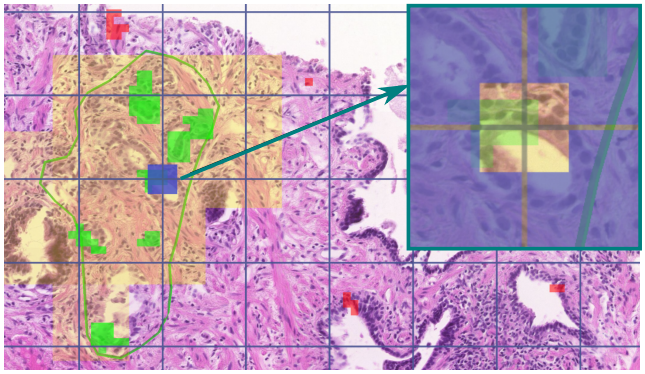

**Figure 8:** *Hematoxylin-eosin stained prostate cancer tissue image with additional data layers: pathologist's annotations (green outline), probability (yellow), explainability (red and green), and sampling for unambiguous regions along with sampling grid (blue). The blue square indicates a region with unambiguous positive contributions to the classification (shown in the zoomed cutout with inverted sample layer rendering).*

## 8. Discussion

Further, we discuss user feedback collected during the case studies, lessons learned during the implementation, and future work. We have also conducted a performance evaluation showing the benefits of our approach, which is discussed in the supplementary material.

### 8.1. User Feedback

During the case studies and subsequent discussions with the users, we also collected verbal feedback on xOpat's usability and usefulness and its shortcomings addressable in future research and development.

Both the pathologist and the developers noted fast and flexible data layer capabilities as the cutting-edge feature of xOpat, not available in other tools. The developers stated that before xOpat, they used in-house scripts to generate output for visual inspection of network outputs. Due to the slowness of this approach, they typically generated and visually inspected only small regions of the slides and thus often missed critical parts.

The scripting-based approach also limited communication and sharing of the findings among the development team members. Notably, it took a lot of work to localize the errors in the context of the entire image based on small cut-outs generated with local and often non-reproducible settings. xOpat allows the developers to inspect data layers over entire images quickly. The annotation and sharing features support consistent communication within the research and development teams. The team manager added that using xOpat for sharing findings also eliminates the cherry-picking of results.

The pathologist also mentioned annotation as one of the prominent features of xOpat, appreciating the option to create the annotation presets and add the annotations in a fast manner. However, he also noted missing options for working with multiple hierarchical annotation categories and category adjustment. While xOpat enables adding annotations of multiple categories, hierarchical categoriza-

tion or changing the annotation category once added will require additional development.

Another limitation of xOpat pointed out by the pathologist was the missing tracker of already viewed parts of the slide. He stated that this is a crucial feature, particularly for clinical usage, where the pathologist needs to ensure that he checked the entire slide and did not overlook anything. Such a tracker should collect information about which parts were already viewed and at which resolution.

Finally, both the developers and the pathologist noted that the tool needs a better interface for loading the data, which is crucial for future usage and is already a work in progress.

### 8.2. Lessons Learned

In pathology research, it is unfortunately still common to present a few cut-out patches with results of the AI algorithms without broader context, which may be misleading (not unlike quoting a person without context). WSI viewers usually do not account for the need to communicate the research results through their interface nor support a visualization means to do so. With xOpat, we enable sharing the results via interactive links showing the detailed patches and the whole slide image. We hope these interactive links will complement static images in their presentations or research papers, providing the necessary context to evaluate the new algorithms better and thus increase trust in them.

Furthermore, the data size and variety require the ability of a system to adjust flexibly rather than having an all-in-one solution. We designed a context-free viewer that delegates the data management to an external system and provides many ways to adjust or extend the data fetching process (see Advanced Data API that we added to OpenSeadragon version 3.2.0). Its RESTful API also enabled *reproducibility*, a crucial property for sharing and testing.

Concerning the UI, we noticed flaws in most annotation environments in *spatial* and *conceptual separation* and focused on minimizing the required steps when annotating the images. To improve intuitiveness, we implemented tutorials and action-based notifications that help novice users learn to use the system and perform actions directly in system messages.

