Cracking Instance Jigsaw Puzzles: An Alternative to Multiple Instance Learning for Whole Slide Image Analysis (Supplementary Material)

Xiwen Chen[1*] Peijie Qiu[2*] Wenhui Zhu[3*] Hao Wang[1] Huayu Li[4] Xuanzhao Dong[3]
Xiaotong Sun[5] Xiaobing Yu[2] Yalin Wang[3] Abolfazl Razi[1] Aristeidis Sotiras[2†]
[1] Clemson University, [2] Washington University in St. Louis, [3] Arizona State University,
[4] University of Arizona, [5] University of Arkansas

A. Proofs in Section 3

A.1. Proof of Proposition 1

Without loss of generality, we define any random permutation/shuffling of a bag of instances as $\mathbf{X}_\sigma = \mathcal{S}[\mathrm{X}] = \{\mathbf{x}_{\sigma(1)}, \mathbf{x}_{\sigma(2)}, \cdots, \mathbf{x}_{\sigma(n)}\} \in \mathbb{R}^{n\times d}$, where σ is a permutation of indices $\{1, 2, \cdots, n\}$. If we further define a permutation matrix $\mathbf{P}_\sigma \in \mathbb{R}^{n\times n}$, we have

$$\mathbf{X}_\sigma = \mathbf{P}_\sigma \mathbf{X},$$

where each row and each column in $\mathbf{P}_\sigma$ has exactly one element equal to 1, with the other elements being zero. $(\mathbf{P}_\sigma)_{ij} = 1$ implies the i-th instance of $\mathbf{X}_\sigma$ from the j-th instance of $\mathbf{X}$. It is easy to verify that $\mathbf{P}_\sigma$ is an orthonormal matrix such that $\mathbf{P}_\sigma^\top \mathbf{P}_\sigma = \mathbf{P}_\sigma \mathbf{P}_\sigma^\top = \mathbf{I}$. This is because $\mathbf{P}_\sigma^\top = \mathbf{P}_\sigma^{-1}$ denotes the inverse process of permutation, which should restore the original order of instances.

We start with introducing the following lemma and corollary to support the proof of **Proposition 1**.

Lemma 1. *For any element-wise activation function* $\mathrm{act}(\cdot)$*, the following permutation equivalence holds:*

$$\mathrm{act}(\mathbf{P}_\sigma \mathbf{X}) = \mathbf{P}_\sigma \,\mathrm{act}(\mathbf{X}).$$

Proof. Since the activation function is applied element-wise to the permuted input, we have

$$\begin{aligned}\mathrm{act}(\mathbf{P}_\sigma \mathbf{X}) &= [\mathrm{act}(\mathbf{x}_{\sigma(1)}), \mathrm{act}(\mathbf{x}_{\sigma(2)}), \cdots, \mathrm{act}(\mathbf{x}_{\sigma(n)})] \\ &= [\mathrm{act}(\mathbf{x_1})_{\sigma(1)}, \mathrm{act}(\mathbf{x}_2)_{\sigma(2)}, \cdots, \mathrm{act}(\mathbf{x}_n)_{\sigma(n)}] \\ &= \mathbf{P}_\sigma \,\mathrm{act}(\mathbf{X}).\end{aligned}$$

This completes the proof. □

Corollary 1. *Lemma 1 can also be extended to the following form:*

$$\begin{aligned}\mathrm{act}(\mathbf{X}^\top \mathbf{P}_\sigma^\top) &= \mathrm{act}(\mathbf{P}_\sigma \mathbf{X})^\top \\ &= (\mathbf{P}_\sigma \,\mathrm{act}(\mathbf{X}))^\top \\ &= \mathrm{act}(\mathbf{X}^\top)\mathbf{P}_\sigma^\top.\end{aligned}$$

A.1.1. ABMIL

The standard attention pooling ($\text{Attn-Pool}(\cdot)$) in ABMIL without any positional encoding can be described as

$$\text{Attn-Pool}(\mathbf{X}) = \underbrace{\mathrm{softmax}\left(\mathbf{W}^\top \tanh(\mathbf{V}\mathbf{X}^\top)\right)}_{\mathbb{R}^{1\times n}} \mathbf{X},$$

where $\mathbf{V} \in \mathbb{R}^{L\times d}$ and $\mathbf{W} \in \mathbb{R}^{L\times 1}$ are learnable weight matrices. We want to prove that

$$\text{Attn-Pool}(\mathbf{X}_\sigma) = \text{Attn-Pool}(\mathbf{X}).$$

Applying Corollary 1, the above attention pooling for a permuted bag of instances can be described as

