XC: Exploring Quantitative Use Cases for Explanations in 3D Object Detection

Sunsheng Gu, **Vahdat Abdelzad,** **Krzysztof Czarnecki**
University of Waterloo
ssgu@uwaterloo.ca
vahdat.abdelzad@uwaterloo.ca
krzysztof.czarnecki@uwaterloo.ca

Abstract

Explainable AI (XAI) methods are frequently applied to obtain qualitative insights about deep models' predictions. However, such insights need to be interpreted by a human observer to be useful. In this paper, we aim to use explanations directly to make decisions without human observers. We adopt two gradient-based explanation methods, Integrated Gradients (IG) and backprop, for the task of 3D object detection. Then, we propose a set of quantitative measures, named Explanation Concentration (XC) scores, that can be used for downstream tasks. These scores quantify the concentration of attributions within the boundaries of detected objects. We evaluate the effectiveness of XC scores via the task of distinguishing true positive (TP) and false positive (FP) detected objects in the KITTI and Waymo datasets. The results demonstrate improvement of more than 100% on both datasets compared to other heuristics such as random guesses and number of LiDAR points in bounding box, raising confidence in XC's potential for application in more use cases. Our results also indicate that computationally expensive XAI methods like IG may not be more valuable when used quantitatively compare to simpler methods.

1 Introduction

Recent development in deep neural networks (DNNs) has led to the state of the art performance on 2D [8, 9, 16, 21, 22] and 3D [12, 27, 37] object detection tasks. However, despite of the improvement in model performance, the lack of model interpretability remains a significant drawback. This issue has sparked interest in *explainable artificial intelligence* (XAI). The primary goal of these XAI methods is to uncover the logic behind the models' decisions [14].

Several recent methods have been proposed to generate interpretable visual explanations [28, 36, 30, 25, 32]. They are usually analyzed by a human observer to understand the model's reasoning for some particular decisions. Hence, explanations are very useful for debugging and diagnosis. However, it would be impossible for a human to analyze each instance in a large dataset. To analyze explanations on a large scale, it is necessary to derive quantitative measures

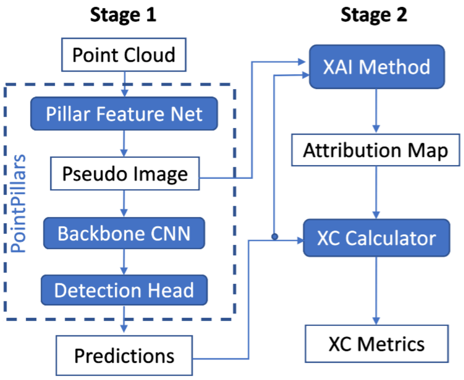


Figure 1: Overview of the XC calculation process: PointPillars first process the input point cloud once, then an XAI method computes feature attribution map for specific predictions using the pseudo image as input, and XC metrics are obtained.

1st Workshop on eXplainable AI approaches for debugging and diagnosis (XAI4Debugging@NeurIPS2021).

from the explanations. There have been a few works in this direction [24, 6, 23], but the input data experimented on are 2D image, text, or tabular data. We have not found any experiments on explanations-related quantitative measures applied to LiDAR point cloud input or to the task of object detection. Our motivation is to fill in the gap by exploring quantitative usage of explanations for LiDAR-based object detection.

We study explanations in the form of attribution maps for a 3D object detector named PointPillars [12] and use them for the problem of distinguishing true positive (TP) vs. false positive (FP) predictions. Attributions, in the context of XAI, denote the influence of input features on model output. We generate attribution maps using two XAI methods: 1) Integrated Gradients (IG) [32], selected for its axiomatic properties; 2) backpropagation [28], selected for its low computational cost. Our main contributions are [1]:

- We demonstrate that quantitative usage of explanation for 3D object detection is a promising direction for future research: We propose a set of quantitative metrics called *Explanation Concentration* (XC), which measures the concentration of attributions within each predicted object. XC can be used to classify the TP vs. FP predictions effectively, often achieving more than 100% improvement compared to simple heuristics such as number of LiDAR points within a predicted box.
- We discover that XC scores derived from backpropagation can perform better than those derived from IG.
- We propose a new score that can identify TP vs. FP pedestrian predictions better than the individual XC scores and object class score. This score is generated by combining the XC scores with object class score using a MLP.

