REVISITING SUBSAMPLING AND MIXUP FOR WSI CLASSIFICATION: A SLOT-ATTENTION-BASED APPROACH

Anonymous authors
Paper under double-blind review

ABSTRACT

Whole Slide Image (WSI) classification requires repetitive zoom-in and out for pathologists, as only small portions of the slide may be relevant to detecting cancer. Due to the lack of patch-level labels, Multiple Instance Learning (MIL) is common practice for training a WSI classifier. One of the challenges in MIL for WSI is the weak supervision coming only from the slide-level labels, often resulting in severe overfitting. In response, researchers have considered adapting patch-level augmentation or applying mixup augmentation, but their applicability remains unverified. Our approach augments the training dataset by sampling a subset of patches in the WSI without significantly altering the underlying semantics of the original slides. Additionally, we introduce an efficient model (*Slot-MIL*) that organizes patches into a fixed number of slots, the abstract representation of patches, using an attention mechanism. We empirically demonstrate that the subsampling augmentation helps to make more informative slots by restricting the over-concentration of attention and to improve interpretability. Finally, we illustrate that combining our attention-based aggregation model with subsampling and mixup, which has shown limited compatibility in existing MIL methods, can enhance both generalization and calibration. Our proposed methods achieve the state-of-the-art performance across various benchmark datasets including class imbalance and distribution shifts.

1 INTRODUCTION

Multiple Instance Learning (MIL) (Dietterich et al., 1997; Maron & Lozano-Pérez, 1997) is a variant of weakly-supervised learning where a unit of learning problem is a bag of instances. A bag in MIL contains different numbers of instances, and a label is only assigned to the bag level and unknown for the instances in the bag. Many real-world problems can be formulated as MIL, including drug-activity prediction (Dietterich et al., 1997), document categorization (Andrews et al., 2002), point-cloud classification (Wu et al., 2015), and medical image processing (Li et al., 2021).

One of the main challenges in MIL is that we are often given a limited number of labeled bags for training in real-world applications. Consider a Whole Slide Image (WSI)[1] classification problem, which is our primary interest in this paper, and a popular example of MIL in the medical imaging domain. A single WSI (bag) usually contains more than tens of thousands of patches (instances), but the labels are only given to the WSI level (i.e., whether a WSI includes patches corresponding to disease), so the total number of labels given is way smaller than it is needed to train a deep learning model to process tons of patches. Moreover, as for the medical imaging domain, it is common that the WSIs from a certain type of disease are scarce in training data, leading to the class imbalance problem. Due to these problems, a naïve method for WSI classification is likely to suffer from severe overfitting or failure in distribution shifts.

For a usual classification problem, there are several data augmentation techniques to reduce overfitting and thus improve generalization performances. Following the standard protocol in image classification, combining simple augmentations such as rotating, flipping, and cutting on patches of

[1]Following words are interchangeable throughout the paper: WSI and bag; patch and instance.

WSI might be a valid choice (Hendrycks et al., 2019). Mixup augmentation (Zhang et al., 2017) might be another option, where a classifier takes a convex combination of two instances as an input and a convex combination of corresponding labels as a label for training.

Nevertheless, such augmentation techniques are not trivial to apply for WSI classification. Augmenting every patch in a WSI is highly burdensome due to the massive patch count in a WSI. Moreover, due to the high cost of processing numerous patches in WSIs, it is common to first encode the patches to the set of features using a pretrained encoder from common image datasets such as ImageNet (Deng et al., 2009), so the augmentations directly applied to patches are not feasible in such settings. To overcome these innate restrictions, previous works tried to suggest several augmentations specialized for WSI. For example, Zhang et al. (2022) inflated the number of bags by splitting a WSI into multiple chunks and assigning the same label as the original WSI. In order to unify the number of patches per WSI while not losing information, Chen & Lu (2023) introduced additional patch classifier which is used as a guidance to choose important patches from WSIs, so that they can be selectively used for the mixup augmentation. Although both aforementioned methods tackled the innate problem of WSI, they require either knowledge distillation or two-stage training to stabilize training and achieve better performance.

We propose a slot-attention based MIL method which aggregates patches into a fixed number of slots based on the attention mechanism (Vaswani et al., 2017; Lee et al., 2019; Locatello et al., 2020). As the varying number of patches are summarized into an identical number of slots for every WSI, we can directly adopt mixup on slots. Also, we revisit the subsampling augmentation in WSI and empirically prove that subsampling helps to mitigate overfitting by involving more crucial patches for decision-making. By integrating subsampling and mixup into our proposal, we can achieve SOTA performance on various datasets. We solved the innate problem of MIL for WSIs through a simple method that does not require any additional process and, thus, is easily applicable.

Our contributions can be outlined as follows:

- We propose a slot-based MIL method, a computationally efficient pooling-based model for WSI classification, exhibiting superior performance with fewer parameters.
- We unveil the power of subsampling, which surpasses other previous augmentations suggested for MIL. We also provide a detailed analysis of the effect of subsampling in regularizing attention scores.
- As our slot-based method aggregates patches into a fixed number of slots that well represent the WSI with the help of subsampling, we can directly adopt mixup to slots. It does not require any extra layer or knowledge distillation, which was essential in previous methods. By doing so, our method obtains SOTA performance with better calibrated predictions.

2 Related Works

2.1 Deep MIL Models

To classify a WSI, aggregating entire patches into meaningful representations is crucial. The simplest baseline is to use mean (Pinheiro & Collobert, 2015) or max pooling operations (Feng & Zhou, 2017) under the assumption that the average or maximum representation across instances can effectively represent the whole bag. ABMIL (Ilse et al., 2018) first adopted attention mechanisms (Bahdanau et al., 2014; Luong et al., 2015) in MIL classification problems and proposed the gated attention to yield instance-wise attention scores with additional non-linearities. However, ABMIL does not consider how patches are related and interact with each other. DSMIL (Li et al., 2021) utilized two streams of networks for effective classification. The first network finds a critical instance using max pooling, and the second network constructs the bag's representation based on attention scores between the critical instance and the others. TRANSMIL (Shao et al., 2021) proposed the TPT module which consists of two transformer layers (Vaswani et al., 2017) and a positional encoding layer. It adopted self-attention for the first time while trying to reduce the complexity by adopting Nystrom-attention (Xiong et al., 2021). ILRA (Xiang & Zhang, 2022) used learnable parameters to aggregate patches into smaller subsets and re-expand to patches based on the attention. It repeats this process to acquire meaningful features.

Table 1: Comparison of augmentation methods for MIL classification. Intra means augmentation within a single WSI, and Inter means augmentation between WSIs. Models are labeled considering where the main augmentation occurs.

Model	Intra	Inter	Requirements & Limitations
REMIX (Yang et al., 2022)	✗	✓	Within same label only
MIXUP-MIL (Gadermayr et al., 2022)	✓	✗	Little gain in performance
DTFD-MIL (Zhang et al., 2022)	✓	✗	Knowledge distillation
RANK-MIX (Chen & Lu, 2023)	✗	✓	Pre-training for teacher model
Subsampling + Slot-Mixup (Ours)	✗	✓	None of above

2.2 MIXUP

Mixup (Zhang et al., 2017) is a simple augmentation strategy where one interpolates pixels of two images and corresponding labels with the same ratio. As mixup provides a smoother decision boundary from class to class, it improves generalization and reduces the memorization of corrupted labels. While being simple to implement, mixup has been reported to improve classifiers across various applications, especially for imbalanced datasets or datasets including minority classes (Galdran et al., 2021; Hwang et al., 2022). Manifold mixup (Verma et al., 2019) adopts the idea of mixup into the feature level and further improves classification performance. As a typical MIL model utilizes pre-extracted features, mixup in MIL context means manifold mixup to be precise.