$$\begin{aligned}&\text{Attn-Pool}(\mathbf{X}_\sigma) \\ &= \mathrm{softmax}\left(\mathbf{W}^\top \tanh(\mathbf{V}(\mathbf{P}_\sigma \mathbf{X})^\top)\right)(\mathbf{P}_\sigma \mathbf{X}) \\ &= \mathrm{softmax}\left(\mathbf{W}^\top \tanh(\mathbf{V}\mathbf{X}^\top \mathbf{P}_\sigma^\top)\right)(\mathbf{P}_\sigma \mathbf{X}) \\ &= \mathrm{softmax}\left(\mathbf{W}^\top \tanh(\mathbf{V}\mathbf{X}^\top)\right)(\mathbf{P}_\sigma^\top \mathbf{P}_\sigma \mathbf{X}) \\ &= \text{Attn-Pool}(\mathbf{X}).\end{aligned}$$

This completes the proof that attention-based pooling in ABMIL is permutation-invariant.

A.1.2. TransMIL

The standard self-attention without positional encoding can be described as follows:

$$\text{Self-Attn}(\mathbf{X}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \mathbf{K} = \mathbf{X}\mathbf{W}^K, \mathbf{V} = \mathbf{X}\mathbf{W}^V,$$

*Equal contribution
†*Corresponding Author*

where $\mathbf{W}^Q$, $\mathbf{W}^K$, $\mathbf{W}^V \in \mathbb{R}^{d\times d_k}$ are learnable weight matrices. Applying Lemma 1 and Corollary 1, the self-attention evaluated on a permuted bag of instances can be described as

$$\begin{aligned}&\text{Self-Attn}(\mathbf{X}_\sigma)\\ =&\,\text{softmax}\left(\frac{(\mathbf{P}_\sigma\mathbf{Q})(\mathbf{P}_\sigma\mathbf{K})^\top}{\sqrt{d_k}}\right)(\mathbf{P}_\sigma\mathbf{V})\\ =&\,\text{softmax}\left(\frac{(\mathbf{P}_\sigma\mathbf{Q})\mathbf{K}^\top}{\sqrt{d_k}}\right)(\mathbf{P}_\sigma^\top\mathbf{P}_\sigma\mathbf{V})\\ =&\,\mathbf{P}_\sigma\,\text{Self-Attn}(\mathbf{X}).\end{aligned}$$

The above result can be easily extended to a transformer layer with self-attention blocks.

Final Pooling. If we consider a global average pooling or max pooling after the final transformer layer, we have

$$\begin{aligned}\text{avgpool}\,(\text{Self-Attn}(\mathbf{X}_\sigma)) &= \text{avgpool}\,(\text{Self-Attn}(\mathbf{X}))\\ \text{maxpool}\,(\text{Self-Attn}(\mathbf{X}_\sigma)) &= \text{maxpool}\,(\text{Self-Attn}(\mathbf{X}))\end{aligned}$$

Although the self-attention is permutation-equivariant instead of permutation-invariant, applying the permutation-invariant global average pooling or max pooling on top of it ensures permutation invariance [16, Sec. 3].

Class Token. In the case of adding a class token $\mathbf{x}_{cls}$ (instead of final pooling) to the instances as in TransMIL [14], the permutation-invariance of TransMIL is trivial to verify. This is because the attention between the output of the class token is invariant to the permutation of the input tokens.

A.1.3. General Attention-based Pooling Mechanisms

The above verification of permutation invariance can also be extended to other attention-based pooling mechanisms, which typically involves a dot product between input instances, *i.e.,* $\mathbf{X}^\top\mathbf{X}$.

Lemma 2. *For the orthonormal matrix $\mathbf{P}_\sigma$, we have the following permutation-invariant property:*

$$\begin{aligned}\mathbf{X}_\sigma^\top\mathbf{X}_\sigma &= (\mathbf{P}_\sigma\mathbf{X})^\top(\mathbf{P}_\sigma\mathbf{X})\\ &= \mathbf{X}^\top\mathbf{P}_\sigma^\top\mathbf{P}_\sigma\mathbf{X}\\ &= \mathbf{X}^\top\mathbf{X}.\end{aligned}$$

Lemma 2 in conjunction with Lemma 1 and Corollary 1 are the key for proving the permutation-invariance in the case of ABMIL and TransMIL. The same principal can be generalized to verify the permutation invariance in more general attention-based pooling mechanisms. However, this is beyond the score of this paper, we leave it to the future exploration.