2 Related work

Explanation methods A simple form of explanation is an input feature saliency map obtained via backpropagation: one computes the partial derivatives of the model output with respect to the input features [28]. These partial derivatives are a measure of importance (also called *attributions*) for the input features. Deconvolution [36] and Guided-backprop [30] are similar methods. Integrated Gradients (IG) [32] is another gradient-based explanation method that computes multiple inputs along a straight-line path from a baseline input to the original input through affine transformations. IG then combines gradients from all these inputs to get feature attributions. The baseline input is defined as an input which generates zero output. In the case of image input, the baseline is defined as a black image with pixel values all equals to zero. Sundararajan *et al.* [32] demonstrate that IG satisfies several axiomatic properties for explanations such as sensitivity (attribution values for nonzero features are nonzero, and attribution values for zero-valued features are zero) and completeness (sum of attributions equals to model output value).

Quantitative usage of explanations Several attempts have been made to utilize explanations quantitatively. Ross *et al.* [24] use the gradient of the log of model output with respect to input features as attributions. They design a loss term using prior knowledge about feature relevance to punish the model for having attributions for irrelevant features, hence forcing the model to be "right for the right reasons" by focusing on the relevant features only. Rieger *et al.* [23] and Du *et al.* [5] also propose similar methods to constrain feature attribution values. Erion *et al.* [6] realize that such prior knowledge about input feature relevance would not always be available, hence they propose a new loss named “pixel attribution prior" (PAP) which only aims to reduce variations in attribution values for neighbouring pixels. Their reasoning is that a robust model should not be heavily influenced by small variations in a few disconnected pixels; rather, it should learn patterns from patches of pixels and neighbouring pixels should have similar attributions.

LiDAR-based object detection A popular technique for point cloud-based object detection is to first divide the 3D point cloud into grids called voxels, learn features for each voxel, then apply convolutional layers to the voxel-wise features to extract higher level features. Techniques used for 2D object detection, such as region proposals and anchor boxes, can then be applied to the extracted feature maps. VoxelNet [37], SeCOND [35], and PointPillars [12] all partly adopted this strategy.

[1] Our code is available at `https://github.com/SunshengGu/XC_eval_pcdet`.

PointPillars' fast inference speed (62 Hz) makes it a desirable choice for deployment on embedded hardware, such as in autonomous vehicles.

False positive detection Several studies have been conducted on identifying false positive (FP) predictions. Hendrycks and Gimpel [10] demonstrated that the softmax class probability could effectively distinguish true positive (TP) and false positive (FP) predictions in multiple datasets, with performance far exceeding random guess. Chen *et al.* [2] also used class score to filter out FP pedestrians predictions iteratively. Methods designed for identifying adversarial attacks or detecting out-of-distribution (OOD) samples may also be applied to detect FP samples. Some notable works are feature squeezing [34], LID [18], GraN [17], ODIN [15] and Lee *et al.* [13].

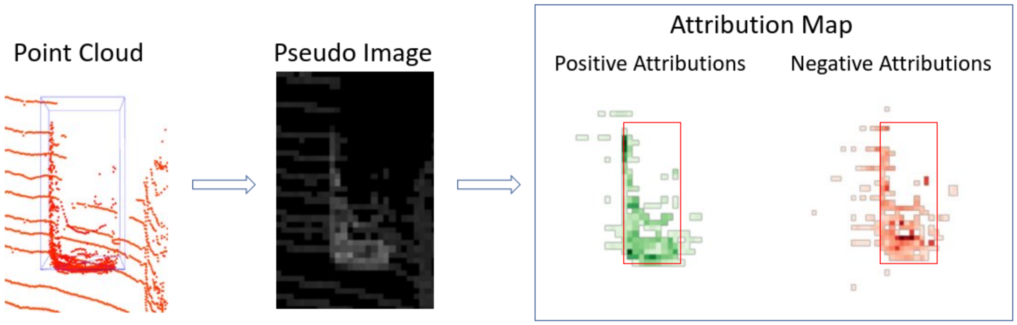


Figure 2: Cropped IG attribution map visualization for a car prediction. The blue box in the point cloud is the bounding box for the ground truth object. The red box in the attribution map is the 2D projection of the car prediction bounding box. Positive attributions are indicated by green pixels and negative attributions are indicated by red pixels; darker color means greater magnitude. Pixels with attribution values less than 0.1 are filtered out and appear white. Note that the attribution map has the same resolution as the pseudo image.

