Appendix

5.1 Graph Neural Networks

All GNN models have an initial linear layer to encode node and edge information, this layer does not use adjacency information. All models have several block layers stacked to each other, using the node-block, edge-blocks and global-blocks from a Graph Nets architecture [11].

The GCN model is composed of several GCN blocks, where each is a node-pooling and aggregation operation followed by a node-block. This is essentially a edge-conditioned GCN block, only node representations are learned.

The MPNN model is composed of a MPNN block, which is a edge-block followed by a node-block. The model learns node and edge representations.

As an update function for each block and sub-blocks, we use a 2-layer multilayer perceptron of constant size with relu activation functions and LayerNorm [9] at the end.

To do graph-level prediction tasks, all GNNs have a global-block layer at the end of the network with three modifications: 1)) we first perform the update of node, edge and global information 2) we expose the node and edge activations for CAM and 3) we only use the global representation of the resulting graph. For regression tasks we use a linear layer for prediction along with the mean square error loss function. For binary classification tasks we use linear layer with sigmoid activation along with a binary cross entropy loss function.

GNNs were implemented using tensorflow 2.0 [3] with the Sonnet and Graph Nets modules.

5.2 Statistical analyses

All box plots display the first quartile, median, and third quartile. The whiskers extend to points within 1.5 IQRs, and observations which fall outside of this range are displayed independently.

Fig 3: statistics are derived from the mean attribution AUROC of model architectures after selecting for top 10% of trained models through a hyper-parameter scan.

Fig 4: statistics are derived from the PARC scores over 8 repeated runs.

Fig 5: statistics are derived from the PARC scores over 10 random seeds for each model, technique and dataset.

Fig 6: error bars are calculated across 1600 perturbation samples: 10 perturbations for 40 example molecules from the test set from each of 4 classification tasks.

5.3 Tests for Statistical Significance

To report the statistical significance, we perform one-tail t-tests for mean comparisons, and Bartlett's tests for variance comparisons. For each claim, we then report the Holm-Bonferroni corrected p-value for each pairwise comparison.

5.4 Architecture hyperparameter scan

For each combination of graph neural network model, task type, and attribution method, we performed a hyper-parameter scan of 195 architectures, with 7 random seeds per architecture. The parameters varied were:

Number of message layers: 1, 2, 3, 4, 5.
Edge feature size: 8, 20, 32.
Node feature size: 20, 30, 35, 40, 50.
Embedding/Global layer size: 25, 50, 75, 100.

Each architecture was trained to 300 epochs using the Adam optimizer with a learning rate of 3e-4 and batch size of 256 graphs for graph-level tasks and batch size of 1 for node-level tasks. During training we saved the weights with the best train loss and used those for the evaluation. Training an architecture took on average 5 minutes on a 8-core CPU. Evaluating a task on all attribution techniques took on average 3 minutes. For more complex experiments, each Faithfulness experiment took on average one hour per random seed, model and task. Stability experiments took on average 10 minutes per model and task.

Dataset	# graphs	Node statistics			Train/test ratio	Source
		Min	Median	Max		
Benzene	12000	4	22	25	80/20	Constructed from [43]
Logic7 (Fluoride AND Carbonyl)	4326	5	22	25	90/10	[43, 29]
Logic8 (Unbrached Alkane AND Carbonyl)	8671	5	22	25	90/10	[43, 29]
Logic10 (Amine AND Ether AND Benzene)	8687	6	22	25	90/10	[43, 29]
Crippen	1127	4	23	119	80/20	Constructed from [15]
Ester / Benzene	3000	5	22	25	66/33	Constructed from [43]
Piperdine / Benzene	3000	5	22	25	66/33	Constructed from [43]
Unbrnchd Alkane / Benzene	3000	5	22	25	66/33	Constructed from [43]
Morpholine / Benzene	3000	5	22	25	66/33	Constructed from [43]
Ester / Phenyl	3000	5	22	25	66/33	Constructed from [43]

Table S1: Graph-level prediction datasets. Summary statistics for datasets relating to graph-level attributes.