2.3 AUGMENTATIONS IN WSI CLASSIFICATION

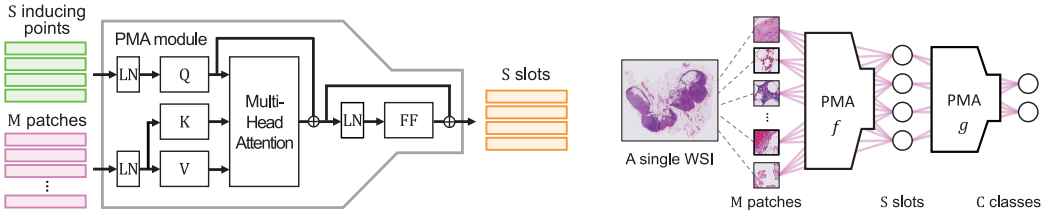
Recent papers have focused on developing augmentation methods to enhance the limited number of WSIs and proposed techniques to prevent over-fitting. REMIX (Yang et al., 2022) uses k-means clustering in order to select important patches from a WSI. Then, it creates new bags by mixing important patches from different WSIs considering Euclidean distance. However, this method can only be applied between WSIs with the same label. DTFD-MIL (Zhang et al., 2022) creates pseudo-bags by splitting WSI into a subset of patches and assigns the same labels as the original one. When training its first-tier model with pseudo-bags, it selects the crucial patches based on gradient (Selvaraju et al., 2016). Then, with those selected patches, it trains the second-tier model. Adopting manifold mixup directly is not applicable as the size of WSIs varies per slide. RANK-MIX (Chen & Lu, 2023) solves this problem by adding linear layers, so-called *teacher*, on existing models to distinguish important patches. With the help of this layer, we can now unify the number of patches per WSIs. It attempts augmentation between WSIs for the first time and proves the effectiveness through performance gain. Nevertheless, it needs pre-training for the teacher model, and its performance without self-training shows little margin.

Orthogonal to mixup, there are some studies (Combalia & Vilaplana, 2018; Breen et al., 2023) to sample essential patches for classification, as utilizing whole patches is computationally heavy and inefficient. Although the authors highlighted that random subsampling improves generalization, they did not clearly explain the correlation between subsampling and attention-based models. We revisit this simple but important idea and adapt it to our model.

3 MAIN CONTRIBUTION

3.1 SLOT-MIL: A SLOT-ATTENTION-BASED ARCHITECTURE FOR MIL

As it is widely assumed that WSI inherently possesses a low-rank structure (Xiang & Zhang, 2022), it is desirable to aggregate a WSI into a smaller subset of patches while preserving the underlying semantics. From this perspective, we adopt the idea of inducing points (Lee et al., 2019) and slots (Locatello et al., 2020), which are the set of learnable vectors used to encode a set of inputs into a fixed-sized feature array. For brevity, we call this module Pooling by Multi-head Attention (PMA), following Lee et al. (2019), with some modifications. Considering the MIL problems where patches lose absolute position information in the pre-processing stage, we omit positional embedding.



(a) PMA module architecture

(b) The overall model structure

Figure 1: (a) The PMA module summarizes M patches into S slots, which is the same number as S learnable vectors called *inducing points*. (b) A single WSI is divided into M patches, which then pass through two PMA modules, resulting in logit for each class.

A bag of M instances is denoted by $\boldsymbol{B} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_M]^\top \in \mathbb{R}^{M \times d_{\boldsymbol{h}}}$, where $\boldsymbol{h}_i$ is the feature vector of the i^{th} instance in the bag computed from a pre-trained encoder and $d_{\boldsymbol{h}}$ is its dimension. We also denote *inducing points* as $\boldsymbol{I} = [\boldsymbol{i}_1, \ldots, \boldsymbol{i}_S]^\top \in \mathbb{R}^{S \times d_{\boldsymbol{i}}}$, where S is the number of slots with dimension of $d_{\boldsymbol{i}}$. For simplicity, we describe our module with single-head attention, but in practice, we adopt the multi-head attention mechanism as recommended in Vaswani et al. (2017). The layer normalization (Ba et al., 2016; Xiong et al., 2020) is applied to the inducing points and the feature vectors before the attention operation,

$$\bar{\boldsymbol{I}} = \mathrm{LN}(\boldsymbol{I}),\ \bar{\boldsymbol{B}} = \mathrm{LN}(\boldsymbol{B}), \tag{1}$$

where LN is the layer norm operator. Then, we compute the query, key, and value matrices as

$$\boldsymbol{Q} = \bar{\boldsymbol{I}}\boldsymbol{W}^Q,\ \boldsymbol{K} = \bar{\boldsymbol{B}}\boldsymbol{W}^K,\ \boldsymbol{V} = \bar{\boldsymbol{B}}\boldsymbol{W}^V, \tag{2}$$

where $\boldsymbol{W}^Q \in \mathbb{R}^{d_{\boldsymbol{i}} \times d}$, $\boldsymbol{W}^K \in \mathbb{R}^{d_{\boldsymbol{h}} \times d}$, and $\boldsymbol{W}^V \in \mathbb{R}^{d_{\boldsymbol{h}} \times d}$ are learnable parameters, and d is the hyperparameter. Then the updated slot matrix $\boldsymbol{S}$ is computed as

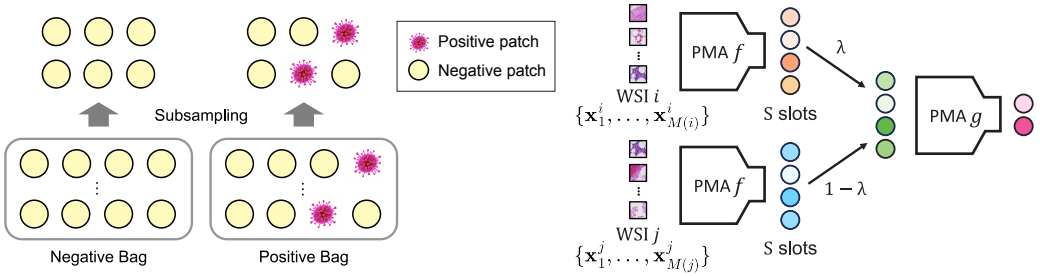
$$\boldsymbol{S}' = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top/\sqrt{d})\boldsymbol{V} + \boldsymbol{Q}, \quad \boldsymbol{S} = \mathrm{MLP}(\mathrm{LN}(\boldsymbol{S}')) + \boldsymbol{S}'. \tag{3}$$

As demonstrated in Lee et al. (2019) and Locatello et al. (2020), the resulting slot matrix $\boldsymbol{S}$ serves as a decent fixed-length summary of the patch features in WSIs. Also, a PMA module is permutation invariant as the slots are not affected by the order of the patches. After summarizing patches into the slots with the first PMA module $f(\cdot)$, we use another PMA $g(\cdot)$ to aggregate slots into the classification logits, as illustrated in Figure 1b. The only difference from the first PMA is the dimension of the final output, which is set to one, letting the output from $g(\cdot)$ be directly used as classification logits. We name the whole pipeline, using two PMAs to get logits for a WSI classification, as *Slot-MIL*. This simple yet powerful model utilizes attention throughout the whole decision process while maintaining the number of parameters relatively smaller than the existing models.