Architecture-wise, our iterative development resulted in a design based on three abstraction components: *the core* (management utilities and crucial components), *modules* (or libraries, components with direct impact), and *plugins* (components with implicit impact). We believe that other plugin-supporting viewers should consider having at least these three abstraction components as well.

### 8.3. Future Work

In our future research, we plan to include missing features pointed out by the users in the evaluation—i.e., implement data loading UI and user progress tracking. Based on the evaluation, we have already included a simple tracker highlighting the regions explored on the minimap (see the interactive examples). But we would also like to explore the possibilities of the storytelling feature for automatically tracking the users' actions for provenance purposes and to uphold reproducible research. We also plan to further improve annotation sharing and add full-scale support for vector data rendering, currently supported only through annotations functionality. Another possible area for future work regarding sharing and collaboration is the support of live multi-user collaboration sessions. However, this requires further research regarding technological setup and user interactions within such a setting.

We are also working on the **Playground Plugin** that will enable the interactive execution of custom algorithms on a server (similarly to computational notebooks), show real-time results, and store and further process them. We already have a working demo allowing us to run real-time Python code on viewed tiles.

Finally, we also plan to improve xOpat's security and optional safe mode, since some features (e.g., remote shader loading or runtime protocol customization) allow cross-site scripting attacks.

## 9. Conclusion

In this paper, we presented xOpat, a browser-based WSI viewer for explainable open digital pathology. xOpat suits the needs of digital pathology researchers, where it can significantly speed up the workflows. It addresses multiple limitations of the existing tools and aims to deliver easy integration into the existing environments. It further provides open and extensible architecture and supports collaboration among different users.

xOpat fulfills the requirements of domain experts who collaborate on designing novel AI-based image segmentation and annotation methods. Together with collaborating domain experts, we identified three typical users—AI algorithm developers, research, and clinical pathologists—and elicited their requirements on the novel tool. The implementation is based on OpenSeadragon, a widely used library for high-resolution images. xOpat is highly configurable and provides a context-free and data-independent environment that supports multi-layer data visualizations that gives insight into AI decisions. To our knowledge, its features are unmatched by any other currently available tool. We kindly advise readers to visit the `muni.cz/go/xopat-showcase` with interactive examples to showcase the tool.

In the future, we would like to extend xOpat with several features that would further increase its applicability in clinical practice. Although numerous tools are already available and well adapted for clinical purposes, an open and unified framework could improve both the clinical and the research digital pathology and help bridge the gap between research and practice. XOpat can also help increase trust in AI algorithms, streamline the collaboration of different research teams, and contribute to the open science movement.

### Acknowledgements

This work has been supported by EOSC-Life project supported by EU Horizon 2020, grant agreement no. 824087, as a part of WP1 Demonstrators under APPID 1228 "Cloudification of BBMRI-ERIC CRC-Cohort and its Digital Pathology Imaging"; by the Czech Ministry of Health (MMCI 00209805) and the Czech Ministry of Education, Youth and Sports (LM2018125—BBMRI-CZ). Computational resources were supplied by the project 'e-Infrastruktura CZ' (e-INFRA LM2018140) provided within the Projects of Large Research, Development and Innovations Infrastructures program.

## References


[AAB*84] ADELSON, E. H., ANDERSON, C. H., BERGEN, J. R., et al. "Pyramid Methods in Image Processing". *RCA Engineer* 29.6 (1984), 33–41 1.

[ABM*12] ALLAN, CHRIS, BUREL, JEAN-MARIE, MOORE, JOSH, et al. "OMERO: Flexible, Model-Driven Data Management for Experimental Biology". *Nature Methods* 9.3 (Feb. 28, 2012), 245–253. ISSN: 1548-7105. DOI: 10.1038/nmeth.1896. pmid: 22373911 2.

[Ape] APERIO. *Aperio ImageScope – Pathology Slide Viewing Software*. URL: https://www.leicabiosystems.com/digital-pathology/manage/aperio-imagescope/ (visited on 03/24/2022) 2.

[BLF*17] BANKHEAD, PETER, LOUGHREY, MAURICE B., FERNÁNDEZ, JOSÉ A., et al. "QuPath: Open Source Software for Digital Pathology Image Analysis". *Scientific Reports* 7.1 (2017), 16878. ISSN: 2045-2322. DOI: 10.1038/s41598-017-17204-5 1, 2.