A.1.4. Permutation-variance with Positional Encoding

We define a bag of instances with positional encoding as

$$\mathbf{X}_{\text{PE}} = \mathbf{X} + \mathbf{PE},$$

where $\mathbf{PE} = [\text{PE}_1, \text{PE}_2, \cdots, \text{PE}_n]$, with PE_n corresponding to the positional encoding for the n-th instance in a bag. Likewise, the permuted n instances with positional encoding is denoted as

$$(\mathbf{X}_{\text{PE}})_\sigma = \mathbf{X}_\sigma + \mathbf{PE}.$$

In the case of any permutation-invariant MIL (denoted as $\text{MIL}(\cdot)$), we have

$$\begin{aligned}\text{MIL}((\mathbf{X}_{\text{PE}})_\sigma) &= \text{MIL}(\mathbf{P}_\sigma^\top(\mathbf{X}_{\text{PE}})_\sigma)\\ &= \text{MIL}(\mathbf{P}_\sigma^\top(\mathbf{X}_\sigma + \mathbf{PE}))\\ &= \text{MIL}(\mathbf{P}_\sigma^\top(\mathbf{P}_\sigma\mathbf{X} + \mathbf{PE}))\\ &= \text{MIL}(\mathbf{X} + \mathbf{P}_\sigma^\top\mathbf{PE}).\end{aligned}$$

For any non-trivial $\mathbf{PE} \neq \mathbf{I}$,

$$\mathbf{X} + \mathbf{P}_\sigma^\top\mathbf{PE} = \mathbf{X} + \mathbf{PE},$$

if and only if $\mathbf{P}_\sigma = \mathbf{I}$. This immediately suggests that there is no non-trivial permutation $\mathbf{P}_\sigma$ and $\mathbf{PE}$ to ensure the permutation invariance in MIL when adding positional encoding. Hence, we conclude that models with positional encoding are not generally permutation-invariant.

A.2. Proof of Theorem 2

Theorem 2. *Incorporating positional information can lower the classification-error upper bound. This is because*

$$H(\mathbf{Y}|\mathbf{X}) \geq H(\mathbf{Y}|\mathbf{X}, \mathbf{P}),$$

where $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n\}$ denotes the positional coordinates associated with each instance on the raw WSIs.

To prove **Theorem 2**, we first introduce the upper bound of the classification error in Lemma 3.

Lemma 3. *([7]) The Bayesian classification error $P_e = \int P_X(x)\,[1 - \max_{\mathbf{Y}} P(\mathbf{Y}|\mathbf{X})]\,d\mathbf{X}$, is bounded by:*

$$P_e \leq \frac{1}{2}H(\mathbf{Y}|\mathbf{X}),$$

where $P(\mathbf{Y}|\mathbf{X})$ is the posterior probability of the class label $\mathbf{Y}$ given a bag $\mathbf{X}$. $H(\cdot)$ denotes the entropy.

Remark 1. *According to Lemma 3, **without** incorporating the position information, the upper bound of the classification error is directly presented as $H(\mathbf{Y}|\mathbf{X})$. When incorporating the position information $\mathbf{P}$, the upper bound is presented as $H(\mathbf{Y}|\mathbf{X}, \mathbf{P})$.*

Below, we begin to prove **Theorem 2** by showing that $H(\mathbf{Y}|\mathbf{X},\mathbf{P})$ is a tighter error bound than $H(\mathbf{Y}|\mathbf{X})$, *i.e.,* $H(\mathbf{Y}|\mathbf{X},\mathbf{P}) \leq H(\mathbf{Y}|\mathbf{X})$.

Proof. According to the definition of entropy [3], $H(\mathbf{Y}|\mathbf{X})$ can be presented as

$$H(\mathbf{Y}|\mathbf{X}) = -\mathbb{E}_{\mathcal{X},\mathcal{Y}}\left[\log P(\mathbf{y}|\mathbf{x})\right].$$

Likewise, $H(\mathbf{y}|\mathbf{x},\mathbf{p})$ is presented as

$$H(\mathbf{Y}|\mathbf{X},\mathbf{P}) = -\mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log P(\mathbf{y}|\mathbf{x},\mathbf{p})\right].$$

By their definition,

$$\begin{aligned} & H(\mathbf{Y}|\mathbf{X}) - H(\mathbf{Y}|\mathbf{X},\mathbf{P}) \qquad (1)\\ =& -\mathbb{E}_{\mathcal{X},\mathcal{Y}}\left[\log P(\mathbf{y}|\mathbf{x})\right] + \mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log P(\mathbf{y}|\mathbf{x},\mathbf{p})\right]\\ =& -\mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log P(\mathbf{y}|\mathbf{x})\right] + \mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log P(\mathbf{y}|\mathbf{x},\mathbf{p})\right]\\ =& -\mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log P(\mathbf{y}|\mathbf{x}) - \log P(\mathbf{y}|\mathbf{x},\mathbf{p})\right]\\ =& -\mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x},\mathbf{p})}\right] \end{aligned}$$

We observe that $p(\mathbf{y}|\mathbf{x},\mathbf{p}) = \frac{p(\mathbf{y},\mathbf{p}|\mathbf{x})}{p(\mathbf{p}|\mathbf{x})}$. Now, plugging it into Eq. 1, we have

$$\begin{aligned} & -\mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log \frac{p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y}|\mathbf{x},\mathbf{p})}\right] \qquad (2)\\ =& -\mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log \frac{p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y},\mathbf{p}|\mathbf{x})}\right]. \end{aligned}$$

Note that Eq. 2 is also the definition of the ***conditional mutual information*** $I(\mathbf{Y};\mathbf{P}|\mathbf{X})$, which should always be **non-negative** [3]. Here, we will still provide further proof.

