A APPENDIX

A.1 IMPLEMENTATION DETAILS

All code for this work was implemented in Python 3.8, using the PyTorch library for the machine learning functionality. Hyperparameter tuning was carried out using the Optuna libary. Some local experiments were carried out on a Dell XPS Windows laptop, utilising a GeForce GTX 1650 graphics card with 4GB of VRAM. GPU support for machine learning was enabled through CUDA v11.0. Other longer running experiments, such as hyperparameter tuning, were carried out on a remote GPU node utilising a Volta V100 Enterprise Compute GPU with 16GB of VRAM. The longest model to train was the GNN for the CRC dataset, which took just under half an hour. The longest running experiment was the hyperparameter analysis for the CRC dataset at just under 40 hours. This was due to high number of samples for the local surrogate methods and the fact that the results were average over 10 different models. The randomisation of data splits was fixed using seeding; the seed values are provided in the code for each experiment. We use the following sources for our data:

- The annotated SIVAL dataset was downloaded from the publicly accessible page: `http://pages.cs.wisc.edu/~bsettles/data/`.
- The MNIST dataset was access directly from the PyTorch Python library: `https://pytorch.org/vision/stable/datasets.html#mnist`.
- The CRC dataset was downloaded from the publicly accessible page: `https://warwick.ac.uk/fac/cross_fac/tia/data/crchistolabelednucleihe/`.
- The Musk dataset was downloaded from the publicly accessible page: `https://archive.ics.uci.edu/ml/datasets/Musk+%28Version+2%29`
- The Tiger, Elephant and Fox datasets were downloaded from the publicly accessible page: `https://archive.ics.uci.edu/ml/datasets/Musk+%28Version+2%29`

A.2 MODELS

In this work, we trained four different models for each dataset. In this section, we provide further details on each of the models. Each model was tuned independently for each dataset, so in the later sections we provide details of the specific architectures for each dataset. We also tuned the learning rate, weight decay, and dropout for each of the models independently for every dataset. Again, these are given in the later sections for each specific dataset.

MI-Net Each instance is embedded to a fixed size, and then these embeddings are aggregated to give a single bag embedding. This aggregation can either take the mean (mil-mean) or max (mil-max) of the instance embeddings. The bag embedding is then classified to give an overall bag prediction. For this model, we tuned the number and size of fully connected (FC) layers, as well as the choice of aggregation function.

mi-Net A prediction is made for each individual instance, and then these are aggregated to a single outcome for the bag as a whole. Similar to the MI-Net model, we tuned the FC layers and the aggregation function.

MI-Attn Each instance is embedded to a fixed size, and then the mil-attn block produces an attention value for each instance embedding. The instance embeddings are then aggregated to a single bag embedding by performing a weighted sum based on the attention values. The bag embedding is then classified to give an overall bag prediction. The mil-attn block is the same as per the MIL attention mechanism proposed by Ilse et al. (2018), i.e., using a single hidden layer. We tuned the the number and size of the FC layers, as well as the size of the hidden attention layer.

MI-GNN The GNN model treats the bag as a fully connected graph and uses graph convolutions to propagate information between instances. Initially, the original instances are embedded to a fixed size. These instance embeddings are then passed onto a $\text{GNN}_{\text{embed}}$ block and a $\text{GNN}_{\text{cluster}}$ block, the output of which is used to reduce the graph representation down into a single embedding using differentiable pooling (gnn-pool). The final bag representation is then classified to give an overall bag prediction. For more details see Tu et al. (2019). We tuned the number and size of the embedding, GNN, and classifier layers.

A.3 EVALUATION METRICS

In this section, we provide further details on how we use normalised discounted cumulative gain at n (NDCG@n) to evaluate the interpretability methods for datasets with instance labels, and how we use area under the perturbation curve with random orderings (AOPC-R) to evaluate the interpretability methods for datasets without instance labels.

NDCG@n To use NDCG@n, we first need a ground truth ordering to compare to. For a particular class, we can place each instance into one of three groups based on their ground truth instance labels: supporting instances, neutral instances, and refuting instances. The ideal instance ordering for that class would then have all of the supporting instances at the beginning, followed by all the neutral instances, and then all of the refuting instances at the end. Then, given interpretability outputs $\{\phi_1, \ldots, \phi_k\}$ for a particular class, we rank the outputs from highest to lowest, i.e., in order of how much they support that class. This importance ordering is then compared to the ground truth ordering using the follow metric:

$$\text{NDCG@n} = \frac{1}{\text{IDCG}} \sum_{i=1}^{n} \frac{\text{rel}(i)}{log_2(i+1)},$$

$$\text{where IDCG} = \sum_{i=1}^{n} \frac{1}{log_2(i+1)}.$$

IDCG is the ideal discounted cumulative gain that normalises the scores across different values of n. The relevance function rel(i) is as follows:

$$\text{rel}(i) = \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ instance in the ranking supports the class,} \\ -1 & \text{if the } i^{\text{th}} \text{ instance in the ranking refutes the class,} \\ 0 & \text{otherwise.} \end{cases}$$

AOPC-R Although originally designed for single instance supervised learning, here we adapt AOPC-R for MIL. Given an importance ordering, AOPC successively removes the most relevant instances (i.e., those at the start of the given importance ordering), and measures the change in prediction. A better ordering will show a more rapid decrease in prediction, as instances that are more supportive will be removed first. This rate of decrease in prediction with respect to class c for a classifier F and bag $X = \{x_1, \ldots, x_k\}$ is measured as follows: given the ordered bag $O_X = \{o_1, \ldots, o_k\}$ (which is just X ordered by instance importance, with the most important instances at the start),

$$\text{AOPC} = \frac{1}{k-1} \sum_{i=1}^{k-1} F_c(X) - F_c(X_{MoRF}^{(i)}), \tag{6}$$

$$\text{where } X_{MoRF}^{(0)} = X, \tag{7}$$

$$\text{and } X_{MoRF}^{(i)} = X_{MoRF}^{(i-1)} \setminus \{o_i\}. \tag{8}$$

To normalise the results, one approach is to measure the average difference in AOPC for the given ordering to the AOPC of several random orderings:

$$\text{AOPC-R} = \frac{1}{r} \sum_{i=1}^{r} \text{AOPC}(O_X) - \text{AOPC}(O_{R_i}), \tag{9}$$

where r is the number of random orderings, and O_{R_i} is the i^{th} random ordering of X. The repeated calls to F_c make AOPC-R very expensive to compute. This can be reduced by examining the p first perturbations rather than all $k-1$ possible perturbations, but that was not something we investigated in this work (we kept $p = k$ and $r = 10$). Furthermore, due to the use of random orderings, this evaluation metric is inherently stochastic, meaning not only do we have variance in the orderings produced in the methods (e.g., from random sampling), we also have variance due to measurement. NDCG@n does not have either of the issues (i.e., its cheap to compute and deterministic), however it requires (at least some) instance labels.