[CKB09] COCKBURN, ANDY, KARLSON, AMY, and BEDERSON, BENJAMIN B. "A Review of Overview+detail, Zooming, and Focus+context Interfaces". *ACM Computing Surveys* 41.1 (2009). ISSN: 0360-0300. DOI: 10.1145/1456650.1456652 4.

[Cod] CODEPLEX FOUNDATION. *OpenSeadragon – An Open-Source, Web-Based Viewer for High-Resolution Zoomable Images*. URL: https://openseadragon.github.io (visited on 10/12/2022) 4, 7.

[Com] COMPUTATIONAL PATHOLOGY GROUP. *Automated Slide Analysis Platform*. URL: https://computationalpathologygroup.github.io/ASAP/ (visited on 10/12/2022) 2.

[GEMF13] GHAZNAVI, FARZAD, EVANS, ANDREW, MADABHUSHI, ANANT, and FELDMAN, MICHAEL. "Digital Imaging in Pathology: Whole-Slide Imaging and Beyond". *Annual Review of Pathology: Mechanisms of Disease* 8.1 (2013), 331–359. ISSN: 1553-4006, 1553-4014. DOI: 10.1146/annurev-pathol-011811-120902 1.

[Gle] GLENCOE SOFTWARE. *PathViewer: An interactive visualization, analysis, and annotation tool specifically tailored for the digital pathology workflow*. URL: https://www.glencoesoftware.com/products/pathviewer/features/ (visited on 10/26/2022) 2.

[Hea] HEALTH, TRIBUN. *Tribun Suite*. URL: https://www.tribun.health/th-suite (visited on 10/17/2022) 1.

[HMK*17] HOLZINGER, ANDREAS, MALLE, BERND, KIESEBERG, PETER, et al. "Machine Learning and Knowledge Extraction in Digital Pathology Needs an Integrative Approach". *Towards Integrative Machine Learning and Knowledge Extraction*. Ed. by HOLZINGER, ANDREAS, GOEBEL, RANDY, FERRI, MASSIMO, and PALADE, VASILE. Lecture Notes in Computer Science. Springer, 2017. ISBN: 978-3-319-69774-1. DOI: 10.1007/978-3-319-69775-8_2 1.

[HMP*22] HOLLANDI, REKA, MOSHKOV, NIKITA, PAAVOLAINEN, LASSI, et al. "Nucleus Segmentation: Towards Automated Solutions". *Trends in Cell Biology* 32.4 (2022), 295–310. ISSN: 09628924. DOI: 10.1016/j.tcb.2021.12.004 1.

[HPS20] HANNA, MATTHEW G., PARWANI, ANIL, and SIRINTRAPUN, SAHUSSAPONT JOSEPH. "Whole Slide Imaging: Technology and Applications". *Advances in Anatomic Pathology* 27.4 (2020), 251–259. ISSN: 1072-4109. DOI: 10.1097/PAP.0000000000000273 1.

[HRM*20] HOFFER, JOHN, RASHID, RUMANA, MUHLICH, JEREMY L., et al. "Minerva: a light-weight, narrative image browser for multiplexed tissue images". *Journal of Open Source Software* 5.54 (2020), 2579. DOI: 10.21105/joss.02579. URL: https://doi.org/10.21105/joss.02579 2.

[Hru22] HRUŠKA, JAKUB. "Visualization of Hidden Layers in Convolutional Neural Networks". Supervisor: Tomáš Brázdil. Diploma Thesis. Masaryk University, Faculty of Informatics, Brno, 2022. URL: https://is.muni.cz/th/huq0h/ 6.

[Ind] INDICA LABS INC. *HALO – Image Analysis Platform for Quantitative Tissue Analysis in Digital Pathology*. URL: https://indicalab.com/halo/ (visited on 10/11/2022) 2.

[JKW*22] JESSUP, JARED, KRUEGER, ROBERT, WARCHOL, SIMON, et al. "Scope2Screen: Focus+Context Techniques for Pathology Tumor Assessment in Multivariate Image Data". *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2022), 259–269. DOI: 10.1109/TVCG.2021.3114786 2.

[JSH*13] JEONG, W., SCHNEIDER, J., HANSEN, A., et al. "A Collaborative Digital Pathology System for Multi-Touch Mobile and Desktop Computing Platforms". *Computer Graphics Forum* 32.6 (2013), 227–242. DOI: 10.1111/cgf.12137. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12137 2.