We can apply Jensen's Inequality,

$$\begin{aligned} & -\mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\log \frac{p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y},\mathbf{p}|\mathbf{x})}\right] \qquad (3)\\ \overset{(a)}{\geq}& -\log \mathbb{E}_{\mathcal{X},\mathcal{Y},\mathcal{P}}\left[\frac{p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y},\mathbf{p}|\mathbf{x})}\right]\\ =& -\log \mathbb{E}_{\mathcal{X}}\mathbb{E}_{\mathcal{P},\mathcal{Y}|\mathcal{X}}\left[\frac{p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y},\mathbf{p}|\mathbf{x})}\right]\\ \overset{(b)}{=}& -\log 1 = 0. \end{aligned}$$

Here, Eq. 3(b) is because,

$$\begin{aligned} & \mathbb{E}_{\mathcal{P},\mathcal{Y}|\mathcal{X}}\left[\frac{p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y},\mathbf{p}|\mathbf{x})}\right] \qquad (4)\\ =& \sum_{\mathbf{y},\mathbf{p}|\mathbf{x}\in\mathcal{P},\mathcal{Y}|\mathcal{X}} p(\mathbf{y},\mathbf{p}|\mathbf{x})\frac{p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})}{p(\mathbf{y},\mathbf{p}|\mathbf{x})}\\ \overset{(a)}{=}& \sum_{\mathbf{y},\mathbf{p}|\mathbf{x}\in\mathcal{P},\mathcal{Y}|\mathcal{X}} p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})\\ =& \sum_{\mathbf{p}|\mathbf{x}\in\mathcal{P}|\mathcal{X}} p(\mathbf{p}|\mathbf{x}) \sum_{\mathbf{y}|\mathbf{x}\in\mathcal{Y}|\mathcal{X}} p(\mathbf{y}|\mathbf{x})\\ =&1. \end{aligned}$$

The right term of (a) Eq. 4 can be interpreted as this: we denote a new random variable $\mathbf{Z} : p(\mathbf{z}) = p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x}), \mathbf{z} \in \mathcal{P},\mathcal{Y}|\mathcal{X}$. This is the outer product distribution which assigns probability $p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})$ to each $(\mathbf{p},\mathbf{y}|\mathbf{x})$. Hence, the right term of (a) in Eq. 4 becomes the sum of the probabilities of all possible outcomes of a random variable $\mathbf{Z}$, which has to be 1.

Now, we go back to the inequality Eq. 3(a). Apparently, when $p(\mathbf{y},\mathbf{p}|\mathbf{x}) = p(\mathbf{p}|\mathbf{x})p(\mathbf{y}|\mathbf{x})$, the equality holds. Therefore, we can conclude that $H(\mathbf{Y}|\mathbf{X}) - H(\mathbf{Y}|\mathbf{X},\mathbf{P}) \geq 0$, and the equality holds *if and only if* $\mathbf{P}$ and $\mathbf{Y}$ are conditionally independent given $\mathbf{X}$. The proof is completed.

□

B. More Discussion on Siamese Network

B.1. Network Architecture

The detailed network architectures of the two variants of the proposed method are shown in Fig. S1.

Ours [Trans.] Given extracted features from a bag $\mathbf{X} \in \mathbb{R}^{n\times d}$, we first apply a `MLP` for reducing its dimensions. Specifically, the MLP is able to be presented as:

$$\mathbf{X} \in \mathbb{R}^{n\times d} \xrightarrow{\mathtt{g}(\cdot)} \mathbf{X} \in \mathbb{R}^{n\times d/2} \xrightarrow{\mathtt{g}(\cdot)} \mathbf{X} \in \mathbb{R}^{n\times 128}, \qquad (5)$$

where $\mathtt{g}(\cdot)$ denotes a linear layer with a ReLU activation function. This operation can substantially reduce the computational cost. Afterward, we perform `squaring` and apply `PPEG` as described in TransMIL [14]. Then, we `flatten` the positioned features and feed them into two transformer blocks, and we obtain $f(\mathbf{X})$ or $f(S[\mathbf{X}])$ here. Further, we apply `Average Pooling` for aggregating all instance features for the final bag-level classification.

