FastPathology: An open-source platform for artificial intelligence-based digital pathology

André Pedersen[1,3,5], Ingerid Reinertsen[5], Marit Valla[1,3,4], and Erik Smistad[2,5]

[1] Department of Clinical and Molecular Medicine, NTNU, Trondheim, Norway
[2] Department of Circulation and Medical Imaging, NTNU, Trondheim, Norway
[3] Clinic of Surgery, St.Olav's Hospital, Trondheim, Norway
[4] Department of Pathology, St.Olav's Hospital, Trondheim, Norway
[5] Department of Health Research, SINTEF, Trondheim, Norway

1 Purpose

Whole slide microscopy images (WSIs) used in digital pathology are often extremely large. A $40\times$ WSI can have approximately $200k \times 100k$ color pixels resulting in an uncompressed size of ~ 56 gigabytes [1]. This exceeds the amount of RAM and GPU memory on most systems. Thus, special data handling is required to store, read, process and display these images.

With the increasing integration of digital pathology into clinical practice worldwide, there is a need for tools that can assist clinicians in their daily practice. Today, there are many open-source softwares for visualizing and performing image analysis on WSIs, e.g. QuPath [2], Cytomine [3], Orbit [7]. Still, most of these alternatives do not support deployment of trained deep neural networks, which is known to outperform traditional image processing methods on complex image analysis tasks such as classification, segmentation and object detection.

As most developers that work with Convolutional Neural Networks (CNNs) train their models in Python using frameworks like TensorFlow and Keras, it is important that the platform supports deployment of these models. A solution to this may be to deploy models using Python directly, using the same libraries, as it is done in Orbit. However, inference is quite optimized in Python, since the the actual inference engines, such as TensorFlow are written in C++ and have Python bindings. The Python language itself is not fast and is not well-suited for large scale, high performance software development. Also, there are currently no good solutions for visualizing WSIs in Python. Most existing platforms have chosen to use Java/Groovy as the main language. While boasting good multi-platform support and being a modern object-oriented language, the performance of Java compared to C and C++ is debated. It is possible to deploy TensorFlow-based models in Java, using the DeepLearning4J library, but its support for layers and architectures is currently quite limited. The core of most inference engines, such as TensorFlow, is written in C++. We argue that due to the high-performance demand of processing and visualizing WSIs, modern C++ together with GPU libraries such as OpenCL and OpenGL are better suited to create such software. Thus, we propose to use and extend the existing high-performance C++

framework FAST [6] to develop an open-source platform for reading, visualizing and processing WSIs using deep neural networks.

2 Methods

FAST [6] was introduced in 2015 as a framework for high performance medical image computing and visualization using multi-core CPUs and GPUs. In 2019 [5], it was extended with deep neural network inference capabilities using multiple inference engines such as TensorFlow, OpenVINO and TensorRT. In this abstract, we describe an application called FastPathology made with FAST which consists of a graphical user interface (GUI) and open trained neural networks for analysing digital pathology images. Several components were made and added to FAST to enable processing and visualization of WSIs. The application runs on both Windows and Ubuntu Linux and is available online at `https://github.com/SINTEFMedtek/FAST-Pathology`.

Reading whole slide images - WSIs are typically stored in proprietary formats made by various digital pathology scanner vendors. The lack of a common industry format has spawned an open-source C library called OpenSlide [4] which can read some of these proprietary formats. Since these images are very big, they are usually stored as tiled image pyramids. OpenSlide was added to FAST to enable reading of these files and thereby accessing the raw color pixel data. OpenSlide uses the operating systems' virtual memory mechanisms which enables one to open and read very large files without exhausting the system memory (RAM) by streaming data on demand from disk to RAM.

Rendering whole slide images - High performance image rendering usually requires a GPU implementation. Since, GPUs also have a very limited memory size, WSIs can not fit into the GPUs memory. To this end, a virtual memory system for WSIs on the GPU was implemented in FAST using OpenGL. In this system, only the required tiles at the required resolution in the image pyramid are transferred to the GPU memory as textures. Which tiles and which resolution that are required at a given time is automatically determined based on the current position and amount of zoom of the current view of the image. Reading tiles from disk and streaming them to the GPU takes time. Therefore, the tiles are cached and put in a queue. The user can manually specify a maximum size in bytes. Every time a tile is used, it is added to the back of the queue. Whenever the queue is larger than its limit, tiles are removed from the front of the queue and its texture deleted. Before a given tile is ready to be rendered, the best available tiles already cached with a lower resolution are displayed instead. The lowest resolution of the image pyramid is always present in GPU memory. Thus, the WSI will always be displayed, even when higher resolution tiles are being loaded. The user can thus easily pan and zoom with the mouse to visualize all parts of a WSI with low latency and a fixed GPU memory usage.