A.4 ADDITIONAL RESULTS

In this section, we detail our additional results on the Musk and Tiger, Elephant & Fox (TEF) classical MIL datasets. Although these datasets do not have instance labels, since they are widely used MIL datasets, it is useful to see how our interpretability methods perform on them. Each of these datasets are instance independent, and as they have a low number of instances per bag, we adapted our local surrogate methods to allow sampling with repeats (otherwise we cannot sample enough coalitions). In our previous experiments, we restricted the local surrogate methods to sampling without repeats, which was not an issue with the larger bag sizes in the SIVAL, 4-MNIST-Bags, and CRC datasets. For each of these classic datasets, we used the same model hyperparameters that we used for SIVAL (see Appendix A.6), i.e., we didn't retune the training parameters or model architectures. However, we did tune the parameters for the interpretability methods, which we discuss in Appendix A.5. We give the interpretability results as well as the model performance for Musk (Table A1), Tiger (Table A2), Elephant (Table A3), and Fox (Table A4) below.

Table A1: Musk interpretability AOPC-R results. Note that we are using the Musk1 dataset, rather than Musk2, as the latter has much larger bag sizes, making the use of AOPC-R infeasible.

Methods	MI-Net	mi-Net	MI-Attn	MI-GNN	Overall
Model Acc	0.871 ± 0.024	0.836 ± 0.027	0.793 ± 0.029	0.793 ± 0.028	0.823 ± 0.016
Inherent	N/A	**0.108** ± **0.010**	0.002 ± 0.015	0.003 ± 0.006	0.036 ± 0.011
Single	0.123 ± 0.010	**0.108** ± **0.010**	0.113 ± 0.010	**0.090** ± **0.010**	0.108 ± 0.010
One Removed	0.108 ± 0.009	0.107 ± 0.010	0.088 ± 0.008	0.076 ± 0.009	0.095 ± 0.009
Combined	**0.124** ± **0.010**	0.107 ± 0.010	**0.116** ± **0.010**	0.088 ± 0.010	**0.109** ± **0.010**
RandomSHAP	0.115 ± 0.009	0.104 ± 0.010	0.110 ± 0.009	0.077 ± 0.009	0.102 ± 0.009
GuidedSHAP	0.122 ± 0.009	0.106 ± 0.010	**0.116** ± **0.010**	0.084 ± 0.009	0.107 ± 0.009
RandomLIME	0.113 ± 0.009	0.107 ± 0.010	0.110 ± 0.009	0.080 ± 0.009	0.102 ± 0.009
GuidedLIME	0.117 ± 0.009	0.106 ± 0.010	0.102 ± 0.008	0.077 ± 0.009	0.101 ± 0.009
MILLI	0.117 ± 0.009	0.105 ± 0.010	0.109 ± 0.009	0.084 ± 0.010	0.104 ± 0.009

Table A2: Tiger interpretability AOPC-R results.

Methods	MI-Net	mi-Net	MI-Attn	MI-GNN	Overall
Model Acc	0.827 ± 0.024	0.807 ± 0.029	0.807 ± 0.019	0.800 ± 0.028	0.810 ± 0.005
Inherent	N/A	0.124 ± 0.005	0.001 ± 0.007	0.000 ± 0.003	0.042 ± 0.005
Single	**0.132** ± **0.005**	0.125 ± 0.005	0.120 ± 0.005	**0.082** ± **0.003**	0.115 ± 0.004
One Removed	0.123 ± 0.005	**0.127** ± **0.005**	0.114 ± 0.004	0.081 ± 0.003	0.111 ± 0.004
Combined	0.130 ± 0.005	**0.127** ± **0.005**	**0.124** ± **0.005**	**0.082** ± **0.003**	**0.116** ± **0.004**
RandomSHAP	0.124 ± 0.004	0.122 ± 0.005	0.121 ± 0.005	0.080 ± 0.003	0.112 ± 0.004
GuidedSHAP	0.130 ± 0.005	0.125 ± 0.005	0.121 ± 0.005	**0.082** ± **0.003**	0.115 ± 0.004
RandomLIME	0.128 ± 0.005	0.125 ± 0.005	0.119 ± 0.005	0.081 ± 0.003	0.113 ± 0.004
GuidedLIME	0.129 ± 0.005	0.125 ± 0.005	0.122 ± 0.005	**0.082** ± **0.003**	0.115 ± 0.004
MILLI	0.128 ± 0.005	0.125 ± 0.005	0.120 ± 0.004	0.081 ± 0.003	0.114 ± 0.004

We find that there is very little difference in interpretability performance for each of our proposed methods across these four datasets. This is to be expected, as the instances are independent, therefore the independent-instance methods as well as the local surrogate methods are able to identify the important instances, i.e., there is little to be gained by sampling coalitions when each instance can be understood in isolation. However, this reinforces the applicability of all of our proposed methods. We note that the attention and GNN inherent interpretability methods perform poorly in these experiments — this is because they are unable to condition their outputs on a specific class (i.e., they can only answer *which* questions, not *what* questions). We also note that the performance is much worse on the Fox dataset. Here, the underlying MIL models perform poorly, so it is unsurprising that the interpretability methods also perform poorly.

Table A3: Elephant interpretability AOPC-R results.