3 Explanation in the form of attributions

As mentioned before in Section 2, explanation can be presented in the form of input feature attributions, where the magnitude of the attributions corresponds to feature importance, and the sign of attributions indicates positive or negative influence on the model output. In our approach, attributions are generated by IG [32] and backprop [28] for PointPillars [12]. PointPillars has three parts as shown in Fig. 1: a pillar feature net which learns a 2D bird's eye view (BEV) voxelized feature map called the *psuedo image* from the original point cloud, a backbone 2D CNN to extract more features from the psuedo image, and SSD [16] as detection head to generate 3D object predictions.

IG's effectiveness on image data is well-recognized. However, there are a fixed number of pixels at fixed locations in an image; whereas in a point cloud, both the quantity and location of the points can vary greatly. IG's input transformations may not be meaningful for point cloud. Hence, we chose to generate attributions using the pseudo image as input, so that IG could be directly applied. Note that our pipeline in Fig. 1 is not limited to IG or backprop explanations only, any other XAI method that produces explanations in the form of feature attributions could be applied too.

There could be many predictions in the same pseudo image. To explain the different predictions separately, each attribution map is generated for a specific predicted box and its class label. In Fig. 2, the IG attributions generated for a car prediction is shown. For this particular example, the positive attributions highlight features that increase the model's belief that the object is a car, whereas the negative attributions reduce this belief.

4 Explanation concentration (XC)

We propose a new set of scores called *Explanation Concentration* (XC) which measures the concentration of attributions within a given predicted object's boundaries. As shown in Fig. 2, many pixels outside of the object have noticeable attributions too, indicating that context features can influence model output as well. The XC scores are partly motivated by previous studies which demonstrate that overly-relying on context can hurt model performance for both image classification [26] and 3D object detection [27].

The process of computing the XC scores is as follows. First, denote the i^{th} predicted box in a pseudo image as $Pred_i$. Then denote the sum of positive attributions across all channels in the pixel at location (x, y) of the attribution map as $a^+_{(x,y)}$. The idea is to avoid having positive and negative channel values cancelling each other out at a specific pixel.

Using thresholds to eliminate noisy signals is common practice in computer vision. For example, in the Canny edge detection algorithm [1], thresholds are used to mask out weak edges. We apply the same idea to the attribution values. Denote a pixel-wise threshold as a_{thresh}, and use it to filter out insignificant attributions (in other words, the sum $a^+_{(x,y)}$ can be called *significant* if it exceeds a_{thresh}). An indicator function $I^+_{(x,y)}$ is defined such that it equals to 1 if $a^+_{(x,y)} >= a_{thresh}$, 0 otherwise. One way to quantify the concentration of attributions within the predicted box $Pred_i$ is by summing. Two new variables are defined for each $Pred_i$ in a pseudo image:

$$s^+_i = \sum a^+_{(x,y)} \times I^+_{(x,y)},\ \forall (x, y) \text{ in } Pred_i \tag{1}$$

$$S^+_i = \sum a^+_{(x,y)} \times I^+_{(x,y)},\ \forall (x, y) \text{ in pseudo image} \tag{2}$$

Now the stage is set for defining an XC score by summing:

$$XC\_s^+_i = s^+_i / S^+_i \tag{3}$$

$XC\_s^+_i$ is thus the proportion of the positive attributions which lie within $Pred_i$.

The other way to quantify the concentration of attributions for $Pred_i$ is by counting. For this purpose, we count the number of pixels in the pseudo image having significant attributions, rather than summing them:

$$c^+_i = \sum I^+_{(x,y)},\ \forall (x, y) \text{ in } Pred_i \tag{4}$$

$$C^+_i = \sum I^+_{(x,y)},\ \forall (x, y) \text{ in pseudoimage} \tag{5}$$

$$XC\_c^+_i = c^+_i / C^+_i \tag{6}$$

Computing XC by counting might be helpful, because often there are a few outlier pixels with high magnitude attribution values outside of the bounding bounding box (see Fig. 3 for example). When XC is computed by counting, these outliers will not have a big effect; but when XC is computed by summing, the values of these outlier pixels can skew the resulting XC value towards the lower end. One may also compute a similar set of scores by considering the negative attributions only: call them $XC\_s^-_i$ and $XC\_c^-_i$ for a certain box $Pred_i$.

We observed that pixels located at object boundaries often get labelled as "outside of the box". Hence, when calculating any XC scores, the predicted boxes are enlarged by a small margin m on all sides, so that pixels at object boundaries are labeled as inside the predicted box.