Ours [CNN] Similar to **Ours [Trans.]**, we first apply `MLP`, `squaring`, and `PPEG` to obtain the positioned instance features. Then, we apply two residual blocks [6], where each blocks have the same input channel and output channel of 128. We obtain the block from `pytorch` official implementation (`https://github.com/pytorch/vision/blob/main/torchvision/models/resnet.py`). Subsequently, we `flatten` the features and apply `average pooling` again for the final bag-level classification.

B.2. 1D CAM Derivation

Class activation mapping (CAM) [19] is the most commonly used method to visualize the most discriminative areas that the model focuses on to make its decision. Typically, the CAM is computed for any network ended with a global average pooling and linear classifier, which is the primary reason why we use global average pooling for both CNN and Transformer. Below, we derive the 1-dimensional CAM in our case. First, we reshape the output from either

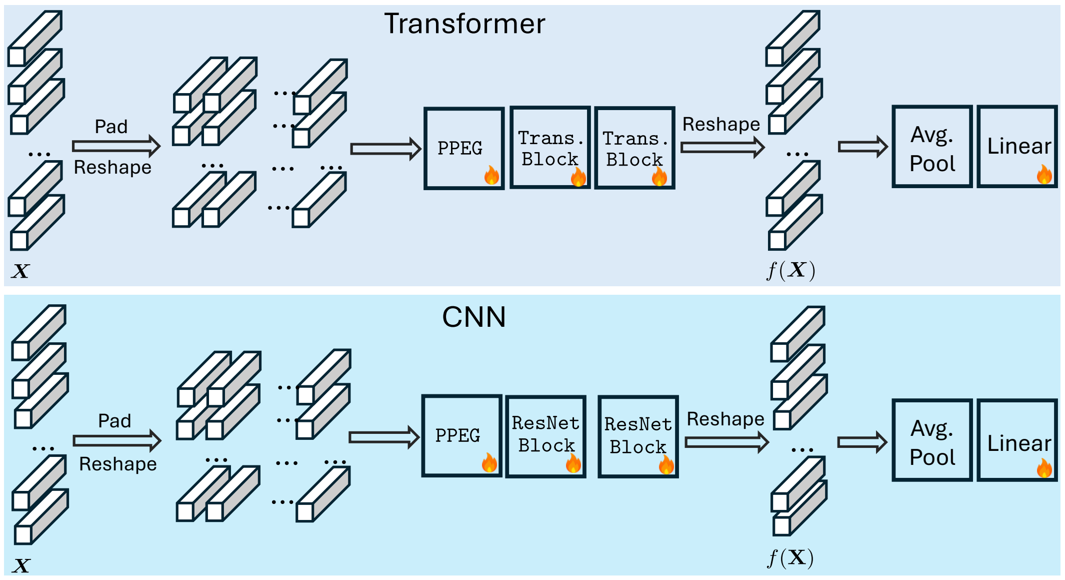

Figure S1. The network architecture of CNN ans Transformer used in the proposed method.

our CNN or Transformer as a 2D matrix with L dimensions: $F \in \mathbb{R}^{n\times L}$, where n is the number of tiles in an WSI. We use subscript i to index the feature vector corresponding to the i-th instance: $F_i \in \mathbb{R}^l$. The predicted classification logit for the c-th class $\texttt{logit}_c$ reads

$$\texttt{logit}_c = \underbrace{\sum_{l=1}^{L} w_l^c \cdot \underbrace{\frac{1}{n}\sum_{i=1}^{n} F_i}_{\text{average pooling}}}_{\text{linear classifier}}$$
$$= \frac{1}{n}\sum_{i=1}^{n}\sum_{l=1}^{L} w_l^c F_i,$$

where w_l^c is the l-th unit corresponding to the c-th class in the weight of the linear classifier. CAM is then defined as

$$\mathsf{CAM}_i^c = \sum_{l=1}^{L} w_l^c F_i,$$

which is the importance of the activation for the i-th instance leading to the classification of an WSI to class c.

We provide the simplest solution to visualize the importance of each tile contributing to the final classification of an WSI via CAM. However, plain CAM relies on the global average pooling to obtain the importance. In a more general cases, where the network architecture may not contain global average pooling, the importance of each tile can also be obtained by using other variants of CAM, *e.g.,* Grad-CAM [13], Grad-CAM++ [1], etc. However, this is not the main focus of this paper, hence we leave it for future work.

