BMC Bioinformatics

SOFTWARE

Open Access

Slideflow: deep learning for digital histopathology with real-time whole-slide visualization

James M. Dolezal[1*], Sara Kochanny[1], Emma Dyer[1], Siddhi Ramesh[1], Andrew Srisuwananukorn[2], Matteo Sacco[1], Frederick M. Howard[1], Anran Li[1], Prajval Mohan[3] and Alexander T. Pearson[1*]

*Correspondence: james@slideflow.dev; alexander.pearson@bsd.uchicago.edu

[1] Section of Hematology/Oncology, Department of Medicine, University of Chicago Medical Center, Chicago, IL, USA
[2] Division of Hematology, Department of Internal Medicine, The Ohio State University Comprehensive Cancer Center, Columbus, OH, USA
[3] Department of Computer Science, University of Chicago, Chicago, IL, USA

Abstract

Deep learning methods have emerged as powerful tools for analyzing histopathological images, but current methods are often specialized for specific domains and software environments, and few open-source options exist for deploying models in an interactive interface. Experimenting with different deep learning approaches typically requires switching software libraries and reprocessing data, reducing the feasibility and practicality of experimenting with new architectures. We developed a flexible deep learning library for histopathology called Slideflow, a package which supports a broad array of deep learning methods for digital pathology and includes a fast whole-slide interface for deploying trained models. Slideflow includes unique tools for whole-slide image data processing, efficient stain normalization and augmentation, weakly-supervised whole-slide classification, uncertainty quantification, feature generation, feature space analysis, and explainability. Whole-slide image processing is highly optimized, enabling whole-slide tile extraction at 40x magnification in 2.5 s per slide. The framework-agnostic data processing pipeline enables rapid experimentation with new methods built with either Tensorflow or PyTorch, and the graphical user interface supports real-time visualization of slides, predictions, heatmaps, and feature space characteristics on a variety of hardware devices, including ARM-based devices such as the Raspberry Pi.

Keywords: Digital pathology, Computational pathology, Software toolkit, Whole-slide imaging, Explainable AI, Self-supervised learning

Background

Histopathology slides of patient tissue and tumor specimens serve many purposes and are increasingly being captured and stored in digital formats. The advent of deep learning models for analyzing digital histopathology images has unlocked a new dimension from which we can extract clinically meaningful information [1]. These models not only accelerate and enhance pathology clinical workflows but also detect subtle morphological features that escape the human eye, improving diagnostic efficiency and accuracy [2, 3]. Furthermore, deep learning models allow genomic subtype classification directly

from digital histopathology images [4–7], and they show promise as tools for patient risk stratification [8], prognosis [9], and treatment selection [10]. Digital histopathology may also increase accessibility of hematoxylin and eosin (H&E) and immunohistochemistry staining in low-resource settings through virtual staining, and these tools offer the ability to boost clinical workflow efficiency and provide advanced diagnostics that may otherwise be unattainable due to a limited number of trained pathologists or their absence altogether [11]. Despite the far-reaching potential of deep learning applications in digital histopathology, development complexity and access to computational resources remain barriers to widespread adoption. There is a growing need for accessible and efficient open-source software that functions as a platform to perform these analyses.

Efficient software design is crucial for deep learning research toolkits in digital histopathology. Model performance improves with the amount of training data provided [12], but data storage requirements may impose practical limitations on dataset sizes and experimental scope. Unified and efficient data storage can reduce data redundancy and optimize the computational resources needed for training models. Computationally efficient software can reduce data processing and storage requirements while shortening training time, enabling groups with varying equipment capabilities to train their own models or access pretrained models for novel research objectives.

Additionally, it is vital for deep learning models that aim to support clinical decision-making to be transparent and interpretable [13]. Explaining how a model has reached its decision and the level of certainty associated with a prediction can help build clinician trust, which may help foster greater adoption of these tools into clinical practice. Software that seamlessly integrates explainability and uncertainty quantification presents a significant advantage in promoting the potential clinical utility of these deep learning tools.

As computational tools continue to gain importance in biological sciences, it is essential to ensure their accessibility to researchers with diverse computational backgrounds. This can be achieved through comprehensive documentation, intuitive code design, and active project development. When creating analytical tools for clinical applications, it is also important to consider that the target end-users may not possess extensive computational expertise. A graphical user interface (GUI) can offer an accessible entry point for such users, facilitating the deployment of deep learning tools in clinical settings.

In the broader machine learning literature, there is an abundance of software libraries to assist with developing fast and robust deep learning applications, decreasing barriers to research entry, streamlining development workflows, and improving speed of iteration and innovation. However, there are many processes unique to computational pathology research that make direct utilization of generic deep learning libraries challenging, including whole-slide image processing, stain normalization, and pathology-specific algorithms such as tissue segmentation, cell identification, and multiple-instance learning. There is a need for accessible software libraries that provide access to commonly utilized computational pathology workflows that can serve as a foundation for research and development.

At present, most computational pathology research papers either share code that is custom written for the specific research application and not immediately generalizable

to other datasets or research questions, or do not share code at all [14]. There are several existing software libraries which seek to address this problem by providing a generalizable toolkit for pathology artificial intelligence research. Some of these libraries have not been updated in several years [15, 16], while others continue to see active development and appear to have ongoing support and community utilization. TIAToolbox is a robust, PyTorch-based computational pathology library that provides tools for whole-slide image processing and stain normalization, tile-based classification, and both tissue and nucleus segmentation [17]. TIAToolbox also offers a clean and well-engineered code base and detailed, user-friendly documentation. Although not under active development, CLAM is a popular PyTorch-based deep learning repository that includes whole-slide image processing, feature extraction using a ResNet50 model pretrained on ImageNet, and attention-based multiple-instance learning [18]. PathML is a PyTorch-based deep learning library which provides whole-slide image processing for a variety of slide formats, model architectures for nucleus detection and weakly supervised tile-based classification, and detailed online documentation [19]. DeepPath is a Tensorflow-based library which provides whole-slide image processing for SVS slides and support for training Inception-v3 weakly supervised tile-based classification models [20]. Histolab is a whole-slide image preprocessing library for SVS slides designed to assist with downstream deep learning tasks, and thus does not include tools for model development [21]. Finally, MONAI is a PyTorch-based framework for deep learning in healthcare imaging primarily designed for radiology images, but which has a pathology working group seeking to expand support for pathology applications [22]. An interactive user interface for flexible model deployment to whole-slide images, accessible by someone with limited or no programming experience, is not readily available in any of these libraries.

Deep learning pathology research has seen tremendous advances over the past several years. State-of-the-art methods now include self-supervised learning, multiple-instance learning, generative adversarial networks, and uncertainty quantification. To maximize relevance for researchers, libraries designed for computational pathology researchers should provide highly flexible and efficient data processing pipelines, a design which supports fast implementation of new algorithms, and detailed documentation that illustrate how the library can be expanded as new methods emerge. Designing a flexible software platform that can grow and scale with a constantly evolving field is a challenging task, but would offer numerous benefits for researchers such as reduced software development time, easier access to state-of-the-art algorithms, and greater reproducibility. We have thus sought to develop a deep learning framework engineered from the ground up for flexibility, scalability, and ease of use, with an emphasis on both model development and interactive whole-slide image model deployment.

Recognizing the limitations and unmet needs in current computational pathology tools, we introduce Slideflow. Slideflow is a comprehensive Python package designed to bridge these gaps, delivering an end-to-end suite of user-friendly tools for building, testing, explaining, and deploying deep learning models for histopathology applications. Slideflow aims to overcome challenges in computational efficiency, accessibility, and interpretability while empowering researchers and clinicians to harness the full potential of evolving deep learning approaches for digital histopathology.

Table 1 Slide scanner compatibility

Vendor	File format
Aperio	SVS
Philips	TIFF
Mirax	MRXS
Hamamatsu	NDPI
Leica	SCN
Ventana	BIF, TIF
Trestle	TIF
Sakura	SVSLIDE
Olympus	TIFF

Images from Philips and Olympus slide scanners must be exported from the scanner in TIFF format, as iSyntax and VSI files are currently unsupported

Implementation

Technical overview

Slideflow is a Python package providing a library of deep learning tools implemented with both PyTorch and Tensorflow backends. It has been developed to provide an end-to-end toolkit for building and deploying deep learning histopathology applications for scientific research, including efficient data processing, model training, evaluation, uncertainty quantification, explainability, and model deployment in a graphical user interface (GUI). Slideflow includes a whole-slide user interface, Slideflow Studio, for generating predictions and heatmaps for whole-slide images in real time. Slideflow is easy to deploy, with distributions available on the Python Packaging Index (PyPI) and pre-built docker containers on Docker Hub.

Whole-slide image processing

Slideflow supports nine slide scanner vendors (Table 1) and includes two slide reading backends – cuCIM [23], an efficient, GPU-accelerated slide reading framework for TIFF and SVS slides, and VIPS [24], an OpenSlide-based framework which adds support for additional slide formats. The first step in processing whole-slide images (WSI) for downstream deep learning applications is slide-level masking and filtering, a process that determines which areas of the slide are relevant and which areas should be ignored. Slides can be manually annotated with Regions of Interest (ROIs) using the provided whole-slide visualization tool, Slideflow Studio, or using alternative programs such as QuPath [25] or Aperio ImageScope [26]. Tissue detection can be performed with Otsu's thresholding [27], which masks areas of background. Pen marks and out-of-focus areas can be masked with Gaussian blur filtering, as implemented by the scikit-image library [28].[1] Parameters for Otsu's thresholding and Gaussian blur filtering are customizable, with default values determined through empirical testing on slides with artifacts. Additionally, Slideflow includes an API for custom slide-level masking, with an example demonstrating how to use this API to apply a DeepFocus [29] model for detection of out-of-focus regions detailed in the online documentation [30].

[1] Gaussian blur filtering by default is performed with $\sigma = 3$ and threshold $= 0.02$ at one-fourth the target magnification.

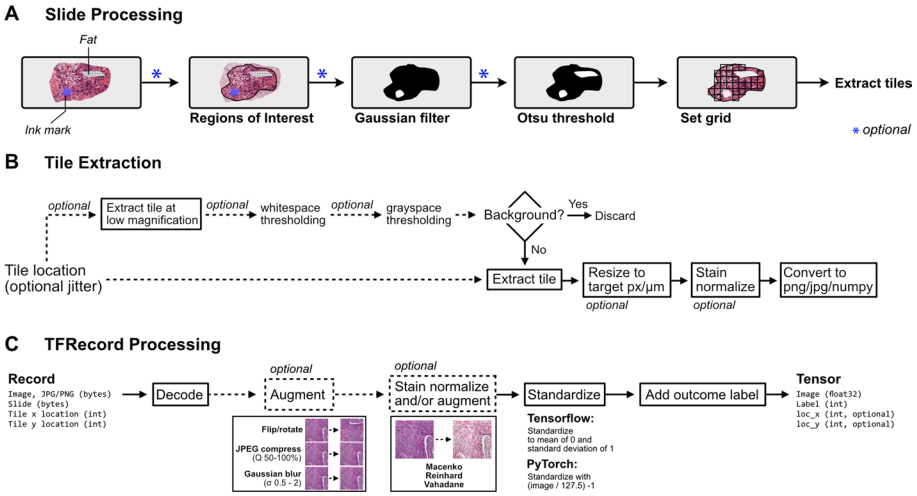

Fig. 1 Schematic of data flow during whole-slide image tile extraction and image processing. **a** Schematic of initial slide processing and grid preparation. Whole-slide images can be annotated with Regions of Interest (ROI) to include only relevant areas of a slide for subsequent analysis. Optional slide filtering steps, including Gaussian blur filtering and Otsu's thresholding, may be applied at this step. The remaining areas of the slide are sectioned into a grid in preparation for tile extraction. **b** Data flow during tile extraction. If tile-level filtering is to be performed, such as whitespace or grayspace filtering, a low-magnification image at the highest pyramidal layer is taken at the given location and used for background filtering. If the tile passes filtering, the full-magnification image extracted, optionally resized to match a target micron size, stain normalized, and converted into a PNG image, JPEG image, or Numpy array. Extracted tiles can be saved to disk as individual files, or buffered into TFRecords for faster dataset reading. **c** Schematic of data flow when reading from TFRecords. Image tiles are buffered in TFRecords with JPEG or PNG compression and stored with slide and location metadata. During dataset iteration, data is decoded and converted to Tensors. Augmentation, including random flipping/rotation, random JPEG compression, and random Gaussian blur, can be applied at this step. If stain normalization was not performed during tile extraction, stain normalization can be applied at this step when iterating through a TFRecord dataset in real-time. Images are then standardized, and slide names are converted to ground-truth outcome labels using a provided CSV annotations file or Pandas DataFrame

After slide-level masking and filtering, WSIs are tiled into smaller sub-regions. A schematic of data flow during tile extraction is shown in Fig. 1. Image tiles can be buffered into binary TFRecord format, an efficient storage format that improves dataset iteration speed compared to iterating directly from WSIs. Image data can be encoded using either JPEG (lossy) or PNG (lossless) compression. Image tiles are extracted in a grid, with optional overlap or jitter for data augmentation. During tile extraction, images tiles can undergo an additional filtering step using either grayspace or whitespace filtering, potentially identifying tiles with high background content that were not successfully removed by Otsu's thresholding. Grayspace filtering is performed by converting images into the hue, saturation, value (HSV) spectrum, classifying pixels with saturation below a given threshold[2] as gray, and discarding the tile as background if the fraction of pixels identified as gray exceeds a prespecified threshold.[3] Whitespace filtering is performed by calculating the brightness of each pixel (average of red, green, and blue channels),

[2] Default grayspace saturation filtering threshold is 0.05.