5 Results and discussions

5.1 Evaluation metrics and implementation details

The objective of our experiment is to evaluate XC's performance on a meta classification task: classifying predictions as either FP or TP. The predicted objects are categorized as TP or FP based on KITTI's conventions [7]. To evaluate the performance of a specific score, we could simply apply a score threshold: if the score is above the threshold then the corresponding prediction is TP, otherwise it's FP. Then we can evaluate the resulting detection accuracy. However, the resulting accuracy is a function of the threshold. To remove the effect of threshold selection and evaluate the performance of different XC scores more fairly, we compute area under the precision recall curve (AUPR) [19] and area under the receiver operating characteristics curve (AUROC) [3], both are threshold-independent performance measures for binary classification.

In binary classification tasks, typically one class is treated as the "positive" class, whereas the other class is treated as the "negative" class. We may choose to treat either the TP boxes or the FP boxes as the positive class. The AUROC metric treats both classes equally and can reflect the score's (e.g., one of the XC scores) ability in correctly identifying both the positive and negative classes. On the other hand, the AUPR metric puts more emphasis on the score's ability to correctly identify the positive class. For both AURP and AUROC, higher values indicate better performance.

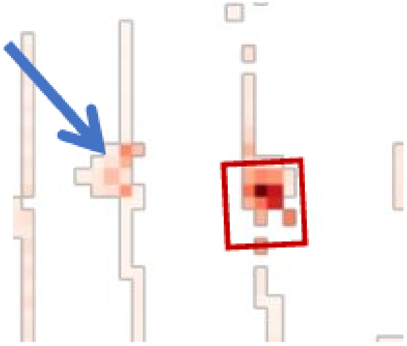


Figure 3: Negative attribution map for a TP pedestrian prediction (the red box). The patch of outlier attributions is pointed out by the arrow.

We evaluate the XC scores on three PointPillars [12] models trained on the KITTI dataset [7] and on one PointPillars model trained on the Waymo dataset [31]. We use OpenPCDet's [33] implementation of PointPillars for our experiments and adapt their default settings for training. The first three models are trained for 80 epochs on the KITTI dataset, with 3712 frames for training and 3769 frames for validation. Due to time and resource constraints, we obtain only one more model trained for 30 epochs on 20% of the Waymo dataset, with 31616 frames for training and 7997 frames for validation. To obtain attribution values, we use Captum [29], a model interpretability library developed for PyTorch [20]. We explore both IG [32] and backprop [28] as explanation methods.

For KITTI, the XC scores are obtained on the predicted objects from the 3769 validation frames. There are on average 67k predicted objects produced by each model. One pixel in the pseudo image encodes point features in a 0.16m $\times 0.16$m $\times 4.00$m pillar. For Waymo, we comptue XC scores for 94k predicted boxes sampled from 800 validation frames. One pixel in the pseudo image represents point features in a 0.33m $\times 0.33$m $\times 6.00$m pillar. For both datasets, we apply $m = 0.2$m to the predicted boxes and apply $a_{thresh} = 0.1$ to the attribution maps.