Tissue segmentation - As WSIs are extremely large, applying a sliding window method across the image might take several minutes, especially when using deep neural networks. Thus, removing redundant regions such as glass

would be of interest. In FAST, we have implemented a simple tissue detector, that inputs a low-resolution version of the WSI, and thresholds the RGB image color space based on how far away a specific RGB triplet is from the color white. Morphological opening is then performed in order to filter away smaller fragments around and within the tissue. All parameters were empirically determined tuned on a local breast cancer data set, but they also perform well on WSIs from other scanners and on sections containing tissue from other organs.

Neural network processing - Inference of neural networks is done through FAST by loading a trained model stored on disk as described in [5]. Currently, the only neural network added to the application is patch-wise classification. This is done by tiling the image into smaller patches at a size and magnification level selected by the model that is in use. These tiles are then batched in a given size and propagated through the network in parallel. The predictions are then visualized as small squares with different colors for different classes and varying opacity dependent on confidence. Simultaneously, new sets of patches are processed on a different thread. FAST also supports multiclass classification.

Graphical user interface - In order to use the WSI functionality in FAST, a good graphical user interface (GUI) is required. The GUI of FAST-Pathology is implemented using Qt 5. Our goal is to make it user-friendly and applicable in diagnostic pathology. Currently, the GUI is split in to two windows. The right window shows the OpenGL window for visualizing WSIs as well as segmentations and results from CNN prediction. The left window comprises a simple taskbar with buttons for reading WSIs, applying algorithms or trained models on the WSIs, toggling between different results and images, and changing the opacity. It is also possible to remove the background class in the tumor prediction, as it might be of interest to only study the final segment with the WSI. If a new image is selected, all previous results are removed from the visualization. It is also possible to deploy multiple trained models simultaneously. This may be beneficial if the pipeline involves several steps.

Storing results - Storing results from different image analysis is an important part of a WSI analysis platform. Currently, it is possible to store the tissue segmentation and predictions on disk using the metaimage (.mhd/.raw) format in FAST.

3 Results

Since this is an early implementation, results are preliminary. However, Fig. 1 shows a screenshot of the current GUI where a WSI is being processed by a neural network performing patch-wise classification to determine whether a patch contains tumor tissue or not.

As an example, memory usage when reading, zooming around and panning the view of a WSI (uncompressed size of ∼40.7 GBs and full resolution of ∼105k×138k pixels) for 2.5 minutes was 394.3 MB RAM and ∼0.7 GB GPU RAM. The time from opening to visualizing a WSI was around ∼0.5 seconds and only approximately 80.8 MB of RAM was used. However, speed may vary

dependent on hardware. A short video of the current version of the platform can be found on YouTube `https://youtu.be/mepLWfORWQg`.

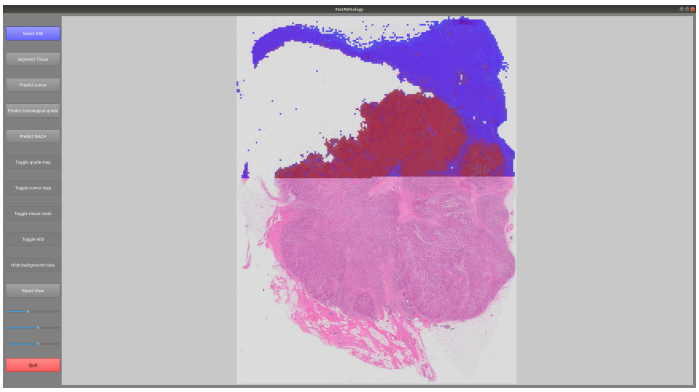

Fig. 1. Screenshot of the GUI on Ubuntu 18.04 as a WSI is being processed by a neural network performing patch-wise classification displayed as a colored overlay.

4 Conclusion

Applying state-of-the-art deep neural networks on WSIs, we have developed an open-source, AI-based platform intended for use in diagnostic pathology and research within digital pathology. The platform is implemented in C++ using the FAST framework. We have presented an early implementation. Key functionalities are already included, and new features are currently being made. We encourage the community to contribute to this open-source project on GitHub.

References

1. Bándi, P., et al.: From Detection of Individual Metastases to Classification of Lymph Node Status at the Patient Level: The CAMELYON17 Challenge. IEEE Transactions on Medical Imaging **38**(2), 550–560 (2019)
2. Bankhead, P., et al.: Qupath: Open source software for digital pathology image analysis. BioRxiv (01 2017)
3. Marée, R., et al.: Cytomine: An open-source software for collaborative analysis of whole-slide images. Diagnostic Pathology **1**(8) (2016)
4. Satyanarayanan, M., et al.: OpenSlide: A vendor-neutral software foundation for digital pathology. Journal of Pathology Informatics **4**(1), 27 (2013)
5. Smistad, E., et al.: High performance neural network inference, streaming, and visualization of medical images using fast. IEEE Access **7**, 136310–136321 (2019)
6. Smistad, E., et al.: FAST: framework for heterogeneous medical image computing and visualization. International Journal of computer assisted radiology and surgery **10**(11), 1811–1822 (2015)
7. Stritt, M., et al.: Orbit image analysis: An open-source whole slide image analysis tool (08 2019)