[3] Default grayspace fraction threshold is 0.6.

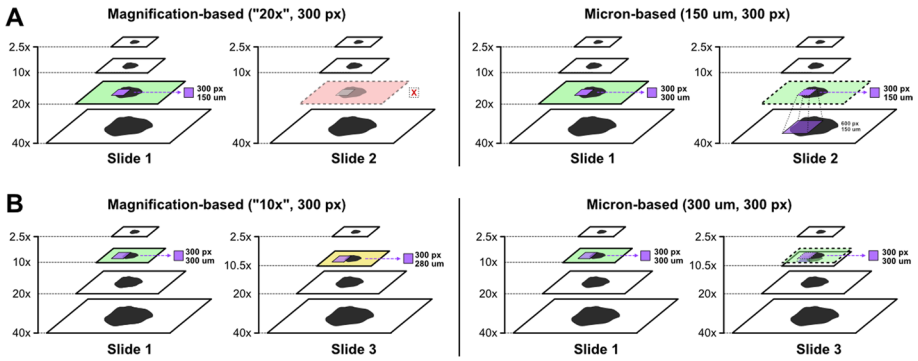

Fig. 2 Magnification-based vs micron-based tile extraction. **a** Comparison between magnification-based and micron-based tile extraction at 20x effective optical magnification. In this example, slide 1 has internal pyramid images stored at 2.5x, 10x, 20x and 40x magnification. Slide 2 has images stored at 2.5x, 10x, and 40x. Magnification-based strategies extract tiles at a matching layer in the image pyramid for a target optical magnification. In this example, Slide 2 is missing the 20x magnification layer, so tiles could not be extracted at 20x magnification. In comparison, with micron-based tile extraction, image tiles would be extracted at the 40x layer for Slide 2 and resized to an effective optical magnification of 20x. **b** Comparison between magnification-based and micron-based tile extraction at 10x effective optical magnification. In this scenario, the "10x" layers in Slide 1 and Slide 3 have slightly different effective optical magnifications – 10x and 10.5x – due to slide scanner differences. With magnification-based tile extraction, image tiles extracted from Slide 1 and Slide 3 would have slightly different effective optical magnification. With micron-based tile extraction, tiles would be extracted at the 10.5x magnification layer from Slide 3 and resized to match the same effective optical magnification as Slide 1 (10x). This strategy ensures that all image tiles have the same effective optical magnification

classifying pixels with brightness above a given threshold[4] as white, and discarding the tile as background if the proportion of pixels identified as white exceeds a prespecified threshold. Default filtering thresholds were determined through empirical testing, as previously described, and are user-customizable [31]. Tile-level background filtering is performed at lower magnification for computational efficiency.

The effective optical magnification at which tiles are extracted can be determined by either designating a pyramid level or a tile width in microns (Fig. 2). If a pyramid level is used, image tiles are extracted at a specified pixel size. Not all slides have the same optical magnifications available as pyramid layers, so this approach may require that some slides are skipped (Additional file 1: Fig. 1). Furthermore, this approach restricts the magnification levels that can be explored for downstream analysis to only what is available in each slide. Alternatively, if a tile width is specified in microns, image tiles will be extracted from the nearest higher-magnification pyramid layer and downsized to the target micron width, allowing researchers to granularly specify effective optical magnification. For this added flexibility, micron-based tile extraction is preferred. When using cuCIM, images are resized using bilinear interpolation, and when using VIPS, images are resized using Lanczos interpolation [32]. Tile extraction is heavily optimized and multiprocessing accelerated, enabling whole-slide tile extraction at 40x magnification in as little as 2.5 s per slide. Fast tile extraction enables researchers to quickly experiment with different magnification levels.

[4] Default whitespace brightness filtering threshold is 230.

All parameters needed to reproduce processed slide data are logged during tile extraction, assisting with data lineage tracking and management. Visual summary reports containing images of WSIs overlayed with tile masks and selected tiles from each WSI are automatically generated after tiles have been extracted, allowing researchers to quickly assess the quality of masking, filtering, and extracted images. This step can identify potential dataset issues, such as out-of-focus slides, suboptimal background or artifact removal, or low-quality ROIs. Slideflow Studio, an interactive whole-slide graphical user interface (GUI) included with Slideflow, can be used to preview slide masking, background filtering, and stain normalization settings in real-time. Interactive visualization assists with rapid determination of optimal slide processing parameters if further tuning of these settings is required. Slideflow Studio is described in more detail in the "Tissue and cell segmentation" section.

Stain normalization and augmentation

Digital hematoxylin and eosin (H&E) stain normalization, which is used to help reduce biasing batch effects incurred by differences in staining color and intensity among slides, can be applied to image tiles either during tile extraction or in real-time during model training. Available stain normalization methods include Reinhard [33], Macenko [34], and Vahadane [35], as well as a masked variant for Reinhard—where normalization is only applied in non-white areas—and fast variants for both Reinhard and Macenko normalizers, with the brightness standardization step disabled. The Reinhard and Macenko normalizers have native Numpy/OpenCV, Tensorflow, and PyTorch implementations to improve computational efficiency and enable GPU acceleration. Stain normalizers include several default reference fits and can be optionally fit to user-defined images.

Real-time stain augmentation can be performed during training when using the Reinhard or Macenko normalizers. Slideflow includes a novel stain augmentation approach performed by dynamically randomizing the stain normalization target, using a preset standard deviation around the normalization fit values. As the randomized stain matrix targets are centered around normalized values, the result is a combination of both normalization and augmentation. This approach differs from the stain augmentation described by Tellez et al., in which authors performed stain augmentation in the deconvoluted stain matrix space without normalization [36].

Additionally, Slideflow includes an option for using contextual information from the corresponding WSI during Macenko stain normalization. With contextual normalization, staining patterns across a slide are used to inform normalization of a constituent image tile. The deconvoluted H&E channels for a given image tile are normalized using the maximum H&E concentrations calculated from a context image, rather than calculating these maximum concentrations from the image being transformed. When this option is used, the context image is a thumbnail of the WSI, with background and areas outside ROIs removed to prevent pen marks and other artifacts from interfering with stain deconvolution. Contextual normalization is not recommended when artifact removal is suboptimal or ROIs are unavailable.

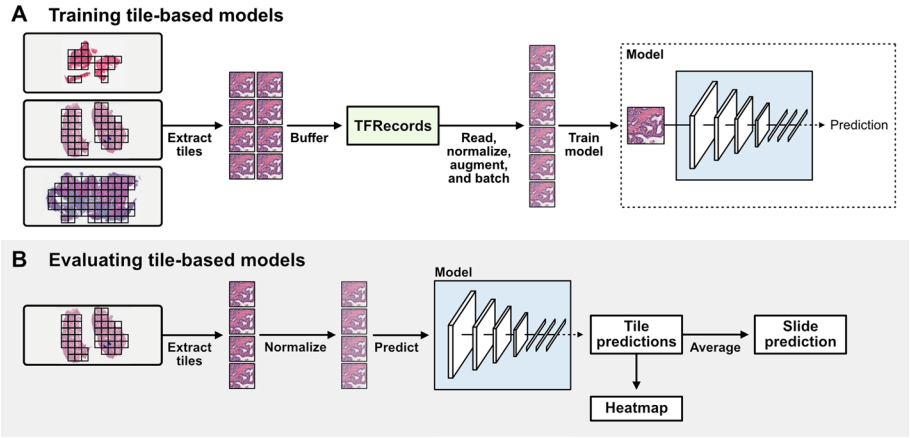

Fig. 3 Summary of approach for tile-based deep learning models. **a** Image tiles are extracted from whole-slide images after any slide processing and buffered into TFRecords. During training, tiles are read from TFRecords, augmented, stain normalized, standardized, and batched. In this weakly-supervised approach, models are trained using ground-truth labels for each tile determined from the label of the corresponding slide. **b** When evaluating a tile-based model, image tiles do not need to be buffered into TFRecords. Image tiles can be extracted from slides, processed and stain normalized, and predictions are generated for each tile from a slide. The final slide-level prediction is the average prediction from all tiles. Tile-level predictions can be visualized as a heatmap

Training weakly-supervised, tile-based models

Slideflow includes several tools for training deep learning classification, regression, and time-to-event models for WSIs. Weakly-supervised, tile-based models use a deep learning image classifier to generate predictions for all tiles in a slide, and the final slide-level prediction is calculated by averaging tile predictions (Fig. 3). Although this method may be limited with highly heterogeneous tumors or when salient morphological features are sparse, it has proven to be an effective approach for a variety of biological applications [6, 37–39]. In addition to averaging, Slideflow supports calculating slide- and patient-level predictions using a variety of aggregation functions, as well as arbitrary user-defined functions.

Slideflow includes dataset organization tools to easily support the standard train, validate, test paradigm used for biomarker development. A held-out test set is first designated, either manually or using one of several tools for dataset splitting. Several approaches can be used for separating the remaining dataset into a training and validation splits, including fixed splitting, bootstrap, *k*-fold cross-validation, and site-preserved *k*-fold cross-validation [40].

Model architecture, loss function, and training hyperparameters are then configured; a list of included pre-configured model architectures is provided in Additional file 1: Table 1. Slideflow also supports custom models and loss functions using either Tensorflow or PyTorch. Models can be trained to single categorical, multi-categorical, continuous, or time-to-event outcomes. Multi-modal models can also be trained with additional arbitrary input, such as clinical variables, using late multimodal fusion via concatenation at the post-convolutional layer. Hyperparameters can be optionally tuned using either grid-search sweeps or Bayesian hyperparameter optimization, which uses the SMAC3 [41] package. Hyperparameter search spaces can be easily customized, and several pre-configured search spaces are provided.

During training, images can undergo augmentation to broaden the training domain and promote generalizability (Fig. 1C). Images can undergo random cardinal rotation and random horizontal or vertical flipping, random JPEG compression,[5] and random Gaussian blur,[6] and stain augmentation. When using Tensorflow, images are standardized to a mean of 0 and standard deviation of 1, and when using PyTorch, images are standardized to a range of 0–1.

By default, mini-batch balancing is used to ensure equal representation of all slides in each batch and equal representation of all classes (Additional file 1: Fig. 2). Throughout training, slides and classes will be oversampled to enable this balancing. This mini-batch balancing can be customized or disabled. During training, one epoch is defined as the total number of image tiles available in the training dataset divided by the batch size. The user can specify the interval at which validation checks are performed in the middle of an epoch. Early stopping can be configured to trigger when the exponential moving average of either loss or accuracy plateaus, or models can be trained for a prespecified number of epochs. Training can be distributed across multiple GPUs for computational efficiency. Training progress is monitored locally using Tensorboard [42] or remotely using Neptune.ai [43].

