Sequential Attention-based Sampling for Histopathological Analysis

Tarun G
Indian Institute of Science, Bangalore, India
tarung@iisc.ac.in

Naman Malpani
Indian Institute of Science, Bangalore, India
namanmalpani@iisc.ac.in

Gugan Thoppe
Indian Institute of Science, Bangalore, India
gthoppe@iisc.ac.in

Devarajan Sridharan
Indian Institute of Science, Bangalore, India
sridhar@iisc.ac.in

Abstract

Deep neural networks are increasingly applied in automated histopathology. Yet, whole-slide images (WSIs) are often acquired at gigapixel sizes, rendering them computationally infeasible to analyze entirely at high resolution. Diagnostic labels are largely available only at the slide-level, because expert annotation of images at a finer (patch) level is both laborious and expensive. Moreover, regions with diagnostic information typically occupy only a small fraction of the WSI, making it inefficient to examine the entire slide at full resolution. Here, we propose SASHA – *S*equential *A*ttention-based *S*ampling for *H*istopathological *A*nalysis – a deep reinforcement learning approach for efficient analysis of histopathological images. First, SASHA learns informative features with a lightweight hierarchical, attention-based multiple instance learning (MIL) model. Second, SASHA samples intelligently and zooms selectively into a small fraction (10-20%) of high-resolution patches to achieve reliable diagnoses. We show that SASHA matches state-of-the-art methods that analyze the WSI fully at high resolution, albeit at a fraction of their computational and memory costs. In addition, it significantly outperforms competing, sparse sampling methods. We propose SASHA as an intelligent sampling model for medical imaging challenges that involve automated diagnosis with exceptionally large images containing sparsely informative features[1].

1 Introduction

Deep learning models have emerged as a major frontier in automated medical imaging diagnosis. For example, such models accurately identify tumorous tissue in whole-slide histopathological images (WSI) [14, 10, 12, 19]. Yet, hardware and memory bottlenecks render it impractical to analyze WSIs acquired at gigapixel sizes. Moreover, analyzing WSI-s at high resolution is inefficient because regions with diagnostic information (e.g., tumor cells) make up only a small fraction of the entire image, and are not clearly identifiable at low resolutions. As a result, methods that predict slide-level labels from low-resolution WSIs suffer from performance bottlenecks [29, 16]. By contrast, methods that incorporate fine-grained "patch-level" information perform better [28, 37]. Yet, these latter methods require significant resources and expert annotations of WSIs at the patch level.

The Multiple Instance Learning (MIL) framework [5], addresses these limitations elegantly. In MIL, the WSI is divided into non-overlapping patches, with each patch being encoded into a high-dimensional feature vector. The feature vectors are aggregated using simple methods, like max- or mean-pooling, and this "bag" of features is slide-level labels. Furthermore, diagnostic regions can be identified with weighted aggregation methods, like Attention-Based deep Multiple Instance Learning (ABMIL) [15], which assigns patch-level attention scores, based on their relevance.

[1]Model implementation is available at: `https://github.com/coglabiisc/SASHA`

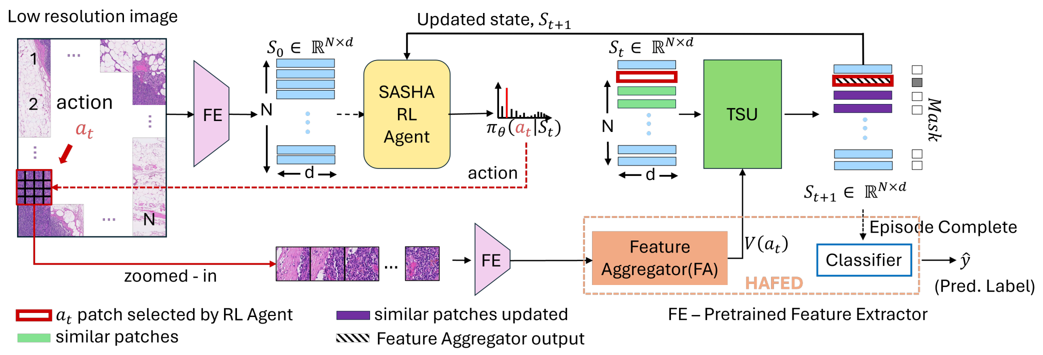

Figure 1: SASHA: An attention-augmented deep RL model. Starting with low-resolution WSI features $S_0 \in \mathbb{R}^{N\times d}$, at each time step t, the deep RL agent selects a patch a_t (red outline) to zoom in at high-resolution, based on policy $\pi(a_t|S_t)$. High-resolution features $V(a_t) \in \mathbb{R}^d$ are extracted and aggregated using a Heirarchical Attention-based Feature Distiller (HAFED, Section 3.2). The states S_t of patches with features similar to a_t are updated with a Targeted State Updater (TSU, Section 3.3). At the end of each episode, the classifier (Section 3.2) predicts the presence or type of cancer.

Nonetheless, a major limitation of MIL-based approaches is the need to process most WSI patches at high resolution. Because tumors typically occupy only a small portion of the slide, feature extraction for the majority of non-tumor patches becomes computationally inefficient, increasing inference time and memory usage. To address this, Zhao et al. [40] proposed a deep reinforcement learning (RL)-based approach – RLogist, that first examines WSIs at low resolution and then selectively zooms into diagnostically relevant high-resolution patches to predict slide-level labels.

Although reportedly much faster ($\sim$4x) than conventional MIL approaches at inference time [40], RLogist suffers from major drawbacks: i) Poor accuracy: Because RLogist samples only a fraction of the whole slide, it lags current sota models that sample the whole WSI at high resolution by $\sim$10-15%; ii) Weakly diagnostic features: Image features are extracted with a generic ResNet-50 [13] model pretrained on ImageNet [6], thereby rendering these features poorly diagnostic of pathology; iii) Training inefficiency: The RL agent training suffers from bottlenecks, such as convergence issues arising from the RL policy network being trained concurrently with the classification network.

Here, we address these limitations by combining RL with multi-attention MIL in a framework we call SASHA - **S**equential **A**ttention-based **S**ampling for **H**istopathological **A**nalysis (Fig. 1). Our study makes the following key contributions:

- We design an attention-augmented deep RL agent that scans WSI patches at low resolution, and zooms into selected patches to extract meaningful, high-resolution features.
- To improve diagnostic accuracy, we propose a Hierarchical Attention-based FEature Distiller (HAFED), which uses multiple attention heads to generate label-informed MIL features.
- For stable convergence of the RL policy, we train a slide-level label classifier as part of the HAFED model, and freeze the classifier during RL training.
- For faster training and inference, we exploit similarities among patch features to perform a concerted state update of correlated instances with a targeted state updater (TSU).

We show that SASHA achieves state-of-the-art performance on two cancer benchmarks [3, 26], comparable to models that analyze the entire WSI at high resolution, while requiring only $\sim 6\%$ of their memory for WSI representation and with upto $8\times$ faster inference times.

2 Related Work

Deep learning approaches for automated histopathology have a rich and detailed history (reviewed in [34, 27]). Here, we review specifically those methods that inspire key aspects of our solution and, therefore, form benchmarks for performance comparisons.

Multiple Instance Learning (MIL). MIL is a weakly supervised learning paradigm widely applied in WSI analysis. Unlike fully supervised learning, where each instance is associated with a label, MIL involves dividing the slide into a "bag" of instances (patches); the model is then trained with bag-level labels. MIL is, therefore, suitable for classification problems where slide-level, but not patch-level, labels are available. Early MIL approaches relied on instance selection to identify the most relevant patches within a bag ([9]). Recent models, such as attention-based MIL (ABMIL) [15], assign importance weights to patches, allowing the model to focus on diagnostically relevant regions.

Advances in attention-based MIL. A key limitation of these approaches is that each patch is treated independently of the others, even though many may share strong feature correlations. Recent models exploit correlations among instance representations using a self-attention mechanism (e.g., TransMIL [32]). Other approaches cluster instances based on their attention scores [22]. Yet, these attention mechanisms may capture only some types of diagnostic features, leading to overfitting and lack of generalization. To overcome this, Attention-Challenging Multiple Instance Learning (ACMIL) [39] employs multiple attention blocks, each focusing on distinctive diagnostic features. Inspired by these approaches, our model incorporates multiple attention heads for diagnostically relevant feature extraction and a correlation-based state update rule for efficient feature updates. Moreover, earlier approaches largely process every patch at high resolution, which is computationally inefficient. ZoomMIL [36] addresses this by recursively sampling a fixed number of patches per resolution based on top-k attention scores. However, its sampling strategy depends on predefined attention criteria, which motivates the use of adaptive and context-aware patch selection methods.

Deep Reinforcement Learning (DRL) in medical imaging. Conventional RL learns a policy for actions that maximizes the expected reward in novel, unstructured environments. DRL augments RL with the representational power of deep neural networks. The power of DRL was originally demonstrated in challenging strategy and video games (e.g., Deep-Q, AlphaGo), [25, 33]. DRL is particularly useful in domains with limited annotations, such as automated medical diagnosis, and has been successfully applied to various medical imaging tasks, including localization of anatomical landmarks in medical scans [1], lesion detection [23, 35], 3D image segmentation of MRI scans [20] and tumor classification in WSIs [40, 41]. In such tasks, a DRL agent learns an optimal search strategy by exploring different portions of the environment (here, image) and receiving sparse feedback in the form of classification accuracy, without the need for analyzing the entire image at high resolution.

DRL for histopathology. Diagnosis of WSIs by human histopathologists involves viewing the slide – at a low (“scanning”) resolution – and zooming into suspected tumor regions – at high resolution – for reliable diagnosis. In this context, a recent approach, RLogist [40], models the scanning process of histopathologists as a Markov Decision Process (MDP) using an RL agent. The agent learns an efficient observation strategy, selectively zooming into only a few high-resolution WSI patches before making a classification decision, thereby reducing computational overhead and inference time. Similarly, PAMIL [41] addresses the shortcomings of instance-based clustering methods using an RL agent to form pseudo-bags (aggregates) of related instances from high-resolution WSIs. It models pseudo-bag formation as an MDP, forming new pseudo-bags based on past knowledge from the previously formed bags, informed by the label error arising from bag-level predictions.

Feature extraction for histopathology. Deep neural networks such as ResNet [13] and Vision Transformers (ViT) [7], pre-trained on ImageNet [6], are commonly used for feature extraction in computer vision tasks. Their exposure to millions of natural images enables them to identify generic, relevant features, for specialized downstream tasks. Yet, pretraining on natural images is not ideal for medical image diagnosis. [2]. For instance, histopathology images lack a canonical orientation, exhibit low color variation due to standardized staining protocols, and their diagnostic information varies with magnification. To address this, Kang et al. [17] introduced ResNet-50 and ViT-S models with self-supervised pretraining on large WSI datasets (e.g., TCGA, TULIP).

3 Model description

Our framework combines multi-attention multiple instance learning (MIL) with reinforcement learning (RL), to identify diagnostic patches in the WSI. We also employ a targeted update rule that exploits correlations among instances to efficiently update the state. Each step in the model – and how it addresses shortcomings with existing methods – is described subsequently.