Dataset	# nodes	Edge statistics (degree)			Train/test ratio	Source
		Min	Median	Max		
BA-shapes	700	4	8	178	100/0	Constructed from [51]
BA-community	1400	4	8	130	100/0	Constructed from [51]
Tree-grid	1231	2	6	14	100/0	Constructed from [51]

Table S2: Node-level prediction datasets. Summary statistics for datasets relating to node-level attributes, each dataset is a single graph.

From the exhaustive hyperparameter scan we selected a architecture that was sufficient for near-perfect predictive performance for all tasks and which we use for all subsequent experiments. This architecture is a 3-layer GNN, with 20 dimensions for edges, 50-dimensions for nodes and a global layer size of 100. We use l2 regularization at a value of 1e-5 for all linear layers of the network.

5.5 Dataset specifications

Molecules for the spurious correlation experiment (Section 4.2) were selected for diversity from a drug-like dataset [43]. We employ a maxmin selection strategy on the molecule graph, using an sparse encoding that represents the presence of all possible subgraphs up to length 7, and a jaccard distance function. We used this strategy to select 2000 molecules that obey each logic combination for the training set, and an additional 1000 molecules for the test set. RDKit[2] was used to perform this selection. This selection strategy allows our train and test set to represent a wide variety of subgraphs which minimizes spurious correlations.

5.6 Additional Attribution Accuracy Findings

5.7 Attribution technique hyperparameters

Attribution methods can be sensitive to hyperparameters and to post-processing (e.g. normalization). In our experiments, IG and SmoothGrad and Attention rely on hyperparameters. We tuned hyperparameters for IG and SmoothGrad by computed attribution scores on several configurations for a single task (Figure S6). We found that IG is not reliable when the number of integration steps is too low. When the number of steps is 200 the performance is quite stable. We compared different types of techniques used as input for SmoothGrad, we did not consider IG for SmoothGrad due to computational cost. In general, we found that SmoothGrad(GradInput) sometimes improves performance over GradInput and tends to degrade or have a negligible effect on GradCAM. For when the number of