A design decision needs to be made when performing the softmax computation in (3). In the original softmax operation used in attention mechanisms (Vaswani et al., 2017; Lee et al., 2019), normalization is applied to the key dimension, meaning that the patch features compete with each other for a slot. In contrast, in slot attention (Locatello et al., 2020), the softmax normalization is applied to the query dimension, resulting in slots competing with each other for a patch feature. Interestingly, empirical observations indicate that neither of these two options significantly outperforms the other. As a result, a hybrid approach has been explored, where both normalization schemes are combined. In practice, this means using key normalization for half of the attention heads and query normalization for the other half. This hybrid approach has been found to enhance the stability of the training process and even improve the performance in some datasets.

3.2 Subsampling

In this paper, we mainly consider a binary classification problem, where the goal is to build a classifier that takes a bag of feature matrix $\boldsymbol{B}$ as an input and predicts a corresponding binary class $y \in \{0, 1\}$. A common assumption in MIL is that each instance in a bag has its corresponding latent target variables $y_1, \ldots, y_M$, which are usually unknown. The label of a bag is positive if at least one of its instances is positive, and negative otherwise, i.e., $y = 1 - \prod_{m=1}^{M}(1 - y_m)$.



(a) Subsampling in MIL (b) Slot-Mixup

Figure 2: (a) Subsampling generally maintains the original label when patches are abundant, which is natural in WSIs. (b) Illustration of *Slot-Mixup*. As two WSIs i and j with different numbers of patches $M(i)$ and $M(j)$, respectively, are summarized into the same number of S slots, applying MIXUP is straightforward. Two PMA modules f are identical.

One straightforward augmentation method in this scenario is subsampling patches from a bag. Figure 2a provides an overview of its procedure, where we first randomly select a subset of patches from a WSI without replacement and then utilize only these chosen subsets for training. It is worth mentioning that several works have touched upon this concept to some extent: Zhang et al. (2022) proposed a method where a single WSI is split into k subsets in order to make pseudo-bags working as augmented data for training a WSI classification model; Chen & Lu (2023) also proposed to draw a fixed number of patches from WSIs, mainly for the purpose of matching the number of patches between WSIs for the mixup augmentation.

We reveal the efficacy of subsampling itself in the context of WSI classification or presumably for generic MIL. A question naturally arises on whether subsampling may alter the semantic of a MIL; we argue that this is rare, due to the following reasoning. In case of a positive WSI slide, for a subsampled bag to be negative, one should subsample all the patches with the negative latent patch labels, which is highly unlikely considering the typical number of patches included in a WSI. All the patches are of negative latent labels for a negative WSI, so subsampling would not change the slide label anyway. Unlike the previous augmentation strategies, subsampling does not incur additional training costs or extra requirements and modifications to the classification model. Actually, it reduces the time complexity of each step of training from $\mathcal{O}(SM)$ to $\mathcal{O}(pSM)$, where $p \in [0, 1]$ is the subsampling rate.

While we interpret subsampling as a data augmentation technique and thus do not apply it during inference, an alternative perspective is to view subsampling as an instance of dropout (Srivastava et al., 2014). From this standpoint, one might consider performing Monte Carlo (MC) inference, where predictions from multiple subsampled inputs are averaged (Gal & Ghahramani, 2016). In Table 11 of the appendix, we present the results of our model inferred in this way. However, to summarize, both the inference without subsampling and the Monte-Carlo type inference do not exhibit notable differences in their performance.

3.3 SLOT-MIXUP: MIXUP AUGMENTATION USING SLOTS

As mixup was originally designed for a typical single instance learning problem, applying them for WSI classification as well as generic MIL problems is not straightforward. An apparent problem arises from the fact that, unlike single-instance learning, the unit of prediction here is a bag of patches. There exists an ambiguity of what patches to mix as two slides typically have different numbers of patches. A naïve method, as indirectly adopted in Chen & Lu (2023), is to choose a fixed number of patches from two slides. However, as there is no available patch-wise information or annotation, it is difficult to see whether the mixing of arbitrarily chosen patches results in a semantically meaningful augmentation that would enhance the generalization ability of a model trained with them.

On the other hand, in the case of our Slot-MIL model, given two WSIs, the model can summarize them into the identical number of slots encoding the essential information required for classification.

Therefore, having the slot summaries of two WSIs offers a better option to apply mixup. Specifically, let $\boldsymbol{S}_i$ and $\boldsymbol{S}_j$ be the slots computed from Slot-MIL model (3) for i^{th} and j^{th} slides. Then we draw the mixing ratio $\lambda \sim \text{Beta}(\alpha, \alpha)$ and compute the mixed slot set and corresponding label as,

$$\widetilde{\boldsymbol{S}}_{ij} \quad = \lambda \boldsymbol{S}_i + (1-\lambda)\boldsymbol{S}_j, \quad \widetilde{y}_{ij} = \lambda y_i + (1-\lambda) y_j. \tag{4}$$

The application of mixup in this manner offers several advantages in contrast to its prior adaptation for WSI classification. By operating within the context of a fixed number of slots, the inherent ambiguity associated with the selection of patches for mixing is effectively eliminated. Concurrently, since these slots serve as concise summarizations of the individual instances, the act of mixing slots is anticipated to yield more meaningful augmented data. Additionally, akin to the principles of manifold mixup, the mixing process transpires at the slot level, thus obviating the need for the recomputation of patch features each time augmentation is applied during the training of data. This novel approach is termed *Slot-Mixup*, and Figure 2b illustrates the procedure. Our experimental results empirically demonstrate its efficacy in addressing the WSI classification problem.

Slots are calculated through learnable inducing points and thus change during training. In order to mix slots that well represent WSIs, we empirically find that starting Slot-Mixup after some epochs is effective. In our experiments, We call this heuristic LATE-MIX and studied the effect of the number of epochs to start mixup as a hyperparameter L.

3.4 SLOT-MIXUP WITH SUBSAMPLING

We combine two augmentation strategies, subsampling and Slot-Mixup, into our Slot-MIL model. As we empirically validate with various benchmarks, two augmentation techniques work well in synergy, resulting in more accurate and better-calibrated predictions. Throughout the paper, we call the combined augmentation strategy of subsampling and Slot-Mixup as SubMix.

4 EXPERIMENTS AND RESULTS

Datasets. We present experimental findings on three datasets: **(1) TCGA-NSCLC**, a subtype classification problem where all slides are of positive cancer. It is a balanced dataset, comprising 528 LUAD and 512 LUSC slides, and ensures slides from the same patient do not overlap between the train and test sets. **(2) CAMELYON-16**, consisting of 159 positive slides and 238 negative slides, where positive patches occupying less than 10% of the tissue area in positive slides. Both train and test sets are class-imbalanced. **(3) CAMELYON-17**, comprising 145 positive slides and 353 negative slides. Distribution shifts exist between train and test splits, as the train and test data are sourced from different medical centers (Litjens et al., 2018). Unless otherwise specified, all the patches in the slides are encoded into features by the ResNet (He et al., 2016) pre-trained with ImageNet. More details for the dataset can be found in Appendix A.1.