Three pretrained classification models, trained on thyroid cancer (BRAF-RAS gene expression score), breast cancer (Estrogen Receptor [ER]-positive vs. ER-negative), and lung cancer (adenocarcinoma vs. squamous cell carcinoma), have been made available on Hugging Face [44–46].

Evaluating weakly-supervised tile-based models

After training, performance metrics will be automatically calculated from the validation dataset at the tile-, slide- and patient-levels, including accuracy, area under receiver operator curve (AUROC), and average precision (AP). Slide-level predictions are calculated by averaging the one-hot predictions for all tiles from a slide, and patient-level predictions are calculated by averaging all tile-level predictions across all slides from a given patient. Tile-, slide-, and patient-level predictions are saved during both training and evaluation, allowing the researcher to calculate custom metrics if desired. Saved models can also be applied to held-out test sets, calculating test-set metrics, or to single slides for inference.

Predictive heatmaps are generated by overlying tile-level predictions onto a slide and can also be generated for either a single slide or multiple slides. Heatmap calculation is accelerated by parallelized tile extraction with multiprocessing. Heatmaps can be rendered and exported as images or interactively viewed with real-time navigation in Slideflow Studio, the whole-slide GUI.

Uncertainty quantification

Estimation of confidence and uncertainty is essential for medical AI applications, as reliable uncertainty quantification can facilitate clinical decision-making and potentially improve patient safety [47, 48]. Slideflow enables uncertainty estimation using the Monte

[5] 10% chance of compression at a random quality level between 50–100%

[6] 50% chance of Gaussian blur with sigma between 0.5–2.0.

Carlo dropout paradigm [49]. With this approach, models are built with dropout layers enabled during both training and inference. During inference, a single image undergoes multiple forward passes in the network, with the resulting distribution representing the final prediction (mean) and uncertainty (standard deviation). Tile-level uncertainty can be translated into slide-level uncertainty and used for subsequent confidence thresholding, using our previously described uncertainty thresholding algorithm [31]. Briefly, from a given set of validation predictions, an optimal uncertainty threshold is determined, below which predictions are more likely to be correct compared to predictions with higher uncertainty. Predictions with uncertainty above this threshold are then discarded. This thresholding is performed first at the tile level and then at the slide level. Thresholds are determined for a given training dataset using nested cross-validation, to prevent data leakage. This uncertainty estimation and confidence thresholding approach improves accuracy for high-confidence predictions and guards against domain shift.

Image features and feature space analysis

Converting images into feature vectors provides an avenue for feature space analysis and more advanced, whole-slide classification models such as multiple-instance learning (MIL). Three feature generation methods are provided for processing image tiles into numerical vectors: pretrained networks, finetuned classifiers, and self-supervised learning (Fig. 4). For all methods, features can be calculated for single image tiles, a single slide, or image tiles read from TFRecords.

Several pretrained networks can be used for converting images into feature vectors. Slideflow includes an API for calculating layer activations at any arbitrary neural network layer for models pretrained on ImageNet, by specifying an architecture name and layer name. If multiple layers are specified, activations at each layer will be calculated and concatenated. An API is also included for easily generating features using the pretrained, pathology-specific CTransPath [50] and RetCCL [51] networks. Finetuned classifiers trained with Slideflow can also be used for feature generation, calculating activations at any arbitrary neural network layer. Finally, an API is included for easily training the self-supervised learning model SimCLR, providing another avenue for feature generation through contrastive pretraining.

Once features are calculated for a dataset of image tiles, several tools are provided for flexible dimensionality reduction and feature space visualization using UMAP [52]. UMAP plots can be quickly generated and labeled with outcomes, clinical variables, and suspected confounders such as site. Corresponding image tiles can be overlaid onto the UMAP plots to aid in interpretability, resulting in visualizations we refer to as mosaic maps. Mosaic maps are generated by separating a given UMAP projection into a grid (default 50×50), and for each grid space, plotting the corresponding image tile for any of the points in the grid space. Tile selection within a grid space can either be random (default), or the tile nearest to the centroid for that grid space can be chosen. Mosaic maps can be viewed alongside labeled UMAPs, providing insights into the relationship between morphologic image features, the outcome of interest, and other relevant clinical factors. This process can also help identify potential sources of confounding and bias. Mosaic maps can be exported as a high-resolution figure or viewed interactively with Slideflow Studio.

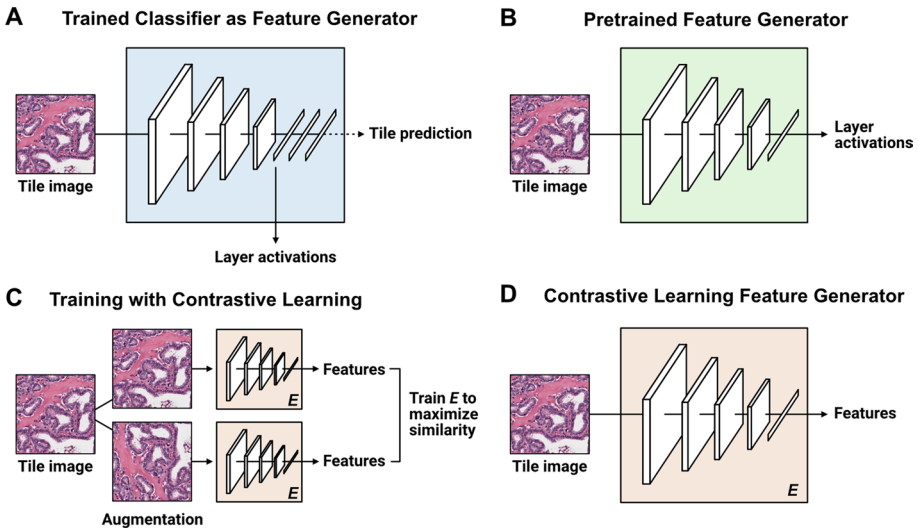

Fig. 4 Feature generation methods. **a** Trained classifiers can be converted into feature generators by specifying a neural network layer and calculating activations at the given layer. **b** Several pretrained models can be used as feature generators, including non-pathology pretrained models (*e.g.* ImageNet), or pathology-specific pretrained models (*e.g.* CTransPath). **c** Self-supervised contrastive learning (SimCLR) can be used to train a feature generator without requiring ground-truth labels for classification. **d** Features can be calculated from a model trained using self-supervised learning

Weakly-supervised multiple-instance learning

Tile-based whole-slide classification models may not be well-suited for datasets where relevant histopathological features are expected to be sparse or heterogenous in a slide, or when pathologist-annotated regions of interest are unavailable. Attention-based multiple-instance learning (MIL) models provide an avenue for weakly-supervised whole-slide classification that is theoretically still robust when relevant features are sparse or regions of interest are unavailable [53]. These models generate predictions from bags of image tile feature vectors, and models will learn to ignore uninformative image tiles through attention weighting. The performance of these models is in part dependent upon the quality of features calculated from image tiles.

Three types of MIL models can be trained in Slideflow, including traditional MIL [54], attention-based MIL [55], and clustering-constrained attention-based MIL (CLAM) [18]. An API is provided for fast conversion of image tiles into feature vectors and training of MIL models from these generated features. Training is executed using the FastAI framework [56], including options for either cosine annealing learning rate scheduling or one-cycle learning rate scheduling [57]. For CLAM, an option is provided to use the originally published training loop instead of the FastAI trainer, if desired. Trained MIL models can be used to generate attention heatmaps for WSIs, highlighting areas of the slide weighted with high attention. As with predictive heatmaps for tile-based classification models, attention heatmaps can be generated and exported as high-resolution images.

Generative adversarial networks

A growing body of evidence is showing the potential utility of Generative Adversarial Networks (GANs) for histopathological applications. GANs can reproduce realistic

synthetic histology images that have been used for training augmentation, stain and color normalization, image enhancement, and explainability. Slideflow includes an API for training and using StyleGAN2 [58] and StyleGAN3 [59] with optional class conditioning from image tiles saved in TFRecords. Model architectures, training paradigms, and configuration options are all equivalent to their original implementations. The API provides a method for training these GANs from preprocessed images already stored in TFRecords, without requiring additional data processing or alternative formatting, and additionally provides an interface for calculating predictions from generated images using a trained classifier.

For class-conditional GANs, several tools are provided for generating images from intermediate classes using feature space embedding interpolation. To generate an image in between two classes, we calculate the associated class embedding for each class, and then perform a linear interpolation to achieve an intermediate embedding. These intermediate embeddings can be used for class or layer blending applications [60]. Images generated from trained GANs can be exported as raw PNG or JPG images, saved in TFRecord format, or visualized in real-time using Slideflow Studio.

Three pretrained conditional GANs, trained on thyroid cancer (conditioned on BRAF-like vs. RAS-like gene expression), breast cancer (conditioned on Estrogen Receptor [ER]-positive vs. ER-negative), and lung cancer (conditioned on adenocarcinoma vs. squamous cell carcinoma), have been made available on Hugging Face [61–63].

Model explainability

Explainable artificial intelligence (XAI) approaches are an increasingly important component of medical imaging research. These techniques can provide insights into what image features models have learned, support biological plausibility of model predictions, and improve model trust [13]. Slideflow includes four methods for model explainability: heatmaps, mosaic maps, gradient-based pixel attribution (saliency maps), and conditional generative adversarial networks (cGANs).

Heatmaps, including both predictive and attention heatmaps, provide an avenue for quickly assessing the areas of a WSI relevant to the final prediction. Described above, heatmaps can be calculated for a single slide or a dataset of multiple slides, and either exported as a high-resolution figure or interactively viewed.

Mosaic maps provide a tool for feature space exploration at arbitrary neural network layers. With pathologist interpretation, they can highlight morphologic associations between outcome variables and offer insights into the spatial relationship of image features among classes.

Gradient-based pixel attribution approaches highlight the pixels in an image that were relevant for a given neural network model prediction. These attribution heatmaps, or saliency maps, can be calculated through a variety of provided methods, including Grad-CAM [64], vanilla gradients [65], integrated gradients (and variations thereof) [66], and XRAI [67]. Saliency maps can be displayed as raw heatmaps or as overlays onto the associated image. Utility functions are included for rapid comparison between different saliency map methods. Saliency maps can also be interactively viewed for WSIs in real-time using Slideflow Studio.

cGANs offer a dataset-level explainability approach for trained neural networks models, using the paradigm we have recently described [60]. The synthetic histology generated by cGANs can illustrate morphologic features associated with outcome classes and can provide insights into the importance of larger histopathological features not amenable to localization with saliency maps, such as differences in architecture, stroma, colloid, and staining patterns.

Tissue and cell segmentation

Slideflow also provides tools for building and using tissue and cell segmentation models. Binary and multiclass tissue segmentation models can be trained from labeled regions of interest using a variety of architectures, including U-Net, DeepLabV3, FPN, and others, as implemented by Iakubovskii [68]. During training, whole-slide thumbnails and paired regions-of-interest masks at a user-defined microns-per-pixel magnification are augmented with random cropping, flipping, and cardinal rotation. Once trained, tissue segmentation models can be deployed for generating regions of interest or used for slide-level masking and quality control, via either a programmatic interface or Slideflow Studio. A pretrained tumor identification tissue segmentation model, trained on 8,122 slides and paired pathologist-annotated regions of interest from The Cancer Genome Atlas [69], is available on Hugging Face [70]. Example tumor regions of interest generated from this model on an external dataset of head and neck cancer whole-slide images from the University of Chicago is shown in Additional file 1: Fig. 3.

Whole-slide cell identification and segmentation is performed using Cellpose [71]. Both pretrained and user-trained models can be used to generate cell segmentation masks and centroids using either a programmatic interface or Slideflow Studio. Cell diameter is configured in microns, rather than pixels, to improve generalizability across slides with different magnifications. Cell segmentation is applied to whole-slide images in tile-wise fashion, with segmentation masks stitched together to form a whole-slide mask. Masks and centroids can be used to guide tile extraction, so that each extracted tile represents a single detected cell/nucleus, or be exported in NumPy format for downstream use and further analysis.