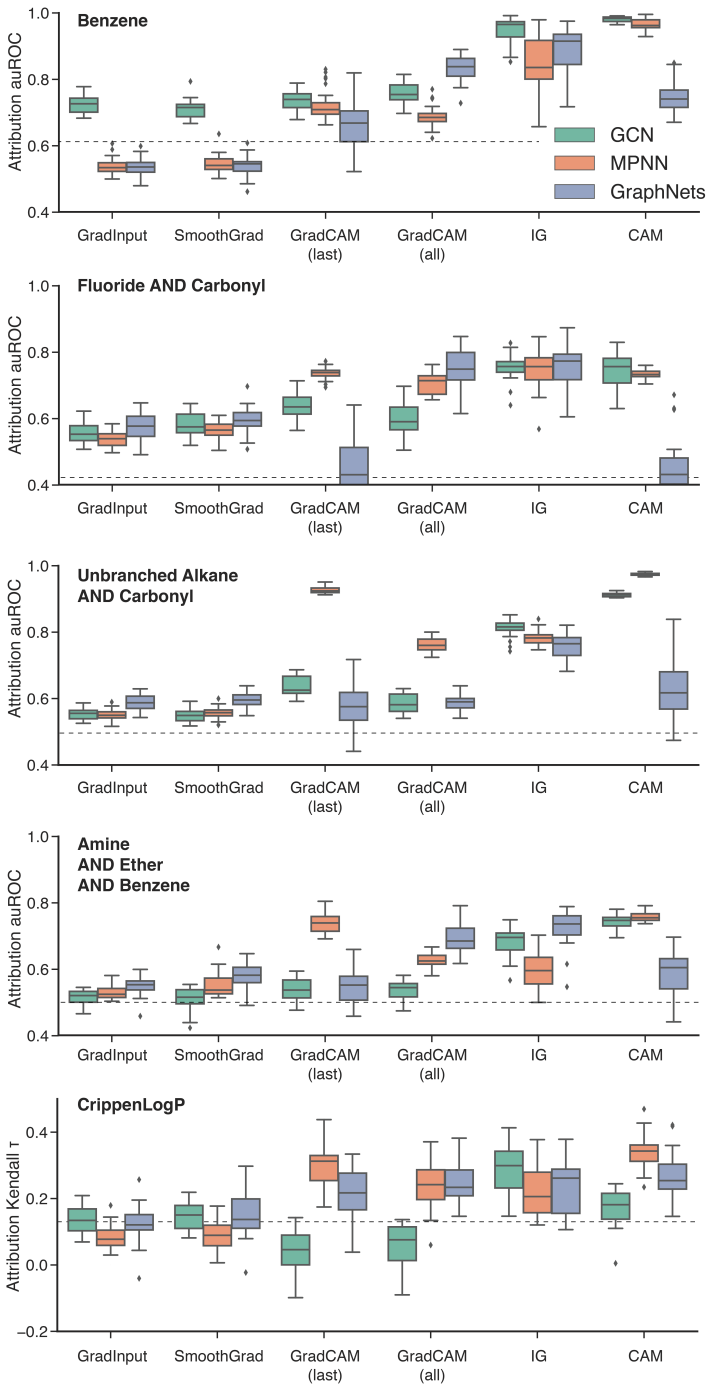

Figure S1: Attribution accuracy for all models, attribution methods, and tasks. This box-plot representation shows the values present in Figure 3. Attribution accuracy is plotted for all models and all attribution methods, across all tasks. Box plots represent the distribution of accuracies as specified in the main text.

samples is equal to 50, 100, and 200, the attribution performance seems comparable. For Attention, we used a mean reduction function for the number of blocks with a single attention head to have similar settings as in [51]. We think scores could be improved by expanding the hyperparameters considered for Attention.