3.1 Problem Statement and Overview

For our task, the model seeks to predict binary target labels $Y \in \{0, 1\}$ for a given Whole Slide Image (WSI) X. The latter is a 3-channel RGB image that is available in the dataset at multiple different levels of magnification, whereas the former either indicates the presence (versus absence) of cancer or indicates one of two subtypes. We notate the size (in pixels) of the image at magnification

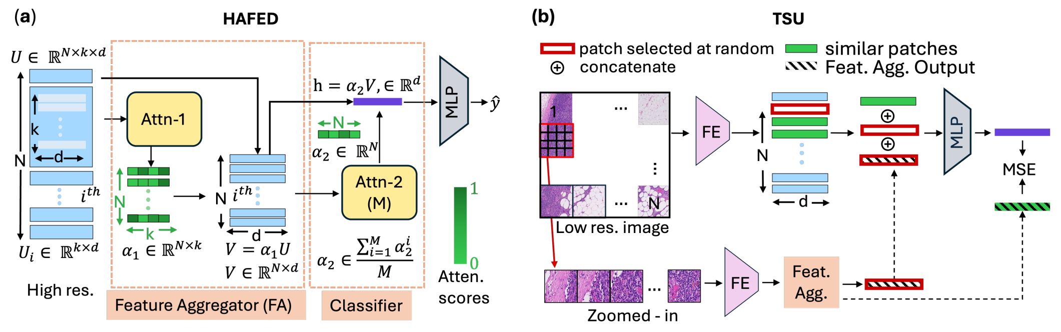

Figure 2: Components of SASHA. (**a**) HAFED. During training, features from each patch in each WSI are extracted and aggregated with a hierarchical, attention-based feature distiller (HAFED). High-resolution input $U \in \mathbb{R}^{N\times k\times d}$, undergoes two levels of attentional filtering, first operating across the k high resolution patches in each low resolution patch $\alpha_1 \in \mathbb{R}^{N\times k}$ and then operating across the N low resolution patches $\alpha_2 \in \mathbb{R}^N$ to produce the final classifier prediction $\hat{Y}$. During inference, only the sampled patch's high-resolution features are analyzed. (**b**) TSU. Following this the state of all patches whose features are correlated with the sampled patch are concurrently updated, with a targeted state updater (TSU) (see sections 3.2 and 3.3 for details).

level $m \in \{1, 2, 3\}$ as G^m. The goal of our approach is to scan the image at the lowest resolution G^3, which we call the "scanning-level" as in [40], while zooming into and analyzing only select portions of the image at a higher resolution G^1. For this, each WSI at G^3 is divided into multiple non-overlapping patches, such that a given WSI X is represented as $X = \{x_1, \ldots, x_N\}$, where N is the total number of patches at low resolution G^3; note that N can vary across different WSI images. Now, when zoomed in, these patches are magnified to a high-resolution; for this, we choose the highest resolution G^1 in all datasets tested. Therefore, each patch x_i at the low-resolution comprises of k patches at the high-resolution; we denote these as $\{x_{i1}, \ldots, x_{ik}\}$.

Given these definitions, the three main components of the algorithm are as follows (Fig. 1):

Hierarchical Feature Distiller: This hierarchical attention-based network extracts diagnostic features from the k high-resolution patches for each low-resolution patch and creates a d dimensional feature representation for it. The output of this network is used to update WSI "state" and to train a downstream WSI label classifier; see Section 3.2 for details.

Targeted State Updater: This network maintains a state of the entire WSI, comprising a d dimensional feature representation for each low-resolution patch. As each patch is zoomed in to high-resolution, the state vector is updated, but only for instances whose features are correlated with those of the sampled patch; see Section 3.3 for details.

Deep RL Agent: This agent comprises a policy network and a value network. Based on the current state, the former network produces a probability distribution over a set of actions from which the agent selects the next patch to zoom in, whereas the latter network is used to compute a generalized advantage estimate for each state-action combination. Both networks are optimized using the Proximal Policy Optimization (PPO) algorithm; see Section 3.4 for details.

Preprocessing and Feature Extraction. Prior to training and testing, we preprocess the WSI and extract features using a pre-trained model, as follows: To isolate histopathological tissue from the glass slide background on the WSI we apply established segmentation methods [22] at low resolution. The extracted tissue is then divided into non-overlapping patches of size $(256 \times 256 \times 3)$, again at both resolutions; these operations were performed using the code provided as part of the CLAM [22] GitHub repository. For the first level of feature extraction, we utilize a pre-trained ViT-based encoder [7], as described in the Related Work (Section 2). Each input patch is encoded into a feature embedding vector of dimension d. We denote the input feature representation of a WSI at low-resolution as $Z \in \mathbb{R}^{N\times d}$, and at high-resolution as $U \in \mathbb{R}^{N\times k\times d}$.

3.2 Hierarchical Attention-based Feature Distiller (HAFED)

A key first step with diagnosis of large WSIs is to learn informative, yet parsimonious, feature representations for the high-resolution patches. Such representations are critical for the RL agent to update WSI state, as it makes decisions about which patch to visit next, as well as for the final classification. Recent advances in attention-based MIL models compute such informative representations for high-resolution whole-slide images (WSIs) [15]. However, in an RL setting, the state representation for the WSI at each time step comprises a mix of information both at low and high-resolutions, depending on the specific patches zoomed in (see next). It is, therefore, desirable for both the low-resolution and high-resolution feature representations to have matched dimensionality; such a design choice considerably simplifies downstream computations (e.g., classification). To achieve this, we propose a Hierarchical Attention-based FEature Distiller (HAFED) (Fig. 2a).

HAFED is a two-stage attention-based model designed to operate directly on high-resolution feature patches while producing representations compatible with low-resolution patches. In the first stage (Fig. 2a, Feature Aggregator), the model takes as input high-resolution features $U \in \mathbb{R}^{N \times k \times d}$, and learns to estimate attention scores for each of the k sub-patches, reflecting the latter's relative importance in each of the N low resolution patches; a stochastic instance masking strategy is applied to avoid over-fitting [39]. A weighted aggregation of the sub-patches based on these attention scores produces a compressed feature representation $V \in \mathbb{R}^{N \times d}$, thereby reducing the dimensionality of the high-resolution features and aligning them with the dimensionality of the low-resolution features $Z \in \mathbb{R}^{N \times d}$. In the second stage (Fig. 2a, Classifier), a second attention mechanism operates across the N patch-level features in V to identify diagnostically important patches at the slide level. A weighted aggregation of the N features, guided by the second-stage attention scores, yields a final slide-level embedding $h \in \mathbb{R}^d$ that summarizes the entire WSI. This d-dimensional embedding is then used for downstream WSI classification. We employed multiple (M) attention branches, and the model is trained with the sum of three losses [39]: i) a similarity loss to learn distinct attention weight patterns in each branch, ii) the label loss associated with each branch, and iii) overall label loss.

During training, both stages of the model are trained end-to-end by visiting all of the N patches at high resolution. Thus, attention weights, and feature representations, in both the first and second stage of the model are learned in a "label-aware" manner using the ACMIL loss (classifier and dissimilarity losses). During inference, a single high-resolution patch a_t is visited at each timestep t and its compressed feature representation $V(a_t) \in \mathbb{R}^d$ is used to update the slide state. While previous approaches [40] employ generic, label-agnostic feature extractors HAFED identifies important regions at both the sub-patch (first-stage) and patch (second-stage) levels, which enables efficient learning of diagnostically-relevant and parsimonious feature representations from high-resolution WSIs.

3.3 Targeted State Updater (TSU)

In our deep RL model, each WSI is represented by a "state" $S_t \in \mathbb{R}^{N \times d}$ at each timestep t in an episode. The RL agent makes sequential decisions of which patch to visit based on S_t. We describe, here, how this state is computed and updated over successive time steps: Scanning begins with the low-resolution feature representation Z, which serves as the initial state S_0. While Z provides a coarse global view of the WSI, it often lacks the fine-grained diagnostic details necessary for accurate classification. Obtaining high-resolution features for all patches is computationally expensive; thus, at each step t, we selectively zoom into a single patch at high resolution to gather additional information. The selected patch, denoted as a_t, yields k high-resolution sub-patches, which are processed through the first attention model of HAFED. A weighted aggregation of the sub-patches produces a refined high-resolution feature vector $V(a_t) \in \mathbb{R}^d$ that captures detailed local information.

The challenge then is how to effectively integrate this newly acquired high-resolution information into the overall state representation S_t without corrupting previously acquired information. One possible approach (e.g., Zhao et al. [40]) involves concatenating $V(a_t)$ with every patch in S_t, training an MLP (multi-layer perceptron) with this augmented state as input to produce an updated global state representation S_{t+1} that gradually converges toward the feature representation obtained from all high-resolution patches. We argue that such a global state update is far from ideal. Because only one high resolution patch is observed in each time step, updating instances globally risks corrupting the state, especially when instances unrelated to the sampled patch are concurrently updated. Moreover, information from the sampled patch is likely to be relevant, primarily, for updating instances that share features similar to it [32, 22].

To exploit these correlations, we propose a targeted, similarity-based state update (TSU) model (Fig. 2b). For the selected patch a_t, we compute the cosine similarity between $S_t(a_t)$ and all other patch representations in S_t. We then define a set C containing the indices of patches whose cosine similarity with $S_t(a_t)$ exceeds a threshold (τ, a hyperparameter), excluding a_t itself. Only the patches in C are updated: $V(a_t)$ is concatenated with the representations of patches in C and passed through

Algorithm 1 Agent-Environment Loop at Time Step t

```
Input: Current state $S_t \in \mathbb{R}^{N \times d}$; Set $I$ of sampled patch indices, upto time $t$
Parameters: Policy network $\pi_{\theta_t}$; HAFED feature aggregator model $f_H(.;\theta_H)$ and classifier $f_C(.;\theta_C)$; TSU state update model $f_S(.;\theta_S)$
Output: Updated state $S_{t+1}$ and action $a_{t+1}$ at time $t+1$
1: $a_t \sim \pi_{\theta_t}(. \mid S_t)$ {Sample patch index $a_t$ from the policy distribution}
2: $I \leftarrow I \cup \{a_t\}$ {Store selected patch index}
3: if training then
4:     $\hat{y_t} \leftarrow f_C(S_t;\theta_C)$ {predict WSI label with HAFED classifier}
5:     Update $\pi_\theta$ with a PPO algorithm; intermediate reward $r_t = -CE(y, \hat{y_t})$
6: end if
7: $V(a_t) \leftarrow f_H(U_{a_t};\theta_H)$ {extract features $V(a_t) \in \mathbb{R}^d$ from high-resolution zoomed in features, $U_{a_t}\mathbb{R}^{k \times d}$ of the sampled patch, with the HAFED model}
8: $S_{t+1}(a_t) \leftarrow V(a_t)$ {update the sampled patch with its high-resolution feature representation}
9: $C \leftarrow \{i : \cos\angle(S_t(i), S_t(a_t)) \geq \tau\}$ {identify set of patch indices whose cosine similarity with $S_t(a_t)$ exceeds a threshold}
10: for each patch index $i$ in $C \setminus I$ do
11:     $S_{t+1}(i) \leftarrow f_S([S_t(i), S_t(a_t), V(a_t)];\theta_S)$ {targeted update of similar patches with the TSU model}
12: end for
13: return $S_{t+1}, a_{t+1}$
```

an MLP trained to minimize the MSE loss against the corresponding entries in $V(C)$. Additionally, $S_t(a_t)$ is directly replaced with $V(a_t)$ because its exact high-resolution representation is available. In the next time step, the patch a_t is masked out, and future actions are sampled from the remaining unmasked patches; this strategy is applied during both the training and evaluation stages. With ablation studies (Table 2), we show that this selective update strategy outperforms naive approaches.