Methods	MI-Net	mi-Net	MI-Attn	MI-GNN	Overall
Model Acc	0.857 ± 0.017	0.863 ± 0.017	0.867 ± 0.016	0.853 ± 0.020	0.860 ± 0.003
Inherent	N/A	0.130 ± 0.006	0.000 ± 0.006	0.000 ± 0.003	0.043 ± 0.005
Single	**0.127 ± 0.004**	**0.131 ± 0.006**	**0.127 ± 0.005**	0.105 ± 0.004	**0.122 ± 0.005**
One Removed	0.117 ± 0.004	0.129 ± 0.006	0.105 ± 0.005	0.103 ± 0.004	0.114 ± 0.005
Combined	0.126 ± 0.004	0.130 ± 0.006	**0.127 ± 0.005**	**0.106 ± 0.004**	**0.122 ± 0.005**
RandomSHAP	0.124 ± 0.004	0.126 ± 0.006	0.119 ± 0.005	0.102 ± 0.004	0.118 ± 0.005
GuidedSHAP	**0.127 ± 0.004**	0.130 ± 0.006	0.124 ± 0.005	0.105 ± 0.004	**0.122 ± 0.005**
RandomLIME	0.124 ± 0.004	0.128 ± 0.006	0.121 ± 0.005	0.104 ± 0.004	0.119 ± 0.005
GuidedLIME	0.123 ± 0.004	0.130 ± 0.006	0.118 ± 0.005	0.104 ± 0.004	0.119 ± 0.005
MILLI	0.124 ± 0.005	0.128 ± 0.006	0.121 ± 0.005	0.103 ± 0.005	0.119 ± 0.005

Table A4: Fox interpretability AOPC-R results.

Methods	MI-Net	mi-Net	MI-Attn	MI-GNN	Overall
Model Acc	0.600 ± 0.011	0.610 ± 0.017	0.620 ± 0.023	0.580 ± 0.013	0.603 ± 0.007
Inherent	N/A	**0.050 ± 0.002**	0.001 ± 0.002	0.000 ± 0.001	0.017 ± 0.002
Single	0.044 ± 0.002	0.049 ± 0.002	0.041 ± 0.002	0.021 ± 0.001	0.039 ± 0.002
One Removed	0.043 ± 0.002	0.049 ± 0.002	0.040 ± 0.002	0.021 ± 0.001	0.038 ± 0.002
Combined	0.044 ± 0.002	**0.050 ± 0.002**	0.041 ± 0.002	0.021 ± 0.001	0.039 ± 0.002
RandomSHAP	0.044 ± 0.002	0.049 ± 0.002	0.041 ± 0.002	0.021 ± 0.001	0.039 ± 0.002
GuidedSHAP	**0.045 ± 0.002**	**0.050 ± 0.002**	**0.042 ± 0.002**	0.021 ± 0.001	**0.040 ± 0.002**
RandomLIME	0.044 ± 0.002	**0.050 ± 0.002**	0.041 ± 0.002	0.021 ± 0.001	0.039 ± 0.002
GuidedLIME	0.044 ± 0.002	**0.050 ± 0.002**	0.041 ± 0.002	0.021 ± 0.001	0.039 ± 0.002
MILLI	0.043 ± 0.002	0.049 ± 0.002	0.041 ± 0.002	**0.022 ± 0.001**	0.039 ± 0.002

A.5 INTERPRETABILITY METHOD HYPERPARAMETER SELECTION

In this section we discuss our method for hyperparameter selection in the interpretability methods, and detail the hyperparameters that we found to be most effective. An advantage of the inherent interpretability and independent-instance methods is that they do not have hyperparameters, i.e., we only had to select hyperparameters for the local surrogate interpretability methods. First, we discuss our choice of hyperparameters for LIME, and then our choice of hyperparameters for MILLI.

LIME hyperparameters When using the LIME weight kernel, there are two hyperparameters to tune. Firstly, the distance measures that are commonly used are L2 and cosine distance. In our experiments, we found very little difference between each of these distance measures, therefore we arbitrarily chose to use L2 distance in all of our experiments. The second hyperparameter is the kernel width, which determines the weighting of coalitions, i.e., a large kernel width means all coalitions are weighted more evenly, and a small kernel width prioritises larger coalitions. We chose to use a kernel width for all of our experiments that is determined by the average bag size, such that the half coalition (i.e., $|z| = 0.5k$) is weighted at 0.5.

MILLI hyperparameters For MILLI, there were three hyperparameters to tune: the number of samples, α, and β. For the number of samples, we generated sample size plots such as Figure 4 (also see Appendix A.9), and chose the number of samples to be at the point where all the methods had reasonably converged. For α and β we ran a grid search over the possible values, and chose the best performing pair of parameters. The hyperparameters were tuned for each dataset, except for the TEF datasets, in which we only tuned on Tiger and then used the same hyperparameters across all three datasets. In Table A5, we provide the chosen hyperparameters for each dataset, as well as the expected coalition size $\mathbb{E}[|z|]$ determined by α and β. For the CRC, Musk and TEF datasets, the chosen parameters produce small values for $\mathbb{E}[|z|]$, i.e., the sampling is heavily biased towards smaller coalitions, which is expected as these are instance independent datasets. Conversely, the parameters for the 4-MNIST-Bags focus on sampling larger coalitions that are able to capture the instance interactions in the dataset. Although the SIVAL dataset is instance independent, the parameters also focus on sampling larger coalitions. This could be because, although the instances are independent, it may be difficult to classify an object from just one instance, i.e., several instances are needed to make the correct decision. We also note that, in all cases, $\alpha < 0.5$, i.e., the sampling is biased towards instances ranked lower in the initial importance ordering used by MILLI. Our explanation for this is that it is easy to understand the contribution of discriminatory instances, as they have a large effect on the model prediction, but it is more difficult to understand the contribution of non-discriminatory instances, as they have much less of an effect. Therefore, more samples containing non-discriminatory instances are required to properly understand their effect, hence biasing the sampling towards them.

Table A5: MILLI hyperparameters.

