A Feature Importance Explanation Methods

We briefly review several FI explanation methods and explain how they are used in this paper. These methods can be classified as gradient-based (1-2), attention-based (3), and perturbation-based (4-7). Note that when computing derivatives of model outputs for explanation methods, we use the logit of the predicted class rather than the predicted probability for purposes of numerical stability.

1. Vanilla Gradient (VGrad) [49]. This method offers an explanation of model behavior in terms of the gradient of the model output with respect to the input, $\nabla_x f_\theta(x)_{\hat{y}}$. When computing scores for a bounding box vector representation, we sum up the gradient for each element.
2. Expected Gradients (ExpGrad) [15]. This method estimates the integral in Integrated Gradients [54] by Monte Carlo sampling in order to speed up computation, and it uses the data distribution to obtain baseline inputs. That is, the explanation

$$\tilde{e} = \mathbb{E}_{\alpha\sim\text{Unif}(0,1)}\mathbb{E}_{x'\sim D}\left[(x'-x)\circ\nabla_x f_\theta\big(x'+\alpha(x-x')\big)_{\hat{y}}\right]$$

   is estimated with a single sample of α and $x' \sim D$ using the training dataset D. We consider alternative baselines x' later.
3. Attention weights (AttWeight) [25, 63, 52]. This approach treats attention weights in a model as an explanation of model feature importance. For the Up-Down model [2], we use its sole set of top-down attention weights, but early experiments suggest this is not an effective method and we do not explore it further.
4. Leave-one-out omission (LOO) [32]. An LOO explanation assigns a score to feature j as the difference in the function output on the original input and an input with feature j replaced. Any Replace function may be used with LOO. Hence $\tilde{e}_j = \text{diff}\big(f(x),\ \text{Replace}(x, \vec{1}_{-j})\big)$ where diff measures the difference in function outputs, and $\vec{1}_{-j}$ is the ones vector with element j set to 0.
5. Keep-one-in omission (KOI). The complement of leave-one-out, this method scores each feature by computing the effect of replacing all features except that a given feature.
6. SHAP [37]. We use the model-agnostic Kernel SHAP method, a generalization of LIME which assigns scores to features by fitting a linear model on perturbations of an input in order to predict the effect of each feature perturbation on the model output. Specifically, Kernel SHAP obtains an explanation by solving a weighted regression problem where model outputs are predicted based on the presence of features in the input:

$$\arg\min_{\tilde{e}} \mathbb{E}_{s\sim\mathcal{D}_s}\ \pi(s)\big(f_\theta(x_s)_{\hat{y}} - f_\theta(x_{\vec{0}})_{\hat{y}} - \tilde{e}^T s\big)^2 \tag{8}$$

   where s is a random binary mask over features, $x_s = \text{Replace}(x, s)$, the "null" input $x_{\vec{0}} = \text{Replace}(x, \vec{0})$, and π is the Shapley kernel [37]. Any Replace function may be used with SHAP.
7. Average Effect (AvgEffect). This method follows SHAP exactly except for the use of a regression. To estimate a feature's importance, we aim to compute the expected difference between model outputs with that feature observed vs. replaced:

$$\tilde{e}_j = \mathbb{E}_{s_1,s_0\sim\mathcal{D}_s}\text{diff}\big(f_\theta(x_{s_1})_{\hat{y}},\ f_\theta(x_{s_0})_{\hat{y}}\big) \tag{9}$$

   where s_1 and s_0 are versions of a random binary vector s where some element has been set to 1 in s_1 and 0 in s_0. In practice this expectation is estimated via Monte Carlo sampling. As long as elements of s are sampled independently, this method gives the same result as Kernel SHAP when the number of samples is large, but results will differ when the sample size is small.

B Replace Functions

We explain the tuning process for Replace functions in this section. As the Replace function is used in both obtaining model FI and data augmentation, we tune the Replace functions using the Align and Align+Suff-Human objectives with the LOO explanation method, and we select the function with the highest average Dev set performance. For the full sequential tuning process across all hyperparameters, see Appendix F. We consider five different Replace functions: All-Zeros, All-Negative-Ones, Gaussian, Marginal Distribution, and Shuffling. The first two functions simply replace

Table 6: $\mathrm{Replace}$ Functions Tuning

Method	Align	Align+Suff-Human
All-Zeros	68.41	68.55
All-Negative-Ones	68.29	70.67
Gaussian	68.80	69.94
Marginal Dist	67.54	69.25
Shuffle	43.10	46.10

features with zeros or negative ones. Gaussian function adds zero-mean Gaussian noise to input features with the standard deviation calculated using all features within the current batch. Marginal Distribution replaces a feature (a bounding box) with a randomly sampled feature (another bounding box) from the current batch. The Shuffle function shuffles elements of the input representation across all bounding boxes that need replacement within one sample (within and across bounding boxes). We find that All-Negative-Ones $\mathrm{Replace}$ function has the highest average accuracy on the Dev set, and we use it for all situations where replacement is needed (see Table 6).