3.4 Sequential Attention-based Sampling for Histopathological Analysis

We train a reinforcement learning (RL) agent to select which patches to zoom into, leveraging feature distillation with the HAFED model and state update with the TSU model. We formulate the patch selection process as a sequential decision-making problem, where an agent interacts with a slide-specific environment over multiple steps (Algorithm 1). During each training episode, at each timestep t, the agent observes the WSI state S_t, which is initialized with the low-resolution feature $S_0 = Z$. Based on S_t, the policy network π_θ produces a probability distribution over a set of actions from a discrete action space $\{1, 2, \dots, N\}$. The agent samples an action a_t, corresponding to the next low-resolution patch to zoom in, according to the current policy $\pi_{\theta_t}(\cdot \mid S_t)$ with a stochastic sampling rule. The zoomed-in patch is passed through HAFED to obtain its high-resolution representation $V(a_t)$. This representation is then used by the TSU to selectively refine the current state S_t, by modifying only those patches that are similar to a_t in the feature space. This produces an updated state S_{t+1}. The classifier trained together with HAFED then uses the updated state S_{t+1} to produce a slide-level prediction $\hat{y}_t$. This prediction is compared against the ground truth label y to compute a time-step-wise reward, based on a negative cross-entropy (CE) loss. This feedback is used to update the policy network using Proximal Policy Optimization (PPO) [31] algorithm. We employed Generalized Advantage Estimation (GAE) in computing the advantage function [30], whose output was trained to predict the reward until the end of the episode. Both the policy and value network were modeled on standard architectures [15] (details in Appendix A.2.3). The same steps were followed during evaluation, except that the action with the highest probability based on the policy network's output was chosen deterministically.

4 Experiments

4.1 Dataset and implementation details

We evaluate the effectiveness of SASHA with two publicly available benchmark datasets CAMELYON16 breast cancer [3] and TCGA-NSCLC lung cancer dataset [26]. While the CAMELYON16 labels correspond to normal versus pathological WSIs, TCGA-NSCLC labels correspond to distinguishing two different types of cancers, adenocarcinoma versus squamous cell carcinoma. We followed standardized train-test splits for both datasets. In addition, we also evaluate SASHA on multi-class classification tasks using the BRACS and Camelyon+ datasets. Further details on the datasets, hyperparameter selection, implementation, and training are in Appendices A.1 and A.2.

	Method	CAMELYON16			TCGA - NSCLC		
		Accuracy	AUC	F1	Accuracy	AUC	F1
Sample 100%	Max Pooling	0.933 ±0.007	0.971±0.001	0.911±0.008	0.881±0.010	0.952±0.008	0.882±0.010
	ABMIL	0.947±0.008	0.958±0.016	0.928±0.011	0.883±0.045	0.947±0.021	0.884±0.045
	CLAM_SB	0.949±0.012	0.976±0.008	0.931±0.015	0.905±0.019	0.961±0.009	0.907±0.018
	TransMIL	0.936±0.006	0.953±0.008	0.912±0.008	0.898±0.015	0.964±0.004	0.899±0.016
	DTFD	0.944±0.014	0.977±0.008	0.944±0.014	0.911±0.012	0.962±0.010	0.911±0.022
	ACMIL	0.941±0.015	0.970±0.011	0.924±0.017	0.906±0.025	0.959±0.006	0.907±0.026
	HAFED (ours)	0.963±0.008	0.980±0.003	0.951±0.011	0.923±0.011	0.966±0.015	0.925±0.010
k	Zoom-MIL (k=300)	0.780 ±0.017	0.700 ±0.029	0.738 ±0.015	0.908 ±0.004	0.967 ±0.004	0.908 ±0.004
10%	RLogist-0.1	0.824	0.829	–	0.828	0.892	–
	Zoom-MIL (k=80)	0.753±0.02	0.743±0.007	0.743±0.012	**0.907±0.002**	**0.969±0.012**	**0.907±0.002**
	SASHA-0.1 (ours)	**0.901±0.021**	**0.918±0.014**	**0.856±0.031**	0.897±0.023	0.956±0.023	0.898±0.024
20%	RLogist-0.2	0.862	0.879	–	0.839	0.903	–
	Zoom-MIL (k=160)	0.759±0.001	0.794±0.001	0.755±0.001	0.912±0.005	0.970±0.001	0.912±0.005
	SASHA-0.2 (ours)	**0.953±0.017**	**0.979±0.008**	**0.937±0.024**	**0.912±0.010**	**0.963±0.014**	**0.914±0.011**

Table 1: Performance comparison on the CAMELYON16 and TCGA-NSCLC datasets. The top 7 rows reflect models that process all patches at high resolution ("Sample 100%"). Underlined values indicate the best performance among these methods. Rows 9-11 and 12-14 reflect models that selectively sample either "10%" or "20%" of the patches, respectively. **Bold** values indicate the best performance among these methods at the respective sampling fraction.

4.2 Baselines and Metrics

We evaluate and compare the performance of our model against two types of baselines. First, we evaluate a range of different sota methods that analyze all patches in the WSI at high resolution, including MaxPooling, ABMIL [15], CLAM [22], TransMIL [32], DTFD [38], and ACMIL [39]. We also include comparisons with our hierarchical attention-based (HAFED) model (Section 3.2) with the full-resolution WSI. Yet, these methods serve only as an upper bound on achievable performance.

For a fair comparison, we also evaluate models that sample only a fraction of the WSI. Specifically, we compare our approach with RLogist [40] when trained to sample exactly 10% and 20% of patches at high resolution, under identical experimental settings. We also compare with a non-RL-based Zoom-MIL [36] model that samples exactly k = 80, 160, or 300 patches at low resolution (5x $\rightarrow$ 20x). The choice of k values aligns with the patch statistics of the dataset: WSIs contain $\sim$800 low-resolution patches on average; thus, $k = 80$ and $k = 160$ correspond to 10% and 20% of the available patches, on average, providing a fair comparison with the RLogist-0.1/SASHA-0.1 and RLogist-0.2/SASHA-0.2 settings, respectively. $k = 300$ follows the default configuration used in the original Zoom-MIL paper [36]. This setup allows us to systematically analyze the effect of varying observation budgets on model performance. In addition, we also compare against ACMIL and DTFD under the same observation budgets, with random sampling of high-resolution patches (Table 2); details in Appendix A.3. Models were trained with the official codebase provided by the respective studies, except for RLogist. The RLogist codebase is insufficient for replication; therefore, we report the performance metrics *as is* from the original study [40].

Our performance metrics include Accuracy, AUC, and F1 scores for normal versus pathological tissue classification (CAMELYON16) or for classifying between the different tumor types (TCGA-NSCLC). In addition, we expect that selective sampling models, such as ours, would exhibit significantly faster inference times compared to models that examine all patches at high resolution; therefore, we also evaluate and compare inference times (Fig. 3c) . Finally, we also compute a metric of "WSI compressibility". We define this as a ratio of the original WSI image size, at full resolution, to the size the of WSI representation used by each model's classifier to make a final decision. This metric has important real-world implications in terms of efficient archival and storage of WSI images. We compare models of comparable complexity and performance (CLAM, TransMIL, ACMIL, DTFD, SASHA) in terms both WSI compressibility and overall model size.

4.3 Classification Performance, Inference Time and Compressibility

Classification Performance. SASHA consistently outperforms the sota competing deep RL model – RLogist – by a large margin, for both the 10% and 20% observation budgets (Table 1, SASHA-0.1 and SASHA-0.2, vs RLogist-0.1 and RLogist-0.2, respectively). Improved classification performance was reflected in all 3 metrics quantified: Accuracy, AUC, and F1-score. Zoom-MIL performed comparably with SASHA-0.2 for the TCGA benchmark, but underperformed heavily for CAMELYON-16. Moreover, our HAFED model performed on par with the best sota models (e.g. ACMIL, DTFD) that

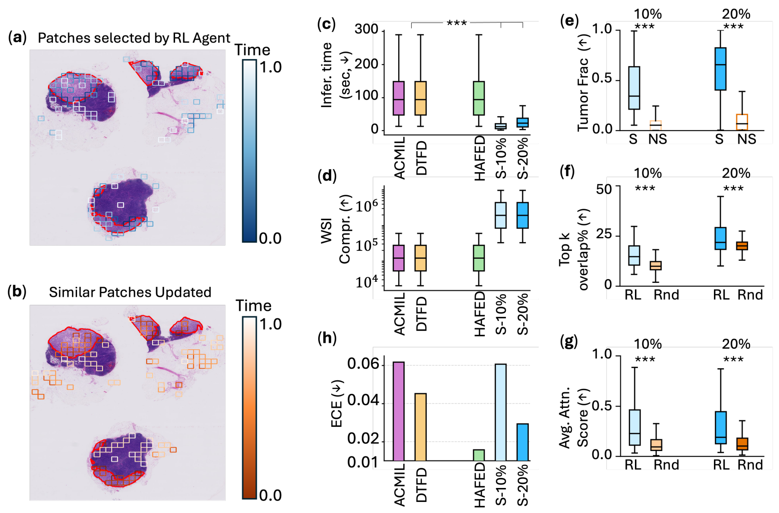

Figure 3: (**a-b**) Patch selection and update strategy by SASHA's RL agent. Box color corresponds to time fraction in episode. (**c-d, h**) Inference time, compressibility and expected calibration error (ECE) for SASHA and HAFED compared to other models. S-10% and S-20% refer to SASHA-0.1 and -0.2 models, respectively. (**e**) Tumor fraction in patches sampled by SASHA's RL agent (S) versus non-sampled patches (NS). (**f-g**) Top-k attention score overlap fraction and average attention score for patches sampled by SASHA's RL agent versus a random policy (Rnd). In (e-g) left and right pairs of bars reflect 10% and 20% observation budgets, respectively. *** indicates $p < 0.001$.

analyze the full, high-resolution WSI (Table 1, underlined). Interestingly, even with sampling 5x fewer patches, SASHA-0.2 nears ceiling performance achievable with HAFED, when all 100% of patches are sampled at high resolution (Table 1, compare SASHA-0.2 with HAFED).

Inference time. The average inference time per WSI for the CAMELYON16 dataset, including both pre-processing and feature extraction time, was $\sim$117 seconds, using HAFED (Fig. 3c). Other sota attention-based methods like ACMIL and DTFD also exhibited similar average inference times per WSI. The most expensive operation was feature extraction for each of the high resolution patches. By contrast, our model samples only a few patches at high-resolution, thereby leading to a substantial decrease in overall inference time, $\sim$14 seconds and $\sim$26 seconds per WSI for SASHA-0.1 and SASHA-0.2 – an $8\times$ and $4\times$ speedup, respectively – compared to the second fastest model (Fig. 3c).

WSI compressibility. Similarly, a comparison of the WSI compressibility revealed a significant ($> 16\times$) improvement, compared to models that examine the entire slide at high resolution (Fig. 3d). The main reason for this saving is the fact that our HAFED model encodes features for both low- and high-resolution patches in a uniform $N \times d$ dimensional space; by contrast, other models encode high-resolution features in a $N \times k \times d$ dimensional space, where k reflects the ratio of the zoomed-in to the scanning resolution size. The full set of results is shown in Appendix A.4. These results highlight the efficiency of our method in balancing performance with computational and representational costs.

4.4 Ablation Study and Limited Sampling

We performed a systematic ablation study removing each component of the model in turn – or replacing it with a naive variant – and evaluating it with the CAMELYON16 dataset. These included: a) replacing the WSI-pretrained ViT with a ResNet50, used commonly for feature extraction [41, 40], b) replacing ViT with the CONCH medical image-encoder[24], pretrained on 1.17M image caption pairs, c) replacing multi-branch attention with single branch in the Classifer [15], d) replacing the targeted state update policy with a global state update [40], e) removing the TSU and updating the state only of the sampled (local, high-resolution) patch, and f) and selecting an action based on a random policy (see "Variant"s in Table 2). In every case, we observed sharp drops in accuracy, ranging from 7-45%, relative to the baseline SASHA model; similar drops occurred in AUROC and F1 scores Table 2. The strongest impact occurred when changing the RL policy (Table 2, f). The next strongest impact occurred when changing the feature extractor to ResNet-50 (a), but this impact was mitigated when using the CONCH encoder (b). Intermediate levels of impact occurred when changing the classifier to single-branch attention (c) or altering the state update method (d-e).