Baselines. We compare mean and max-pool, ABMIL (Ilse et al., 2018), DSMIL (Li et al., 2021), TRANSMIL (Shao et al., 2021), ILRA (Xiang & Zhang, 2022) for models without augmentation. For comparing augmentation methods, DTFD-MIL (Zhang et al., 2022), and RANK-MIX (Chen & Lu, 2023) are used. We describe the detailed hyperparameter settings in Appendix A.2.

Implementation details and evaluation metrics. We follow the hyperparameter setting by Li et al. (2021) for simplicity. We use Adam optimizer with a learning rate of 1e-4 and trained for 200 epochs. For additional information, please refer to Appendix A.3. Since TCGA-NSCLC does not provide the official train-test split, for fair comparison, we divide the dataset into train:valid:test=60:15:25 ratio, and conduct 4-fold cross-validation. As CAMELYON-16 and CAMELYON-17 have the official train-test splits, we only divide the train set into train:valid=80:20 ratio for 5-fold cross-validation. All train and valid sets are divided by using stratified k-fold. Reporting cross-validation results based on the best validation area under the ROC Curve (AUC) has been a convention in the literature, but considering small-sized valid sets, the results can be quite inconsistent. So, we report the average test performance based on the top ten valid AUCs for each fold. We also measure negative log-likelihood (NLL) for calibration measure.

We clarify hyperparameters for SUBMIX as follows: (number of Slots, Subsampling rate, Late Mix) = (S, p, L) : (16, 0.4, 0.2), (4, 0.2, 0.2), (16, 0.1, 0.2) for CAMELYON-16, CAMELYON-17, and TCGA-NSCLC, respectively. The best performance altering α within $0.2, 0.5, 1.0, 2.0$ is reported.

Table 2: Comparing augmentations in WSI classification

Method/Dataset	TCGA-NSCLC		
	ACC (↑)	AUC (↑)	NLL (↓)
ABMIL	0.832±0.039	0.884±0.044	0.708±0.274
ABMIL + DTFD-MIL	0.834±0.034	0.893±0.030	0.980±0.178
ABMIL + Sub ($p = 0.1$)	**0.852**±0.021	**0.920**±0.021	**0.513**±0.127
ABMIL + Sub ($p = 0.2$)	0.844±0.024	0.914±0.023	0.572±0.123
ABMIL + Sub ($p = 0.4$)	0.835±0.027	0.899±0.027	0.640±0.103
DSMIL	0.831±0.022	0.897±0.021	0.737±0.098
DSMIL + RankMix	0.820±0.026	0.894±0.032	0.623±0.119
DSMIL + Sub ($p = 0.1$)	**0.853**±0.026	**0.922**±0.022	**0.598**±0.197
DSMIL + Sub ($p = 0.2$)	0.850±0.032	0.919±0.024	0.623±0.173
DSMIL + Sub ($p = 0.4$)	0.845±0.025	0.913±0.023	0.617±0.093

Table 3: Comparison of the Subsampling, Slot-Mixup, and SubMix.

Method/Dataset	TCGA-NSCLC		
	ACC (↑)	AUC (↑)	NLL (↓)
Slot-MIL	0.852±0.025	0.914±0.016	1.001±0.202
+ Sub ($p = 0.1$)	0.870±0.017	0.929±0.019	0.789±0.326
+ Sub ($p = 0.2$)	**0.873**±0.020	**0.931**±0.019	0.751±0.333
+ Sub ($p = 0.4$)	**0.873**±0.020	**0.931**±0.023	0.866±0.313
+ Slot-Mixup ($\alpha = 0.2$)	0.857±0.022	0.915±0.019	0.684±0.099
+ Slot-Mixup ($\alpha = 0.5$)	0.855±0.019	0.914±0.021	0.686±0.161
+ Slot-Mixup ($\alpha = 1.0$)	0.856±0.020	0.915±0.021	0.657±0.137
+ SubMix ($p = 0.2, \alpha = 0.2$)	0.871±0.019	0.930±0.020	0.547±0.141
+ SubMix ($p = 0.2, \alpha = 0.5$)	0.869±0.022	**0.931**±0.020	**0.496**±0.135

$L = 0.1$ means that we start mixup after 10% of total epochs. The comparison for augmentation methods shares an identical baseline setting for fairness.

4.1 Subsampling and Mixup

4.1.1 Subsampling versus Other Augmentation Methods

We first compare subsampling with existing augmentation methods in Table 2. We applied DTFD-MIL and RANK-MIX to ABMIL and DSMIL, respectively, following original papers. As they either split a WSI or select subset patches by the extra network, it is natural to compare with our subsampling method. With only subsampling, we can achieve performance on par with the above two methods. The result shows the importance of subsampling, which is quite unexplored yet. Optimal results are reported. NLL in MIL area is higher than the natural image area, as they are weakly supervised. The model tends to have high confidence even when it fails to predict correctly. Full results varying subsampling rate p are in Appendix B.1.

4.1.2 Why Subsampling Works for MIL?

Given that a slide-level label can be determined by just one positive patch, it is natural for a model to predominantly concentrate on a small subset of the slide. However, as the training progresses, attention-based models tend to increasingly focus on smaller subsets, leading to a degradation in generalization. As the combination of positive patches evolves through iterations due to subsampling, the model learns to allocate attention to important patches with more equitable weights. Subsequent to the findings presented in Figure 3, the empirical analysis establishes a correlation between overfitting and attention scores across patches. To achieve this, we normalize the sum of attention across all patches to 1 and select the top 100 patches with the highest attention scores. For these patches, we calculate the entropy as $E = -\sum_a p(a) \log_2 p(a)$, where $p(a)$ represents the normalized attention scores. A higher entropy signifies a more even distribution of attention scores. Consequently, due to the excessive concentration of attention scores on specific patches, a model that lacks subsampling encounters overfitting during training. Visualizations at the slide-level further strengthen the argument that attention is appropriately distributed to important areas in alignment with pathologists' labels, as discussed in Appendix B.2. Moreover, as empirically verified in Appendix B.8, this phenomenon persists irrespective of the model type or dataset balance.

4.1.3 Slot-Mixup with Subsampling

In Table 3, we present a comparative analysis of the effects of subsampling, Slot-Mixup, and SubMix when applied to our Slot-MIL model. Subsampling demonstrates an ability to enhance generalization on unseen test data; however, it only marginally improves over-confident predictions compared to the baseline. Conversely, training with mixup exhibits a similar AUC to the baseline but achieves a lower NLL due to the generation of intermediate labeled data, which contributes to a smoother decision boundary. Combining both techniques, which we refer to as SubMix, allows us to improve generalization performance while maintaining relatively well-calibrated predictions. As a result, for the remaining experiments, we adopt SubMix as our primary augmentation method.

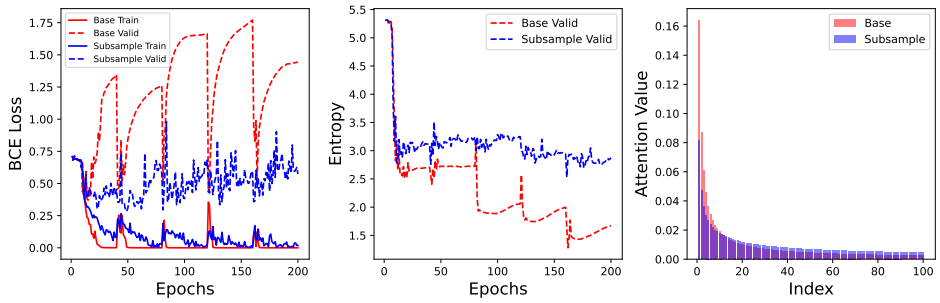


Figure 3: Results of Slot-MIL + SUB on CAMELYON-16. **(left)** Subsampling improves generalization performance. **(middle)** Without subsampling, entropy gets smaller as training proceeds. This means over-concentration of attention to specific patches. **(right)** Mean of Top 100 attention for whole test sets. Index 1 means the highest attention score among patches.