C Differentiable SHAP

In this section, we show how to differentiate through SHAP explanations while respecting the theoretical properties that SHAP explanations provide. In Appendix D, we discuss how to limit the computational burden of computing perturbation-based explanations during model training.

Kernel SHAP [37] values are obtained via a weighted linear regression as follows: To explain a model $f : \mathcal{X} \to \mathcal{Y}$, one defines a data distribution $\mathcal{D}_s$ over binary feature masks for randomly replacing features with some reference value (denoted in our paper by the $\mathrm{Replace}$ operation). In SHAP, these reference values are either (1) randomly drawn from the marginal data distribution over that feature, or (2) preset by the user to a fixed value for all features. We choose the second option based on $\mathrm{Replace}$ function tuning. The closed-form solution for SHAP values is then given by a weighted least-squares regression [37]:

$$\tilde{e} = (S^T W S)^{-1} S^T W Y \tag{10}$$

where the row vector S_i is drawn from $\mathcal{D}_s$, W is a diagonal weight matrix with elements $W_{ii} = \pi(S_i)$, and Y_i is the difference in function outputs on x_{S_i} and the "null" input, $f_\theta(x_{S_i}) - f_\theta(x_{\vec{0}})$. This formulation can also satisfy the additivity constraint that the explanation weights sum to the difference $f_\theta(x) - f_\theta(x_{\vec{0}})$. This is done by adding a "data point" S_i that is all ones, with its weight W_{ii} manually set to a large value. The resulting explanation is differentiable w.r.t. θ by virtue of being differentiable w.r.t. Y.

D Varying Compute Budgets in Feature Importance Methods

In Table 10, we show the performance of each FI method for improving CLEVR-XAI dev ID accuracy with UpDn. Surprisingly, we find that accuracy improvements do not increase with a higher compute budget for the FI method. Below, we describe how the compute budget can vary for each method.

Vanilla Grad and Attention have invariable compute budgets, as measured in terms of the number of forward+backward passes. However, the other methods have variable budgets. Expected Gradients depends straightforwardly on the number of sampled α values, since we use the same negative-ones baseline feature value for all points (one forward and one backward pass per sample).

To compute a perturbation-based explanation, we need to compute f_θ at least once per feature in x. This is because we need to measure the effect of replacing each feature separately. With SHAP, we need at least one sample S_i per feature in order for $\tilde{e}$ to be identifiable (equivalently, for $S^T W S$ to be invertible). However, a *complete explanation is not needed for every datapoint during training*. Instead, we can estimate feature importance for only a subset of features for each datapoint in a given batch. This allows us to greatly limit the computational cost of explanation supervision. In fact, we can use *as little as a single sample per data point*. The strength of SGD-based training in this context is that, over the course of training, a large number of feature importance estimates will be computed and penalized against human explanations.

Table 7: Resplit Sensitivity Test.

	Resplit 1		Resplit 2		Resplit 3	
FI Method	ID Acc.	OOD Acc.	ID Acc.	OOD Acc.	ID Acc.	OOD Acc.
Baseline	69.99	50.40	69.21	57.73	69.08	58.75
VisFIS	72.00	52.79	71.03	60.28	70.70	62.14

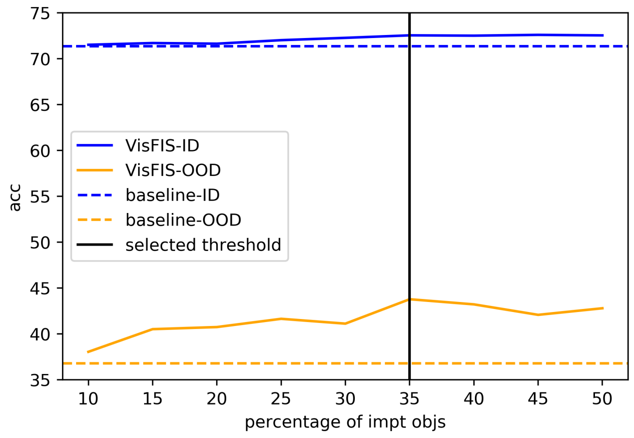

Figure 4: Threshold ablation on CLEVR-XAI.