Dataset	Sample Size	α	β	$\mathbb{E}[\|z\|]$
SIVAL	200	0.05	-0.01	13
4-MNIST-Bags	150	0.05	0.01	16
CRC	1000	0.008	-5.0	2
MUSK	150	0.3	-1.0	2
TEF	150	0.3	0.01	3

A.6 SIVAL EXPERIMENT DETAILS

Dataset For the SIVAL dataset, each instance is represented by a 30-dimensional feature vector, and there are around 30 instances per bag. We chose 12 of the 25 original classes to be the positive classes, and randomly selected 30 images from each of the other 13 classes to form the negative class, meaning overall we had 13 classes (12 positive and one negative). In total, we had 60 bags for each of the 12 positive classes, and 390 bags for the single negative class, meaning the class distribution was $\approx 5.4\%$ for each positive class and $\approx 35.1\%$ for the negative class. The arbitrarily chosen positive classes were: *apple*, *banana*, *checkeredscarf*, *cokecan*, *dataminingbook*, *goldmedal*, *largespoon*, *rapbook*, *smileyfacedoll*, *spritecan*, *translucentbowl*, and *wd40can*. We normalised each instance according to the dataset mean and standard deviation. No other data augmentation was used. The dataset was separated into train, validation, and test data using an 80/10/10 split. This was done with stratified sampling in order to maintain the same data distribution across all splits.

Training When training models against the SIVAL dataset, we used a batch size of one, i.e., a single bag of, on average, 30 instances. We trained the models to minimise cross entropy loss using the Adam optimiser; the hyperparamater details for learning rate (LR), weight decay (WD) and dropout (DO) are given in Table A6. We utilised early stopping based on validation loss — if the validation loss had not decreased for 10 epochs then we terminated the training procedure and reset the model to the point at which it caused the last decrease in validation loss. Otherwise, the maximum number of epochs was 100. The tuned architectures for each model are given in Tables A7 to A10, and the results for each model are comapred in Table A11.

Table A6: SIVAL hyperparameters.

Model	LR	WD	DO
Mi-Net	5×10^{-3}	1×10^{-3}	0.45
mi-Net	5×10^{-4}	1×10^{-5}	0.25
MI-Attn	1×10^{-3}	1×10^{-5}	0.15
MI-GNN	5×10^{-4}	1×10^{-5}	0.2

Table A7: SIVAL MI-Net architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	30	128
2	FC + ReLU + DO	128	256
3	mil-max	256	256
4	FC	256	13

Table A8: SIVAL mi-Net architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	30	512
2	FC + ReLU + DO	512	256
3	FC + ReLU + DO	256	64
4	FC	64	13
5	mil-mean	13	13

Table A9: SIVAL MI-Attn architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	30	128
2	FC + ReLU + DO	128	256
3	FC + ReLU + DO	256	128
4	mil-attn(256) + DO	128	128
5	FC	128	13

Table A10: SIVAL MI-GNN architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	30	128
2a $\text{GNN}_{\text{embed}}$	SAGEConv + ReLU + DO	128	128
	SAGEConv + ReLU + DO	128	256
	SAGEConv + ReLU + DO	256	64
2b $\text{GNN}_{\text{cluster}}$	SAGEConv + Softmax	128	1
3	gnn-pool	64	64
4	FC + ReLU + DO	64	128
5	FC	128	13

Table A11: SIVAL model results. The mean performance was calculated over ten repeat trainings of each model, and the standard error of the mean is given.

Model	Train Accuracy	Val Accuracy	Test Accuracy
MI-Net	**0.984 ± 0.004**	**0.850 ± 0.009**	**0.819 ± 0.012**
mi-Net	0.967 ± 0.005	0.835 ± 0.010	0.808 ± 0.011
MI-Attn	0.972 ± 0.007	0.830 ± 0.011	0.813 ± 0.012
MI-GNN	0.932 ± 0.014	0.803 ± 0.014	0.781 ± 0.019

A.7 4-MNIST-BAGS EXPERIMENT DETAILS

Dataset In the 4-MNIST-Bags experiments, the bag sizes were draw from a normal distribution, with a mean of 30 and a variance of 2. We used 2500 training bags, 1000 validation bags, and 1000 test bags. The instances in the training bags were only drawn from the original MNIST training split, and the instances in the validation and test bags were only drawn from the original MNIST test split, i.e., there was no overlap between training, validation, and test instances. The classes were balanced, so, on average, there were 625 bags per class in the training data, and 250 bags per class in the validation and test data. We normalised the MNIST images using the PyTorch normalise transformation, with a mean of 0.1307 and a stand deviation of 0.3081. No other data augmentation was carried out.

Training The training procedure was the same as for the SIVAL experiments: a batch size of one, early stopping with a patience of ten, and a maximum of 100 epochs. The training hyperparamater details are given in Table A12. For each model, we first passed the MNIST instances through a convolutional architecture to produce initial instance embeddings. This encoder was not tuned for each model (i.e., the architecture was fixed, but the weights were learnt). The architecture for this encoder is given in Table A13, and then the model architectures are given in Tables A14 to A17. The encoder produces features vectors with 800 features, therefore the input size to each of the models is 800. The model results are given in Table A18.

Table A12: 4-MNIST-Bags hyperparameters.

Model	LR	WD	DO
MI-Net	1×10^{-4}	1×10^{-3}	0.3
mi-Net	1×10^{-4}	1×10^{-4}	0.3
MI-Attn	1×10^{-4}	1×10^{-4}	0.15
MI-GNN	5×10^{-5}	1×10^{-5}	0.3

Table A13: 4-MNIST-Bags convolutional encoding architecture. For the convolutional (Conv2d) and pooling (MaxPool2d) layers, the numbers in the brackets are the kernel size, stride, and padding.

Layer	Type	Input	Out
1	Conv2d(5, 1, 0) + ReLU	1	20
2	MaxPool2d(2, 2, 0) + DO	20	20
3	Conv2d(5, 1, 0) + ReLU	20	50
4	MaxPool2d(2, 2, 0) + DO	50	50

Table A14: 4-MNIST-Bags MI-Net architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	800	128
2	FC + ReLU + DO	128	512
3	mil-mean	512	512
4	FC	512	4

Table A15: 4-MNIST-Bags mi-Net architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	800	512
2	FC + ReLU + DO	512	128
3	FC + ReLU + DO	128	64
4	FC	128	4
5	mil-mean	4	4

Table A16: 4-MNIST-Bags MI-Attn architecture.

Layer	Type	Input size	Output size
1	FC + ReLU + Dropout	800	64
2	FC + ReLU + Dropout	64	256
3	mil-attn(64) + Dropout	256	256
4	FC + ReLU + Dropout	256	64
5	FC	64	4

Table A17: 4-MNIST-Bags MI-GNN architecture.