Table 4: Results on CAMELYON-17 (distribution shifts); Experiments are conducted based on features extracted by pre-trained ResNet-18. DTFD-MIL is applied to ABMIL. Training time and inference time are reported in seconds per epoch. We measure FLOPs with a bag size of 10,000.

Method/Dataset	CAMELYON-17						
	ACC (↑)	AUC (↑)	NLL (↓)	Train	Inference	FLOPs	Model Size
Meanpool	0.621±0.090	0.661±0.034	**0.622**±0.110	9.20	4.76	1,024	1,026
Maxpool	0.669±0.066	0.602±0.033	0.693±0.108	9.40	4.98	1,024	1,026
ABMIL	0.813±0.023	0.784±0.013	1.056±0.488	9.59	4.79	1.32G	132,483
DSMIL	0.735±0.044	0.745±0.027	1.894±0.847	10.15	4.84	3.30G	331,396
TRANS-MIL	0.727±0.063	0.739±0.059	2.521±0.472	36.92	9.02	50.31G	2,147,346
ILRA	0.716±0.027	0.713±0.036	1.196±0.198	27.53	7.98	9.92G	1,555,330
Slot-MIL	**0.832**±0.006	0.813±0.016	1.181±0.425	12.53	5.63	5.45G	1,590,785
DTFD-MIL	0.793±0.026	0.819±0.014	1.206±0.387	15.09	6.90	-	-
Slot-MIL + RankMix	0.823±0.032	0.817±0.025	0.640±0.124	30.91	5.64	-	-
Slot-MIL + SubMix	0.805±0.016	**0.835**±0.015	0.633±0.035	22.74	5.66	-	-

4.2 WSI CLASSIFICATION

4.2.1 WSI CLASSIFICATION UNDER DISTRIBUTION SHIFT

CAMELYON-17 experiences distribution shifts due to the use of different scanners in separate medical centers for training and testing, compounded by variations in data collection and processing methods. As shown by Wiles et al. (2021), traditional pre-training methods do not effectively address this issue. However, recent work by Lee et al. (2022) suggests that selective fine-tuning of specific layers offers a promising solution.

Our Slot-MIL model demonstrates state-of-the-art (SOTA) performance, particularly in a distribution-shifted domain, as illustrated in Table 4. Furthermore, the SubMix augmentation technique significantly enhances performance, affirming the efficacy of our approach even in the absence of pre-training or fine-tuning. It is important to note that the use of the AUC metric is preferred over AUC due to its robustness in measuring performance across various threshold settings. SubMix, in particular, exhibits substantial improvements in this regard.

4.2.2 STANDARD WSI CLASSIFICATION

Table 5 shows the results for standard WSI classification, where distribution shift does not exist as the WSIs from different scanners exist in both the train and the test set. Comparison without augmentation is above the double line, and with augmentation is below the double line. *Slot-MIL* shows SOTA performance with large margin in both datasets when augmentation is not adopted. Also, SUBMIX works better than RANKMIX proving the validity of our method. Considering that RANKMIX needs additional training for teacher which requires around 2x training complexity than

Table 5: Results on CAMELYON-16 and TCGA-NSCLC. RankMix and SubMix is applied to Slot-MIL, but omitted for brevity. We utilize the open-source features provided by Zhang et al. (2022) and Li et al. (2021) for ResNet-50 and SimCLR experiments, respectively.

Method/Dataset	CAMELYON-16				TCGA-NSCLC			
	ResNet-50		SimCLR		ResNet-18		SimCLR	
	AUC (↑)	NLL (↓)	AUC (↑)	NLL (↓)	AUC (↑)	NLL (↓)	AUC (↑)	NLL (↓)
Meanpool	0.522±0.036	0.971±0.064	0.604±0.003	0.674±0.023	0.798±0.025	0.571±0.047	0.972±0.010	**0.232**±0.055
Maxpool	0.783±0.022	0.942±0.258	0.967±0.002	0.353±0.147	0.802±0.011	0.572±0.023	0.961±0.013	0.608±0.053
ABMIL	0.808±0.034	1.185±0.395	0.972±0.002	0.234±0.033	0.884±0.044	0.708±0.274	**0.981**±0.010	0.263±0.101
DSMIL	0.833±0.063	1.620±1.145	0.968±0.008	0.456±0.182	0.897±0.021	0.737±0.098	**0.981**±0.010	0.324±0.133
TRANSMIL	0.834±0.036	1.654±0.326	0.939±0.010	0.988±0.188	0.893±0.021	1.791±0.559	0.974±0.009	0.381±0.127
ILRA	0.842±0.051	1.157±0.396	0.973±0.007	0.333±0.043	0.901±0.028	0.824±0.117	**0.981**±0.011	0.277±0.103
Slot-MIL	0.893±0.023	1.242±0.979	0.972±0.007	0.294±0.065	0.914±0.016	1.001±0.202	**0.981**±0.011	0.276±0.131
DTFD-MIL	0.844±0.052	1.014±0.255	**0.975**±0.004	0.292±0.035	0.893±0.030	0.980±0.178	**0.981**±0.011	0.309±0.135
RankMix	0.914±0.025	0.525±0.087	0.965±0.012	0.342±0.064	**0.932**±0.021	0.532±0.173	0.980±0.011	0.283±0.014
SubMix	**0.921**±0.020	**0.448**±0.103	**0.975**±0.008	**0.229**±0.071	0.931±0.020	**0.496**±0.135	**0.981**±0.012	0.248±0.105

SUBMIX, our method is efficient and powerful. The results without self-training on RANKMIX are in Appendix B.3 which can be more fair comparison in terms of complexity.

4.3 Further Analysis and Ablation Studies

Additionally, we present an in-depth analysis of the hyperparameters governing our Slot-MIL and SubMix methodologies.

Number of slots, S. We empirically demonstrate that a small number of slots is sufficient to capture the underlying semantics of WSIs. There is minimal difference in performance when the number of slots exceeds a certain threshold. Consequently, we determine the optimal number of slots as (16, 4, 16) for CAMELYON-16, CAMELYON-17, and TCGA-NSCLC, taking into consideration computational efficiency.

Mixup hyperparameter, α. The mixup ratio, λ, varies during each iteration and is sampled from the beta distribution $\mathcal{B}(\alpha, \alpha)$. Across different datasets, we observe that values of α greater than 1 negatively impact performance, as they lead to a more pronounced divergence between the mixed feature distribution and the original distribution.

Subsampling rate, p. The subsampling rate is directly related to the proportion of positive patches within positive slides. In the case of TCGA-NSCLC, where positive patches make up approximately 80% of the dataset, even a small value of p suffices to capture the underlying semantics of the original labels. Conversely, in CAMELYON-16, where positive patches constitute only 10% of positive slides, larger values of p prove to be effective. Further experiments with varying values of p are detailed in Appendix B.4.