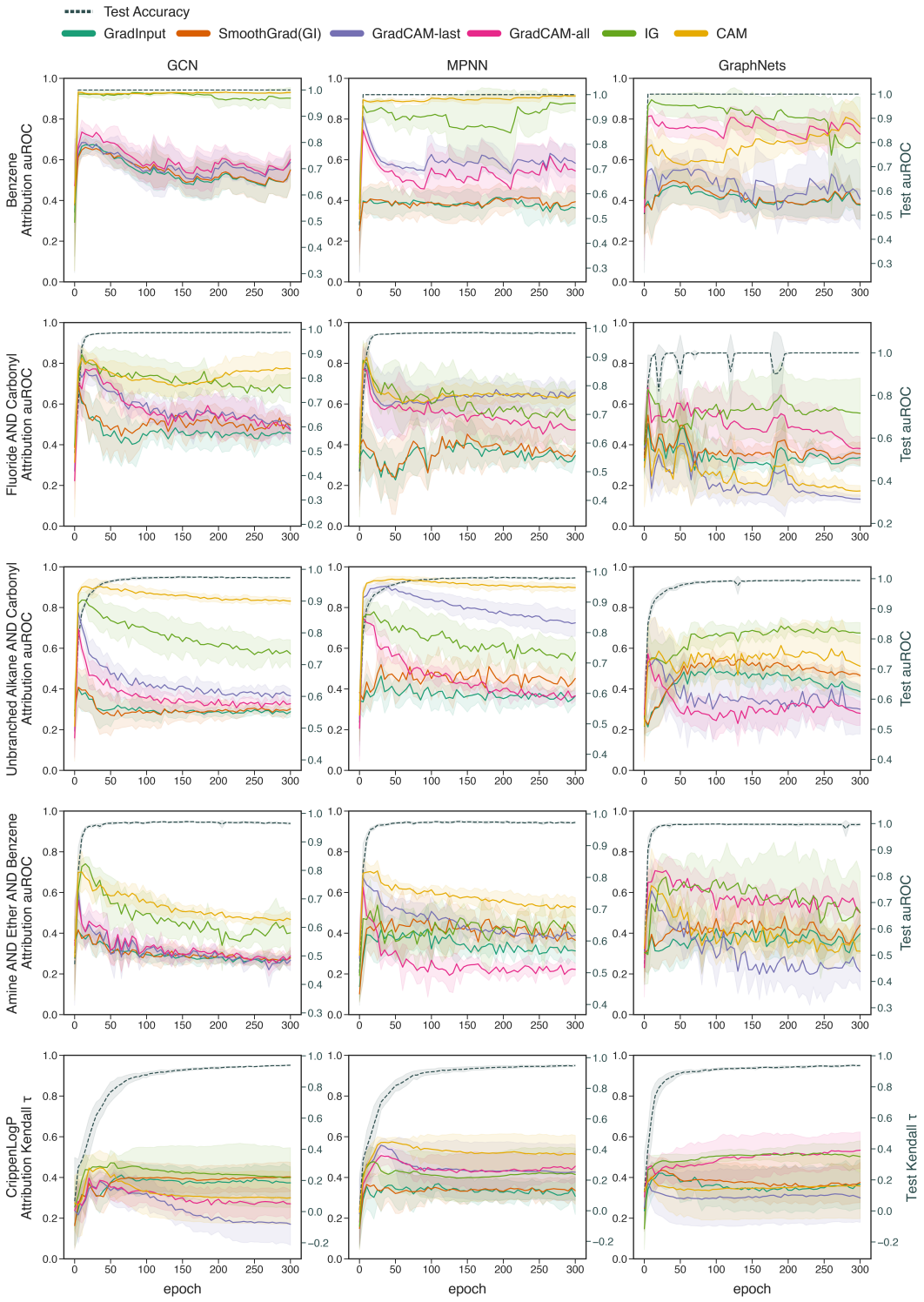

Figure S2: Attribution accuracy as a function of training epoch. Predictive performance on the test set is plotted with a shared x-axis to highlight the differing time courses of generalization performance andv attribution performance.