Whole-slide visualization with slideflow studio

Slideflow includes a visualization tool, Slideflow Studio, for interactively viewing WSIs, focal predictions, heatmaps, saliency, uncertainty, and mosaic maps. Slide imaging data is read using either cuCIM or VIPS, accelerated with multiprocessing, and rendered using OpenGL. Slide processing settings, such as Otsu's thresholding, grayspace filtering, stain normalization, etc., can be previewed in real-time to assist with fast determination of optimal slide processing parameters (Additional file 1: Fig. 4). ROI annotations can be loaded, edited, and added with a lasso selection tool.

Once a model is loaded, right-clicking anywhere on the slide will reveal a preview window showing an image tile extracted at this location, before and after stain normalization (if applicable) (Fig. 5). Saliency maps generated through gradient-based pixel attribution can also be displayed as a heatmap or as an overlay on the extracted image tile. Tile-level predictions and uncertainty are shown in the control panel.

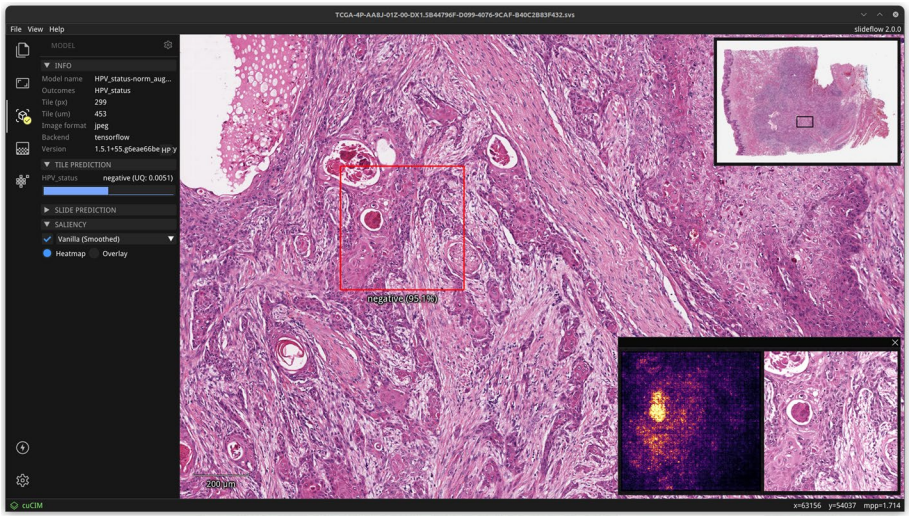

Fig. 5 Real-time predictions and saliency maps from whole-slide images. Slideflow Studio provides an interface for generating both whole-slide and focal tile predictions. Right clicking on an area of the screen extracts a tile at the given location, showing the tile before and after stain normalization (if applicable). Gradient-based pixel attribution methods can be used to render saliency maps for the given image, which can be shown as a heatmap (shown here) or as an overlay (not shown)

Whole-slide predictive heatmaps can be calculated, customized, and displayed for slides once a model is loaded. A "low memory" mode can be enabled in Performance Settings, reducing memory consumption at the cost of slower heatmap calculation. Final slide-level predictions (and uncertainty, if applicable) will be shown in the control panel after the heatmap is calculated.

Mosaic maps can also be interactively viewed in Slideflow Studio (Additional file 1: Fig. 5). Compared with a static figure, this interface permits closer inspection of image tiles at higher resolution through interactive zoom and enables dynamic modification of the mosaic grid size. A popup window in the bottom-right corner shows the user the corresponding UMAP plot, with a red box indicating the current section of the plot in view. Hovering over an image tile will show a larger section from the corresponding slide at that tile location, revealing the surrounding histologic context.

Programmatic interface

Slideflow can be installed via the Python Packaging Index (PyPI), and pre-built Docker containers are available for easier dependency management.

Once installed, the first step to using Slideflow is creating a Project to facilitate organization of raw data, processed data, saved models, and experimental results. Projects are created and loaded using the slideflow.Project class. Locations of whole-slide images, pathologist ROI annotations, and processed TFRecords are defined through a dataset configuration JSON file associated with the Project. Ground truth patient- and slide-level diagnoses and clinical outcomes can be provided either via a CSV file or a Pandas DataFrame. Several example preconfigured projects for lung cancer, breast cancer, and thyroid cancer are provided for testing and functionality exploration; when used,

whole-slide images and associated clinical annotations will be automatically downloaded from public repositories.

Most functions—such as training classification models, self-supervised learning, feature generation, or GAN training—require processed image tiles extracted from whole-slide images at a given magnification and pixel resolution. Data organization and processing is supervised by the slideflow.Dataset class, an instance of which can be generated by using the Project.dataset() method and specifying the desired magnification and target pixel size. Clinical annotations associated with a Project are automatically passed to the dataset and can be used for dataset filtering. Datasets can be easily split into training, test, and validation subsets with Dataset.split(). Whole-slide image processing and tile extraction is performed using the Dataset.extract_tiles() method, with generated TFRecords automatically organized and stored according to the Project dataset configuration. Once tiles are extracted, a tensorflow.data.Dataset or torch.utils.data.DataLoader that dynamically reads and processes imaging data from TFRecords can be easily created using the Dataset.tensorflow() and Dataset.torch() methods, respectively.

Classification models are configured using slideflow.ModelParams, a class which defines the model architecture and training hyperparameters. To train a model, use Project.train(), providing a ModelParams object, a training and validation Dataset, and the column name that contains the ground-truth labels in the clinical annotations. Held-out test sets can be assessed using Project.evaluate(), and predictions can be generated for slides with Project.predict().

The online documentation includes further details about training, including step-by-step tutorials, instructions for the use of custom models and loss functions, and training strongly-supervised models with ROI labels. The documentation also contains further descriptions of the programmatic interface for Project configuration dataset management, multiple-instance learning, feature space analysis, generative adversarial networks, and segmentation [30].

Supported hardware and software environments

Slideflow is a Python package that requires Python 3.7 or greater, and either Tensorflow or PyTorch. Some features, such as GANs and tissue segmentation, require PyTorch. Model training requires a Linux-based operating system, such as Ubuntu or RHEL/CentOS, and is greatly accelerated with a dedicated GPU. Trained models can be deployed using the interactive interface on a variety of operating systems and hardware environments. Testing of this user interface has been performed on Linux-based systems, Windows, macOS (Intel and Apple chip), and ARM-based devices such as the Raspberry Pi and Jetson Nano. A dedicated GPU is recommended, but not required, for model deployment.

Software development processes

Slideflow has been engineered according to industry software quality standards. Source code adheres to PEP 8 style guidelines [72], supported through integrated pylint [73] checking. Functions and classes include Google-style docstrings [74] for documentation, along with type annotations leveraging mypy [75].

The public GitHub repository serves as the platform for issue tracking, feature requests, and community contributions. Pull requests require approval before merging into the master branch, which triggers continuous integration testing on GitHub Actions including flake8 linting. Slideflow additionally includes an integrated test suite, with both unit tests and functional tests covering core data processing, whole-slide image processing, TFRecord manipulation, stain normalization, model training, heatmap and mosaic map generation, and feature space analysis. The online documentation is built from the repository using Sphinx, leveraging code documentation to ensure it stays current with ongoing development work.

Altogether, these processes promote stability, continuity, transparency, and reliability as new capabilities are added, facilitating long-term maintainability across future releases.

Results

The variety of tools offered by Slideflow have been utilized for many diverse research objectives, including gene expression prediction [6], prognostication [76], uncertainty quantification [31], identification of bias and batch effects [40], drug response prediction [77], and generation of synthetic histology for model explainability [60]. In order to illustrate some of these tools on a real dataset, we will present results on a benchmark dataset for Human Papilloma Virus (HPV) status prediction in head and neck squamous cell carcinoma. The training/validation dataset is comprised of 262 patients (151 HPV-negative, 111 HPV-positive) from the University of Chicago, and the held-out external dataset is comprised of 459 patients (407 HPV-negative, 52 HPV-positive) across 26 sites from The Cancer Genome Atlas (TCGA). All patients had one associated WSI.

Slide processing

Pathologists annotated ROIs encircling areas of tumor for all slides, except in cases where the entire sample was determined to be tumor. Otsu and Gaussian blur filtering were explored for this dataset. In some cases, Otsu's thresholding highlighted pen marks as foreground tissue, so both background filtering methods were used in addition to ROIs for all slides. Image tiles were extracted at 299 pixels and seven micron sizes ranging from 76 μm (40x magnification) to 1208 μm (2.5x). Using the cuCIM backend, tile extraction speed ranged from 614 to 2091 tiles/second (0.38–25 slides/second) (Fig. 6, A and B). Tile extraction was also performed using VIPS for comparison, with tile extraction speed ranging between 198 and 1300 tiles per second (0.24–11.5 slides/second). Otsu's thresholding added 0.30 ± 0.12 s per slide, and Gaussian blur filtering added 0.50 ± 0.93 s per slide. Tile extraction with cuCIM was 1.6–3.3 times faster than VIPS (Fig. 6A, B). Example pages from the associated tile extraction PDF reports are shown in Additional file 1: Figs. 6 and 7. TFRecord buffering permitted dataset iteration at 11,453 images/second using Tensorflow and 6,784 images/second using PyTorch. Buffered dataset sizes are shown in Fig. 6C.

Experimentation with tile-based background filtering was performed for comparison but not used for downstream analysis (Fig. 7). Otsu's thresholding identified $6.4\% \pm 8.2\%$ more background tiles than grayspace filtering. Image tiles removed by Otsu's thresholding but not grayspace filtering typically included edge tiles or images

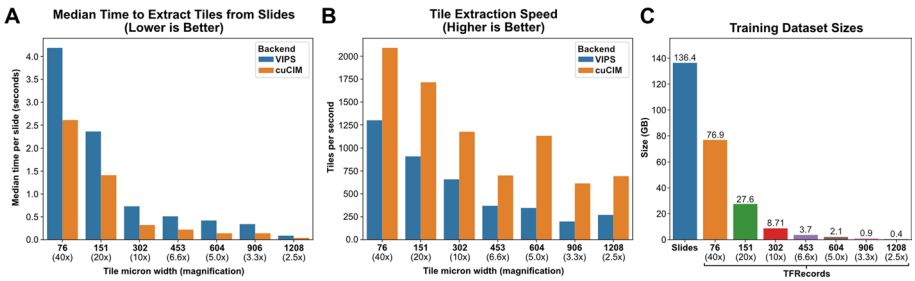

Fig. 6 Whole-slide image processing speed and dataset sizes/ **a** Median time required to extract tiles from WSIs using each of the two available slide processing backends. **b** Average tile extraction speed from WSIs, using each slide processing backend. **c** Size of the full training dataset as raw slides and in buffered TFRecords, at each assessed magnification size. Image tiles were stored in TFRecords using JPEG compression with 100% quality. Benchmarks were performed using an AMD Threadripper 3960X