Variant	Feature	Classifier	TSU	RL Policy	Accuracy	AUC	F1
SASHA Default	✓	✓	✓	✓	0.964	0.980	0.953
(a) ResNet-50	*	✓	✓	✓	0.860	0.817	0.780
(b) CONCH	*	✓	✓	✓	0.930	0.950	0.905
(c) 1^o Att. Branch	✓	*	✓	✓	0.899	0.964	0.851
(d) Global Update	✓	✓	*	✓	0.837	0.824	0.779
(e) Local Update	✓	✓	*	✓	0.860	0.901	0.800
(f) Random Policy	✓	✓	✓	*	0.516	0.550	0.553
(g) ACMIL-0.2 (Rnd.)	✓	✓	*	*	0.838	0.857	0.773
(h) DTFD-0.2 (Rnd.)	✓	✓	*	*	0.862	0.878	0.862

Table 2: Ablation study on CAMELYON16 with the SASHA-0.2 setup using 20% high-res patches. ✓ denotes the default, * the altered component.

Additional ablation experiments are described in Appendix A.3. These results indicate that every component of our model was critical for achieving high performance, comparable to sota models.

Finally, we examined the effect of imposing a limited observation budget on other sota models, and compared the performance of SASHA under the same budget. Specifically, ACMIL and DTFD were trained and tested with a random selection of high-resolution patches under a 20% observation budget for each WSI. In this case also we observed substantial drops in performance relative to the default SASHA model (Table 2, g-h). In other words, randomly sampling a limited fraction of high-resolution patches, even with the best models, failed to outperform SASHA.

4.5 Patch selection strategy: Explainability

When deploying deep RL models in medical settings, it is essential that they learn not only an effective, but also an explainable action policy. For this, we explore whether SASHA's RL agent exhibits a meaningful patch selection strategy prior to classifying WSI images (Fig. 3a and Fig. 3b). We propose that such a strategy must satisfy at least two important requirements: i) In WSIs containing tumor tissue, the RL agent must preferentially sample patches with a large fraction of tumor tissue for high-resolution scans, and ii) an effective RL policy should preferentially sample patches afforded a high attention score by sota models that examine the entire WSI at high resolution. We evaluated each of these criteria in the data. Indeed, the tumor fraction in patches sampled by the RL agent at high resolution (10% with SASHA-0.1 or 20% with SASHA-0.2) significantly exceeds that in a corresponding number of randomly chosen patches, among those not sampled by the agent (Fig. 3e, $p<0.001$). Moreover, the RL policy guides the agent to preferentially samples patches afforded a high attention score by sota models: both the average attention score (Fig. 3g, $p<0.001$) and the fraction of top-k patches sampled under the RL policy were statistically significantly greater than the fraction of patches sampled by a random policy (Fig. 3f, $p<0.001$). In other words, our RL agent's sampling actions were guided by an efficient, intuitive and explainable policy.

4.6 Effect of observation budget: Calibration

We have shown that SASHA achieves near-sota performance at a fraction of the inference time, even with sampling 10% or 20% of the patches at high resolution. Is there any advantage to sampling more patches, other than achieving a marginal improvement accuracy, especially when the inference time increases almost linearly with the proportion of high-resolution patches sampled. Here, we show empirically that sampling a higher fraction of patches yields better calibrated predictions. With well calibrated models, the model's confidence in its prediction (e.g., predictive uncertainty) is a useful indicator for when referral to a specialist is warranted [8]. We quantified the expected calibration error (ECE) [11] as a measure of the degree of calibration of the models. The ECE decreased systematically at the proportion sampled patches increased from 10% (SASHA-0.1) to 20% (SASHA-0.2) to 100% (HAFED). This pattern occurred possibly because providing the model more timesteps to sample the WSI in each RL episode enabled the model to recover from sub-optimal sampling actions in the early timesteps, and to sample the relevant portions of the WSI in the later timesteps. Surprisingly, ECE with SASHA-0.2 was lower than that of even sota attention-based models like ACMIL and DTFD (Fig. 3h). In other words, while a shorter observation budget encouraged efficient learning, a longer budget produced better calibrated predictions.

5 Conclusions and Limitations

We show that an attention-augmented deep RL agent outperforms or performs on par with sota models on WSI classification, albeit at a fraction of their memory and computational costs. Key limitations of the model include a significant impact of the observation budget on model calibration, and the need to train the model with all high resolution patches, which significantly increases training time; other limitations are discussed in Appendix A.6. Nonetheless, we demonstrate explainable, reliable, and calibrated predictions, which could enhance clinicians' trust in model predictions.

Acknowledgements

This research was supported by a Department of Science and Technology SwarnaJayanti Fellowship, a Google Research Grant, a Pratiksha Trust Intramural Grant and a Gore Subraya Bhat Chair Associate Professorship in Digital Health (all to DS), Ministry of Education Scholarships (to TG and NM), an Institute of Eminence (IoE) Grant (to IISc), a Walmart Centre for Tech Excellence at IISc, the Indo–French Centre for the Promotion of Advanced Research Grant (CEFIPRA; Project 7102–1), Kotak IISc FinTech Grant, a DST–SERB Core Research Grant (CRG/2021/008330), and a Pratiksha Trust Young Investigator Award (all to GT). We are very grateful to Karm Patel, Mainak Biswas, Aditi Saxena, and Tej Pratap Yadav for help with model development and simulations, and Raj Jain for help with the original draft of the manuscript. The authors declare no competing interests.

References

[1] Amir Alansary, Ozan Oktay, Yuanwei Li, Loic Le Folgoc, Benjamin Hou, Ghislain Vaillant, Konstantinos Kamnitsas, Athanasios Vlontzos, Ben Glocker, Bernhard Kainz, et al. Evaluating reinforcement learning agents for anatomical landmark detection. *Medical Image Analysis*, 53:156–164, 2019.

[2] Shekoofeh Azizi, Laura Culp, Jan Freyberg, Basil Mustafa, Sebastien Baur, Simon Kornblith, Ting Chen, Nenad Tomasev, Jovana Mitrović, Patricia Strachan, et al. Robust and data-efficient generalization of self-supervised machine learning for diagnostic imaging. *Nature Biomedical Engineering*, 7(6):756–779, 2023.

[3] Babak Ehteshami Bejnordi, Mitko Veta, Paul Johannes Van Diest, Bram Van Ginneken, Nico Karssemeijer, Geert Litjens, Jeroen AWM Van Der Laak, Meyke Hermsen, Quirine F Manson, Maschenka Balkenhol, et al. Diagnostic assessment of deep learning algorithms for detection of lymph node metastases in women with breast cancer. *JAMA*, 318(22):2199–2210, 2017. URL `https://camelyon16.grand-challenge.org/`.

[4] Nadia Brancati, Anna Maria Anniciello, Pushpak Pati, Daniel Riccio, Giosuè Scognamiglio, Guillaume Jaume, Giuseppe De Pietro, Maurizio Di Bonito, Antonio Foncubierta, Gerardo Botti, et al. Bracs: A dataset for breast carcinoma subtyping in h&e histology images. *Database*, 2022:baac093, 2022.

[5] Kausik Das, Sailesh Conjeti, Abhijit Guha Roy, Jyotirmoy Chatterjee, and Debdoot Sheet. Multiple instance learning of deep convolutional neural networks for breast histopathology whole slide classification. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 578–581. IEEE, 2018.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[8] Krishnamurthy Dvijotham, Jim Winkens, Melih Barsbey, Sumedh Ghaisas, Robert Stanforth, Nick Pawlowski, Patricia Strachan, Zahra Ahmed, Shekoofeh Azizi, Yoram Bachrach, et al. Enhancing the reliability and accuracy of ai-enabled diagnosis via complementarity-driven deferral to clinicians. *Nature Medicine*, 29(7):1814–1820, 2023.

[9] Michael Gadermayr and Maximilian Tschuchnig. Multiple instance learning for digital pathology: A review of the state-of-the-art, limitations & future potential. *Computerized Medical Imaging and Graphics*, 112:102337, 2024.

[10] Baris Gecer, Selim Aksoy, Ezgi Mercan, Linda G Shapiro, Donald L Weaver, and Joann G Elmore. Detection and classification of cancer in whole slide breast histopathology images using deep convolutional networks. *Pattern Recognition*, 84:345–356, 2018.

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[12] Zabit Hameed, Sofia Zahia, Begonya Garcia-Zapirain, Jose Javier Aguirre, and Ana Maria Vanegas. Breast cancer histopathology image classification using an ensemble of deep learning models. *Sensors*, 20(16): 4373, 2020.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[14] Le Hou, Dimitris Samaras, Tahsin M Kurc, Yi Gao, James E Davis, and Joel H Saltz. Patch-based convolutional neural network for whole slide tissue image classification. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2424–2433, 2016.

[15] Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International Conference on Machine Learning*, pages 2127–2136. PMLR, 2018.

[16] Eleanor Jenkinson and Ognjen Arandjelović. Whole slide image understanding in pathology: what is the salient scale of analysis? *BioMedInformatics*, 4(1):489–518, 2024.

[17] Mingu Kang, Heon Song, Seonwook Park, Donggeun Yoo, and Sérgio Pereira. Benchmarking self-supervised learning on diverse pathology datasets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3344–3354, 2023.

[18] Bin Li, Yin Li, and Kevin W Eliceiri. Dual-stream multiple instance learning network for whole slide image classification with self-supervised contrastive learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14318–14328, 2021.

[19] Jiayun Li, Wenyuan Li, Anthony Sisk, Huihui Ye, W Dean Wallace, William Speier, and Corey W Arnold. A multi-resolution model for histopathology image classification and localization with multiple instance learning. *Computers in Biology and Medicine*, 131:104253, 2021.

[20] Xuan Liao, Wenhao Li, Qisen Xu, Xiangfeng Wang, Bo Jin, Xiaoyun Zhang, Yanfeng Wang, and Ya Zhang. Iteratively-refined interactive 3d medical image segmentation with multi-agent reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9394–9402, 2020.

[21] Xitong Ling, Yuanyuan Lei, Jiawen Li, Junru Cheng, Wenting Huang, Tian Guan, Jian Guan, and Yonghong et al. He. A comprehensive benchmark dataset for pathological lymph node metastasis in breast cancer: Camelyon+. *Scientific Data*, 12:1381, 2025. doi: 10.1038/s41597-025-05586-5. URL `https://doi.org/10.1038/s41597-025-05586-5`. Dataset available via ScienceDB: CSTR:31253.11.sciencedb.16442.

[22] Ming Y Lu, Drew FK Williamson, Tiffany Y Chen, Richard J Chen, Matteo Barbieri, and Faisal Mahmood. Data-efficient and weakly supervised computational pathology on whole-slide images. *Nature Biomedical Engineering*, 5(6):555–570, 2021.

[23] Gabriel Maicas, Gustavo Carneiro, Andrew P Bradley, Jacinto C Nascimento, and Ian Reid. Deep reinforcement learning for active breast lesion detection from dce-mri. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 665–673. Springer, 2017.

[24] Y Lu Ming, Chen Bowen, FK Williamson Drew, J Chen Richard, and Ivy Liang. Towards a visual-language foundation model for computational pathology. *arXiv preprint*, 2023.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[26] Cancer Genome Atlas Research Network et al. Comprehensive molecular profiling of lung adenocarcinoma. *Nature*, 511(7511):543, 2014. URL `https://www.cancer.gov/ccg/research/genome-sequencing/tcga`.

[27] Linhao Qu, Siyu Liu, Xiaoyu Liu, Manning Wang, and Zhijian Song. Towards label-efficient automatic diagnosis and analysis: a comprehensive survey of advanced deep learning-based weakly-supervised, semi-supervised and self-supervised techniques in histopathological image analysis. *Physics in Medicine & Biology*, 67(20):20TR01, 2022.