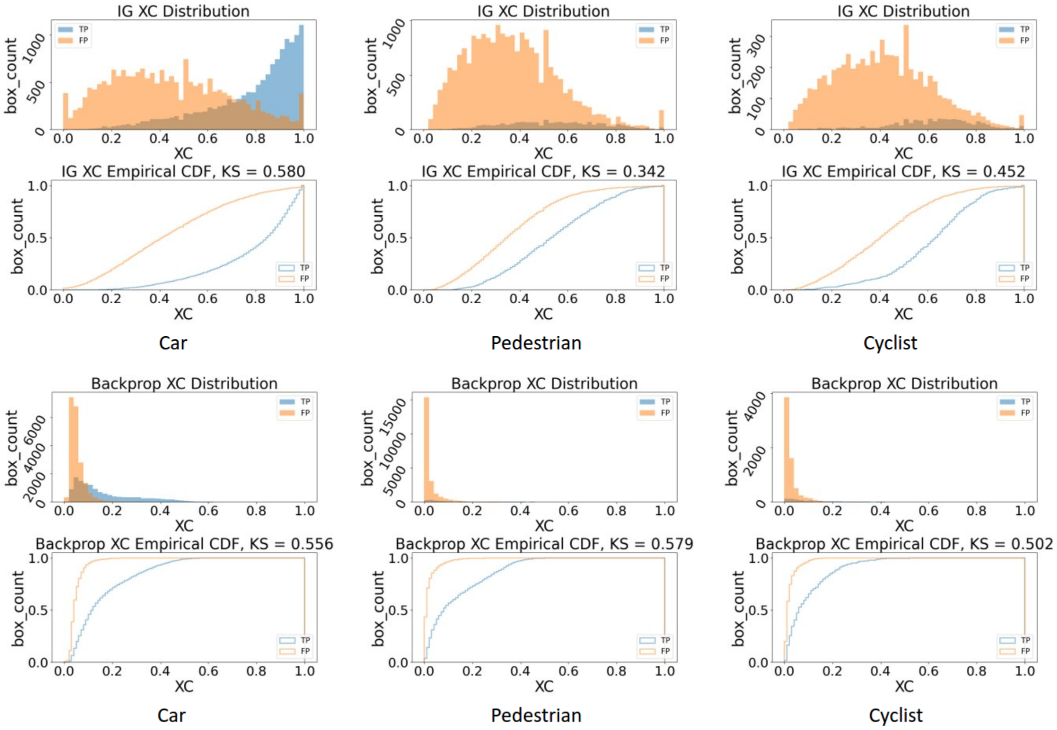


Figure 4: Distribution of XC_c^+ values obtained by IG (upper row) and backprop (lower row) attributions for TP and FP predicted boxes in each of the object class of the KITTI validation set. "KS" means the Kolmogorov-Smirnov statistic [4].

5.2 Distribution of XC values

The distribution of XC_c^+ values obtained from one of the models trained on KITTI [7] is shown in Fig. 4. Both the histograms and the empirical cumulative distribution function plots demonstrate that the XC values of TP instances tend to be greater than those of the FP instances. Another notable observation is that XC values derived from IG are well-dispersed within the range of $[0, 1]$ for both the TP and FP instances, whereas XC values derived from backprop are mostly below 0.5, with almost all FP instances having XC values below 0.2.

5.3 Using XC to identify TP and FP predictions

The results for TP vs. FP box classification using the XC scores on the KITTI dataset [7] are shown in Table 1. Each specific metric in this table is averaged over three models. The four XC scores derived from IG and backprop are evaluated, along with four other box-wise features (random guess, distance of the predicted box to LiDAR sensor, the number of points inside the predicted box, and the highest class score for the predicted box) serving as baselines for comparison. Each of the aforementioned features are evaluated by three metrics on different object classes, making up twelve metrics in total. Note that the object classes in Table 1 (as well as those in other tables in later sections) represent the predicted labels, not the ground truth labels. Also, in OpenPCDet's [33] implementation of PointPillars [12], the class scores are not softmax scores. Rather, they are class-wise sigmoid scores: a pair of "class vs. not class" scores for each object class.

When computing AUROC and AUPR, the TP boxes are treated as the positive instances, but AUPR_op refers to the AUPR value obtained by treating the FP predicted boxes as positive instances. In essence, AUPR reflects TP detection performance, whereas AUPR_op reflects FP detection performance. The expected AUROC value for a random score is 0.5, and the expected AUPR value is the proportion of the positive instances.

Table 1: Comparison of the XC scores' ability to classify TP and FP predictions for different object types in the KITTI dataset. The subscripts "IG" and "B" indicate whether the corresponding XC score is derived from IG or backprop attributions. For each evaluation metric, the XC scores performing worse than number of points are highlighted by underscore and the best performing feature other than the top class score is highlighted in **bold**.