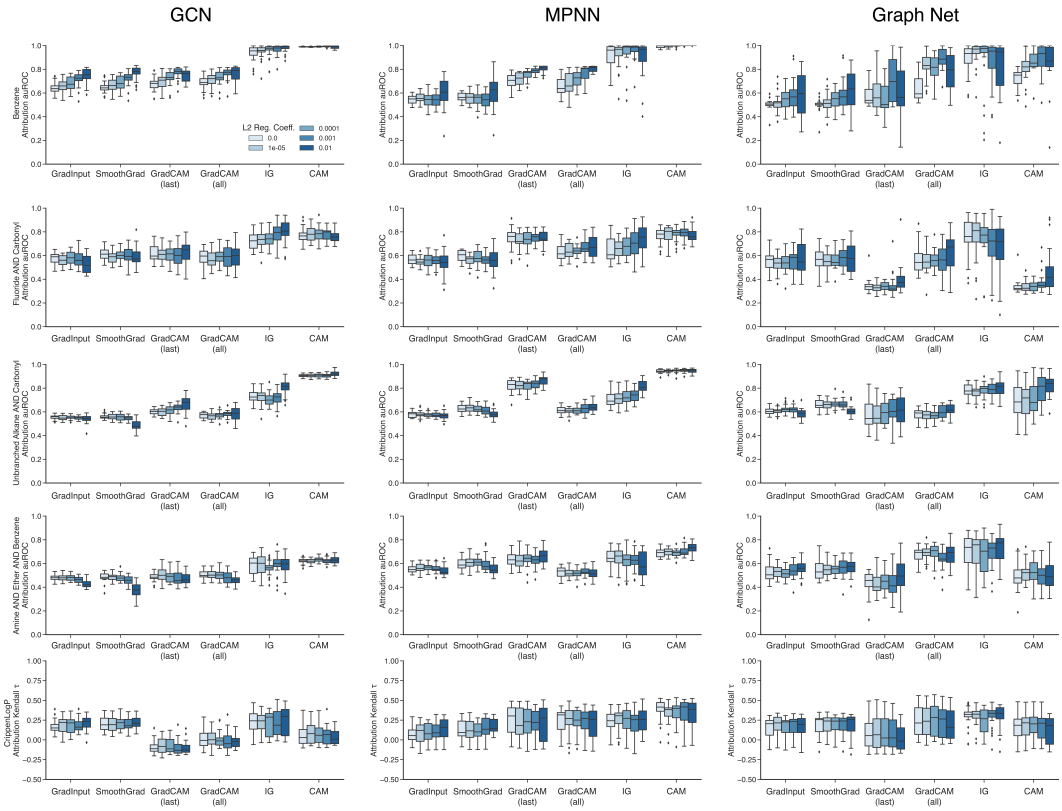

Figure S3: Regularization strength during training affects attribution performance in some, but not all, tasks. Attribution performance in the comparatively simple *Benzene* task is strongly affected by regularization strength of models during training. However, the trend is less clear for more complex Boolean logic tasks, and there is no apparent trend for the regularization *CrippenLogP* task

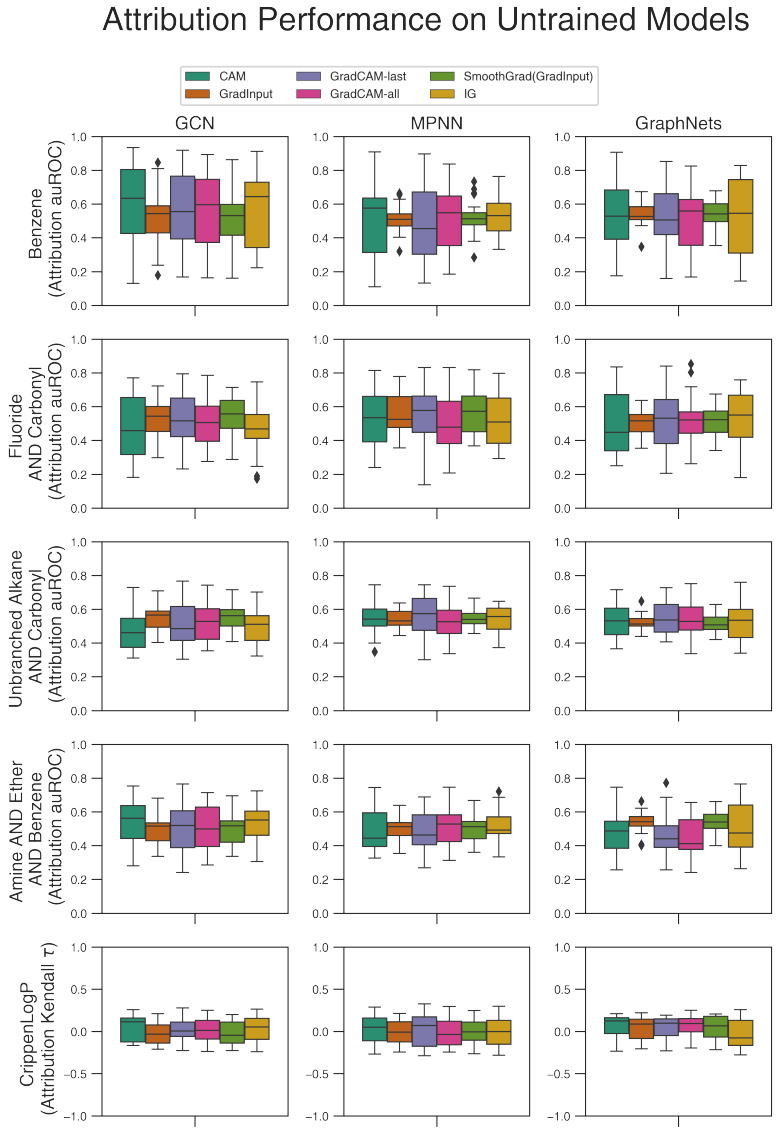

Figure S4: Attribution performance applied to randomly initialized models. When applied to randomly initialized models that have not been fit on data, all attribution methods perform randomly, although with high variance.