[28] Kaushiki Roy, Debapriya Banik, Debotosh Bhattacharjee, and Mita Nasipuri. Patch-based system for classification of breast histology images using deep learning. *Computerized Medical Imaging and Graphics*, 71:90–103, 2019.

[29] Jun Ruan, Zhikui Zhu, Chenchen Wu, Guanglu Ye, Jingfan Zhou, and Junqiu Yue. A fast and effective detection framework for whole-slide histopathology image analysis. *Plos One*, 16(5):e0251521, 2021.

[30] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[31] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL `https://arxiv.org/abs/1707.06347`.

[32] Zhuchen Shao, Hao Bian, Yang Chen, Yifeng Wang, Jian Zhang, Xiangyang Ji, et al. Transmil: Transformer based correlated multiple instance learning for whole slide image classification. *Advances in Neural Information Processing Systems*, 34:2136–2147, 2021.

[33] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[34] Chetan L Srinidhi, Ozan Ciga, and Anne L Martel. Deep neural network models for computational histopathology: A survey. *Medical Image Analysis*, 67:101813, 2021.

[35] Joseph N Stember and Hrithwik Shalu. Reinforcement learning using deep q networks and q learning accurately localizes brain tumors on mri with very small training sets. *BMC Medical Imaging*, 22(1):224, 2022.

[36] Kevin Thandiackal, Boqi Chen, Pushpak Pati, Guillaume Jaume, Drew F. K. Williamson, Maria Gabrani, and Orcun Goksel. Differentiable zooming for multiple instance learning on whole-slide images. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXI*, page 699–715, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19802-1. doi: 10.1007/978-3-031-19803-8_41. URL `https://doi.org/10.1007/978-3-031-19803-8_41`.

[37] Jing Wang, Zhe Xu, Zhi-Feng Pang, Zhanqiang Huo, and Junwei Luo. Tumor detection for whole slide image of liver based on patch-based convolutional neural network. *Multimedia Tools and Applications*, 80: 17429–17440, 2021.

[38] Hongrun Zhang, Yanda Meng, Yitian Zhao, Yihong Qiao, Xiaoyun Yang, Sarah E Coupland, and Yalin Zheng. Dtfd-mil: Double-tier feature distillation multiple instance learning for histopathology whole slide image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18802–18812, 2022.

[39] Yunlong Zhang, Honglin Li, Yunxuan Sun, Sunyi Zheng, Chenglu Zhu, and Lin Yang. Attention-challenging multiple instance learning for whole slide image classification. In *European Conference on Computer Vision*, pages 125–143. Springer, 2024.

[40] Boxuan Zhao, Jun Zhang, Deheng Ye, Jian Cao, Xiao Han, Qiang Fu, and Wei Yang. Rlogist: fast observation strategy on whole-slide images with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 3570–3578, 2023.

[41] Tingting Zheng, Kui Jiang, and Hongxun Yao. Dynamic policy-driven adaptive multi-instance learning for whole slide image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8028–8037, 2024.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction summarize the paper's contribution by differentiating previous approaches from our method. We concisely describe key components of our method – HAFED, TSU and RL Agent – and outline the key role of each component in Section 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss our model's limitations briefly in Section 5 (Conclusions and Limitations). A more extended discussion is provided in Appendix A.6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: The paper does not include any theoretical results. It focuses on model design and empirical evaluation of the proposed method on four datasets.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The key components for the proposed method are described in Section 3.2 and Section 3.3. Details regarding SASHA RL Agent is provided in Section 3.4, with the details about the policy and value network is mentioned in Appendix A.2.3. For the dataset, details are provided in Appendix A.1 and hyper-parameter selection in Appendix A.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The datasets used in experiments study CAMELYON16, TCGA-NSCLC lung cancer, CAMELYON+ and BRACS datasets are publicly available. The codebase provides step by step instructions covering from preprocessing to final evaluation, enabling reproducibility across different seeds. Hyper-parameters details as mentioned in Appendix A.2. The implementation of SASHA is publicly available at [1].

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The details regarding the train-test splits are provided in Appendix A.1 and hyper-parameters and optimizer details are in Appendix A.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Statistical tests are conducted to evaluate patch selection strategy in Section 4.5 and assess the effect of the observation budget in Section 4.6. All values reported in Table 1 are reported as a mean across 3-5 seeds along with the standard deviation across seeds.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The details of the GPU used for simulation and time of execution required for inference is mentioned in Appendix A.5.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The datasets used in the experimental studies CAMELYON16, TCGA-NSCLC lung cancer, CAMELYON+ and BRACS datasets (Section 4.1, Appendix A.1) are publicly available. The original distributors of the datasets have obtained the required ethics approvals for their public distribution.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.

- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The proposed method has potential to assist histopathalogist to diagnose the patient whole slide images (WSIs). The compressibility of WSIs can help to reduce the storage cost while ensuring the reproducibility of results. We have also discussed the importance of explainability, reliability and model calibration to build clinicians' trust in model predictions in Section 4.5 and Section 4.6.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: In our experimental studies we have used publicly available datasets. We do not envision significant potential for misuse of our models.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: Details of the baseline methods used for comparison in our experiments are clearly described in Section 2 and Section 4.2 and sources of all publicly available datasets are provided.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Code base is publicly available at [1], including model details of the proposed method, along with hyperparameters, and configuration files to support reproducibility.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This work does not involve any crowdsourcing or original research with human subjects. Sources of publicly available cancer datasets and their usage terms are described, where appropriate.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve any original research with human subjects. Ethical approvals for acquiring the cancer datasets as well as informed consent from human participants for their redistribution have been obtained by the respective dataset providers (CAMELYON16, TCGA-NSCLC, CAMELYON+ and BRACS, respectively).

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core methodology of the paper does not involve the use of large language models (LLMs).

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

Sequential Attention-based Sampling for Histopathological Analysis

A Appendix

A.1 Dataset details

The CAMELYON16 dataset comprises 399 whole-slide images (WSIs) of lymph node sections. After pre-processing and segmentation, we generated non-overlapping patches of size 256×256 ($\times 3$ RGB channels). On average, this resulted in, on average, $\sim$800 patches per WSI at a scanning resolution of $5\times$, and $\sim$12,500 patches at a high resolution of $20\times$. We used the official test set for evaluation [3]. The training set was further divided to create a validation set (10%) while maintaining a similar proportion of normal and tumor slides in each subset.

The TCGA-NSCLC dataset comprises 1,054 whole-slide images (WSIs) obtained from the TCGA-NSCLC repository under the TCGA-LUSC and TCGA-LUAD projects. Ten low-quality slides without magnification information were discarded, and three slides with errors were ignored. We followed the same test split as in DSMIL [18]. The test set comprised a total of 213 slides: 104 from TCGA-LUAD and 109 from TCGA-LUSC. Following preprocessing, we obtained, on average, $\sim$212 patches per WSI at $5\times$ magnification and $\sim$3,393 patches at $20\times$ magnification. The training set was further divided to create a validation (20%) set while maintaining a similar proportion of labels in each subset.

Dataset	Task	Train		Val		Test	
		Class 1	Class 2	Class 1	Class 2	Class 1	Class 2
CAMELYON16	Classify tumor	140	102	18	9	80	49
TCGA-NSCLC	Classify sub-type	339	319	86	81	104	109

Table 3: Summary of datasets used in this study. CAMELYON16 involves binary classification of normal versus tumor slides, while TCGA-NSCLC involves distinguishing between adenocarcinoma (LUAD) and squamous cell carcinoma (LUSC). The table shows the number of WSIs for each class in the training, validation, and test sets. Here, "Class 1" refers to Normal (for CAMELYON16) and LUAD (for TCGA-NSCLC), while "Class 2" refers to Tumor (for CAMELYON16) and LUSC (for TCGA-NSCLC).

The CamelyonPlus dataset[21], comprises 1,347 whole-slide images (WSIs) derived from the curated CAMELYON16 and CAMELYON17 collections, after removal of low-quality slides and re-labeling. It involves a four-class classification. We created training, validation, and test splits to ensure class balance across all four labels. A summary of the splits and labels can be found in Table 4.

Dataset	Train				Val				Test			
	Neg.	Mi-m.	Ma-m.	ITC	Neg.	Mi-m.	Ma-m.	ITC	Neg.	Mi-m.	Ma-m.	ITC
Camelyon+	608	122	174	36	130	26	38	9	131	26	38	9

Table 4: Summary of Camelyon+ dataset. The WSIs have the following 4-class slide-level labels: Negative (Neg.), Micro-metastasis (Mi-m.), Macro-metastasis (Ma-m.), and Isolated Tumor Cells (ITC). The table shows the number of WSIs for each class in the training, validation, and test sets.

The BRACS (BReAst Carcinoma Subtyping) dataset comprises 547 labeled WSIs obtained from the public repository [4]. This dataset involves a 7-class classification task, characterizing each tumor as one of three benign (BT), two atypical (AT), or two malignant (MT) sub-types. We ignored the "normal" subtype in the benign (BT) class, and each set of subtypes within BT, AT, and MT was grouped into its respective class. Additionally, a small number of slides (7/503) that lacked all relevant levels of magnification were excluded. After preprocessing, we use the official dataset split across train, validation, and test split with details outlined in Table 5.

	Benign (BT)	Atypical (AT)	Malignant (MT)
Train	171	52	135
Val	16	14	21
Test	25	23	16

Table 5: BRACS dataset distribution for 3-class classification (Benign, Atypical, Malignant) across training, validation, and test splits.

A.2 Implementation and Hyper-parameter Selection

A.2.1 Patch feature distillation with HAFED

The first step of SASHA involves feature extraction and aggregation of high-resolution patch features, which specifies the RL agent's state $S(t)$. We use a Vision Transformer (ViT) pretrained on histopathological images to extract features from each zoomed in, high-resolution patch [17]. This ViT was used *as is* without any additional fine tuning, and its output – a 384 dimensional embedding per patch – was provided as input to a hierarchical attention-based feature aggregator and distiller (HAFED).

Layer name	Input Dim	Output Dim	Input Batch	Output Batch
Layer-1	384	128	$U_i = \{U_{i1}, U_{i2}, \ldots, U_{ik}\}$	$A_i = \{a_{i1}, a_{i2}, \ldots, a_{ik}\}$
Layer-2	384	128	$U_i = \{U_{i1}, U_{i2}, \ldots, U_{ik}\}$	$B_i = \{b_{i1}, b_{i2}, \ldots, b_{ik}\}$
Layer-3	128	1	$\text{sigmoid}(A) \cdot \tanh(B)$	${\alpha_1}^i \in \mathbb{R}^k$
$V_i = {\alpha_1}^i \cdot U_i, \quad V_i \in \mathbb{R}^d$				
Layer-4	384	128	$V = \{V_1, V_2, \ldots, V_N\}$	$C = \{c_1, c_2, \ldots, c_N\}$
Layer-5	384	128	$V = \{V_1, V_2, \ldots, V_N\}$	$D = \{d_1, d_2, \ldots, d_N\}$
Layer-6	128	M (5)	$\text{sigmoid}(C) \cdot \tanh(D)$	$\alpha_2 \in \mathbb{R}^{N \times M}$
$\alpha = \frac{1}{M} \sum_{j=1}^{M} \alpha_2^{(:,j)}, \quad \alpha \in \mathbb{R}^N; \quad h = \alpha \cdot V, \quad h \in \mathbb{R}^d$				
Layer-7	384	2	h	logits

Table 6: Architecture of the HAFED model. Layers 1-3 correspond to the feature aggregator (FA) and layers 4-7 correspond to the classifier. N ($\sim$800 for camelyon-16) is the number of patches in low-resolution WSI, k (16) is the number of sub-patches in high-resolution per one patch in low-resolution, d (384) feature embedding dimension of one patch. $U_i \in \mathbb{R}^{16 \times 384}$, $A_i \in \mathbb{R}^{16 \times 128}$, $B_i \in \mathbb{R}^{16 \times 128}$, $C \in \mathbb{R}^{N \times 128}$, $D \in \mathbb{R}^{N \times 128}$, $h \in \mathbb{R}^{384}$.