With our per-explanation compute budget of k model forward+backward passes and an input dimensionality d, we allow for $k < d$ by explaining only k features while keeping the other $d - k$ features constant. With LOO explanations, this simply requires not computing scores for $d - k$ features, which are ignored in the $\mathcal{L}_{\text{align}}$ loss. With SHAP explanations, we pick $d - k$ features to always set to 0 in our random masks s. Then when computing Eq. 10, we drop those constant feature columns from S to obtain a new $k \times k$ matrix, which ensures that $\tilde{e}$ is identifiable.

E Data Details

Dataset License. We conduct experiments on three datasets: CLEVR-XAI [5] under the CC BY-NC-ND 4.0 license, GQA [21], and VQA-HAT [11] both under the CC BY 4.0 license.

Distribution Shift Resplit Sensitivity. Since we randomly construct ID and OOD splits with our distribution shift, we show the robustness of VisFIS across three resplits of CLEVR-XAI dataset here. In Table 7, we see that VisFIS gives significant performance improvements in all resplits. The absolute OOD accuracies vary across resplits, but the size of the OOD performance improvement between VisFIS and the baseline is generally similar, with between a 2.3 and 3.4 percentage point improvement for each split.

Threshold for Human Feature Importance. For all our objectives except for Align, we need to select the threshold for human FI to separate important features from unimportant ones. We select the threshold separately for each dataset mainly based on (1) qualitative visualizations of the important features and (2) the percentage of data without important features. If a data point is without important features given a threshold, we do not use FI supervision objectives for that datapoint, but we do compute the main task objective, $\mathcal{L}_{\text{Task}}$. Although we want the importance features to be reasonable given qualitative visualizations, we don't want to exclude too much data from training. We balance between good qualitative results and relatively few excluded data points by selecting thresholds of 0.85, 0.55, and 0.3 for CLEVR-XAI, VQA-HAT, and GQA respectively. These thresholds exclude 1%, 1%, and 8% of data from training with additional objectives for the three datasets. To ensure that VisFIS is robust to this choice of threshold, we measure its performance improvement across a range of thresholds using UpDn on CLEVR-XAI. In Fig. 4, we present model accuracy as a function of the

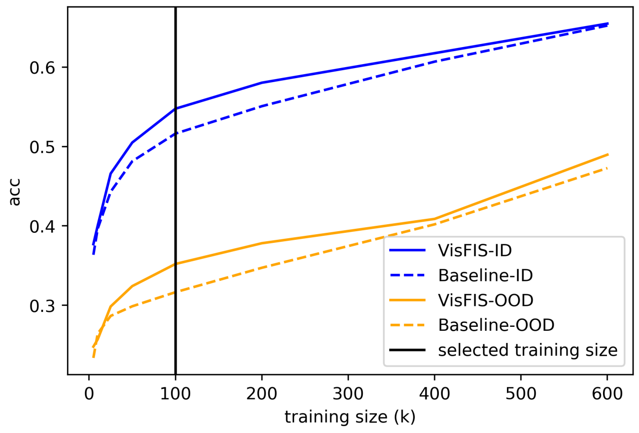

Figure 5: Training Size Ablation on GQA with an UpDn model.

Table 8: Thresholds for categorizing explanation faithfulness and subsequent distribution statistics, for UpDn models on CLEVR-XAI.

		Distribution over Faithfulness	
Metric	Category	Threshold	Data Proportion
Sufficiency	Worst	≥0.25	21%
Sufficiency	Middle	<0.25	25%
Sufficiency	Best	<0.01	53%
Comprehensiveness	Worst	<0.20	32%
Comprehensiveness	Middle	<0.40	41%
Comprehensiveness	Best	≥ 0.40	27%

percentage of objects across images that are deemed as important based on a threshold. The values of the threshold vary from 0.1 to 0.98. The results show that performance improvements in ID and particularly OOD test accuracy are obtainable across a large range of threshold values.

Training Size Ablation. GQA dataset [21] contains 943k training points and 132k validation points. After distribution shift, based on a ratio of 6:1:1.5:1.5, we obtain 645k, 107k, 161k, and 161k data for Train, Dev, Test ID, and Test OOD sets respectively. We then downsample the train set to about 1/6 of its original size. We also exclude a small fraction of data with no ground-truth bounding boxes, and we limit our dev and test sets to 20k points. Thus the final split sizes are 101k train points, 20k Dev, 20k ID Test, and 20k OOD Test. We term this dataset GQA-101k in the main paper. To measure how FI supervision improvements vary with the amount of training data, we compare VISFIS with the baseline for GQA using between 5k and 600k training points. Shown in Fig. 5, the results suggest that supervision is most helpful for improving OOD accuracy when using between 10k and 300k training points, though improvements in OOD accuracy may still be obtained beyond this value.