Late-mix parameter, L. Allowing the slots to learn the underlying representations of WSIs from the un-mixed original training set is crucial. Hence, the application of mixup after a certain number of epochs becomes particularly essential, especially in the case of CAMELYON-16 and CAMELYON-17. Additional details are provided in Appendix B.5.

5 Conclusion

WSI classification suffers extreme overfitting due to a lack of data and weak signal coming only from the slide-level label. In order to solve this problem, previous studies tried to suggest augmentations but their approach is either ineffective or complex. Based on our efficient model Slot-MIL, which aggregates patches into informative slots, we can easily apply mixup Slot-Mixup. Also, we uncover the effect of subsampling on the attention-based model in MIL, which is quite unexplored yet. Utilizing subsampling, and Slot-Mixup concurrently, we achieve SOTA performance in various datasets with superior calibrated-prediction than other models. Although it is still not calibrated well as a natural image, we hope that our model can assist in diagnosing cancer in real-world applications. As we can unify the number of patches in WSIs with subsampling, future research includes mini-batch training which is not investigated well in MIL for WSIs.

ETHICS STATEMENT

This paper does not contain ethical concerns.

REPRODUCIBILITY STATEMENT

All experiments are implemented with PyTorch v1.13.1+cu117 (Paszke et al., 2019) and run over 5 times for reproducibility.

REFERENCES

S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, 2002. 1

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 4

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 2

Jack Breen, Katie Allen, Kieran Zucker, Geoff Hall, Nicolas M Orsi, and Nishant Ravikumar. Efficient subtyping of ovarian cancer histopathology whole slide images using active sampling in multiple instance learning. In *Proceedings of SPIE 12471*, volume 12471. SPIE, 2023. 3

Yuan-Chih Chen and Chun-Shien Lu. Rankmix: Data augmentation for weakly supervised learning of classifying whole slide images with diverse sizes and imbalanced categories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 23936–23945, 2023. 2, 3, 5, 6

Marc Combalia and Verónica Vilaplana. Monte-carlo sampling applied to multiple instance learning for whole slide image classification. *1st Conference on Medical Imaging with Deep Learning (MIDL 2018)*, 2018. 3

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*, pp. 248–255. Ieee, 2009. 2, 13

T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31–71, 1997. 1

Ji Feng and Zhi-Hua Zhou. Deep miml network. In *Thirty-First AAAI conference on artificial intelligence*, 2017. 2

Michael Gadermayr, Lukas Koller, Maximilian Tschuchnig, Lea Maria Stangassinger, Christina Kreutzer, Sebastien Couillard-Despres, Gertie Janneke Oostingh, and Anton Hittmair. Mixup-mil: Novel data augmentation for multiple instance learning and a study on thyroid cancer diagnosis. *arXiv preprint arXiv:2211.05862*, 2022. 3

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1050–1059, 2016. 5

A. Galdran, G. Carneiro, and M. A. G. Ballester. Balanced-MixUp for highly imbalanced medical image classification. In *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention 2021 (MICCAI 2021)*, 2021. 3

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*, pp. 770–778, 2016. 6, 13

Dan Hendrycks, Norman Mu, Ekin D Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*, 2019. 2

I. Hwang, S. Lee, Y. Kwak, S. Oh, D. Teney, J. Kim, and B. Zhang. SelecMix: debiased learning by contradicting-pair sampling. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, 2022. 3

Maximilian Ilse, Jakub Tomczak, and Max Welling. Attention-based deep multiple instance learning. In *International conference on machine learning*, pp. 2127–2136. PMLR, 2018. 2, 6

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pp. 3744–3753. PMLR, 2019. 2, 3, 4

Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022. 8

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*, pp. 14318–14328, 2021. 1, 2, 6, 9, 13, 15

Geert Litjens, Peter Bandi, Babak Ehteshami Bejnordi, Oscar Geessink, Maschenka Balkenhol, Peter Bult, Altuna Halilovic, Meyke Hermsen, Rob van de Loo, Rob Vogels, et al. 1399 h&e-stained sentinel lymph node sections of breast cancer patients: the camelyon dataset. *GigaScience*, 7(6):giy065, 2018. 6

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33:11525–11538, 2020. 2, 3, 4

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 13

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015. 2

Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. *Advances in neural information processing systems*, 10, 1997. 1

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf. 10

Pedro O Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1713–1721, 2015. 2

Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016. 3

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. 2, 6

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of Machine Learning Research*, 15(1):1929–1958, 2014. 5

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 4

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, pp. 6438–6447. PMLR, 2019. 3

Olivia Wiles, Sven Gowal, Florian Stimberg, Sylvestre Alvise-Rebuffi, Ira Ktena, Krishnamurthy Dvijotham, and Taylan Cemgil. A fine-grained analysis on distribution shift. *arXiv preprint arXiv:2110.11328*, 2021. 8

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: a deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

Jinxi Xiang and Jun Zhang. Exploring low-rank property in multiple instance learning for whole slide image classification. In *The Eleventh International Conference on Learning Representations*, 2022. 2, 3, 6, 15

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pp. 10524–10533. PMLR, 2020. 4

Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nyströmformer: A Nyström-based algorithm for approximating self-attention. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021. 2

Jiawei Yang, Hanbo Chen, Yu Zhao, Fan Yang, Yao Zhang, Lei He, and Jianhua Yao. Remix: A general and efficient framework for multiple instance learning based whole slide image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 35–45. Springer, 2022. 3

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*, pp. 18802–18812, 2022. 2, 3, 5, 6, 9, 13

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2, 3

A EXPERIMENTS DETAILS

A.1 DATASET

TCGA-NSCLC consists of 528 LUAD, 514 LUSC excluding low quality WSIs following DSMIL (Li et al., 2021). We divide WSI by 224 X 224 size patch on 20x magnification. Then, Imagenet (Deng et al., 2009) pretrained Resnet-18 (He et al., 2016) is used for feature extraction. Dimension of each patch is 512. For **CAMELYON-16**, we used feature from Zhang et al. (2022). It divides a WSI into 256 x 256 size patch on 20x magnification, while extracting features using Resnet-50. Dimension for each patch is 1024. Before passing features through model, we reduct its dimension to 512 using a linear layer. This is a unified setting for all models. **CAMELYON-17** is extracted by Resnet-18 on 224 X 224 size patch using 20x magnification. For train set, we use WSIs scanned from CWZ, RST, RUMC center. Remains are used for test set.
We also evaluated with **TCGA-NSCLC** feature extracted by SIMCLR-based model offered from (Li et al., 2021). It divides WSI by 224 x 224 size patch on 20x magnification.

A.2 BASELINES

We followed the structure and hyper-parameter of original papers unless mentioned. For ABMIL, we use gated-attention as it performs better than naïve attention. For ILRA, we set rank=64, iteration=4, and changed hidden dimension to 128 as it performs better than 256. For DTFD-MIL, we use 5 pseudo-bags for TCGA-NSCLC, and 8 pseudo-bags for CAMELYON-16, CAMELYON-17. We report best performance within AFS and MaxMinS. For RANK-MIX, we report best performance within $\alpha = 0.5, 1$.

A.3 IMPLEMENTATION DETAILS

We train with 100 epochs for TCGA-NSCLC, and 200 epochs for others. We use CosineAnnealingWarmRestart (Loshchilov & Hutter, 2016) with 5 restarts. For adam optimizer, $wd = 1e-4$, and (beta1, beta2)=$(0.9, 0.999)$. Batch-size is 1 following conventions.