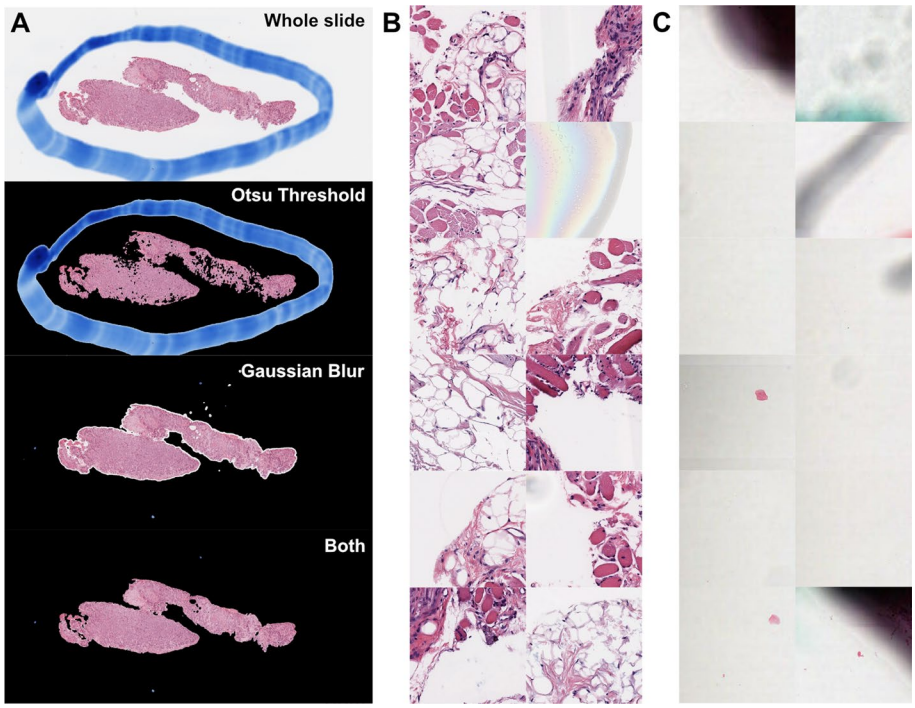

Fig. 7 Comparison between background filtering methods. **a** Comparison between slide background filtering methods. An example whole-slide image with large pen mark artifact is shown, with associated background filtering masks applied. Black areas indicate masked background. With Otsu's thresholding alone, the pen mark is identified as foreground, and several parts of the tissue area are erroneously removed as background. Gaussian blur filtering removes the pen mark, but some smaller areas of background near edges are not removed. Performing Gaussian blur first, followed by Otsu's thresholding, results in the most accurate background identification and removes the pen mark. **b** Example images identified as background using Otsu's thresholding, but not when using grayspace filtering. **c** Example images identified as background using grayspace filtering, but not when using whitespace filtering

with heterogenous background content, such as fat or stroma (Fig. 7B). With rare exception, all image tiles removed by Otsu's thresholding were also removed with grayspace filtering. For 85.2% of slides, there was a < 5% difference in background identified by whitespace filtering and grayspace filtering. In 12.6% of cases, grayspace

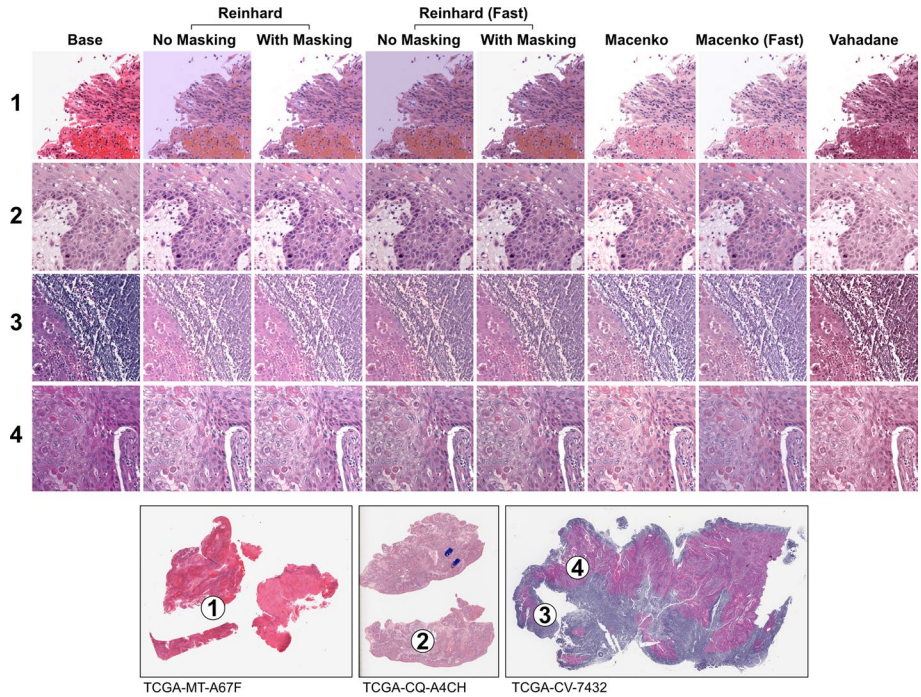

Fig. 8 Comparison between stain normalization methods. Four image tiles from whole-slide images with different staining patterns are shown before and after stain normalization. Three stain normalizers are compared: Reinhard, Macenko, and Vahadane. The Reinhard normalizer has a standard and fast variant (with brightness standardization disabled), and a masked and unmasked variant. Similarly, the Macenko also has a standard and fast variant

filtering removed a median of 10.2% more background than whitespace filtering (Fig. 7C), and in 2.2% of cases, whitespace filtering failed to remove any background tiles.

Three stain normalization methods with several variations were compared, as shown in Fig. 8. The fast variants of the Reinhard methods produced similar results to the standard Reinhard variants, with slight differences in perceived brightness. The masked and unmasked Reinhard variants yielded similar images in most contexts, but with the unmasked variants producing pink-tinted background for image tiles containing high background content. Compared with standard Macenko normalization, context-aware Macenko normalization generally resulted in images with higher perceived contrast but occasionally washed out fine details in bright areas, such as areas containing fat (Additional file 1: Fig. 8). Macenko stain augmentation yielded realistic, artifact-free images with diverse staining hues (Additional file 1: Fig. 9).

Computational efficiency was compared between methods using both Tensorflow and PyTorch, with results shown in Fig. 9. Removal of the brightness standardization step improved peak stain normalization speed by 272% for the Reinhard normalizer and 51% for the Macenko normalizer when using Tensorflow, and 26% and 24% for Reinhard and Macenko normalizers when using PyTorch. Reinhard normalizers exhibited superior computational performance when processed on the GPU, and Macenko normalizers were faster when processed on CPU. The utility of the Vahadane algorithm is limited by long processing times, and could not be used for real-time normalization. Based on a

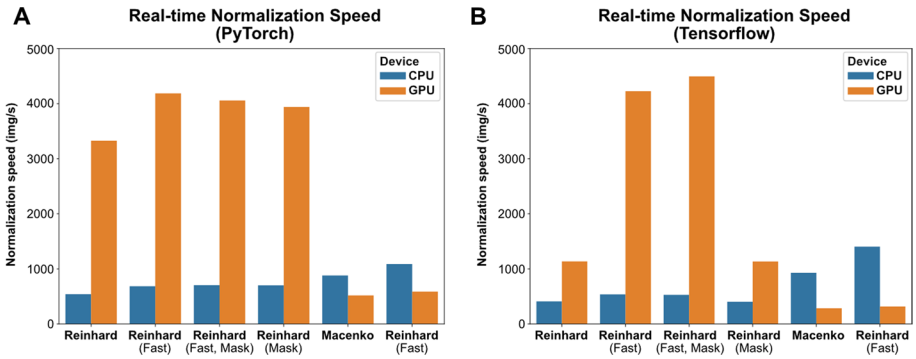

Fig. 9 Real-time stain normalization benchmarks. Stain normalization speed was assessed for each normalization method and device type (CPU, GPU) using both Tensorflow and PyTorch. **a** Benchmark results using PyTorch. **b** Benchmark results using Tensorflow. All benchmarks were obtained using an AMD Threadripper 3960X CPU and A100 40 GB GPU

combination of qualitative assessment and computational efficiency, the standard Macenko stain normalization strategy was chosen for subsequent analyses. Real-time stain augmentation was used during model training.

Weakly-supervised tile-based classification

Weakly-supervised, binary classification models were trained on the institutional dataset of 262 slides to predict HPV status using three-fold cross-validation and the Tensorflow backend. As a first step, models were trained on the first cross-fold at seven magnification levels between 2.5x and 40x using the Xception architecture and a single set of hyperparameters (Fig. 10). The best performance was seen at 6.6X (tile micron width of 453), so this magnification level was used for subsequent analysis. Hyperparameters were tuned on the first cross-fold with Bayesian hyperparameter optimization using the "shallow" search space configuration, a maximum of 50 iterations, and 5 replicate models trained for each hyperparameter combination. Average training time for each model was 2 min 11 s. Ten models were trained using the best performing hyperparameter

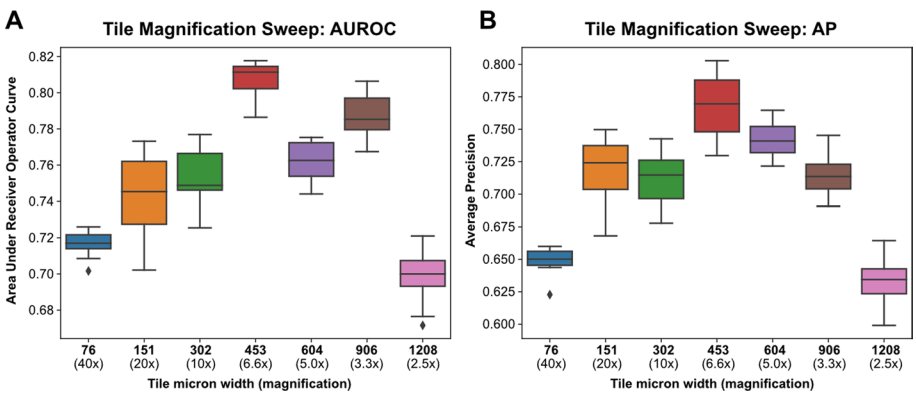

Fig. 10 Tile magnification optimization. Image tiles were extracted from slides in the training dataset at seven magnification sizes, ranging from 2.5x (tile width 1208 microns) to 40x (tile width 76 microns). Using the first training cross-fold, ten replicate models were trained at each magnification size using random weight initialization. **a** Area Under Receiver Operator Curve (AUROC) for models trained at varying magnifications. **b** Average Precision (AP) for models trained at varying magnifications

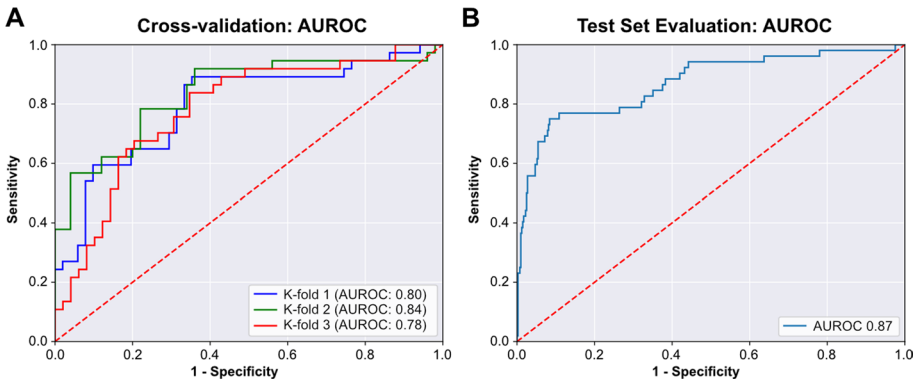

Fig. 11 Area Under Receiver Operator Curve (AUROC) for cross-validation and the held-out test set. **a** AUROC for the three models trained in cross-validation on the single institution, University of Chicago dataset. **b** After cross-validation, a final model was trained on the full University of Chicago dataset. This model was then evaluated on the held-out test set from The Cancer Genome Atlas (TCGA). AUROC for this test set evaluation is shown

combination and compared with ten models trained using the starting hyperparameters. AUROC was not improved using the optimized hyperparameters (0.808 vs. 0.803, P = 0.11), so the initial hyperparameters were used for subsequent analysis (Additional file 1: Table 2). AUROC across the three cross-folds was 0.80, 0.84, and 0.78, with AP of 0.79, 0.83, and 0.77 (Fig. 11). Dropout-based uncertainty quantification was used for confidence threshold determination. A final model was then trained across the full dataset using the previously determined optimal hyperparameters and the Tensorflow backend. When validated on the external test set comprised of 459 slides from TCGA, the model resulted in an AUROC of 0.87 and AP of 0.80. Using uncertainty quantification and confidence thresholding, 89.5% of slides had high-confidence predictions, with an AUROC of 0.88 within high-confidence predictions. Using prespecified prediction and uncertainty thresholds determined from cross-validation, the final model had a test-set accuracy of 90.2%, sensitivity of 77.0%, and specificity of 92.0% within high-confidence predictions.