Categorizing Faithfulness into Worst/Middle/Best Groups. As part of our analysis of how accuracy varies with explanation plausibility, we group datapoint explanations into three faithfulness categories, Worst, Middle, and Best. We select these based on theoretically sensible values of the Sufficiency and Comprehensiveness metric (see Sec. 5 for metric definitions). To be in the Best Sufficiency category, the average Sufficiency score (across explanation sparsity levels) must be at or below 0.01, meaning that the Replaced input must receive a predicted probability no more than one percentage point below the original. For UpDn on CLEVR-XAI, this is about 53% of the data. To be in the Best Comprehensiveness category, removing the top features must lower the predicted probability by at least 0.4 points (on average across explanation sparsity levels). We give the remaining values and data proportions in Table 8.

Table 9: Feature importance method tuning for VisFIS objective with UpDn model on CLEVR-XAI dev set. The accuracy is averaged over five random seeds. See Appendix F for the full tuning details.

Method	accuracy
Vanilla Grad-gt	72.55
KOI-gt	71.92
ExpGrad-pred	72.43

F Training Details

Our implementations makes use of PyTorch [42]. Our UpDn model is optimized with a standard Adam [29], and LXMERT uses Adam with a linear-decayed learning-rate schedule [12]. We use a batch size 64 for UpDn and 32 for LXMERT. For all experiments, we train UpDn for 50 epochs and LXMERT for 35 epochs. UpDn is trained from scratch, while LXMERT uses the default pretrained checkpoint. It takes about an hour to train UpDn on an Nvidia RTX 2080 Ti and about 6 hours for LXMERT on an Nvidia A100.

Hyperparameter Tuning. We detail the tuning steps here. All tuning is done using CLEVR-XAI. The tuning is done in sequential order. We first tune learning rate for the baseline UpDn and LXMERT models. Learning rate is chosen from {1e-2, 5e-3, 1e-3, 5e-4, 1e-4} for UpDn and {5e-4, 1e-4, 5e-5, 1e-5}. We settle with 1e-3 and 5e-5 respectively. We then fix the learning rate and tune the weight λ_i for different objectives. For augmentation objectives, we tune the weight with UpDn and use the same weight for LXMERT. The weight for augmentation is chosen from {100, 10, 1, 1e-1, 1e-2}, and we end up using weight of 1 for all augmentation objectives. For Inv-FI and Align objectives, we use FI method LOO with all-zeros replacement function, and tune the weight for UpDn and LXMERT separately. For UpDn, the weight is chosen from {100, 10, 1, 1e-1, 1e-2}, and for LXMERT, it is chosen from {1e-3, 1e-4, 1e-5, 1e-6, 1e-7}. We use weight 1 for UpDn and weight 1e-3 for LXMERT+Inv-FI and weight 1e-6 for LXMERT+Align. We also tune the alignment function - Cosine Similarity, KL divergence, L1 distance, and L2 distance - for the Uncertainty and Align objectives and use KL for Uncertainty and Cosine Similarity for Align. In addition, we tune the weight for HINT [47] and SCR [64] with Vanilla Gradient. The weight is chosen {10, 1, 1e-1, 1e-2, 1e-3, 1e-4} for UpDn and {1e-3, 1e-4, 1e-5, 1e-6, 1e-7} for LXMERT. We use 1e-3 for UpDn and 1e-6 for LXMERT. We then fix the objective weights and tune the $\mathrm{Replace}$ function (results in Table 6). Finally, we tune the FI method to use for Inv-FI and Align objectives. KOI-gt works the best with Inv-FI, and Expected Gradient-pred for Align. The numbers for Inv-FI and Align in Table 11 are obtained with KOI-gt and Expected Gradient-pred respectively. We then tune VisFIS with KOI-gt, Expected Gradient-pred, and, for fair comparison with other relevant works, Vanilla Gradient-gt. It turns out that Vanilla Gradient gives the greatest performance gain, and we choose it for all our experiments with VisFIS (see Table 9).

Stop Gradient. When backpropagating through model explanations, we apply a stop gradient for particular FI supervision methods in order to avoid influencing how the model handles the full input (which should be used principally for the task loss $\mathcal{L}_{\text{Task}}$). For FI methods that involve baseline output $f_\theta(x)$ or "null" output $f_\theta(x_{\vec{0}})$, which includes Excepted Gradient, LOO, KOI, and SHAP, we stop the gradient at $f_\theta(x)$ and $f_\theta(x_{\vec{0}})$.

G Additional Results

G.1 Which Objectives Are Affected by Random Supervision?

Design. In earlier experiments, we find that VisFIS does not improve performance with random supervision. Here, we further explore how each of the four additional objective terms in VisFIS is individually influenced by random supervision. To assess the effect of random supervision on each objective, we give random supervision to one of the objectives and normal supervision to the other three on CLEVR-XAI with UpDn.