B ADDITIONAL EXPERIMENTS

B.1 SUBSAMPLING VERSUS OTHER AUGMENTATION METHODS

Table 6: With varying subsampling rate p, our method shows robust performance gain compared to previous methods, such as DTFD-MIL or RANKMIX, regardless of the model architecture.

Method/Dataset	CAMELYON-16			TCGA-NSCLC		
	ACC (↑)	AUC (↑)	NLL (↓)	ACC (↑)	AUC (↑)	NLL (↓)
ABMIL	0.821±0.015	0.808±0.034	1.185±0.395	0.832±0.039	0.884±0.044	0.708±0.274
+ DTFD-MIL(AFS)	0.847±0.016	**0.844**±0.052	1.014±0.255	0.834±0.034	0.893±0.030	0.980±0.178
+ SUB $p = 0.1$	0.829±0.019	0.812±0.046	**0.780**±0.093	**0.852**±0.021	**0.920**±0.021	**0.513**±0.127
+ SUB $p = 0.2$	0.843±0.010	0.837±0.038	0.837±0.154	0.844±0.024	0.914±0.023	0.572±0.123
+ SUB $p = 0.4$	**0.851**±0.009	**0.844**±0.037	1.015±0.207	0.835±0.027	0.899±0.027	0.640±0.103
DSMIL	0.839±0.012	0.833±0.063	1.620±1.145	0.831±0.022	0.897±0.021	0.737±0.098
+ RANKMIX	0.851±0.013	0.865±0.036	**0.538**±0.070	0.820±0.026	0.894±0.032	0.623±0.119
+ SUB $p = 0.1$	0.853±0.021	0.851±0.046	0.780±0.287	**0.853**±0.026	**0.922**±0.022	**0.598**±0.197
+ SUB $p = 0.2$	**0.866**±0.022	**0.870**±0.046	0.730±0.278	0.850±0.032	0.919±0.024	0.623±0.173
+ SUB $p = 0.4$	0.848±0.031	0.851±0.052	0.750±0.211	0.845±0.025	0.913±0.023	0.617±0.093

B.2 PATCH LEVEL ANNOTATION

This is attention visualization on CAMELYON-16 as it have patch-level annotation. Experts' annotation is outlined with blue. First row contains a split of a original WSI. We use our model Slot-MIL. When we apply subsampling, model detects tumor area more accurately. This helps to make informative slots. Deeper shades of blue indicate higher attention, while shades closer to white indicate

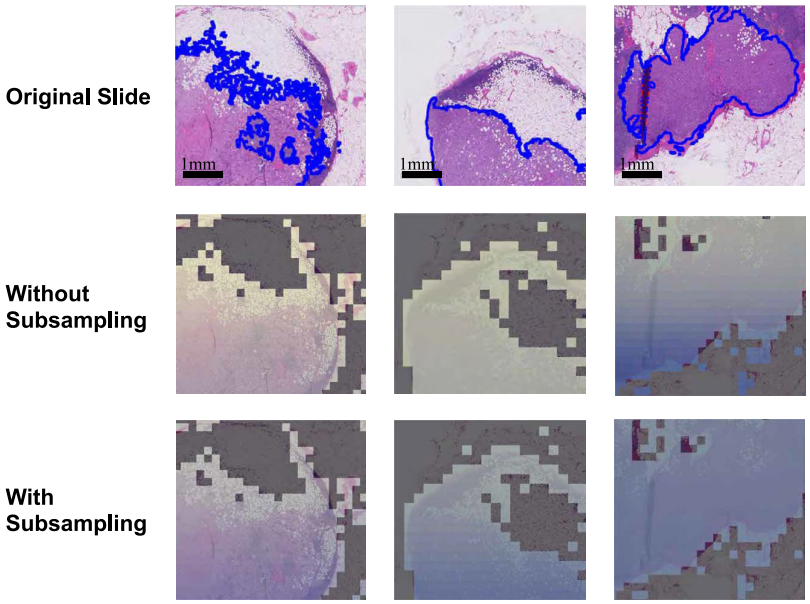


Figure 4: **(a),(b)** With the help of subsampling, Slot-MIL detects tumor area more accurately. **(c)** Without subsampling, the attention score tends to concentrate on a small area. With subsampling, the attention is distributed more evenly, aligning with experts' annotations.

lower attention. Gray patches represent the background, which is excluded in the pre-processing stage.

B.3 Comparing sub-mix and rank-mix

Without self-training, which is more fair comparison in terms of training complexity, SUBMIX shows better performance than RANKMIX. Also, the performance difference between Appendix B.1 on RANKMIX shows the superiority of our model *Slot-MIL*, as RANKMIX is model-agnostic method that trains with the help of baseline model. All the experiments are done above the *Slot-MIL*.

Table 7: Without self-training, SUBMIX works way better than RANKMIX.

Method/Dataset	CAMELYON-16			TCGA-NSCLC		
	ACC (↑)	AUC (↑)	NLL (↓)	ACC (↑)	AUC (↑)	NLL (↓)
RANKMIX	0.870±0.023	0.914±0.025	0.525±0.087	0.873±0.018	0.932±0.021	0.532±0.173
RANKMIX w/o self-training	0.856±0.015	0.883±0.035	0.550±0.103	0.860±0.030	0.926±0.030	0.542±0.194
SUBMIX	0.890±0.020	0.921±0.020	0.448±0.103	0.873±0.023	0.929±0.019	0.511±0.141

B.4 Ablation on Subsampling Rate

CAMELYON-16 works well with high subsampling rate p, and TCGA does not affected by p that much as it consists of around 80% positive patches.

B.5 Ablation on Late Mix

The late mix ablation is done on Slot-MIL with SUBMIX augmentation. So the **baseline** Slot-MIL means **Slot-MIL + SUBMIX with** $L = 0$. We omit this on table for brevity. $\alpha = 0.5$ for all experiments. $p = 0.2$ for CAMELYON-16, and $p = 0.4$ for CAMELYON-17.

Table 8: Optimal subsampling rate differs per dataset. But merely any subsampling rate works better than not applying it.

Method/Dataset	CAMELYON-16			TCGA-NSCLC		
	ACC (↑)	AUC (↑)	NLL (↓)	ACC (↑)	AUC (↑)	NLL (↓)
Slot-MIL	0.834±0.047	0.893±0.023	1.242±0.979	0.852±0.025	0.914±0.016	1.001±0.202
+ SUB $p = 0.1$	0.869±0.016	0.905±0.022	0.508±0.043	0.870±0.017	0.929±0.019	0.789±0.326
+ SUB $p = 0.2$	0.873±0.021	0.911±0.021	0.600±0.093	0.873±0.020	0.931±0.019	0.751±0.333
+ SUB $p = 0.4$	0.881±0.024	0.919±0.022	0.731±0.286	0.873±0.020	0.931±0.023	0.866±0.313

Table 9: Starting mixup from initial epoch is not recommended especially in CAMELYON-16, and CAMELYON-17.