As described in Section 3.2, the first stage of HAFED acts as a feature aggregator (FA); the architecture of this network is shown in Table 6. We employed a single attention branch, which takes high resolution features $U \in \mathbb{R}^{N \times k \times d}$ and assigns attention scores for each of the k sub patches. The inner product between the attention scores and U produces $V \in \mathbb{R}^{N \times d}$, which is in same dimensionality as the low-resolution features ($Z \in \mathbb{R}^{N \times d}$). The second stage of HAFED acts as a classifier, which is trained end-to-end with the feature aggregator. Here, we use five attention branches for the CAMELYON16 dataset and one for the TCGA-NSCLC dataset; the number of branches was determined as a hyperparameter based on the respective validation set (Table 7). This stage takes V as input, assigns attention scores across the N patches, and computes the inner product between the scores and V to produce a 384-dimensional embedding representing the entire WSI. The architecture of this network is shown in Fig. 2a. We applied masking to the top four and top eight patches with a probability of 0.4 in the first and second stages, respectively (Table 7).

HAFED is trained with an annealed learning rate schedule starting at 4 $\times 10^{-4}$, with the AdamW optimizer and a weight decay of 1 $\times 10^{-4}$. The HAFED module is trained end-to-end using the following loss components: (1) a cross-entropy loss between the final predicted label and the ground truth label, (2) a cross-entropy loss applied independently to the predictions from each attention branch in second stage and ground truth label, and (3) a dissimilarity loss that encourages diversity among the attention scores of the different branches in the second stage. This dissimilarity loss encourages each branch to focus on distinct tumor features, thereby improving the model's ability to capture complementary diagnostic features[39].

During inference, at each time step, the first stage of HAFED (FA), receives as input k zoomed-in sub-patch images corresponding to the selected action. It assigns attention scores to estimate the relative importance of each sub-patch and aggregates their features by computing a weighted sum based on these scores; this is then used to update the state (see next subsection, Appendix A.2.2). In the final time step, the second stage of HAFED (classifier), takes the state representation S_T and predicts the label $\hat{y}$. In addition to the final classification, the second stage is also used to assign rewards during SASHA' RL agent training (see subsection Appendix A.2.3).

Model	**Hyper-parameters**	**Selection Criterion**	**SASHA - 0.1 / 0.2**			
			CAMELYON16		**TCGA-NSCLC**	
			Range	**Values**	**Range**	**Values**
HAFED	Attn-2 in HAFED (M)	Validation	1-5	5	1-5	1
	Attn -1 Masking Probability	Validation	0.4 - 0.6	0.4	0.4 - 0.6	0.4
	Attn -2 Masking Probability	Validation	0.4 - 0.6	0.4	0.4 - 0.6	0.4
TSU	Cosine Threshold in TSU (τ)	Validation	0.88 - 0.98	0.9 / 0.95	0.8 - 0.92	0.9/0.9
Actor & Critic	Discount Factor	Validation	0.8-1.0	1.0	0.8-1.0	1.0
	Clipping Parameter	Default	-	0.1	-	0.1
	Entropy Regularization Coefficient	Default	-	0.001	-	0.001
	Gradient Clipping Threshold	Default	-	0.5	-	0.5

Table 7: Hyperparameters for the CAMELYON16 and TCGA-NSCLC datasets. In the Values column, entries separated by a slash (/) represent values for SASHA-0.1 and SASHA-0.2 models respectively; a singleton entry reflects the same value across both SASHA variants.

A.2.2 Updating WSI state with TSU

The second step of SASHA involves updating the RL agent's state with the high-resolution patch representation extracted with HAFED. During training, the states of similar patches are concurrently updated with a targeted state update (TSU) rule. Specifically, this included all patches whose low-resolution representation's similarity to the currently selected patch, exceeded a threshold value (τ), based on cosine similarity; τ was tuned as a hyperparameter (Table 7)

The architecture of this TSU network is shown in Table 8. For each patch a_τ that crosses the similarity threshold, the TSU model takes as input a concatenated vector comprising (i) its low-resolution features $Z_{a_\tau} \in \mathbb{R}^d$, (ii) the low-resolution feature representation of the selected patch a_t, $Z_{a_t} \in \mathbb{R}^d$ and (iii) the high-resolution feature representation of the selected patch $V_{a_t} \in \mathbb{R}^d$. Note that while (i-ii) are obtained directly from pretrained ViT, (iii) is obtained from the FA output of the HAFED model. Concatenating these 3 features results in a $3d$-dimensional input $(3 \times 384 = 1152)$, based on which the model predicts a d-dimensional high-resolution feature of the patch a_τ, $\hat{V}_{a_\tau}$. The state of the selected patch alone is updated by directly copying its high-resolution HAFED (FA) output V_{a_t}, into its (corresponding) entry in the WSI state $S(t)$. In this manner, the TSU integrates information from the low-resolution representation of patches similar to the currently visited patch, with the high-resolution representation of visited patches to update the WSI state.

During training the TSU network is trained by sampling patches randomly, and updating their states based on the predicted high-resolution feature of each patch ($\hat{V}$) and its actual high-resolution feature (V) obtained from HAFED's feature aggregator. Once a patch state is updated – either directly, by visiting it, or indirectly, by the similarity-based rule – its index is masked out, and it is no longer visited during training. The TSU model is trained for 500 epochs with an Adam optimizer and a learning rate of 1 $\times 10^{-4}$. The best model was chosen based on the minimum of the validation loss.

During inference, the low-resolution patch to visit at each time step a_t is selected by RL agent. The state of patches similar to the currently visited patch are updated with the same sequence of steps as described for the training. A key difference is that, in this case, only the directly visited patch indices are masked out, and the RL agent is free to select, for its action in the next time step ($t + 1$), any of the patches that have not been visited directly in the previous time steps ($0, \ldots, t$).

Layer	**Input Dim**	**Output Dim**	**# Parameters**
FC-1	1152	768	885,504
FC-2	768	384	295,296
Layer Norm	-	384	384

Table 8: Architecture of the TSU model. The input is a $3d = 1152$ dimensional vector ($d = 384$; see text for details). The final output is d-dimensional. FC: fully connected layer with bias.

A.2.3 Actor and Critic

The final component of SASHA is a reinforcement learning (RL) agent responsible for sampling, sequentially, the most relevant patches for high-resolution analysis. This selection is formulated as a sequential decision-making problem, where the agent learns to identify diagnostically informative regions within each slide. We adopt the Actor-Critic framework and train the agent using the Proximal Policy Optimization (PPO) algorithm, a widely used on-policy method known for its stability and effectiveness in complex RL tasks [31].

The purpose of the actor network is to learn a policy – a conditional probability distribution over actions given the current state S_t. In our setup, the environment is described as, the state is initialized with low-resolution features ($Z \in \mathbb{R}^{N \times d}$), the action is selected from a discrete action space $A \in \{1, 2, \ldots, N\}$ (indices of low-resolution patches), and the reward function is specified as the negative of cross-entropy (CE) loss between the predicted WSI label, at time step t, and the ground truth. Given the definitions of state, action and reward, the actor and critic networks are designed as follows.

The Actor network follows the same architecture as Layers 1–3 in Table 6. A softmax is applied to the output attention scores to obtain a categorical distribution (i.e., the policy) over the N patches, from which an action a_t (i.e., a patch index) is sampled during training. To reduce redundancy, once a patch is selected, its index is masked in all subsequent time steps – by setting its corresponding logit to negative infinity – effectively removing it from consideration for future visits. the rationale behind this policy is that zooming into the same patch multiple times will not yield additional information, as the patch content remains unchanged. After selecting a_t, the Target State Updater (TSU) module is invoked to update the current state S_t to the next state S_{t+1}, thereby incorporating the high-resolution information from the selected patch into the state representation.

The Critic network also mirrors the backbone of actor network i.e architecture of Layers 1–3 in Table 6, with an additional final layer that takes the aggregated state representation ($\alpha \cdot S_t$), and predicts the reward-to-go or expected return given the current state S_t. The reward-to-go is used in calculating the generalized advantage estimate (GAE, [30]). During training the actual reward-to-go from the given state is known, and the critic network is trained by minimizing MSE loss between the predicted and actual value.

Both networks are trained for 15 epochs using the AdamW optimizer with an annealing learning rate starting from 1×10^{-5} and a weight decay of 1×10^{-3}. Network hyperparameters are reported in Table 7. During training of the Actor and Critic networks, all other components of SASHA are frozen (i.e., HAFED, TSU), and only the actor and critic networks are updated. We select the best policy across epochs as the one that minimizes the validation loss while simultaneously maximizing the sum of the validation AUROC and F1 scores ($\min_\pi[\text{Loss} + 2 - (\text{AUROC} + \text{F1})]$). This formulation ensures that the selected policy balances both accuracy (AUROC and F1) and calibration (validation loss).

During inference, given the current state S_t, the Actor network outputs a policy in the form of a categorical probability distribution over the N patches. The action a_t is then selected deterministically as the patch index with the highest probability. In a later section (Appendix A.3.3) we explore the effect of alternative, stochastic selection policies during inference.

A.3 Additional Control Experiments

A.3.1 Multiclass Classification Performance of SASHA

We extended the SASHA framework to evaluate its performance on multiclass classification tasks using the Camelyon+ and BRACS datasets (see Appendix A.1 for a description of the datasets).

For both datasets, the SASHA framework was trained using experimental settings identical to those of the binary classification tasks. The same patch-level sampling strategy and aggregation mechanism were employed to ensure consistency in evaluation. To mitigate class imbalance, we employed a resampling mechanism that oversamples minority classes (e.g., ITC for the Camelyon+ dataset) during training.

The performance comparison (Table 9) shows that the methods (HAFED, SASHA) perform comparably with other sota methods, even in this multiclass setting. As before, sampling only 20% of the patches (SASHA-0.2) achieves performance that is comparable to methods that sample all of the WSI patches at high resolution (e.g., DTFD, ACMIL).

	Method		CAMELYON+			BRACs	
		Accuracy	AUC	F1	Accuracy	AUC	F1
100%	DTFD	0.848±0.022	0.921±0.023	0.804±0.04	0.570±0.035	0.782±0.032	0.570±0.035
	ACMIL	0.832±0.013	0.898±0.005	0.786±0.023	0.534±0.046	0.749±0.014	0.596±0.059
	HAFED (ours)	0.866±0.014	0.934±0.01	0.847±0.015	0.544±0.03	0.747±0.01	0.575±0.036
	SASHA-0.1 (ours)	0.869±0.008	0.905±0.003	0.848±0.013	0.596±0.001	0.729±0.001	0.532±0.010
	SASHA-0.2 (ours)	0.871±0.005	0.935±0.006	0.852±0.006	0.598±0.016	0.762±0.010	0.533±0.079

Table 9: Comparison of multiclass classification performance of different methods on the Camelyon+ and BRACS datasets. Reported metrics include weighted-averaged Accuracy, AUC, and F1-score.

A.3.2 Alternative Models under an Observation Budget

We compared the performance of SASHA with competing sota models like ACMIL [39] and DTFD [38] under the same observation budget of either 10% or 20%. That is, we predicted slide labels by having these sota models examine only a limited fraction of patches in each WSI at high resolution, matching the fraction of the WSI sampled by SASHA.