Layer	Type	Input size	Output size
1	FC + ReLU + Dropout	800	64
2	FC + ReLU + Dropout	64	64
3a $\text{GNN}_{\text{embed}}$	SAGEConv + ReLU + Dropout	64	128
	SAGEConv + ReLU + Dropout	128	128
	SAGEConv + ReLU + Dropout	128	128
3b $\text{GNN}_{\text{cluster}}$	SAGEConv + Softmax	64	1
4	gnn-pool	128	128
5	FC + ReLU + Dropout	128	64
6	FC	64	4

Table A18: 4-MNIST-Bags model results. The mean performance was calculated over ten repeat trainings of each model, and the standard error of the mean is given.

Model	Train Accuracy	Val Accuracy	Test Accuracy
MI-Net	**0.995 ± 0.001**	0.972 ± 0.002	0.971 ± 0.002
mi-Net	0.993 ± 0.001	**0.973 ± 0.002**	**0.974 ± 0.002**
MI-Attn	0.991 ± 0.002	0.967 ± 0.003	0.967 ± 0.003
MI-GNN	0.984 ± 0.002	0.968 ± 0.002	0.966 ± 0.002

A.8 CRC EXPERIMENT DETAILS

Dataset In the CRC dataset, there are 100 microscopy images, and each image is annotated with four classes of nuclei: epithelial, inflammatory, fibroblast, and miscellaneous. Of these 100 images, 50 are in the negative class (no epithelial nuclei), and 50 are in the positive class (at least one epithelial nuclei). Each image is 500 x 500 pixels, and was split into 27 x 27 pixel patches to give 324 patches per slide. The patches were created by applying a non-overlapping grid over the image, where each cell of the grid was a 27 x 27 pixel region. The alternative would be to extract patches centred on each marked nuclei (as done by Ilse et al. (2018)), however this method then requires the nuclei to be marked for unseen data. This accounts for the difference between our results and the results of Ilse et al. (2018) — extracting the patches using a fixed grid will include patches that do not contain any marked nuclei, increasing the amount of non-discriminatory data in each bag and thus making the problem harder to learn. We removed slide background patches by using a brightness threshold — any patch with an average pixel value above 230 (using pixel values 0 to 255) was discarded. This left an average of 264 patches per image, and one image was discarded as it had zero foreground patches (this was also manually verified). The images were normalised using the dataset mean (0.8035, 0.6499, 0.8348) and standard deviation (0.0858, 0.1079, 0.0731). During training, we also applied three transformations to patches at random: horizontal flips, vertical flips, and 90 degree rotations. The dataset was separated into train, validation, and test data using a 60/20/20 split. This was done with stratified sampling in order to maintain the same data distribution across all splits.

Training The training procedure was the same as for the SIVAL and 4-MNIST-Bags experiments: a batch size of one, early stopping with a patience of ten, and a maximum of 100 epochs. The training hyperparamater details are given in Table A19. Following the same procedure as the 4-MNIST-Bags experiments, for each model, we first passed the patches through a convolutional architecture to produce initial instance embeddings. The architecture for this encoder is given in Table A20, and then the model architectures are given in Tables A21 to A24. The encoder produces features vectors with 1200 features, therefore the input size to each of the models is 1200. The model results are given in Table A25.

Table A19: CRC training hyperparameters.

Model	LR	WD	DO
MI-Net	5×10^{-4}	1×10^{-3}	0.3
mi-Net	5×10^{-4}	1×10^{-2}	0.25
MI-Attn	1×10^{-3}	1×10^{-6}	0.2
MI-GNN	1×10^{-3}	1×10^{-2}	0.35

Table A20: CRC convolutional encoding architecture. For the convolutional (Conv2d) and pooling (MaxPool2d) layers, the numbers in the brackets are the kernel size, stride, and padding.

Layer	Type	Input	Out
1	Conv2d(4, 1, 0) + ReLU	3	36
2	MaxPool2d(2, 2, 0) + DO	36	36
3	Conv2d(3, 1, 0) + ReLU	36	48
4	MaxPool2d(2, 2, 0) + DO	48	48

Table A21: CRC MI-Net architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	1200	64
2	FC + ReLU + DO	64	512
3	mil-max	512	512
4	FC + ReLU + DO	512	128
4	FC + ReLU + DO	128	64
4	FC + ReLU + DO	64	2

Table A22: CRC mi-Net architecture.

Layer	Type	Input	Output
1	FC + ReLU + DO	1200	64
2	FC + ReLU + DO	64	64
3	FC + ReLU + DO	64	64
4	FC	64	2
5	mil-max	2	2

Table A23: CRC MI-Attn architecture.

Layer	Type	Input size	Output size
1	FC + ReLU + Dropout	1200	64
2	FC + ReLU + Dropout	64	64
3	FC + ReLU + Dropout	64	256
4	mil-attn(128) + Dropout	256	256
5	FC	256	2

Table A24: CRC MI-GNN architecture.

Layer	Type	Input size	Output size
1	FC + ReLU + Dropout	1200	64
2	FC + ReLU + Dropout	64	128
3a $\text{GNN}_{\text{embed}}$	SAGEConv + ReLU + Dropout	128	128
3b $\text{GNN}_{\text{cluster}}$	SAGEConv + Softmax	128	1
4	gnn-pool	128	128
5	FC + ReLU + Dropout	128	128
6	FC	128	2

Table A25: CRC model results. The mean performance was calculated over ten repeat trainings of each model, and the standard error of the mean is given.

Model	Train Accuracy	Val Accuracy	Test Accuracy
MI-Net	**0.880 ± 0.041**	0.805 ± 0.034	0.795 ± 0.042
mi-Net	0.856 ± 0.041	0.810 ± 0.043	0.795 ± 0.049
MI-Attn	0.870 ± 0.014	**0.860 ± 0.017**	**0.830 ± 0.028**
MI-GNN	0.791 ± 0.039	0.765 ± 0.031	0.770 ± 0.032

A.9 ADDITIONAL EXPERIMENTS

In this section, we provide additional sample size experiments for the SIVAL, 4-MNIST-Bags, and CRC datasets for MI-Net (Figure A1), mi-Net (Figure A2), and MI-GNN (Figure A3). We find the trends are relatively consistent across all the models. GuidedSHAP and MILLI are the most consistent and well performing methods, and MILLI is particularly effective on the 4-MNIST-Bags dataset. One considerable difference is the performance of RandomLIME and GuidedLIME on the CRC dataset for the MI-GNN model; for all other models, both LIME methods perform poorly on the CRC dataset. Further investigation is required to understand why this happens. However, the LIME methods are still outperformed by MILLI and GuidedSHAP on the CRC dataset, even for the MI-GNN model.