	Fluoride AND Carbonyl				Unbranched Alkane AND Carbonyl			
	GCN	MPNN	GraphNets	GAT	GCN	MPNN	GraphNets	GAT
Random Baseline	0.42	0.42	0.42	0.43	0.5	0.5	0.5	0.5
GradInput	0.56	0.55	0.57	0.56	0.56	0.55	0.59	0.52
SmoothGrad(GI)	0.58	0.57	0.61	0.56	0.55	0.56	0.6	0.42
GradCAM-last	0.64	0.74	0.43	0.6	0.64	0.93	0.57	0.55
GradCAM-all	0.6	0.71	0.74	0.57	0.59	0.77	0.58	0.53
IG	0.77	0.76	0.76	0.74	0.82	0.78	0.77	0.68
CAM	0.74	0.73	0.45	0.66	0.91	0.97	0.65	0.84
Attention Weights				0.53				0.53

Figure S5: Attribution performance on supplemental classification tasks. Performance of models and attribution techniques on the Fluorine AND Carbonyl identification task (left) and the Unbranched Alkane AND Caronbyl task (right).

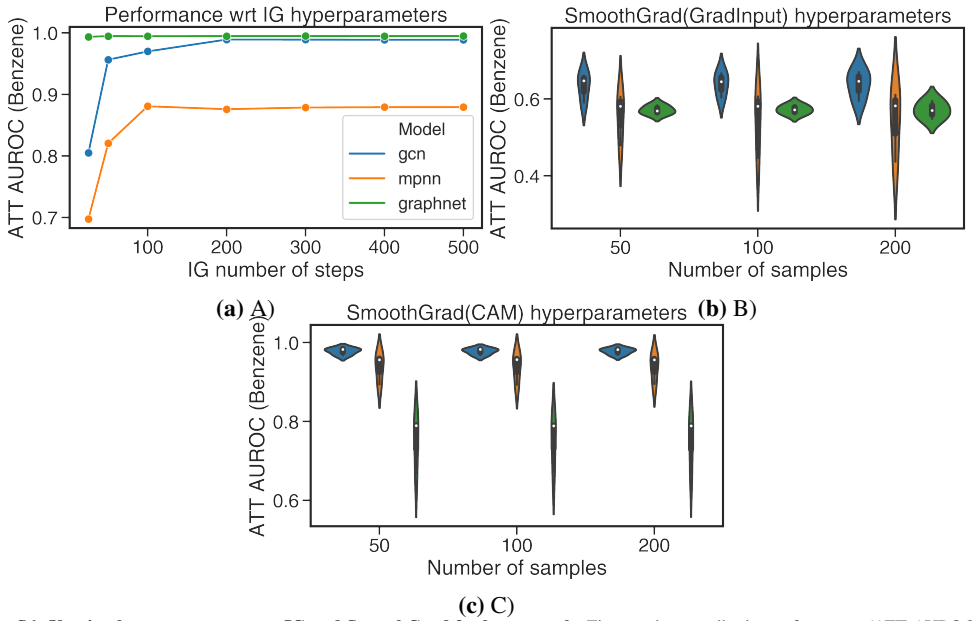

Figure S6: Varying hyperparameters on IG and SmoothGrad for bezene task. Figures show attribution peformance (ATT AUROC) in a variety of settings: A) based on the number of integration steps, B) based on the number of noisy samples used in SmoothGrad for GradInput, C) based on the number of noisy samples used in SmoothGrad for CAM. Colors correspond to model types.