Metrics	Random	Distance	Points	$XC\_s_{IG}^+$	$XC\_c_{IG}^+$	$XC\_s_{IG}^-$	$XC\_c_{IG}^-$	$XC\_s_B^+$	$XC\_c_B^+$	$XC\_s_B^-$	$XC\_c_B^-$	Top Class Score
AUROC												
All	0.5	0.669	0.720	0.843	0.868	0.827	0.869	0.823	0.903	0.822	**0.908**	0.971
Car	0.5	0.770	0.779	0.837	0.857	0.822	0.857	0.708	0.861	0.698	**0.869**	0.964
Pedestrian	0.5	0.779	0.832	0.648	0.699	0.671	0.754	0.864	0.888	0.882	**0.894**	0.958
Cyclist	0.5	0.635	0.780	0.810	0.797	0.806	0.824	0.808	0.843	0.819	**0.855**	0.965
AUPR												
All	0.232	0.372	0.464	0.585	0.653	0.551	0.635	0.597	0.786	0.577	**0.794**	0.926
Car	0.391	0.636	0.666	0.713	0.749	0.697	0.747	0.621	0.830	0.597	**0.836**	0.948
Pedestrian	0.071	0.215	0.250	0.116	0.153	0.136	0.190	0.518	0.557	0.541	**0.572**	0.759
Cyclist	0.083	0.167	0.203	0.225	0.237	0.239	0.270	0.425	0.534	0.450	**0.554**	0.829
AUPR_op												
All	0.768	0.859	0.878	0.941	0.950	0.937	0.952	0.936	0.965	0.937	**0.967**	0.989
Car	0.609	0.829	0.839	0.894	0.903	0.885	**0.905**	0.761	0.888	0.761	0.897	0.974
Pedestrian	0.929	0.976	0.984	0.959	0.967	0.962	0.973	0.986	0.989	0.988	**0.990**	0.996
Cyclist	0.917	0.939	0.976	0.977	0.975	0.978	0.980	0.975	0.979	0.977	**0.982**	0.992

Referring to Table 1, it is clear that the top class score beats all other features in every evaluation metric. However, the improvement brought by XC is certainly non-trivial. One can observe that for all features evaluated, the second best performing feature is most often $XC\_c_B^-$. Although distance to sensor and number of points in predicted box beat the XC scores generated by IG on the three pedestrian class metrics, they are unable to beat any of the XC scores generated by backprop on the pedestrian class.

A very notable case is the AUPR for pedestrian predictions: $XC\_c_B^-$ achieves 129% improvement compared to number of points and 706% improvement compare to random guess. The other three XC scores derived from backprop also achieve more than 100% on AUPR for pedestrian compared to the number of points. Improvements of similar magnitude are also achieved by the backprop XC scores on AUPR for the cyclist class. These observations indicate that the backprop XC scores are much better at correctly identifying the TP predictions for the pedestrian and cyclist classes than simple heuristics such as number of points. Another interesting observation is that the XC scores derived by counting usually outperforms those derived by summing. Such improvement might be due to the outlier attenuation effect mentioned in the second last paragraph of Section 4.

Table 2: Comparison of the XC scores' ability to classify TP and FP predictions for different object types in the Waymo dataset. For each evaluation metric, the XC scores performing worse than number of points are highlighted by underscore and the best performing feature other than the top class score is highlighted in **bold**.

Metrics	Random	Distance	Points	$XC\_s_{IG}^{+}$	$XC\_c_{IG}^{+}$	$XC\_s_{IG}^{-}$	$XC\_c_{IG}^{-}$	$XC\_s_{B}^{+}$	$XC\_c_{B}^{+}$	$XC\_s_{B}^{-}$	$XC\_c_{B}^{-}$	Top Class Score
AUROC												
All	0.5	0.609	0.701	0.714	0.758	0.738	0.766	0.729	0.788	0.721	**0.793**	0.965
Vehicle	0.5	0.703	0.809	0.799	0.821	0.806	0.823	0.754	**0.860**	0.725	**0.860**	0.982
Pedestrian	0.5	0.528	0.614	0.529	0.529	0.529	0.545	0.605	0.638	0.646	**0.651**	0.927
Cyclist	0.5	0.627	0.738	0.766	0.725	0.673	0.689	0.794	0.823	0.799	**0.823**	0.979
AUPR												
All	0.276	0.343	0.476	0.460	0.535	0.483	0.540	0.569	**0.700**	0.542	0.699	0.936
Vehicle	0.377	0.595	0.721	0.626	0.659	0.624	0.654	0.678	**0.827**	0.648	0.824	0.973
Pedestrian	0.176	0.121	0.147	0.183	0.184	0.181	0.192	0.284	0.323	0.326	**0.340**	0.829
Cyclist	0.051	0.072	0.113	0.111	0.096	0.078	0.082	0.279	0.377	0.283	**0.394**	0.791
AUPR_op												
All	0.724	0.811	0.862	0.863	0.881	0.874	**0.884**	0.850	0.876	0.857	0.882	0.983
Vehicle	0.623	0.784	0.873	0.878	0.891	0.883	0.892	0.809	0.891	0.786	**0.893**	0.987
Pedestrian	0.824	0.897	**0.925**	0.841	0.839	0.842	0.846	0.866	0.882	0.889	0.888	0.979
Cyclist	0.950	0.967	0.984	0.984	0.980	0.976	0.978	0.984	0.986	0.985	**0.986**	0.999