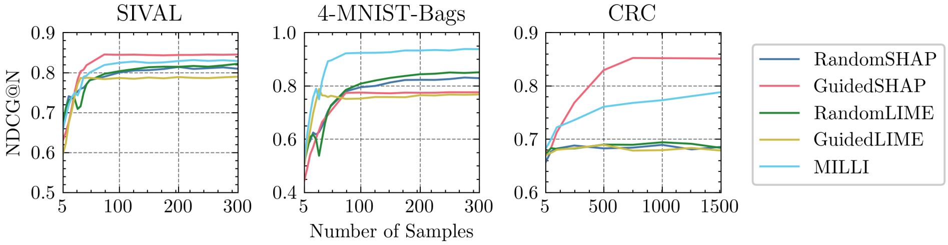

Figure A1: The effect of sample size on interpretability performance for MI-Net.

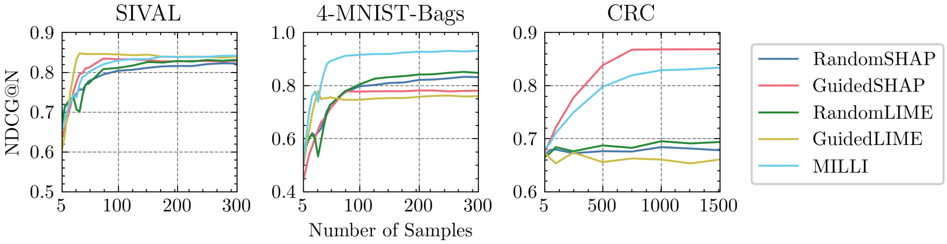

Figure A2: The effect of sample size on interpretability performance for mi-Net.

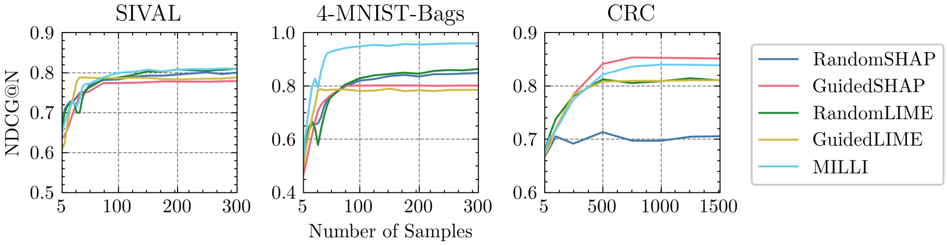

Figure A3: The effect of sample size on interpretability performance for MI-GNN.

A.10 ADDITIONAL OUTPUTS

In this section, we provide additional interpretability outputs from our studies, similar to the one in Section 5. First, we provide additional outputs for the 4-MNIST-Bags dataset. Then, we show the interpretability outputs for the SIVAL and CRC datasets. The SIVAL outputs are created by ranking the instances in a bag according to some interpretability function (i.e., by using MILLI), and then selecting the top n, where n is the known number of key instances in the bag. Then, the corresponding segment for each instance is weighted by the function $log_2(i+1)^{-1}$, where i is the instance's position in the ranking (this is the same scaling used in NDCG@n). The other instances all received a weighting of zero. The brightness of each segment in the output image then corresponds with its relative importance according to the interpretability method. For the CRC interpretability outputs, the important patches are found by using an interpretability method to output the top n patches that support a particular class, where n is the number of known important instances for that class. We then highlight these patches while dimming the other patches to produce an interpretation of the model's decision-making.

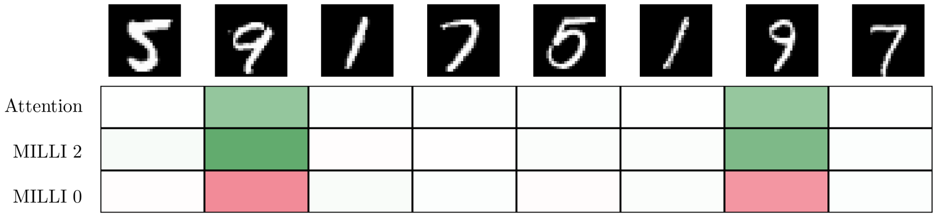

Figure A4: The interpretability output for a class two bag on the 4-MNIST-Bags experiment. The top row shows the attention value for each instance, and bottom two rows show the output of MILLI for classes two and zero respectively (green = support, red = refute). We can see that both Attention and MILLI have identified the **9**s as important, but only MILLI is able to also say that the **9**s support class two and refute class zero.

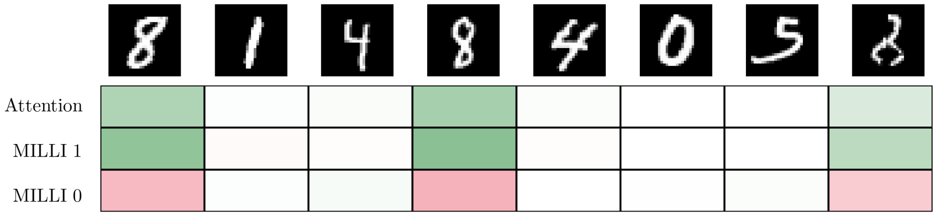

Figure A5: The interpretability output for a class one bag on the 4-MNIST-Bags experiment. We can see that both Attention and MILLI have identified the **8**s as important, but only MILLI is able to also say that the **8**s support class one and refute class zero. Also note that the colours are less strong for the final **8** — the digit is partially obscured, so the model is less confident.