B.3. Computational Complexity

It is obvious that the Siamese network introduces additional computational complexity by having an additional branch compared to MIL methods. However, we argue that the overall computational burden in practice is still acceptable. In practice, most MIL methods do not effectively leverage parallel computation in modern deep learning training, as they typically set the batch size to 1 due to the varying number of instances. However, it is easy to fit a way larger number of bags/instances to most modern GPUs for better parallel acceleration. For the proposed Siamese network solution, we can easily stack the instances before and after shuffling to form a batch size of 2, as they have the same number of instances. By leveraging this parallel acceleration on GPUs, the computation of the proposed method can be as fast as the MIL methods with a single network branch. As shown in Table S1, we also empirically validate this assumption. We set the input bag with a shape $\mathbb{R}^{10000\times 768}$, which is aligned with the output feature dimension of the Swin-VIT extractor. The results illustrate that employing parallel acceleration can reduce the time cost by 41% and 34% for two of our proposed architectures, respectively. *This makes the practical computation complexity of our dual-branch Siamese network is on par with previous MIL methods.*

B.4. Additional Ablation Analysis on CTransPath Features

Please refer to Table S2 for the results, which show a 0.5%-1.2% gain obtained by the proposed model.

Table S1. Time complexity (in *ms*) per iteration by different strategies, benchmarked using input bags containing 10,000 instances, each with 768 dimensions: $\boldsymbol{X} \in \mathbb{R}^{10000\times768}$. The single branch reduces to the same case of standard MIL schemes with only one branch of the neural network; whereas our Siamese solution requires a dual-branch network architecture. Dual-branch **w/o** parallelization denotes the training with a batch size of 1 as in standard MIL training; whereas Dual-branch **w/o** parallelization denotes the training with a batch size of 2.

Strategy	**Ours [Trans.]**	**Ours [CNN]**
single-branch	3.9	1.5
Dual-branch **w/o** Parallelization	7.8	2.9
Dual-branch **w/** Parallelization	4.6	1.9

Table S2. Ablation Analysis on features extracted by CtransPath.

Dataset	Network	$\mathcal{L}_{\text{Equv}}$	acc	f1	AUC	Avg. 3 Metrics
C16	Trans.	✗	95.89	95.62	97.42	96.31
		✓	96.64	96.39	98.00	**97.01**
	CNN	✗	95.21	94.82	96.67	95.57
		✓	96.25	95.99	98.10	**96.78**
TCGA	Trans.	✗	94.53	94.57	97.66	95.59
		✓	95.20	95.19	97.99	**96.13**
	CNN	✗	94.43	94.43	96.99	95.28
		✓	95.11	95.10	97.55	**95.92**

C. A Complete Justification from OT Theory

To make this justification complete and self-contained, we start with the introduction of the forward OT formulation of cracking instance jigsaw puzzles in Sec. C.1 and then introduce our inverse OT formulation and solution in Sec. C.2.

C.1. Forward OT Formulation

Our instance jigsaw puzzles can be formulated as a (forward) OT problem by minimizing the Wasserstein distance (*e.g.,* Earth Mover's Distance) between coordinates associated with the shuffled instances ($\mathbf{p}'$) and those associated with the unshuffled bag ($\mathbf{p}$):

$$\text{EMD}(\mathbf{P}, \mathbf{P}') = \min_{\mathbf{T}\geq 0} \sum_{i,j} \mathbf{T}_{ij}\mathbf{C}^{ij}$$

$$\text{subject to } \sum_i \mathbf{T}_{ij} = \frac{1}{n}, \sum_j \mathbf{T}_{ij} = \frac{1}{n}$$

where $\mathbf{T}$ is the transport flow matrix, and $\mathbf{C}$ is the known cost matrix (*e.g.,* a quadratic cost: $\|p_i - p'_j\|_2^2$).

This forward OT formulation aims at solving the unkown optimal transport plan (*i.e.,* $\mathbf{T}_{\#}$) that can restore the instance ordering from its random arrangement. However, solving this forward OT problem is non-trivial, as it necessitates iterative updates (*e.g.,* Sinkhorn updates [4, 12]). This makes solving the forward OT problem computationally challenging, particularly when deep neural networks are involved in our case.

C.2. Inverse OT Formulation (Proof of Theorem 3)

Our key insight is that solving instance jigsaw puzzles may not require an optimal plan; rather, non-optimal plans can also achieve the same objective. Therefore, we can consider the simplest plan $\tilde{\mathbf{T}}$, *i.e.,* the inverse shuffling operation $\mathcal{S}^{-1}$ (or equivalently, $\mathbf{P}_\sigma^\top$ in a matrix form).