[Med] MEDIA CYBERNETICS. *Image-Pro*. URL: https://www.mediacy.com/imagepro (visited on 03/24/2022) 2.

[Mic] MICROSOFT. *Deep Zoom File Format Overview*. URL: https://docs.microsoft.com/en-us/previous-versions/windows/silverlight/dotnet-windows-silverlight/cc645077(v=vs.95) (visited on 10/12/2022) 4, 7.

[MRS*16] MARÉE, RAPHAËL, ROLLUS, LOÏC, STEVENS, BENJAMIN, et al. "Cytomine: An Open-Source Software For Collaborative Analysis Of Whole-Slide Images". *Diagnostic Pathology* 1.8 (2016). DOI: 10.17629/www.diagnosticpathology.eu-2016-8:151 1, 2.

[PAA*23] PIELAWSKI, NICOLAS, ANDERSSON, AXEL, AVENEL, CHRISTOPHE, et al. "TissUUmaps 3: Improvements in Interactive Visualization, Exploration, and Quality Assessment of Large-Scale Spatial Omics Data". *bioRxiv : the preprint server for biology* (2023). DOI: 10.1101/2022.01.28.478131. URL: https://www.biorxiv.org/content/early/2023/01/16/2022.01.28.478131 2.

[PFP15] PANTANOWITZ, LIRON, FARAHANI, NAVID, and PARWANI, ANIL. "Whole slide imaging in pathology: advantages, limitations, and emerging perspectives". *Pathology and Laboratory Medicine International* (2015), 23. ISSN: 1179-2698. DOI: 10.2147/PLMI.S59826 1.

[PZD*19] PUTTAPIRAT, PARGORN, ZHANG, HAICHUAN, DENG, JINGYI, et al. "OpenHI2 — Open source histopathological image platform". *2019 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*. 2019, 2696–2701. DOI: 10.1109/BIBM47256.2019.8983322 2.

[RFB15] RONNEBERGER, OLAF, FISCHER, PHILIPP, and BROX, THOMAS. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. 2015. arXiv: 1505.04597 [cs] 8.

[RL21] RYDELL, CHRISTOPHER and LINDBLAD, JOAKIM. "CytoBrowser: A Browser-Based Collaborative Annotation Platform for Whole Slide Images". *F1000Research* 10 (Mar. 22, 2021), 226. ISSN: 2046-1402. DOI: 10.12688/f1000research.51916.1. URL: https://f1000research.com/articles/10-226/v1 (visited on 03/14/2023) 2.

[Sec] SECTRA AB. *Sectra Digital Pathology Solution*. URL: https://medical.sectra.com/product/sectra-digital-pathology-solution/ (visited on 10/12/2022) 2.

[SGBP03] SWEDLOW, JASON R., GOLDBERG, ILYA, BRAUNER, ERIK, and PETER K. SORGER. "Informatics and Quantitative Analysis in Biological Imaging". *Science (New York, N.Y.)* 300.5616 (2003), 100–102. DOI: 10.1126/science.1082602. URL: https://www.science.org/doi/abs/10.1126/science.1082602 2.

[SLN*22] SOFRONIEW, NICHOLAS, LAMBERT, TALLEY, NUNEZ-IGLESIAS, JUAN, et al. *Napari/Napari: 0.4.15*. Version v0.4.15. Zenodo, Mar. 10, 2022. DOI: 10.5281/ZENODO.3555620. URL: httpcope://zenodo.org/record/3555620 (visited on 10/03/2022) 2.

[SSV20] STRITT, MANUEL, STALDER, ANNA K., and VEZZALI, ENRICO. "Orbit Image Analysis: An open-source whole slide image analysis tool". *PLOS Computational Biology* 16 (2020), 1–19. DOI: 10.1371/journal.pcbi.1007313 2.

[XB ] XB SOFTWARE LTD. *EnjoyHint*. URL: https://github.com/xbsoftware/enjoyhint (visited on 10/12/2022) 6.

[ZF14] ZEILER, MATTHEW D. and FERGUS, ROB. "Visualizing and understanding convolutional networks". *European conference on computer vision*. Springer. 2014, 818–833. DOI: 10.1007/978-3-319-10590-1_53 8.