After the final model was trained and evaluated on the external test set, a post-hoc analysis was performed to assess the impact of stain augmentation on model performance. Ten replicate models were trained with and without stain augmentation on the full University of Chicago training dataset and then evaluated on the TCGA external test set. This post-hoc analysis demonstrated that stain augmentation resulted in a small but statistically significant improvement in AUROC on the test set, from 0.871 ± 0.008 to 0.882 ± 0.014 ($P = 0.021$, one-sided t-test).

Multiple-instance learning

In order to train multiple-instance learning (MIL) models, image tiles were first extracted from whole-slide images at 6.6X magnification (453 µm width) and normalized with Macenko normalization. Pathologist-annotated ROIs were not used for MIL experiments. Image features were then calculated for all extracted image tiles using the pretrained model CTransPath [50]. MIL models were built using the CLAM single-branch architecture [18] and trained using a single set of default hyperparameters (Additional

file 1: Table 3). Models were trained for a total of 20 epochs. Three-fold cross-validation AUROC for the MIL models was 0.77, 0.78, and 0.79. A final model was trained on the full University of Chicago dataset without validation, and then tested on the TCGA test set. On this held-out test set, the final model had an AUROC of 0.81 and AP of 0.77.

Feature space analysis

Features for the weakly supervised tile-based model were generated and visualized for both datasets using the included feature generation interface. Features are generated through calculation of post-convolutional layer activations for all image tiles, and the resulting feature space is visualized through UMAP dimensionality reduction. UMAP plots of the TCGA and University of Chicago feature spaces show good class separation between HPV-negative and HPV-positive images for both the training and test data (Fig. 12, A and B). The mosaic map for the University of Chicago feature space highlights known biologically-relevant image features associated with HPV status (Fig. 12E). Area 1, enriched for HPV-positive images, shows image tiles with tightly packed cells with scant cytoplasm and surrounding inflammation. Area 3, enriched primarily with HPV-negative images, shows heavy keratinization along with pleomorphic cells with increased cytoplasm. Findings in both of these areas are consistent with known histopathological associations [78, 79]. Area 2, an intermediate zone with both HPV-positive and HPV-negative images, shows image tiles with keratinization, inflammation, and cells with varying cytoplasmic content. Together, this feature space analysis supports the biological plausibility of model predictions.

Features from CTransPath, which were used for training the multiple-instance learning model, were generated and visualized using the included feature generation interface. Separation between HPV-negative and HPV-positive image tiles on the University of Chicago training dataset using CTransPath features is less clear than when using features from the tile-based model (Fig. 12C). On the test dataset from TCGA, there is no clear separation between HPV-positive and HPV-negative image tiles with this visualization (Fig. 12D). This poorer separation between HPV-positive and HPV-negative images in the feature space may help partially explain the discrepancy in performance between the tile-based and MIL models.

Explainability

Heatmaps of model predictions were generated from the final tile-based model at an average rate of 22.1 ± 4.2 s per slide. Example heatmaps of predictions and uncertainty are shown in Fig. 13. In general, areas with low uncertainty and strong predictions for negative HPV status demonstrated high keratinization and cells with pleomorphic nuclei. Low-uncertainty areas with strongly positive HPV status predictions tended to show tightly packed cells with monotonous nuclei and surrounding inflammatory infiltrate. Both of these observations are consistent with known histopathological associations with HPV status [78, 79]. Attention heatmaps were also generated using the final MIL model, as shown in Additional file 1: Fig. 10. Areas with strong, high-confidence predictions from the tile-based model were generally also weighted with high attention

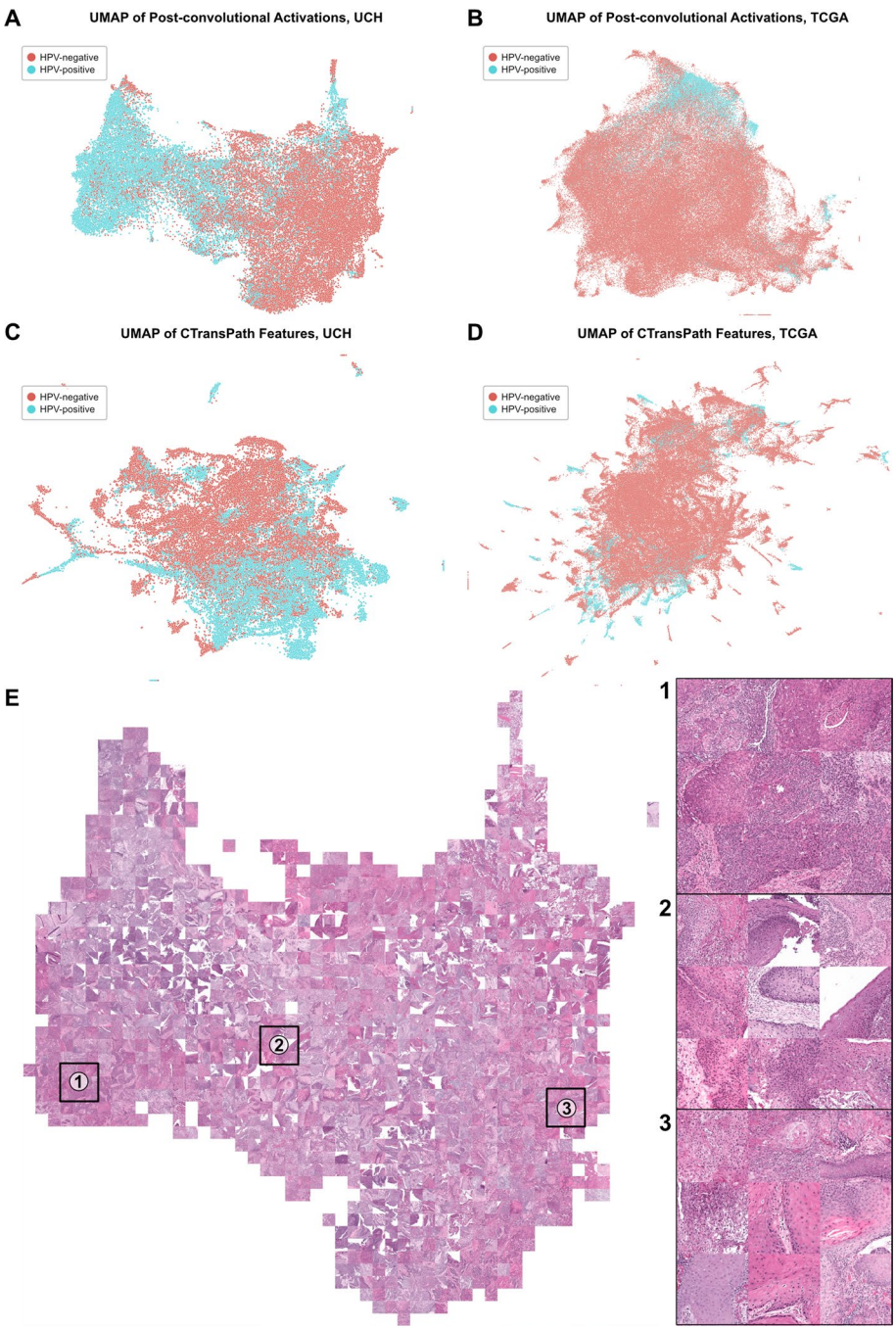

Fig. 12 Feature space visualization for the tile-based and multiple-instance learning models. **a** UMAP plot of post-convolutional layer activations, calculated using the final trained model, for all images in the University of Chicago training dataset. **b** Same as (**a**), but calculated for the TCGA test set. **c** UMAP plot of CTransPath features for all images in the University of Chicago training dataset. **d** Same as (**c**), calculated for the TCGA test set. **e** Mosaic map generated from the UMAP plot shown in (**a**). Three areas are magnified for closer inspection. Area 1 is enriched for HPV-positive images, Area 2 is in a zone of transition between HPV-positive and HPV-negative images, and Area 3 is enriched with HPV-negative images. Image tiles are shown using Macenko stain normalization

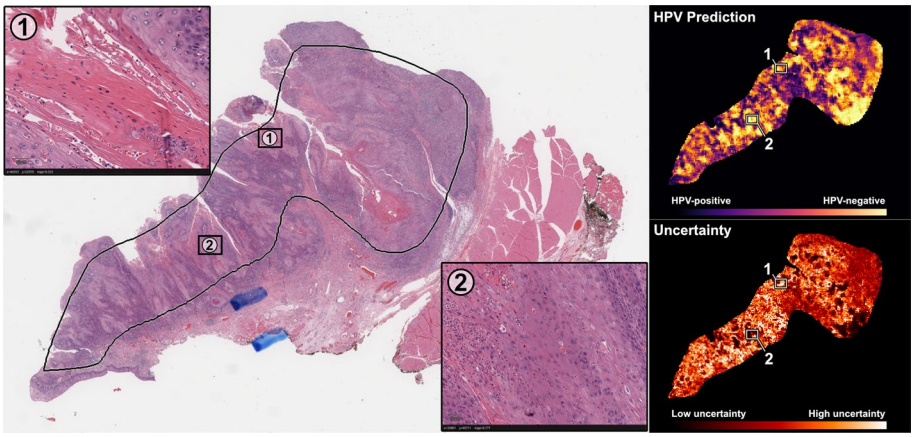

Fig. 13 Prediction and uncertainty heatmaps on an example slide in the external test set. Heatmaps of HPV predictions and uncertainty were generated from the final tile-based model for a randomly selected slide from the test set. Two areas are shown with magnified display. Area 1 is a location with intermediate predictions and high uncertainty, showing mostly stroma and out-of-focus cells in the top-right corner. Area 2 is a location with strong HPV-negative predictions and low uncertainty, showing heavy keratinization and pleomorphic nuclei

from the MIL model. A screenshot of the whole-slide user interface with a loaded heatmap is shown in Fig. 14.

Saliency maps were generated for an example correctly predicted HPV-negative image tile using vanilla gradients, three variations on integrated gradients, and XRAI (Fig. 15A). For this example, all saliency maps highlight the section of the image tile with heavy keratinization, a histopathological factor known to be associated with HPV negativity. Finally, a conditional generative adversarial network (cGAN) was trained on the institutional dataset, conditioned on HPV status, to provide generative explanations for the trained classifier through synthetic histology [60]. The StyleGAN2 architecture was

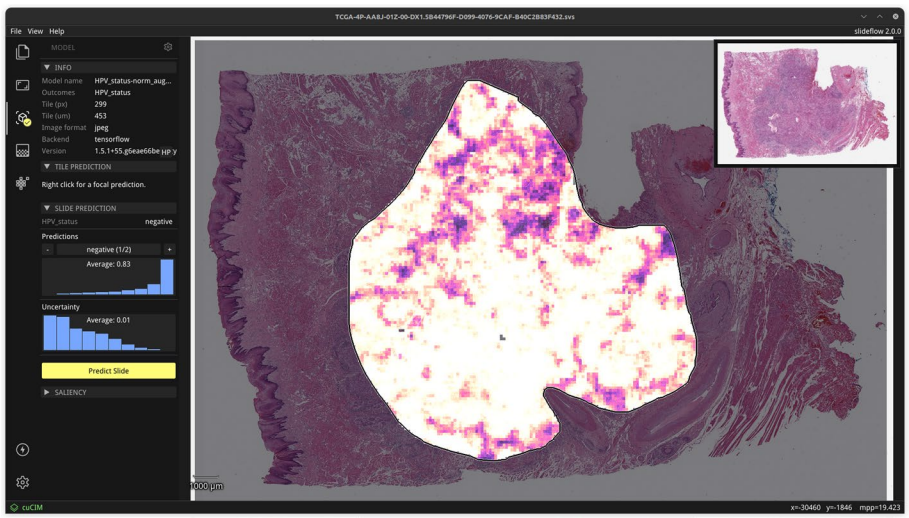

Fig. 14 Interface for viewing and navigating whole-slide heatmaps. Slideflow Studio includes an interface for generating whole-slide predictions and heatmaps, as shown in this figure. Heatmaps can be viewed interactively and exported as both PNG images and Numpy arrays. Heatmap color and display options are customized in another tab of the interface (not shown)