Proof. The inverse optimal transport (OT) problem can be formulated as solving the following optimization problem:

$$\min_\theta \mathcal{L}(\tilde{\mathbf{T}}, \mathbf{T}_{\#}[c_\theta])$$

$$\text{subject to} \quad \mathbf{T}_{\#}[c_\theta] = \min_{\mathbf{T}\geq 0} \sum_{i,j} \mathbf{T}_{ij} c_\theta^{ij},$$

where c_θ is the parameterized cost function, and $\mathcal{L}(\cdot)$ is a loss function, *e.g.,* MSE. $\tilde{\mathbf{T}}$ and $\mathbf{T}_{\#}$ are the observed and optimal transport plan. Some common choices of the loss function $\mathcal{L}$ are mean-squared loss (MSE) and Kullback–Leibler (KL) divergence. However, solving the inverse OT problem requires to solve the forward OT problem to obtain the optimal transport plan, which is intractable in our case. Hence, we approximate the optimal transport plan $\mathbf{T}_{\#}$ with the observed plan $\tilde{\mathbf{T}}$. The above inverse OT objective is simplified to:

$$\min_\theta \sum_{ij} \tilde{\mathbf{T}}_{ij} c_\theta^{ij},$$

where $c_\theta^{ij} = c_\theta(\mathbf{x}_i, \mathbf{x}'_j)$. In the most naive case, $c_\theta(\mathbf{x}_i, \mathbf{x}'_j)$ should be a neural network that takes as input both $\mathbf{x}_i$ and $\mathbf{x}'_j$. Instead, we consider penalizing the L_2 norm between $\mathbf{x}_i$ and $\mathbf{x}'_j$ in the embedding space: $\|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}'_j)\|_2^2$, where f_θ is a neural network.

If we further replace the observed plan $\tilde{\mathbf{T}}_{ij}$ with the inverse shuffling operation in a matrix form $(\text{P}_\sigma^\top)_{ij}$ (see Sec. A.1), the above minimization reduces to

$$\min_\theta \sum_{ij} (\text{P}_\sigma^\top)_{ij} \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}'_j)\|_2^2.$$

Note that $\text{P}_\sigma^\top$ is an orthonormal matrix with exactly one elements equal to 1 at each row and column. Hence, $\sum_i (\text{P}_\sigma^\top)_{ij} = \sum_j (\text{P}_\sigma^\top)_{ij} = 1$ instead of $1/n$. However, this does not affect the results of minimization. Putting everything in a matrix form, we have

$$\min_\theta \frac{1}{2n} \|f_\theta(\mathbf{X}) - \mathbf{P}_\sigma^\top f_\theta(\mathbf{X}_\sigma)\|_2^2,$$

where $\mathbf{X}_\sigma = \mathbf{P}_\sigma \mathbf{X}$. Equivalently, we have

$$\min_\theta \frac{1}{2n} \|\mathbf{P}_\sigma f_\theta(\mathbf{X}) - f_\theta(\mathbf{X}_\sigma)\|_2^2.$$

Replacing the permutation matrix $\mathbf{P}_\sigma$ with the corresponding operator $\mathcal{S}^{-1}$, the above equation reduces to the same as our shuffling equivalence regularization loss. This concludes the proof. $\square$

D. Additional Implementation Details

Reproducibility. The code will be made publicly available upon acceptance. **The core code is provided in the Supplementary file.**

D.1. Implementation Hyperparameters

We adopt the default training setup for baseline models. For our proposed models, we use the AdamW optimizer [9] with an initial learning rate of 5e-4 and a weight decay of 1e-4 for all experiments. Each model is trained for 200 and 20 epochs with a batch size of 1 for WSI classification and survival prediction, respectively. All experiments are implemented in PyTorch (v1.13) [11] and performed on an NVIDIA A100 GPU with 40 GB memory. We present the hyperparameters of all baselines as well as our model in Table S3.

D.2. Dataset for WSI classification

CAMELYON16 dataset is a publicly available collection of WSIs designed to detect metastatic breast cancer in lymph node tissue. It comprises 399 WSIs of lymph node specimens, officially divided into a training set of 270 samples and a test set of 129 samples. Each WSI is associated with a binary label, annotated by expert pathologists, indicating the presence or absence of metastatic cancer. In addition, detailed region-level annotations are provided for cancerous tissue within the WSIs. We employ a threshold-based preprocessing method to filter background information for patch extraction [10, 17]. Each WSI image is cropped into non-overlapping patches of size 256 × 256, resulting in approximately 4.61 million patches at ×20 magnification, with an average of 11,555 patches per slide.

TCGA-NSCLC dataset is also a public dataset for classifying two subtypes of lung cancer (lung squamous cell carcinoma and adenocarcinoma). It includes a total of 1,037 WSIs. Following the preprocessing protocol outlined in CAMELYON16 [10, 17], approximately 13.83 million patches were extracted at ×20 magnification. On average, each WSI yielded 13,335 patches per sample.

We also want to highlight that with three different feature extractors, our experiments are three times the scale compared to previous studies [14, 17].

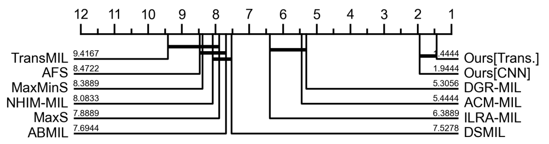

Figure S2. Critical difference diagram based on the Wilcoxon signed-rank test, where the number indicates the average ranks (↑: the higher the better). Methods connected by a single thick line show no statistically significant differences.