Each model was trained with a combination of low-resolution and high-resolution patch features and tested, similarly, with a combination of high and low resolution patch features in the same proportion as of training. Specifically, for a given slide, if N is the total number of low-resolution patches and each low-resolution patch corresponds to k high-resolution patches, then for that slide, the input comprised of a total of $N_L + N_H$ patches comprising $N_L = (N - \lceil f \cdot N \rceil)$ low-resolution and $N_H = \lceil f \cdot N \rceil \cdot k$ high-resolution patches, where $f \in \{0.1, 0.2\}$. Because low and high-resolution patches were of the same pixel dimensionality $(256, 256, 3)$, the pretrained ViT model (Section 3) was used to extract features from each such patch; this yielded a feature vector $W \in \mathbb{R}^{(N_L+N_H)\times d}$ which was provided as input to each model for label prediction, both during training and evaluation.

We tested these models on the CAMELYON16 dataset. First, we created five random splits of the train and validation sets of the dataset. For each dataset split, we performed random patch sampling with three different random seeds. Then, for each model, we computed the average performance across these 15 (5 splits $\times$ 3 seeds) model combinations. In each case, the sum of validation accuracy and F1 scores was used to determine the model for evaluation.

As shown in Table 10, such a strategy of sparsely sampling patches substantially reduced the performance of ACMIL and DTFD models. In each case, SASHA sampling either 10% (0.1 fraction) or 20% (0.2 fraction) of the patches at high resolution outperformed the sota models with the corresponding, identical observation budget. In other words, competing sota models failed to live up to SASHA's performance under a comparable observation budget for high-resolution patches.

A.3.3 Alternative Action Policies during Inference

In the current SASHA implementation, the inference strategy follows a deterministic, greedy policy wherein, for a given state, the action with the highest probability is always selected. This means the

Model	Accuracy	AUC	F1
ACMIL - 0.1	0.841±0.036	0.828±0.034	0.773±0.038
DTFD - 0.1	0.829±0.013	0.835±0.039	0.829±0.013
SASHA - 0.1	**0.901±0.021**	**0.918±0.014**	**0.856±0.031**
ACMIL - 0.2	0.838±0.035	0.857±0.030	0.773±0.043
DTFD - 0.2	0.862±0.027	0.878±0.030	0.862±0.027
SASHA - 0.2	**0.953±0.017**	**0.979±0.008**	**0.937±0.024**

Table 10: Performance of ACMIL (rows 1, 4), DTFD (rows 2, 5) and SASHA (rows 3, 6) models under a limited observation budget of 10% (-0.1) or 20% (-0.2) for high-resolution patches per WSI (see text for details). Values represent averages and standard deviations on the CAMELYON16 dataset across 15 model combinations (5 random data splits $\times$ 3 random seeds for sampling per split).

agent always picks the most likely action as per the learned policy distribution during inference (but not during training, see Appendix A.2.3).

We evaluate the effectiveness of this policy over other stochastic policies, similar to the ones used during learning. Specifically, we introduce more exploration by considering stochastic policy variation, where the agent samples actions from the entire probability distribution over actions given a state.

We explore two different variants – top-k sampling and top-p sampling. In top-k sampling, where only the top k actions with the highest probabilities are retained as candidates for stochastic selection, and the remaining actions are are masked out (assigned zero probability). The agent then samples from this limited distribution, following renormalization. In top-p sampling, actions are sorted in descending order of their probability, and we identify the smallest set of actions such that their cumulative probability exceeds a predefined threshold p. Sampling is then done from this subset, ensuring a balance between diversity and confidence in action selection.

Comparison of different sampling strategies is shown in Table 11. Interestingly, the default deterministic policy of SASHA – a greedy strategy of selecting the action with the highest probability – outperformed all other stochastic policies. This behavior was observed empirically, where the agent consistently selects action with highest probability given current state S_t, resulting in more informative state transitions from S_t to S_{t+1} and improved classification performance, as shown in Table 11 . In addition, the stochastic policy that had the greatest degree of freedom for sampling (Stochastic - full distribution) outperformed, by and large, the other policies (top-k and top-p); this was understandable because the latter policies impose more constraints on the action space for sampling.

Inference Policies	Accuracy	AUC	F1
Stochastic - full distribution	<u>0.884</u>	0.846	<u>0.831</u>
Stochastic - top-k ($k = 3$)	0.876	0.826	0.814
Stochastic - top-k ($k = 5$)	0.868	0.823	0.805
Stochastic - top-p ($p = 0.9$)	0.876	<u>0.859</u>	0.818
Stochastic - top-p ($p = 0.8$)	0.868	0.851	0.795
Deterministic (SASHA default)	**0.953**	**0.979**	**0.937**

Table 11: Comparison of different stochastic policies (top 5 rows), with SASHA's default deterministic policy, during inference. Values denote the performance of the SASHA-0.2 model on the CAMELYON16 dataset. **Bold** and <u>underlined</u>: best and second best performing policies, respectively.

A.3.4 Alternative Reward Schedules during Training

The default SASHA model employs a step-wise reward during training, where the agent receives a reward at every time step of the episode associated with the label prediction loss at that time step (see Appendix A.2.3). In contrast, a previous competing model (RLogist [40]) employed a terminal reward strategy, providing feedback only at the end of each episode.

We tested this alternative reward schedule employing only a terminal reward at the end of each episode, both during training and inference. As shown in Table 12, the RL agent trained with reward at each time step significantly outperformed an agent trained using terminal rewards. We propose that this occurs because our reward formulation – which is designed to minimize the classification loss at each time step – encourages the model to sample more tumor-relevant patches during inference. Confirming this prediction SASHA's default reward schedule sampled, on average, more patches with a high attention (relevance) score, as compared to the terminal reward schedule (Table 12).

Reward Schedule	**Accuracy**	**AUC**	**F1**	**Top-k attention score patches ∩ RL actions**	**Avg attention score of RL actions**
Last time step (Terminal)	0.892	0.941	0.862	15.5	0.0020
Each time step (SASHA default)	**0.953**	**0.979**	**0.937**	**24.25**	**0.0032**

Table 12: Comparison of a terminal reward schedule (row 1) with SASHA's default schedule of reward at each time step (row 2), during training (see text for details). Column 5 tells the average patches per slide that are selected by RL agent are in top-k according to the attention score given by HAFED model. Column 6 tells the average attention score of the selected patches by RL agent. (see Fig. 3f-g for details). Values denote the performance of the SASHA-0.2 model on the CAMELYON16 dataset. **Bold**: the best performing reward schedule.

A.3.5 Analysis of RL Agent Behavior

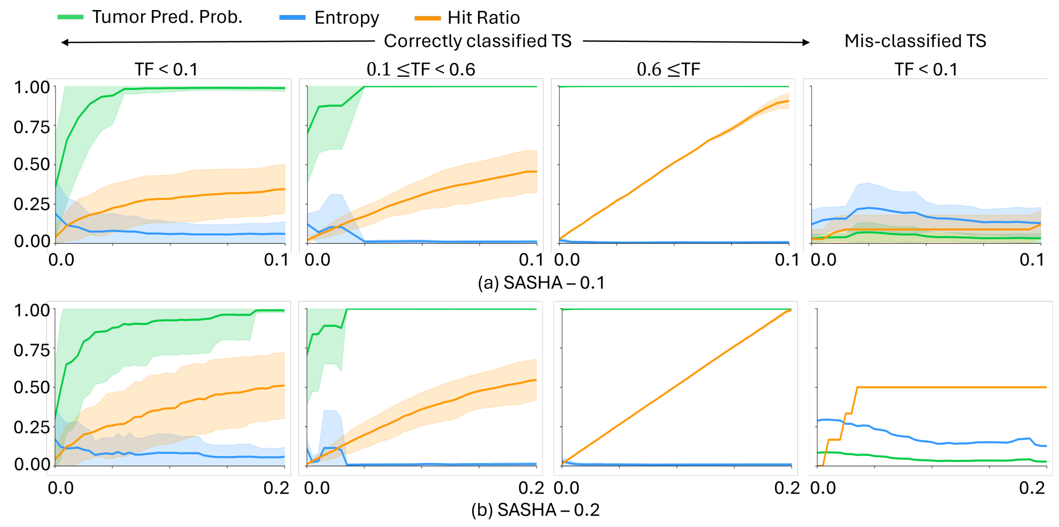

Figure 4: Evaluation of model performance on tumor slides from the test set, categorized into four groups. The first three columns correspond to correctly classified tumor slides (TS) by both SASHA and HAFED across increasing tumor fraction (TF) ranges, while the fourth column represents TS misclassified by SASHA but correctly predicted by HAFED. The x-axis denotes the visited patch fraction, binned with a width of 0.005. For each slide, we compute the average entropy, average tumor class prediction probability, and maximum cumulative tumor patch hit ratio (Eq. (1)) within each bin. These metrics are then averaged across all slides, with solid lines indicating the mean and shaded regions representing the standard deviation.

To better understand the behavior of the RL agent, we perform two additional analyses.

First, we asked: how quickly does the RL agent visit tumor patches, and how does the model's confidence evolve over time steps in an episode? We first performed this analysis for slides that were correctly classified by SASHA, separately for 3 classes of WSIs with varying tumor fractions (TF, Fig. 4, columns 1-3). For this, we defined a cumulative tumor patch Hit Ratio using

$$\text{Hit Ratio}(t) = \frac{\text{Number of tumor patches sampled up to time } t}{\min\left(\text{Observation Budget},\ \text{Total Tumor Patches in WSI}\right)}. \tag{1}$$

In other words, the cumulative hit ratio begins at 0.0 at the start of each episode and would linearly reach 1.0 at the end of each episode for the most efficient sampling strategy, i.e. if every patch sampled by the agent were a tumor patch.

For both variants of the model (SASHA-0.1 and SASHA-0.2), on the slides with very sparse tumor regions (TF<0.1), the classifier's output favoring the tumor class reaches a high value ($\sim$0.8-1.0) even before it vists 5% of all patches at high-resolution (Fig. 4, column 1, green trace). In addition, its predictive entropy drops progressively as more patches are sampled (Fig. 4, column 1, green trace). Surprisingly, even though the peak of the classifier output is reached earlier for SASHA-0.1 as compared to the SASHA-0.2 model, a longer observation budget produces higher accuracies, as well as better calibrated predictions, as shown in the main text Table 1 and Fig. 3-h.

As the budget increases from 10% to 20%, the tumor hit ratio improves correspondingly. With a 10% budget, SASHA tends to misclassify slides with small tumor regions (see F1 score in Table 1), highlighting the importance of increasing the sampling budget. At 20%, SASHA correctly classifies nearly all slides that were correctly predicted by HAFED, demonstrating improved robustness across varying tumor fractions.

As the fraction of tumor patches increases (0.1<TF<0.6 and TF>0.6), the model output favoring the tumor class reaches peak upon visiting progressively miniscule fractions of high-resolution patches (<0.025% or 0.01%, Fig. 4, columns 2-3, green traces). The predictive entropy also goes to near zero at these fractions indicating confident model predictions with very few samples.

For Fig. 4, column 2—where the tumor fraction (TF) lies in the range $0.1 \leq$ TF < 0.6—the tumor hit ratio increases slightly as the budget increases. However, the improvement is minimal because most tumor patches are already captured within the initial 10% of patches selected by the RL agent. This suggests that for WSIs with moderate tumor presence, visiting just 10% of the patches is often sufficient.

Second, we asked: Is the agent able to localize tumor regions even in slides that are ultimately misclassified? Answering this question is important, because robust tumor localization ability could potentially assist histopathologists by prioritizing regions for first-pass analysis in clinical or laboratory settings. For this analysis, we considered slide that were misclassified by SASHA but correctly classified by HAFED.