Results. We show the results in Table 12. The Suff-Human objective is the main reason why VisFIS does not work with random supervision. Uncertainty and alignment objectives with random

Table 10: Feature importance method ablation using the Align objective term, for Updn on the CLEVR-XAI dataset. Budget is the number of additional forward and backward passes used by the method.

	Accuracy @ Compute Budget				
Method	0	1	2	15	30
Attention	71.07	-	-	-	-
Vanilla Grad	-	71.03	-	-	-
Expected Grad	-	-	71.80	71.75	71.54
LOO	-	70.89	71.11	70.99	-
KOI	-	-	71.04	71.16	-
SHAP	-	-	71.05	71.18	71.18
AvgEffect	-	-	71.05	71.03	71.15

Table 11: Objective term ablation for the CLEVR-XAI dataset with an UpDn model

	Accuracy		RRR Metrics			Expl. Metrics		
Objective	ID ↑	OOD ↑	Suff ↑	Inv ↑	Unc ↓	Plau	Suff ↓	Comp ↑
Baseline	71.37	36.80	48.82	77.89	55.17	28.82	34.66	47.56
Saliency Guided	71.50	37.71	73.00	92.17	76.98	13.84	-7.13	21.23
Inv-DA	71.17	35.91	72.53	**93.12**	76.29	14.33	**-7.32**	21.30
Inv-FI	71.41	38.88	45.31	76.34	71.41	28.60	35.79	**48.20**
Uncertainty	71.30	38.34	10.75	86.58	**4.16**	8.56	73.49	41.65
Align	72.04	41.61	61.19	79.51	64.22	37.20	26.86	35.18
Suff-Random	71.73	39.08	73.59	92.59	60.93	17.32	-5.29	22.48
Suff-Human	71.87	40.91	76.94	90.82	81.42	16.27	-6.68	26.45
+ Align	72.42	41.63	**78.55**	89.69	80.02	**35.73**	0.53	27.18
+ Unc	72.33	41.54	77.83	89.70	41.68	23.41	-5.18	37.15
+ Align+Unc+Inv	**72.82**	**43.78**	76.65	91.72	43.64	22.67	-0.30	29.51

Table 12: Random supervision control experiments on UpDn + CLEVR-XAI for different objective terms in VISFIS. We use a fixed set of random explanations for one objective at a time.

Method	ID acc	OOD acc
Baseline	71.30	36.80
VISFIS	72.82	43.78
w/ random Suff-Human	69.93	36.70
w/ random Unc	72.51	41.87
w/ random Align	71.27	39.58
w/ random Inv-FI	72.59	44.20

supervision hurt the performance, but not as much as the sufficiency objective. Note that Suff-Human with random supervision is different from Suff-Random, which has different features mask out across the training process for the same sample. Here, Suff-Human with random supervision has the same (random) features masked out for the entire training process. The invariance objective with random supervision does not hurt the performance at all.

G.2 Accuracy-Plausibility Relationship Across Test Splits, Datasets, and Models

In the main paper Fig. 2, we show how accuracy varies as a function of explanation plausibility and faithfulness for UpDn models on CLEVR-XAI, and we group data points across ID and OOD test splits. Here, we show that the main trends are generally consistent across the choice of explanation metric (Sufficency vs. Comprehensiveness), test split (ID vs. OOD), dataset, and model. Trends across metric and split are shown in Fig. 6, and trends across datasets are shown in Fig. 7. We show results for LXMERT on CLEVR-XAI in Fig. 8. Though the trends weaken slightly in certain settings, we always find that accuracy correlates positively with plausibility for highly faithful explanations, while the relationship is weaker or non-existent for unfaithful explanations.

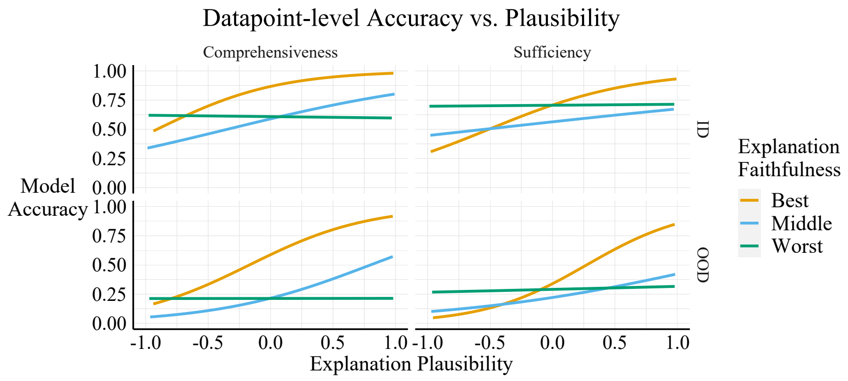

Figure 6: Datapoint level accuracy by explanation plausibility and faithfulness, for CLEVR-XAI models, grouped by faithfulness metric and test split.