D.3. Dataset for WSI Survival Analysis

Both of the following datasets employ a similar preprocessing in that all patches were extracted at ×20 magnification.

TCGA-LUAD dataset is a subset of TCGA-NSCLC dataset that only has a subtype of lung cancer (lung adenocarcinoma). Following the dataset (i.e. `csv` file from `https://github.com/mahmoodlab/Panther`) provided by [15], we processed a subset of the dataset, comprising 463 WSIs from 412 patients.

TCGA-BRCA Dataset is a comprehensive collection curated for the study of Breast Invasive Carcinoma. Following the dataset provided in [8], we processed this dataset to include 931 WSIs from 871 patients.

D.4. Additional Implementation of WSI Survival Analysis

Our implementation strictly adheres to [8], and we use disease-specific survival (DSS). The training framework (e.g., Loss function, the data partition) is implemented based on their repositories: `https://github.com/mahmoodlab/SurvPath`. The features we used is extracted by **UNI** [2] extractor.

E. Statistical Test

Following the recommendation by [5], we utilize the Wilcoxon signed-rank test to compare two classifiers on a single dataset. For summarizing the performance across multiple datasets and feature extractors, we employ a critical difference diagram. In Fig. S2, we present the critical difference diagram with a significance level of $\alpha = 0.05$ (which is a common threshold in hypothesis testing; if the p-value is lower than this value, we treat the performance of two classifiers are significantly different), illustrating that two of our proposed models statistically outperform previous MIL baselines.

References

[1] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolu-

Table S3. Hyperparamters used in the experiments. Here, `Cosine`* denotes cosine decay with 20 epoch linear warmup from 1e-5. `AMP` denotes automatic mixed precision.

Setting	**AB-MIL**	**DS-MIL**	**DTFD-MIL**	**Trans-MIL**	**ILRA-MIL**	**DGR-MIL**	**MHIM-MIL**	**AC-MIL**	**Ours**
Optimizer	Adam	Adam	Adam	Radam	Adam	SGD	Adam	Adam	AdamW
Learning rate	1e-3	1e-4	1e-4	2e-4	1e-4	5e-4	2e-4	1e-4/2e-4	5e-4
Weight decay	0.005	5e-3	1e-4	1e-5	1e-4	1e-4	1e-5	1e-4	1e-4
Scheduler	Cosine*	Cosine	MultiStepLR	LookAhead [18]	Cosine	Cosine*	Cosine	Cosine	LookAhead [18]
Loss	$\mathcal{L}_{bce}$	$\mathcal{L}_{bce}$	$\mathcal{L}_{bce}$ + Tier-2 loss	BCE	$\mathcal{L}_{bce}$	$\mathcal{L}_{bce}, \mathcal{L}_{tri}, \mathcal{L}_{div}$	$\mathcal{L}_{bce}, \mathcal{L}_{con}$	$\mathcal{L}_{bce}, \mathcal{L}_{p}, \mathcal{L}_{d}$	$\mathcal{L}_{bce}, \mathcal{L}_{Equv}$
other	None	Droppath = 0.2	grad. clip = 5	AMP	Xavier initialize	Warmup	None	None	None

tional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018. 4

[2] Richard J Chen, Tong Ding, Ming Y Lu, Drew FK Williamson, Guillaume Jaume, Bowen Chen, Andrew Zhang, Daniel Shao, Andrew H Song, Muhammad Shaban, et al. Towards a general-purpose foundation model for computational pathology. *Nature Medicine*, 2024. 6

[3] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999. 3

[4] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013. 5

[5] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine learning research*, 7:1–30, 2006. 6

[6] 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. 3

[7] Martin Hellman and Josef Raviv. Probability of error, equivocation, and the chernoff bound. *IEEE Transactions on Information Theory*, 16(4):368–372, 1970. 2

[8] Guillaume Jaume, Anurag Vaidya, Richard J Chen, Drew FK Williamson, Paul Pu Liang, and Faisal Mahmood. Modeling dense multimodal interactions between biological pathways and histology for survival prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11579–11590, 2024. 6

[9] I Loshchilov. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[10] Nobuyuki Otsu et al. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975. 6

[11] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[12] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019. 5

[13] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017. 4

[14] 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, 3, 6

[15] Andrew H Song, Richard J Chen, Tong Ding, Drew FK Williamson, Guillaume Jaume, and Faisal Mahmood. Morphological prototyping for unsupervised slide representation learning in computational pathology. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11566–11578, 2024. 6

[16] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017. 2

[17] 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. 6

[18] Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019. 7

[19] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016. 3