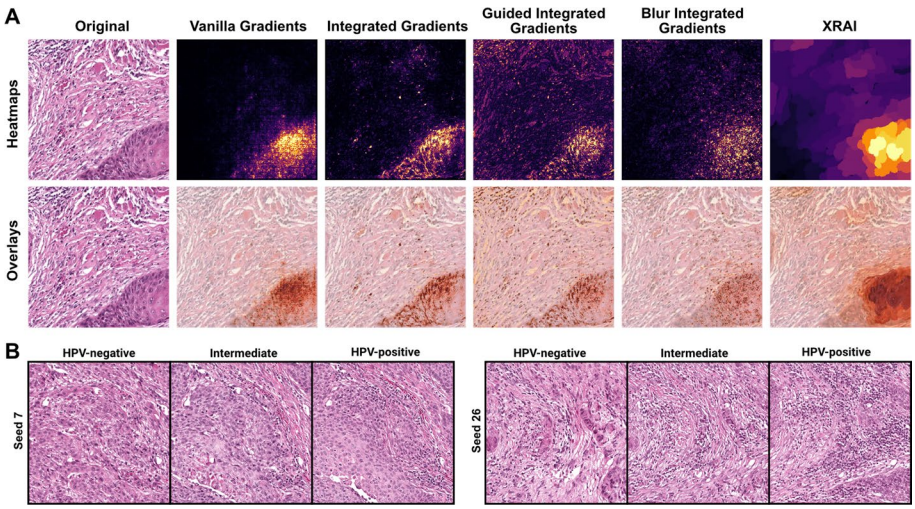

Fig. 15 Example of model explanations using saliency maps and generative adversarial networks. **a** Various gradient-based pixel attribution methods with APIs available in Slideflow were used to generate saliency maps for an example HPV-negative image. **b** A class-conditional GAN based on the StyleGAN2 architecture was trained using Slideflow to generate synthetic histology images belonging to HPV-negative and HPV-positive negative classes, visually highlighting morphologic differences between classes

used, using default hyperparameters from the original implementation [58]. Training was stopped after 15 million images due to divergence with further training. Training took 38 h on four A100 GPUs. Visualizations generated with this method highlighted differences in keratinization and nuclear pleomorphism, which is increased in the synthetic HPV-negative images, and inflammatory infiltrate, increased in synthetic HPV-positive images (Fig. 15B). These differences are consistent with known pathologic associations with HPV status, further supporting the biological plausibility of learned image features [78, 79].

Hardware deployment

The above experiments and benchmarks were performed on a Linux-based workstation with an AMD Threadripper 3960X CPU and NVIDIA A100 40 GB GPU. A breakdown of the total time required to train and evaluate a classification model on this hardware is provided in Additional file 1: Fig. 11. To assess the feasibility of using the whole-slide graphical interface as a deployable tool for WSI analysis, Slideflow Studio was deployed and tested on various Linux workstations, a Windows 10 desktop (with dedicated GPU), an Intel MacBook Pro, an M2 MacBook Pro, a Raspberry Pi 4 (4 GB), and a Jetson Orin Nano. All devices ran Slideflow Studio with usable performance and successfully generated predictions for WSIs. GPU acceleration for model training is only available on systems with a dedicated GPU.

Discussion

Slideflow represents a noteworthy advancement in computational pathology deep learning software, characterized by its versatility, user-friendliness, and a comprehensive array of algorithmic approaches ready for immediate implementation. Among its enhancements over existing tools, Slideflow offers optimized whole-slide image

processing, enabling tile extraction at 40x magnification in as little as 2.5 s per slide. Image tile diameter for tile extraction is defined in microns, rather than pixels, to improve standardization across slide scanners with variable optical magnifications. The platform's data storage format is cross-compatible between both Tensorflow and PyTorch, facilitating versatile experimentation with algorithms from either framework. Slideflow also includes out-of-the-box support for more diverse functions, such as self-supervised learning, generative adversarial networks, segmentation, and highly flexible feature extraction, all utilizing the same unified data storage and streamlined IO pipelines. Included uncertainty quantification and explainability techniques can assist with developing more robust and clinically trustworthy models, and the graphical user interface provides a practical and accessible framework for testing model deployment in a prospective setting. Designed with an intuitive interface that supports easy customization, Slideflow serves as an ideal foundation for researchers with diverse objectives, from trainees with limited programming experience to advanced software developers integrating new methodologies.

Slideflow has been under active development for over 5 years by a dedicated team at University of Chicago and partner institutions. The developers are committed to ongoing long-term support, with plans for regular updates and new releases. Typically a major feature release is targeted quarterly, while minor bug fix and maintenance versions release as needed in between.

Looking ahead, priorities for future development include integration of multi-omics data into multiple-instance learning model training, expanding cell segmentation and classification capabilities, streamlining human-in-the-loop refinement of tissue and cell annotations, augmenting the graphical interface to support no-code model building, and integration of a pretrained model zoo. This roadmap focuses both on capabilities to empower more impactful biomedical research, as well as further improving accessibility and ease-of-use for diverse research teams. With a strong foundation already established, Slideflow is well-positioned to continue advancements as a community platform promoting broader adoption of novel AI tools for computational pathology.

Despite its advantages, Slideflow has several notable limitations. Although data processed in Slideflow is cross-compatible between Tensorflow and PyTorch and many functions can be performed using either deep learning backend, some functions have specific backend requirements. For example, multiple-instance learning and segmentation functions both require PyTorch. There are also several deep learning tasks that have not yet been included in Slideflow, such as the building and analysis of nuclei graphs with graph neural networks (GNNs). Our goal with this library has been to provide a strong foundation upon which new features and additional functionality can be easily built, and we anticipate extending functionality to include GNNs in a future update.

Conclusions

Slideflow is a flexible, end-to-end deep learning toolkit for digital pathology with computationally efficient whole-slide image processing, data storage cross-compatible with Tensorflow and PyTorch, efficient GPU-accelerated stain normalization, and a GUI to support deployment of trained deep learning models. Slideflow includes a variety of digital pathology deep learning methods to support a wide range of research objectives

without switching software environments or reprocessing data. The software is well documented and has been built to support researchers with a range of programming experience in the development of novel deep learning applications for digital histopathology.

Supplementary Information

The online version contains supplementary material available at https://doi.org/10.1186/s12859-024-05758-x.

Additional file 1. Supplementary Figures 1–11 and Supplementary Tables 1–3.

Acknowledgements
Not applicable.

Author contributions
J.M.D., E.D., and S.R. wrote the main manuscript text. J.M.D. and A.S. prepared figures. J.M.D., S.K., E.D., A.S., M.S., F.M.H., A.L., and P.M. contributed to software development. A.T.P. provided overarching guidance, mentorship, and additional clinical insights. All authors reviewed the manuscript.

Funding
This work was funded by NIH grant K12-CA13916013 (JMD), and effort support was also provided via the following grants: NIH R56-DE030958 (ATP), DoD Breakthrough Cancer Research program BC211095 (ATP), Wellcome Leap Q4Bio (ATP), EU Horizon Programme 2021-SC1-BHC (ATP), NIH grant UE5-EB035490 (ATP), NIH grant R01- R01CA276652 (ATP), DoE/NCI Innovative Methodologies and New Data for Predictive Oncology Model Evaluation (IMPROVE) project IAA (ATP), the SU2C Maverick Award Grant (ATP), and grants from ECOG Research and Education Foundation (AS).

Availability of data and materials
Slideflow and Slideflow Studio are available at https://github.com/jamesdolezal/slideflow, via the Python Package Index (PyPI), and Docker Hub (https://hub.docker.com/r/jamesdolezal/slideflow). Slideflow is licensed with GNU General Public License v3.0. The whole-slide image dataset from The Cancer Genome Atlas (TCGA) head and neck squamous cell carcinoma project are publicly available at https://portal.gdc.cancer.gov/projects/TCGA-HNSC. The training datasets from University of Chicago analyzed during the current study are not publicly available due to patient privacy obligations, but are available from the corresponding author on reasonable request. Data can only be shared for non-commercial academic purposes and will require institutional permission and a data use agreement.

Availability and requirements
Project name: Slideflow. Project home page: https://github.com/jamesdolezal/slideflow. Operating system(s): Platform independent. Programming language: Python. Other requirements: Python 3.8 or higher. License: GNU GPL-3.0. Any restrictions to use by non-academics: License needed for commercial use.

Declarations

Ethics approval and consent to participate
The experimental protocol and data collection plan for the University of Chicago dataset was reviewed and approved by the Biological Sciences Division / University of Chicago Medical Center Institutional Review Board (IRB #20-0238). Requirement of informed consent was waived by the Biological Sciences Division / University of Chicago Medical Center Institutional Review Board. All methods were carried out in accordance with relevant guidelines and regulations.

Consent for publication
Not applicable.

Competing interests
ATP reports personal fees from Prelude Therapeutics Advisory Board, personal fees from Elevar Advisory Board, personal fees from AbbVie consulting, and personal fees from Ayala Advisory Board, all outside of submitted work. ATP reports stock options ownership in Privo Therapeutics. All remaining authors report no competing interests.

Received: 17 April 2023 Accepted: 20 March 2024
Published online: 27 March 2024

References

1. van der Laak J, Litjens G, Ciompi F. Deep learning in histopathology: the path to the clinic. Nat Med. 2021;27(5):775–84. https://doi.org/10.1038/s41591-021-01343-4.
2. Campanella G, et al. Clinical-grade computational pathology using weakly supervised deep learning on whole slide images. Nat Med. 2019;25(8):1301–9. https://doi.org/10.1038/s41591-019-0508-1.
3. Raciti P, et al. Novel artificial intelligence system increases the detection of prostate cancer in whole slide images of core needle biopsies. Mod Pathol. 2020;33(10):2058–66. https://doi.org/10.1038/s41379-020-0551-y.