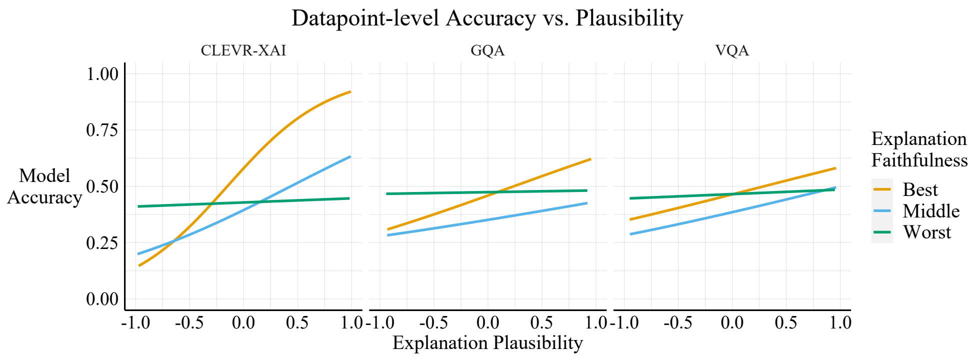

Figure 7: Datapoint level accuracy by explanation plausibility and faithfulness for UpDn models, grouped by dataset.

G.3 Which FI Method Produces the Most Faithful Explanations?

Design. We calculate the Explanation Sufficiency and Explanation Comprehensiveness metrics for the FI methods listed in Appendix A, using either the predicted or ground truth class to select the output logit that is explained. All experiments are conducted on CLEVR-XAI with UpDn. Following guidelines from Hase et al. [20], the UpDn models are trained with Suff-Random objective to make the replaced features in-distribution for the models. For LOO and KOI, we use a budget of 15 and 36 on CLEVR-XAI and VQA-HAT/GQA, which is the same number as the number of bounding boxes. For SHAP, Average Effect, and Expected Gradient, we use a budget of 1000 to reduce noise in deriving each explanation, as these methods involve random sampling. We select the best explanation method for each dataset by taking the best score on average across the two metrics.

Results. We show the results in Table 13. In general, explanations obtained on predicted class are more faithful to the model decisions than those obtained on ground truth class. UpDn attention, LOO-pred, and KOI-pred are among the best across three datasets. SHAP and Average Effect surprisingly are not very faithful across all three datasets. KOI on predicted class is the most faithful one for CLEVR-XAI and VQA-HAT, while LOO on predicted class is the best for GQA. Hence, when calculating explanation metrics, we use KOI on predicted class for CLEVR-XAI and VQA-HAT and LOO on predicted class for GQA.

G.4 Can Explanation Supervision Improve Model Explainability?

Design. To assess the effect of FI supervision on model explainability, we record faithfulness metrics using all of our CLEVR-XAI models. We then plot explanation Sufficiency and Comprehensiveness for each model (averaged across five seeds) to visualize the distribution of faithfulness scores.

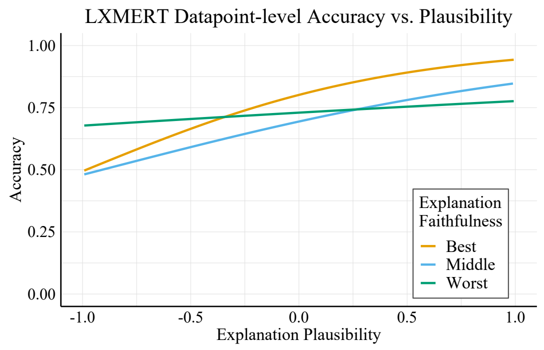

Figure 8: Datapoint level accuracy by explanation plausibility and faithfulness, for LXMERT on CLEVR-XAI, averaged across faithfulness metrics and test splits.