To ensure that the advantages offered by XC are not specific to the KITTI dataset [7], we also present results from Waymo dataset [31] in Table 2. Again, for most evaluation metrics, one of the XC scores is the second best performing feature besides the top class score. In addition, the XC scores obtained from backprop often show 100% or more improvement on AUPR for pedestrian and cyclist predictions compared to the number of points. And again, $XC\_c_{B}^{-}$ is the best performing feature in most cases. Hence, we believe that the advantages of XC are not limited to the KITTI dataset only.

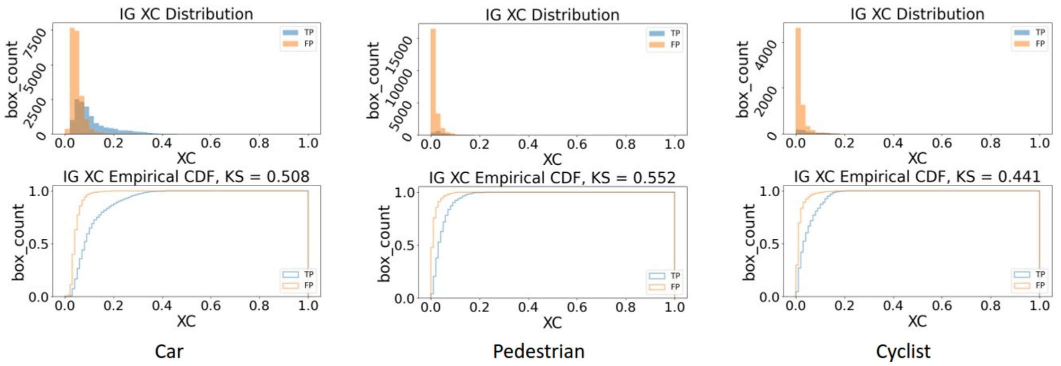
5.4 Why backprop outperforms IG?

IG [32] is designed to reflect feature importance more precisely than other simpler XAI methods such as backprop. Thus, it would be interesting to know why IG-based XC scores underperform backprop-based XC scores in the task of classifying TP vs. FP predictions, especially for the pedestrian and cyclist predictions (see AUPR in Table 1 and Table 2). We suspect that this is due to the difference in XC distribution. In Fig. 4, we present the KS statistic [4] between the distributions of XC values in the TP and FP instances of each object class. The KS statistic is a measure for goodness of fit between two distributions: greater value indicates greater difference between the two distributions. Note that the backprop-based XC scores are able to produce much KS statistic between TP and FP distributions in the pedestrian and cyclist class than the IG-based scores.

Table 3: Average XC performance on distinguishing TP vs. FP predictions for the KITTI dataset. The highest value in each row is highlighted in **bold**.

Metrics	XC_{IG}	Modified XC_{IG}	XC_{B}
AUROC			
All	0.852	0.861	**0.864**
Vehicle	**0.843**	0.795	0.784
Pedestrian	0.693	0.848	**0.882**
Cyclist	0.809	0.812	**0.831**
AUPR			
All	0.606	0.657	**0.689**
Vehicle	**0.726**	0.707	0.721
Pedestrian	0.149	0.321	**0.547**
Cyclist	0.243	0.282	**0.490**
AUPR_op			
All	0.945	**0.952**	0.951
Vehicle	**0.897**	0.848	0.827
Pedestrian	0.965	0.986	**0.988**
Cyclist	0.977	**0.978**	**0.978**

To alter the distribution of IG-based XC scores, we remove the last step in computing IG attributions. As mentioned before in Section 2, IG zeros out attributions for input features with zero value. This is often achieved by multiplying the computed attributions at each input location i by $(x_i - x_i')$, where x is the input and x' is the baseline, which is by default set to zero. We remove this process and re-calculate IG-derived XC scores on the KITTI dataset [7]. Then we evaluate the new XC scores on the binary classification task for TP vs. FP predictions. The average results of all four IG-derived XC scores, all four backprop-derived XC scores (these are averaged over the values presented in Table 1), and of the four modified IG-derived XC scores are presented in Table 3. Note that the results are also averaged over all three models trained on KITTI. The modified IG XC scores resulted in 115% improvement in AUPR for pedestrian predictions and in 16% improvement in AUPR for cyclist predictions compared to the original IG XC scores. Improvement can also be observed in AUROC and AUPR_op for the pedestrian and cyclist classes.