4. Fu Y, et al. Pan-cancer computational histopathology reveals mutations, tumor composition and prognosis. Nat Cancer. 2020;1(8):800–10. https://doi.org/10.1038/s43018-020-0085-8.
5. Schmauch B, et al. A deep learning model to predict RNA-Seq expression of tumours from whole slide images. Nat Commun. 2020;11(1):1. https://doi.org/10.1038/s41467-020-17678-4.
6. Dolezal JM, et al. Deep learning prediction of BRAF-RAS gene expression signature identifies noninvasive follicular thyroid neoplasms with papillary-like nuclear features. Mod Pathol. 2021;34(5):862–74. https://doi.org/10.1038/s41379-020-00724-3.
7. Kather JN, et al. Pan-cancer image-based detection of clinically actionable genetic alterations. Nat Cancer. 2020;1(8):789–99. https://doi.org/10.1038/s43018-020-0087-6.
8. Boehm KM, et al. Multimodal data integration using machine learning improves risk stratification of high-grade serous ovarian cancer. Nat Cancer. 2022;3(6):723–33. https://doi.org/10.1038/s43018-022-00388-9.
9. Cheerla A, Gevaert O. Deep learning with multimodal representation for pancancer prognosis prediction. Bioinformatics. 2019;35(14):i446–54. https://doi.org/10.1093/bioinformatics/btz342.
10. Swanson K, Wu E, Zhang A, Alizadeh AA, Zou J. From patterns to patients: advances in clinical machine learning for cancer diagnosis, prognosis, and treatment. Cell. 2023. https://doi.org/10.1016/j.cell.2023.01.035.
11. Bai B, Yang X, Li Y, Zhang Y, Pillar N, Ozcan A. Deep learning-enabled virtual histological staining of biological samples. Light Sci Appl. 2023;12(1):1. https://doi.org/10.1038/s41377-023-01104-7.
12. Echle A, et al. Clinical-grade detection of microsatellite instability in colorectal tumors by deep learning. Gastroenterology. 2020;159(4):1406-1416.e11. https://doi.org/10.1053/j.gastro.2020.06.021.
13. van der Velden BHM, Kuijf HJ, Gilhuijs KGA, Viergever MA. Explainable artificial intelligence (XAI) in deep learning-based medical image analysis. Med Image Anal. 2022;79: 102470. https://doi.org/10.1016/j.media.2022.102470.
14. Hamilton DG, Hong K, Fraser H, Rowhani-Farid A, Fidler F, Page MJ. Prevalence and predictors of data and code sharing in the medical and health sciences: systematic review with meta-analysis of individual participant data. BMJ. 2023;382: e075767. https://doi.org/10.1136/bmj-2023-075767.
15. Berman AG, Orchard WR, Gehrung M, Markowetz F. PathML: a unified framework for whole-slide image analysis with deep learning. MedRxiv. 2021. https://doi.org/10.1101/2021.07.07.21260138.
16. Muñoz-Aguirre M, Ntasis VF, Rojas S, Guigó R. PyHIST: a histological image segmentation tool. PLoS Comput Biol. 2020;16(10): e1008349. https://doi.org/10.1371/journal.pcbi.1008349.
17. Pocock J, et al. TIAToolbox as an end-to-end library for advanced tissue image analytics. Commun Med. 2022;2(1):1. https://doi.org/10.1038/s43856-022-00186-5.
18. Lu MY, Williamson DFK, Chen TY, Chen RJ, Barbieri M, Mahmood F. Data-efficient and weakly supervised computational pathology on whole-slide images. Nat Biomed Eng. 2021;5(6):555–70. https://doi.org/10.1038/s41551-020-00682-w.
19. Rosenthal J, et al. Building tools for machine learning and artificial intelligence in cancer research: best practices and a case study with the PathML toolkit for computational pathology. Mol Cancer Res. 2022;20(2):2. https://doi.org/10.1158/1541-7786.MCR-21-0665.
20. Coudray N, et al. Classification and mutation prediction from non-small cell lung cancer histopathology images using deep learning. Nat Med. 2018;24(10):10. https://doi.org/10.1038/s41591-018-0177-5.
21. Marcolini A, Bussola N, Arbitrio E, Amgad M, Jurman G, Furlanello C. histolab: a Python library for reproducible digital pathology preprocessing with automated testing. SoftwareX. 2022;20:101237. https://doi.org/10.1016/j.softx.2022.101237.
22. Cardoso MJ et al. MONAI: an open-source framework for deep learning in healthcare; 2022. arXiv: https://doi.org/10.48550/arXiv.2211.02701.
23. Lee G, Bae G, Zaitlen B, Kirkham J, Choudhury R. cuCIM—a GPU image I/O and processing library. 2021. Zenodo. https://doi.org/10.25080/majora-1b6fd038-022.
24. Martinez K, Cupitt JRG. VIPS - a highly tuned image processing software architecture. IEEE Int Conf Image Process. 2005;2:2–574.
25. Bankhead P, et al. QuPath: open source software for digital pathology image analysis. Sci Rep. 2017;7(1):16878. https://doi.org/10.1038/s41598-017-17204-5.
26. Aperio ImageScope. 2023. Available: https://www.leicabiosystems.com/digital-pathology/manage/aperio-imagescope/.
27. Otsu N. A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern. 1979;9(1):62–6. https://doi.org/10.1109/TSMC.1979.4310076.
28. van der Walt S, et al. scikit-image: image processing in Python. PeerJ. 2014;2: e453. https://doi.org/10.7717/peerj.453.
29. Senaras C, Niazi MKK, Lozanski G, Gurcan MN. DeepFocus: detection of out-of-focus regions in whole slide digital images using deep learning. PLoS ONE. 2018;13(10): e0205387. https://doi.org/10.1371/journal.pone.0205387.
30. Dolezal JM. Slideflow documentation; 2023. https://slideflow.dev.
31. Dolezal JM, et al. Uncertainty-informed deep learning models enable high-confidence predictions for digital histopathology. Nat Commun. 2022;13(1):6572. https://doi.org/10.1038/s41467-022-34025-x.
32. Duchon CE. Lanczos filtering in one and two dimensions. J Appl Meteorol Climatol. 1979;18(8):1016–22. https://doi.org/10.1175/1520-0450(1979)018%3c1016:LFIOAT%3e2.0.CO;2.
33. Reinhard E, Adhikhmin M, Gooch B, Shirley P. Color transfer between images. IEEE Comput Graph Appl. 2001;21(5):34–41. https://doi.org/10.1109/38.946629.
34. Macenko M et al. A method for normalizing histology slides for quantitative analysis; 2009, vol. 9, p. 1110. https://doi.org/10.1109/ISBI.2009.5193250.
35. Vahadane A, et al. Structure-preserving color normalization and sparse stain separation for histological images. IEEE Trans Med Imaging. 2016;35(8):1962–71. https://doi.org/10.1109/TMI.2016.2529665.
36. Tellez D, et al. Whole-slide mitosis detection in h&e breast histology using PHH3 as a reference to train distilled stain-invariant convolutional networks. IEEE Trans Med Imaging. 2018. https://doi.org/10.1109/TMI.2018.2820199.

37. Krause J, et al. Deep learning detects genetic alterations in cancer histology generated by adversarial networks. J Pathol. 2021;254(1):70–9. https://doi.org/10.1002/path.5638.
38. Laleh NG, et al. Benchmarking artificial intelligence methods for end-to-end computational pathology. Biorxiv. 2021. https://doi.org/10.1101/2021.08.09.455633.
39. Kather JN, et al. Deep learning can predict microsatellite instability directly from histology in gastrointestinal cancer. Nat Med. 2019;25(7):1054–6. https://doi.org/10.1038/s41591-019-0462-y.
40. Howard FM, et al. The impact of site-specific digital histology signatures on deep learning model accuracy and bias. Nat Commun. 2021;12(1):4423. https://doi.org/10.1038/s41467-021-24698-1.
41. Lindauer M et al. "SMAC3: a versatile bayesian optimization package for hyperparameter optimization; 2021. https://doi.org/10.48550/ARXIV.2109.09831.
42. Abadi M et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems; 2016.
43. neptune.ai. neptune.ai: experiment tracking and model registry; 2022. https://neptune.ai.
44. Dolezal, JM. lung-adeno-squam-v1 (Revision dade98a). Hugging Face. 2022. https://doi.org/10.57967/hf/0089.
45. Dolezal, JM. breast-er-v1 (Revision 17bd7fd). Hugging Face. 2023. https://doi.org/10.57967/hf/1500.
46. Dolezal, JM. thyroid-brs-v1 (Revision 17d17d8). Hugging Face. 2023. https://doi.org/10.57967/hf/1499.
47. Begoli E, Bhattacharya T, Kusnezov D. The need for uncertainty quantification in machine-assisted medical decision making. Nat Mach Intell. 2019;1(1):20–3. https://doi.org/10.1038/s42256-018-0004-1.
48. Kompa B, Snoek J, Beam AL. Second opinion needed: communicating uncertainty in medical machine learning. Npj Digit Med. 2021;4(1):4. https://doi.org/10.1038/s41746-020-00367-3.
49. Gal Y, Ghahramani Z, Dropout as a bayesian approximation: representing model uncertainty in deep learning. In: Proceedings of the 33rd international conference on machine learning, PMLR; 2016. p. 1050–1059. Accessed 28 Mar 2023. https://proceedings.mlr.press/v48/gal16.html.
50. Wang X, et al. Transformer-based unsupervised contrastive learning for histopathological image classification. Med Image Anal. 2022;81: 102559. https://doi.org/10.1016/j.media.2022.102559.
51. Wang X, et al. RetCCL: clustering-guided contrastive learning for whole-slide image retrieval. Med Image Anal. 2023;83: 102645. https://doi.org/10.1016/j.media.2022.102645.
52. McInnes L, Healy J, Melville J. UMAP: uniform manifold approximation and projection for dimension reduction; 2018. arXiv: https://doi.org/10.48550/ARXIV.1802.03426.
53. Gadermayr M, Tschuchnig ME. Multiple instance learning for digital pathology: a review on the state-of-the-art. Limit Future Potential. 2022. https://doi.org/10.48550/arXiv.2206.04425.
54. Ilse M, Tomczak J, Welling M. Attention-based deep multiple instance learning. In: Dy J, Krause A, editors. Proceedings of the 35th international conference on machine learning. Proceedings of machine learning research, vol. 80. PMLR; 2018, pp. 2127–2136. https://proceedings.mlr.press/v80/ilse18a.html.
55. Shao Z et al. "TransMIL: transformer based correlated multiple instance learning for whole slide image classication. In: Neural information processing systems; 2021.
56. Howard J, Gugger S. fastai: a layered API for deep learning; 2020. CoRR https://arxiv.org/abs/2002.04688.
57. Smith LN. A disciplined approach to neural network hyper-parameters: part 1—learning rate, batch size, momentum, and weight decay; 2018. CoRR http://arxiv.org/abs/1803.09820.
58. Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T. Analyzing and improving the image quality of StyleGAN. In: Proceedings. CVPR; 2020.
59. Karras T et al. Alias-free generative adversarial networks; 2021. arXiv. https://doi.org/10.48550/ARXIV.2106.12423.
60. Dolezal JM et al. Deep learning generates synthetic cancer histology for explainability and education; 2022. arXiv: https://doi.org/10.48550/ARXIV.2211.06522.
61. Dolezal JM. breast-er-gan-v1 (Revision db36196). Hugging Face. 2023. https://doi.org/10.57967/hf/1503.
62. Dolezal JM. lung-adeno-squam-gan-v1 (Revision 7e0ea0b). Hugging Face. 2023. https://doi.org/10.57967/hf/1502.
63. Dolezal JM. thyroid-brs-gan-v1 (Revision 2d73248). Hugging Face. 2023. https://doi.org/10.57967/hf/1501.
64. Selvaraju RR, Cogswell M, Das A, Vedantam R, Parikh D, Batra D. Grad-CAM: visual explanations from deep networks via gradient-based localization. Int J Comput Vis. 2019;128(2):336–59. https://doi.org/10.1007/s11263-019-01228-7.
65. Simonyan K, Vedaldi A, Zisserman A. Deep inside convolutional networks: visualising image classification models and saliency maps; 2014.
66. Sundararajan M, Taly A, Yan Q. Axiomatic attribution for deep networks; 2017. CoRR, vol. abs/1703.01365. http://arxiv.org/abs/1703.01365.
67. Kapishnikov A, Bolukbasi T, Viégas FB, Terry M. Segment integrated gradients: better attributions through regions; 2019. CoRR, vol. abs/1906.02825. http://arxiv.org/abs/1906.02825.
68. Iakubovskii P. "Segmentation Models Pytorch," GitHub repository. GitHub; 2019. https://github.com/qubvel/segmentation_models.pytorch.
69. Weinstein JN, et al. The cancer genome atlas pan-cancer analysis project. Nat Genet. 2013;45(10):1113–20. https://doi.org/10.1038/ng.2764.
70. Dolezal JM. tumor-segmentation-v1 (Revision 666fa9d). Hugging Face. 2023. https://doi.org/10.57967/hf/1504.
71. Stringer C, Wang T, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. Nat Methods. 2021;18(1):100–6. https://doi.org/10.1038/s41592-020-01018-x.
72. van Rossum G, Warsaw B, Coghlan N. "Style Guide for Python Code," PEP 8, 2001. https://www.python.org/dev/peps/pep-0008/.
73. "Pylint-code analysis for Python." https://www.pylint.org/.
74. "Google Python Style Guide." https://google.github.io/styleguide/pyguide.html.
75. "Mypy-optional static typing for Python." http://mypy-lang.org/.
76. Howard FM, et al. Multimodal prediction of breast cancer recurrence assays and risk of recurrence. Biorxiv. 2022. https://doi.org/10.1101/2022.07.07.499039.
77. Partin A, et al. Data augmentation and multimodal learning for predicting drug response in patient-derived xenografts from gene expressions and histology images. Front Med. 2023. https://doi.org/10.3389/fmed.2023.1058919.

78. Hennessey PT, Westra WH, Califano JA. Human papillomavirus and head and neck squamous cell carcinoma: recent evidence and clinical implications. J Dent Res. 2009;88(4):300–6. https://doi.org/10.1177/0022034509333371.
79. Xavier SD, Bussoloti Filho I, Lancellotti CLP. Prevalence of histological findings of human papillomavirus (HPV) in oral and oropharyngeal squamous cell carcinoma biopsies: preliminary study. Braz J Otorhinolaryngol. 2005;71(4):510–4. https://doi.org/10.1016/s1808-8694(15)31208-8.

Publisher's Note