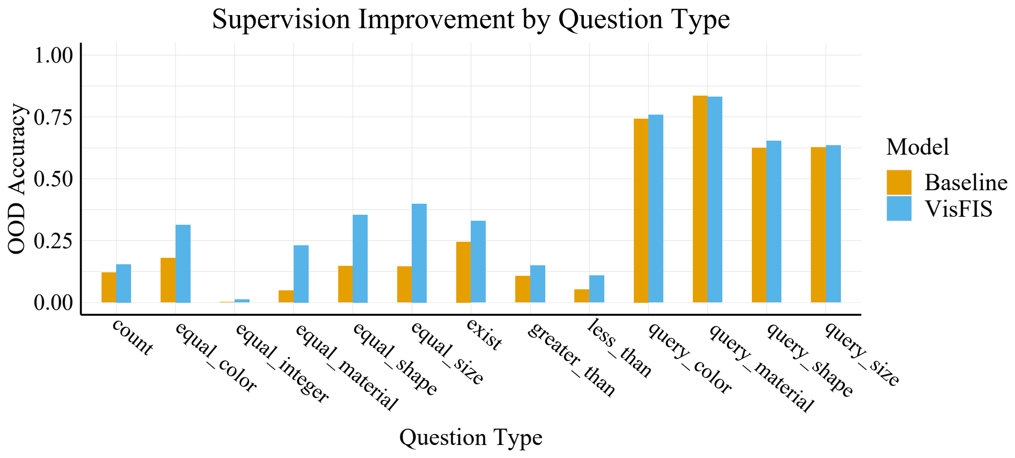

Figure 9: OOD accuracy for the baseline and VisFIS on CLEVR-XAI with UpDn, grouped by question type.

Table 13: FI tuning for explanation metrics with UpDn models on Dev ID data.

	CLEVR-XAI		VQA-HAT		GQA-101k	
FI Method	Suff ↓	Comp ↑	Suff ↓	Comp ↑	Suff ↓	Comp ↑
UpDn Attention	1.19±0.30	20.46±0.72	4.08±4.48	9.06±2.83	0.08±0.40	**11.29±1.65**
Vanilla Grad-pred	8.80±1.79	12.70±1.20	8.06±4.24	5.70±4.33	14.76±1.67	1.60±1.23
Vanilla Grad-gt	5.04±1.08	16.01±0.83	13.21±2.32	5.46±3.44	15.64±2.27	3.00±0.97
ExpGrad-pred	5.15±1.92	12.39±2.05	3.41±5.06	9.20±2.91	0.25±0.38	7.90±1.80
ExpGrad-gt	9.71±1.12	9.78±1.58	6.12±3.85	6.91±3.46	5.00±1.22	4.43±0.99
LOO-pred	-5.01±0.24	14.34±0.82	-2.51±6.97	9.86±3.99	**-3.32±0.33**	9.26±1.82
LOO-gt	2.62±0.73	9.67±0.48	5.75±4.04	4.35±1.82	5.31±1.62	3.14±0.82
KOI-pred	**-5.17±0.32**	**22.43±1.14**	**-3.45±7.34**	**10.61±4.31**	6.05±7.61	7.81±2.68
KOI-gt	3.25±0.73	18.23±0.77	7.71±3.57	5.40±2.28	19.63±6.27	3.62±1.24
SHAP-pred	17.06±0.73	2.86±0.19	36.12±12.57	2.20±3.03	15.25±7.98	0.04±0.14
SHAP-gt	17.07±0.76	2.84±0.19	33.87±12.60	2.11±2.82	13.53±7.07	0.04±0.16
Average Effect-pred	17.35±0.71	2.83±0.18	7.51±3.30	3.79±4.58	1.69±0.14	0.14±0.22
Average Effect-gt	17.46±0.72	2.74±0.20	7.38±3.37	3.88±4.52	1.68±0.14	0.14±0.22

Results. We show results for each model in Fig. 10 (scores also listed in Appendix Table 11). We find that average explanation Sufficiency and Comprehensiveness scores lie along a Pareto frontier, shown by the gray line, which represents a trade-off between better Sufficiency and Comprehensiveness (models better in one metric are worse in the other). Generally, explanation supervision does not improve model explainability relative to unsupervised models, with the exception of the Suff+Unc objective. In the bottom right of the plot, this model demonstrates a better combination of Sufficiency

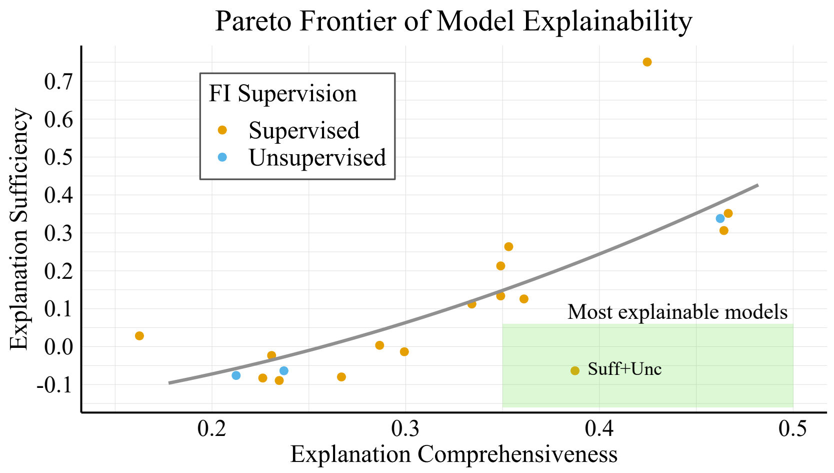

Figure 10: Average model explanation Sufficiency and Comprehensiveness scores (shown for models on in-distribution CLEVR data).