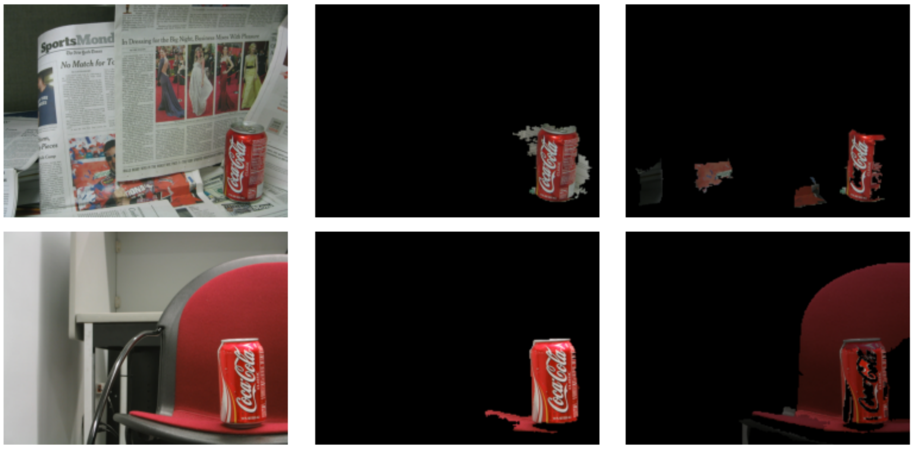

Figure A6: The interpretability output for two examples of the *cokecan* class in the SIVAL dataset. The first column shows the original images, the second column the ground truth segments that contain the object, and the final column is the output from MILLI. The interpretations show the model is relying heavily on the colour red as an indicator of the object's location, as it also picks up on the red in the background as being important.

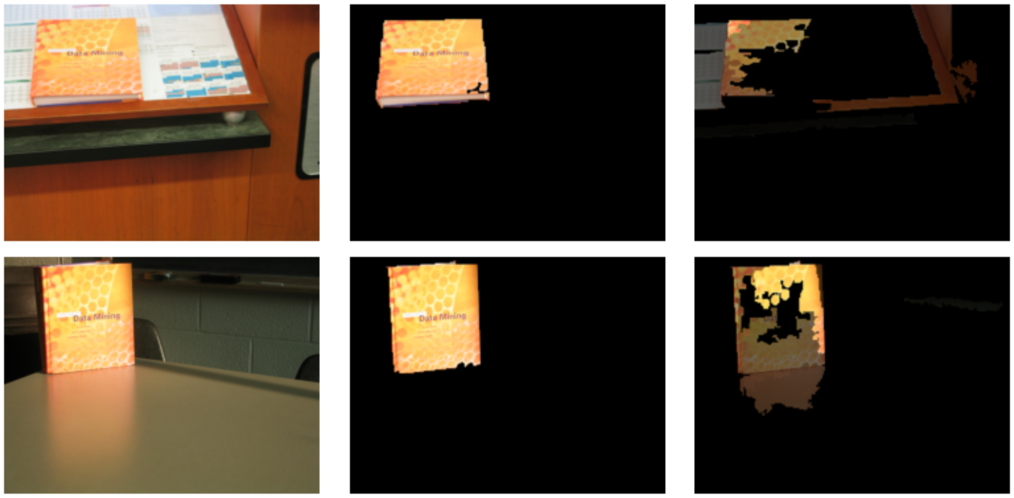

Figure A7: The interpretability output for two examples of the *dataminingbook* class in the SIVAL dataset. Again, the interpretations show the model is relying heavily on the colour orange as an indicator of the object's location, as it also picks up on the similar colours in the background, including the reflection of the book in the table.

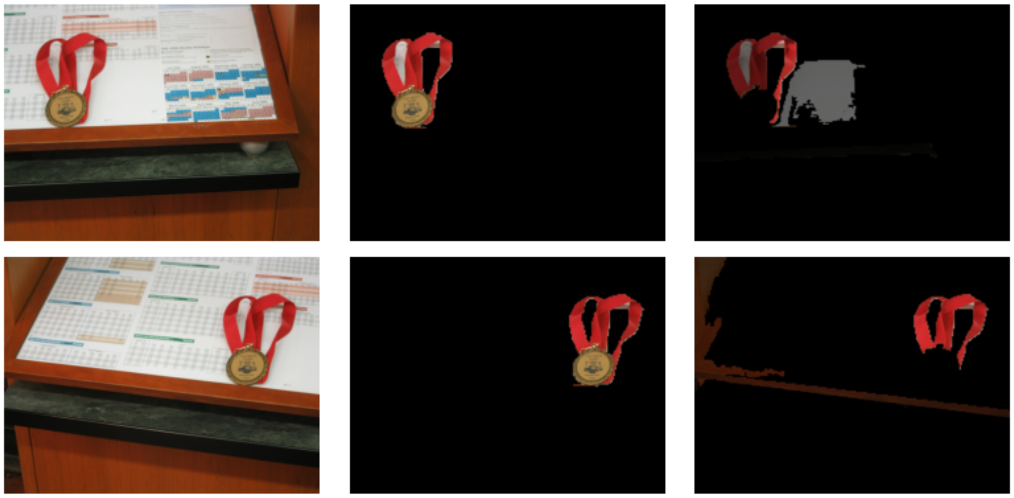

Figure A8: The interpretability output for two examples of the *goldmedal* class in the SIVAL dataset. Here, the interpretations show the model is ignoring the gold medal itself, and instead focusing on the red ribbon, i.e., if it were shown the medal without the ribbon, it would likely misclassify it.