Method/Dataset	CAMELYON-16			CAMELYON-17		
	ACC (↑)	AUC (↑)	NLL (↓)	ACC (↑)	AUC (↑)	NLL (↓)
Slot-MIL	0.867±0.011	0.908±0.016	0.572±0.074	0.785±0.060	0.804±0.036	0.829±0.202
Slot-MIL + $L = 0.1$	0.872±0.022	0.907±0.017	0.509±0.072	0.793±0.004	0.831±0.020	0.641±0.014
Slot-MIL + $L = 0.2$	0.890±0.020	0.921±0.020	0.448±0.103	0.805±0.016	0.835±0.015	0.633±0.035
Slot-MIL + $L = 0.3$	0.881±0.019	0.917±0.019	0.497±0.078	0.799±0.012	0.833±0.010	0.649±0.040

B.6 EXPERIMENT WITH SIMCLR-BASED FEATURES

In order to show *Slot-MIL*'s superiority regardless of feature extraction method and to be consistent with recent papers' highest performance we conduct experiment on SimCLR-based feature provided by DSMIL (Li et al., 2021). One might wonder why not using SimCLR-based feature for main experiments. Considering the fact that training a self-supervised feature on WSIs takes 4 days using 16 Nvidia V100 Gpus (Xiang & Zhang, 2022) and 2 months to be well-optimized (Li et al., 2021), it may not be applicable to all real-world scenarios. In addition, our experiment shows that the SimCLR-based features are so powerful that the simple mean/max pooling reaches nearly the best performance which makes comparison less meaningful. Nevertheless, given that not all tasks possess an optimal feature extractor, we contend that our robust method, which performs effectively across various extractors, holds greater value and is better suited for real-world applications.

Table 10: Results on SimCLR-based features

Method/Dataset	CAMELYON-16			TCGA-NSCLC		
	ACC (↑)	AUC (↑)	NLL (↓)	ACC (↑)	AUC (↑)	NLL (↓)
Meanpool	0.693±0.000	0.604±0.003	0.674±0.023	0.927±0.014	0.972±0.010	**0.232**±0.055
Maxpool	0.920±0.002	0.967±0.002	0.353±0.147	0.920±0.023	0.961±0.013	0.608±0.053
ABMIL	0.921±0.009	0.972±0.002	0.234±0.033	0.933±0.018	**0.981**±0.010	0.263±0.101
DSMIL	0.916±0.012	0.968±0.008	0.456±0.182	0.936±0.017	**0.981**±0.010	0.324±0.133
TRANSMIL	0.889±0.026	0.939±0.010	0.988±0.188	0.924±0.020	0.974±0.009	0.381±0.127
ILRA	**0.923**±0.009	0.973±0.007	0.333±0.043	0.933±0.018	**0.981**±0.011	0.277±0.103
Slot-MIL	0.922±0.008	0.972±0.007	0.294±0.065	**0.937**±0.018	**0.981**±0.011	0.276±0.131
Slot-MIL + SubMix	**0.923**±0.009	**0.975**±0.008	**0.229**±0.071	0.935±0.018	**0.981**±0.012	0.248±0.105

B.7 MC INFERENCE

MC inference performance is measured by getting mean of k-predictions for randomly subsampled subsets per a WSI. We empirically found that k less than 100 degrades performance. With k over 100, MC inference performance quite matches with full patch inference performance. Although MC inference gets better calibrated prediction, is not recommended considering the complexity.

B.8 SUBSAMPLING AND ATTENTION REGULARIZATION

Table 11: Full Patch inference / MC inference comparison on CAMELYON-16

Model/Method	Full Patch			MC Inference(k=100)		
	ACC (↑)	AUC (↑)	NLL (↓)	ACC (↑)	AUC (↑)	NLL (↓)
Sub ($p = 0.2$)	0.873	0.911	0.600	0.857	0.899	0.529
Sub ($p = 0.4$)	0.881	0.919	0.731	0.854	0.899	0.605
SubMix ($p = 0.2, \alpha = 0.5$)	0.869	0.905	0.509	0.853	0.901	0.476
SubMix ($p = 0.4, \alpha = 0.5$)	0.890	0.921	0.448	0.866	0.924	0.403

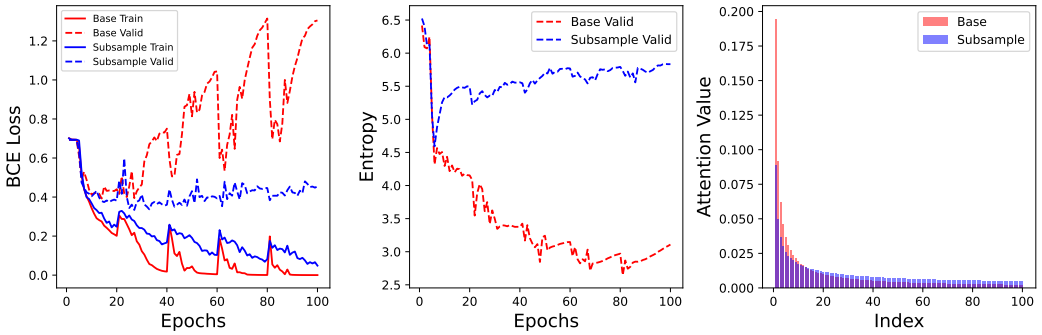


Figure 5: Results of Slot-MIL + SUB on TCGA-NSCLC.

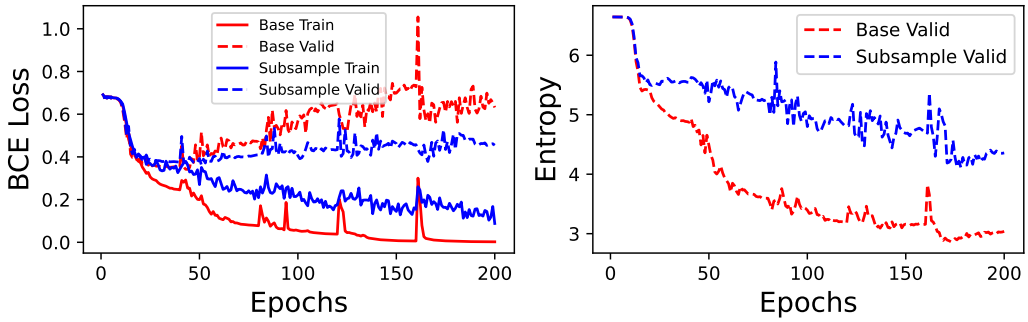


Figure 6: Results of ABMIL on CAMELYON-16.

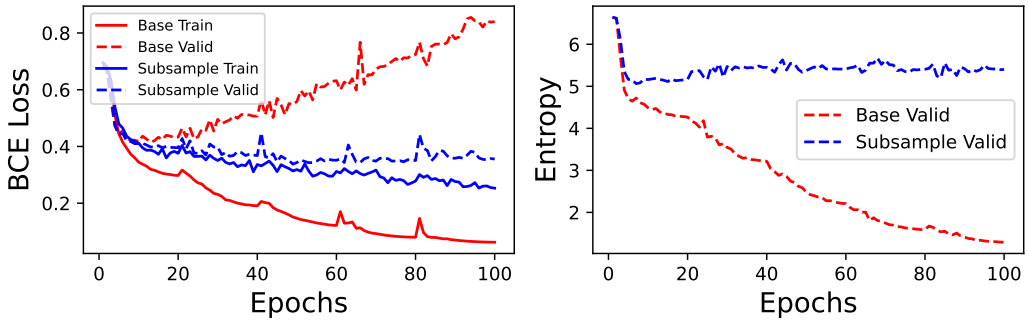


Figure 7: Results of ABMIL on TCGA-NSCLC.