Table 14: Datapoint level faithfulness distributions (in terms of Sufficiency) conditional on datapoint-level and model-level plausibility scores, averaged across CLEVR-XAI models.

		Distribution over Faithfulness		
Model Plausibility	Data Plausibility	Worst	Medium	Best
Low	Low	0.51	0.27	0.22
Low	Middle	0.19	0.41	0.40
Low	High	0.11	0.49	0.40
Middle	Low	0.02	0.13	0.85
Middle	Middle	0.01	0.1	0.89
Middle	High	0.01	0.07	0.92
High	Low	0.24	0.31	0.45
High	Middle	0.20	0.27	0.52
High	High	0.18	0.23	0.60

and Comprehensiveness than other supervised or unsupervised methods, including Saliency-Guided Training [22]. The Suff+Unc model is especially explainable likely because the Sufficiency objective encourages the model to rely on a small number of important features, while the Uncertainty objective encourages the model to become less confident when those important features are removed.

G.5 How Can Models with Low Plausibility Achieve High Accuracy?

Shown in Table 14, we find that models with lower average plausibility show different conditional relationships than models with higher average plausibility, which helps explain why low-average-plausibility models can achieve similar accuracies to high-average-plausibility models. Low-average-plausibility models have low plausibility points with low faithfulness scores, meaning these points are still often accurately predicted and hence do not bring down the average model accuracy. Meanwhile, middle and high-average-plausibility models often have low-plausibility points with highly faithful explanations, meaning these points are often inaccurately predicted, offsetting any gains to average model accuracy that are achieved for points with both highly plausible and faithful explanations.

G.6 Do RRR Metrics Predict OOD Generalization? Additional Datasets and Models

We measure the correlation between RRR metrics (calculated with ID data) and OOD accuracy across a large set of models. We report results additional in Table 16 here for LXMERT models on CLEVR-XAI and UpDn for GQA/VQA. We perform a cross-validation resampling model-level statistics 10k times, using 40 models' metrics as training data and 5 for testing each time. The final metrics we consider are: (1) ID accuracy on its own as a baseline, (2) RRR metrics on their own, (3) ID accuracy plus average model confidence, (4) ID accuracy plus explanation metrics, (5) ID

Table 15: Test accuracy for Updn model on full VQA test set, including all question types.

Method	VQA-HAT ID	VQA-HAT OOD
Baseline	52.22 ± 0.92	38.95 ± 0.91
Suff-Random	52.26 ± 0.90	39.30 ± 0.97
Selvaraju et al. [47]	52.11 ± 1.01	37.95 ± 1.07
Wu and Mooney [64]	52.16 ± 0.94	38.53 ± 0.94
Simpson et al. [50]	52.32 ± 0.91	38.84 ± 1.08
Chang et al. [7]	50.42 ± 1.01	31.29 ± 1.44
Singla et al. [51]	52.93 ± 0.96	39.05 ± 1.64
VisFIS	52.79 ± 0.95	40.49 ± 0.96
w/ Rand. Supervis.	52.21 ± 0.94	37.95 ± 0.99

Table 16: Correlations between metrics and OOD accuracy for additional datasets and model architectures. We derive results from 45 models (differing by seed and objective) per condition.

	UpDn + VQA-HAT		UpDn + GQA-101k		LXMERT + CLEVR-XAI	
Metric	Train	Test	Train	Test	Train	Test
RRR-Suff	0.393	0.627	0.644	0.584	0.464	0.553
RRR-Inv	0.011	0.148	0.549	0.526	0.035	0.160
RRR-Unc	0.470	0.530	0.478	0.459	-0.111	0.024
ID Acc	0.952	0.850	0.908	0.859	0.903	**0.898**
+ Model Conf.	0.957	**0.866**	0.921	**0.876**	0.910	0.858
+ Expl. Metrics	0.956	0.847	0.923	0.873	0.923	0.883
+ RRR-all	0.958	0.846	0.929	0.875	0.920	0.859
All Metrics	**0.965**	0.816	**0.943**	0.832	**0.938**	0.768

accuracy plus RRR metrics, and (6) All Metrics, which uses all available metrics. The results are similar to in the main paper, showing that RRR metrics do not achieve a better correlation with OOD accuracy than ID accuracy does on its own.