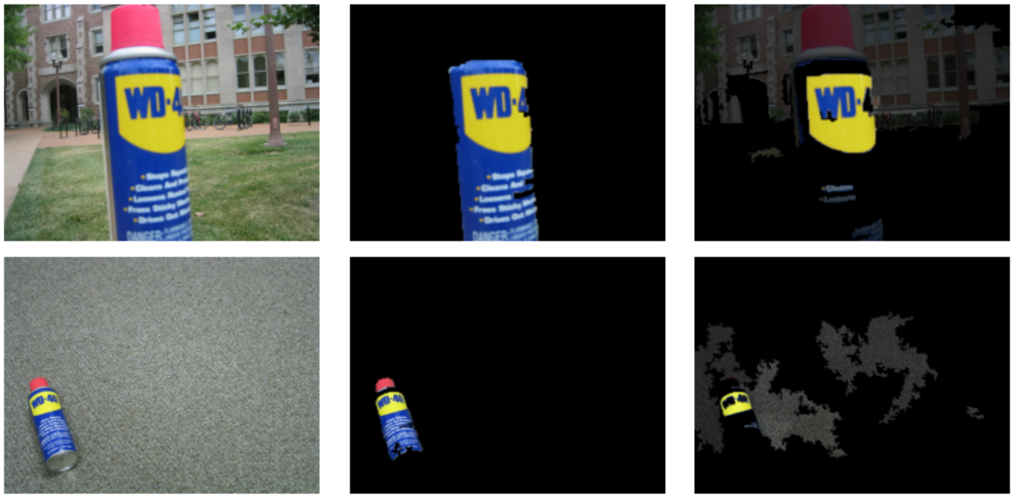

Figure A9: The interpretability output for two examples of the *wd40can* class in the SIVAL dataset. Here, the interpretations show the model is predominantly focusing on the strong yellow colour at the top of the can, and sometimes picks out the letters and red cap. The rest of the bottle is largely ignored, meaning if the top half were obscured, the model would likely misclassify it.

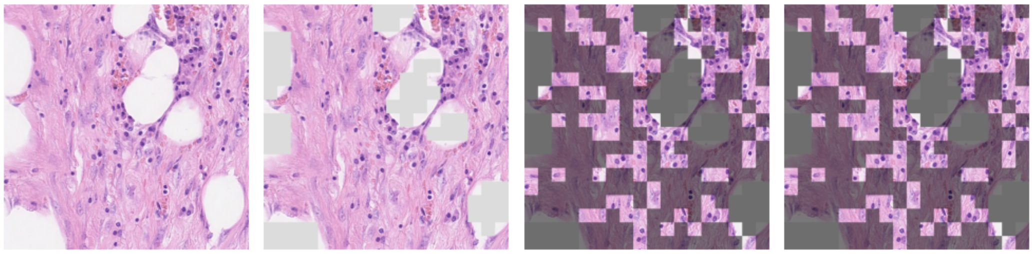

Figure A10: The interpretability output for an image in the CRC dataset. From left to right, the figures shows: the original image, the image with background patches removed, the ground truth patches for the image's label, and the interpretability output from MILLI. In this example, the image is class zero (i.e., non-epithelial), meaning the highlighted patches contain mostly fibroblast and inflammatory nuclei. As shown here, MILLI has identified that the model is using the same patches as the ground truth to make a decision, indicating that the model has learnt to correctly identify fibroblast and inflammatory nuclei as supporting instances for non-epithelial images.

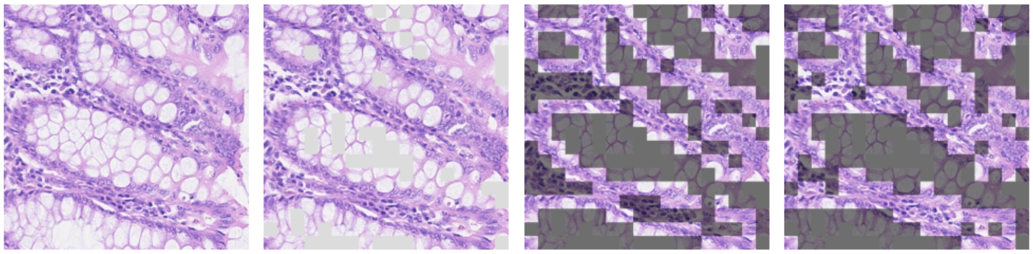

Figure A11: The interpretability output for a positive image in the CRC dataset. In this example, the ground truth patches all contain at least one epithelial nuclei. MILLI has identified that the model is using most of the same patches as the ground truth to make a decision, indicating that the model has learnt to correctly identify epithelial nuclei as supporting class one.

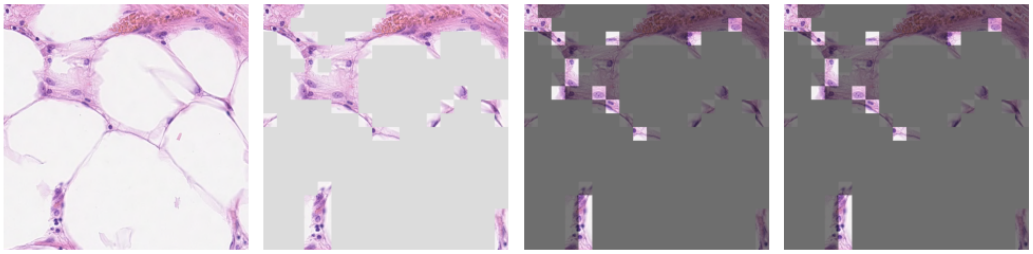

Figure A12: The interpretability output for a negative image in the CRC dataset. In this example, the patches are sparse — there are a lot of background patches that are removed, and the key patches are spread out (as opposed to being connected as in the previous examples). Again, MILLI is able to correctly identify the patches that support the negative class.

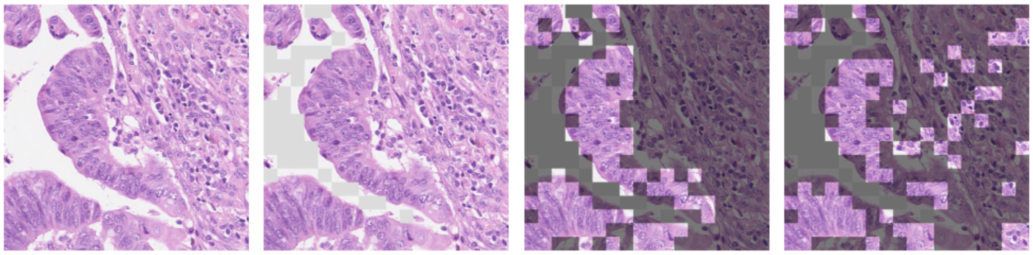

Figure A13: The interpretability output for a positive image in the CRC dataset. In this example, MILLI has identified some of the ground truth patches, but also additional patches that are not labelled as epithelial in the ground truth labelling. As the labelling was not exhaustive, these patches may contain epithelial nuclei without being labelled as such. By using MILLI, it is apparent that the model is using these patches in its decision-making, therefore further investigation into the types of nuclei in these patches would reveal more information about how the model makes its decisions; something that would not be possible without the interpretability output.