We illustrate this with an example slide Fig. 5. For this slide, the RL agent successfully identifies tumor patches, with the first tumor patch being visited at a fraction of 0.0085 and the last at 0.03. For the entire cohort of such slides, tumor patch visits ranged from $0.001 - 0.099$ fraction for SASHA-0.1 and $0.009 - 0.034$ fraction for SASHA-0.2 (Fig. 4, column 4, orange trace). In contrast, when patches are randomly sampled for the same slide, the first tumor patch is typically sampled after visiting 0.141 ± 0.121 fraction of the patches. This demonstrates that the RL agent is able to localize tumor regions, typically with an order of magnitude fewer steps than random sampling.

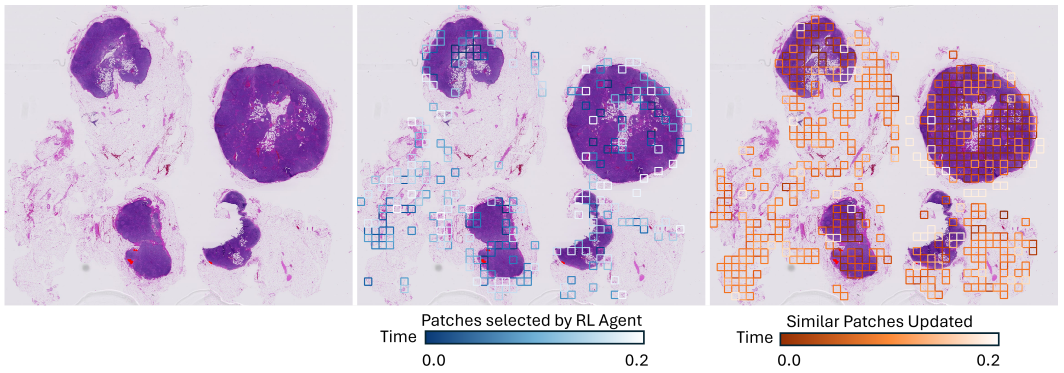

Figure 5: Visualization of the observation path traced by the RL agent for a WSI. The first image shows the tumor-annotated region in the WSI. The second image illustrates the patches selected by the RL agent from the initial to the terminal time steps. The third image presents the similar patches updated by the TSU model.

Method	**Patch**	**FE(HR)**	**FE(LR)**	**Infer**	**TSU**	**RL**	**Total**
DTFD	1.46	116.75	N/A	0.0230	N/A	N/A	116.77
ACMIL	1.46	116.75	N/A	0.0140	N/A	N/A	116.76
HAFED	1.46	116.75	N/A	0.0010	N/A	N/A	116.75
SASHA-0.1	1.46	8.60	4.74	0.0007	2.06	0.105	15.50
SASHA-0.2	1.46	18.23	4.95	0.0007	4.09	0.219	27.49

Table 13: Comparison of inference time for different methods, including the time involved in each computational step. "Patch" denotes the time for dividing the slide into patches, FE denotes Feature Extraction time, with HR representing High Resolution and LR representing Low Resolution. "Infer" refers to the inference time for a single model pass. TSU corresponds to the time required for state update, and RL indicates the time needed for agent policy and value updates. The "Total" time (last column) corresponds to the values shown in Fig. 3c, which does not include the fixed "Patch" time overhead. For SASHA, the value in "Infer." corresponds specifically to the time for one forward pass through the HAFED model.

A.4 Computation of WSI Compressibility, Inference time, and Calibration Error

Whole Slide Images (WSIs) are typically acquired at gigapixel resolution. For laboratories or hospitals maintaining long-term archives, this poses a significant storage challenge. However, we propose that, for accurate tumor diagnosis, it is not necessary to store WSIs as raw images. By storing them as informative feature representations, a considerable reduction in storage requirements can be achieved.

To quantify the data reduction achieved during WSI processing, we define the *WSI compressibility factor* as the ratio between the original image size at full magnification and the size of the final feature representation used for classification. We define the raw image size as $W \times H \times 3$, representing the width, height, and RGB color channels, respectively.

Models such as ACMIL, DTFD, and HAFED operate on high-resolution feature tensors $U \in \mathbb{R}^{N \times k \times d}$, where N is the number of low-resolution patches, k (16) is the number of sub-patches in high-resolution per low-resolution patch, and d (384) is the feature dimension. In contrast, SASHA operates on an initial low-resolution feature set $Z \in \mathbb{R}^{N \times d}$ and iteratively refines it by zooming into diagnostically relevant patches. The final state representation $S_T \in \mathbb{R}^{N \times d}$ can be archived as a compact summary of the WSI, requiring approximately 16 times less storage compared to methods like ACMIL, DTFD, or HAFED, Fig. 3d.

The compressibility factor C is defined as:

$$C = \frac{\text{Image Size}}{\text{Feature Space}} = \begin{cases} \frac{W \times H \times 3}{N \times k \times d} & \text{for ACMIL, DTFD, HAFED} \\ \frac{W \times H \times 3}{N \times d} & \text{for SASHA} \end{cases}$$

Empirically, the compressibility factor is $\sim 5 \times 10^4$ for ACMIL, DTFD, and HAFED, and $\sim 10^6$ for SASHA-0.1 and SASHA-0.2. This indicates that SASHA methods yield significantly more compact feature representations for classification tasks.

Inference time computation: As shown in Figure 3c, SASHA, with its selective sampling strategy, achieves the lowest inference time compared to models such as ACMIL, DTFD, and HAFED, which sample all the patches at high resolution. We report the average inference time per slide for the CAMELYON16 test dataset. The inference time reported in Fig. 3c includes all steps: pre-processing, feature extraction, and classification. Table 13 shows the inference time involved in each step. The column "Patch" refers to the time required to divide the WSI at low resolution into non-overlapping 256$\times$256 patches, while also removing background content such as regions with only the glass slide or insufficient biological tissue. This step is a constant overhead across all methods and is not included in the Total time in Table 13. Feature extraction of high-resolution patches reflects a major inference time bottleneck. Because SASHA visits only a small fraction (10-20%) of patches at high resolution, this yields considerable savings over other approaches that visit all (100%) of patches at high resolution, like DTFD, ACMIL, HAFED. In addition to feature extraction, the TSU is a secondary bottleneck in SASHA because at each timestep, the cosine similarity with all other non-masked patches must be computed, albeit at low resolution. Improving the speed of this step is scope for future work.

Expected Calibration Error (ECE): Expected Calibration Error (ECE) is a metric used to quantify how well a neural network's predicted probabilities align with the true probabilities (or observed frequencies) of its predictions [11]. It measures how reliable a model's confidence levels are in its predictions. A perfectly calibrated model would have an ECE of 0, indicating that the model's confidence scores accurately reflect the true likelihood of its predictions being correct.

To compute ECE, we first divide the predicted confidence scores (i.e., the maximum softmax probabilities) into $M = 10$ equally spaced bins. For each bin, we calculate two quantities: i) Accuracy: the proportion of correctly classified samples in the bin; and ii) Confidence: the average of the predicted probabilities in the bin. These metrics are then used to compute the calibration error as the absolute difference between its accuracy and confidence for each bin, summed across bins [11].

HAFED achieves the lowest Expected Calibration Error (ECE) of approximately 0.015, indicating highly calibrated predictions. In contrast, ACMIL and SASHA-0.1 exhibit the highest ECE values, around 0.06. SASHA-0.2 achieves an intermediate ECE of around 0.03, which is still lower than that of ACMIL and DTFD. This suggests that SASHA-0.2 not only matches HAFED in terms of classification accuracy but also performs better than other sota methods in terms of calibrated predictions.

A.5 Hardware

For feature extraction and model inference, we utilized an NVIDIA GeForce GTX 1080 Ti GPU with 12 GB of memory. Feature extraction was carried out with a batch size of 512. Training for HAFED and TSU was also performed on the same GPU. To accelerate the training of the RL agent described in Section 3.4, we employed an NVIDIA Tesla V-100 GPU with 32 GB of memory.

A.6 Limitations and Extensions

Training time cost - A key limitation of our method is that, despite being RL-based, it performs feature extraction on all high-resolution patches during training. To reduce training time, future improvements would involve developing an end-to-end pipeline that trains the RL agent concurrently with the HAFED and the TSU modules. In such a model, the policy network decides the specific low-resolution patch to sample in real-time, followed by HAFED feature distillation for the zoomed-in patch, and TSU state update for the entire slide, with all steps being trained end-to-end with reward at each time step. This would be followed by the additional steps of training the policy and value networks, and so on, until the end of each episode, for several episodes. This mimics how RL agents learn, conventionally, in novel environments with sporadic rewards. With this method, there would be no need to sample every patch at high resolution during training. Yet, key challenges – with stable, training of multiple modules concurrently – may need to be addressed.

Fixed sampling budget - In our current setup, we employ a fixed sampling budget per slide (e.g., 10% or 20%). However, as shown in Figure 4, we observe that for certain slides, the classifier can become confident in its prediction fairly early in the sampling process, rendering further sampling unnecessary. This suggests that a dynamic budget allocation strategy – guided by the classifier's confidence – could further reduce inference time without compromising performance.

Architecture overhead and parameter scaling - We compare the total number of model parameters—excluding the ViT-based feature extractor (see Appendix A.2.1)—across different methods. DTFD, ACMIL, and HAFED, all of which operate on high-resolution inputs, are relatively lightweight with approximately 82.56K, 84.37K, and 300K parameters, respectively. In contrast, the SASHA pipeline consists of three distinct modules: HAFED, the Target State Updater (TSU), and the Reinforcement Learning (RL) agent, with parameter counts of 300K, 1.1M, and 296K, respectively—summing to approximately 1.7M parameters. This represents a $3.5\times$ increase in parameters compared to HAFED and a $21\times$ increase relative to DTFD and ACMIL.

Filtering normal patches - WSIs typically contain far more normal patches, as compared to tumor patches. This imbalance becomes particularly problematic when the tumor occupies only a small fraction of the slide. In such cases, the RL agent is more likely to encounter and select normal patches – especially in the early steps – leading to inefficient exploration and increased risk of a miss. The sparse distribution of tumor regions makes it difficult for the agent to consistently focus on diagnostically relevant areas within a limited sampling budget. A more efficient strategy would involve the early filtering of normal, redundant, or uninformative patches, allowing the agent to concentrate its computational efforts on a smaller, more informative subset of patches. This could improve both performance and inference efficiency.

Incorporation into clinical workflows - Our method is particularly suited for deployment in clinical settings, from two key standpoints – Explainability and Calibration. First, clinicians are likely to trust decisions made by AI agents only if these decisions can be readily and rationally explained. Our RL agent consistently selects high-attention, tumor-rich patches, as compared to normal patches. Moreover, this selection occurs far more frequently than that of a random policy. These selection strategies make the decisions of the RL agent intuitive and easy to explain to clinicians (Section 4.5).

Second, clinicians are also likely to benefit from a reliable "confidence" score from the AI model, to know when they can, and cannot, trust its predictions. We showed that the expected calibration error (ECE) for SASHA=0.2 were consistently lower than those of competing sota attention-based models like ACMIL and DTFD (Section 4.6). With well-calibrated models (lower ECE), the model's confidence in its predictions (e.g., predictive uncertainty) is reliable. In this case, it becomes a useful indicator for when referral to a clinician or expert is warranted. In other words, by quantifying predictive uncertainty of the SASHA-0.2 classifier with established approaches (e.g., predictive entropy), our RL model could provide clinicians with valuable additional information about when its prediction can (or cannot) be trusted. Finally, inference time accuracy may depend on the extent of domain shift between the test and training clinical samples; developing approaches that combine domain adaptation with efficient sampling may be relevant to address this challenge.