Car

Pedestrian

Cyclist

Figure 5: Distribution of XC_c^{+} values obtained by modified IG (without multiplying attributions by input) attributions for TP and FP predicted boxes in the KITTI validation set.

The improvements on pedestrian and cyclist objects also coincide with a shift in the distribution of XC values. As shown in Fig. 5, the distribution of XC values now appears very similar to that of backprop-based XC values, but very different from that of IG-based XC values. By making feature attributions proportional to input feature magnitude, IG is able to generate attribution maps that capture salient features in the input, which is one reason why IG attribution maps makes more sense to a human observer compared to a blurry backprop attribution map (interested readers may visit Sundararajan *et al.* [32] for more attribution map examples). As a result, IG zeros out most of the attributions outside of the bounding box (because the space outside of object bounding boxes are mostly empty, leading to zero input feature values at such locations), and the remaining attributions are mostly concentrated within the bounding box or at its close proximity (see Fig. 2). Thus, IG is unable to produce mostly very low (< 0.1) XC values for the FP predictions, leading to significant overlap in the values for TP prediction XC scores and FP predictions XC scores, making classification using XC scores difficult.

These observations echo with Erion *et al.*'s [6] claim that IG's choice of a zero-valued baseline is problematic. Take an image of a digit for example, if the background is white but the digit itself is black (i.e., zero), then the zero-valued pixels in fact contains the key features of this image and should not get zero attributions. Similarly for point cloud inputs, not just the presence of points, but also the absence of points in certain locations can help the model classify the object. For instance, pedestrian objects are usually filled with points, whereas car objects have points on its boundaries but are mostly hollow in the middle.

Note that even without multiplying by input values, IG-derived XC scores still cannot beat backprop-derived XC scores on the pedestrian and cyclist classes (see Table 3). Hence, a more expensive method such as IG may produce more visually appealing attribution map, but a less expensive method may have more potential when used quantitatively.

5.5 Combining XC scores with the top class score

In Section 5.3 we demonstrate that the top class score is the best performing box-wise feature in classifying TP vs. FP predictions. In this subsection, we aim to improve its performance by combining it with XC scores generated from backprop attributions. We train classifiers for different object classes in KITTI [7] and perform an ablation study on the box-wise features.

To conduct the ablation study, we build a new dataset D_f from five features (the top classs score and the four XC scores computed based on backprop attributions) for each predicted box produced by one PointPillars [12] model on the KITTI validation set [7]. Since we have three models trained on KITTI, we obtain three D_f datasets. For each D_f, we first group the samples by the predicted object label, then by the number of LiDAR points: those with less than 100 points forms one set, the rest forms another set. Thus, from one D_f, we generate six smaller datasets d_f, two for each of the three object classes.

We then build a 2-layer multilayer perceptron (MLP) with PyTorch [20] as a classifier to be trained on d_f. Layer 1 is of size $(d \times 3)$ and layer 2 is of size (3×1), where d represents the number of input features per instance. ReLU activation is applied after the first layer, and the sigmoid function is applied after the second layer to obtain an output score. We train the MLP to classify a predicted box as TP or FP, using binary cross entropy as the loss function. All input features are normalized based on the following equation prior to being fed into the MLP: $z = (x - \mu)/s$, where x is the original feature value, μ is feature mean value, and s is the standard deviation for that feature. We use the Adam optimizer [11] with learning rate set to 0.001.

For each experiment, we first shuffle d_f and then augment it by duplicating the instances four times. Next we apply 5-fold cross validation to the augmented d_f, obtaining 5 different 80%/20% train/validation splits. For the training instances, we also add a small uniformly distributed noise $U(-0.05, +0.05)$ to each feature to help the MLP generalize better. For each different split, we train the MLP for 12 epochs with batch size = 16 and record three evaluation metrics (AUROC, AUPR, and AUPR_op) of the output score on the validation instances. We repeat the 5-fold cross validation 5 times, obtaining $5 \times 5 = 25$ different values for each of the metrics, and record the average. Note that the above process is repeated for each d_f in the three D_f we have, each evaluation metric is averaged over the three D_f and shown in Table 4.

In Table 4, when the top class score is the only input feature, we evaluate its performance directly and use it as baseline for comparison; when more than one features are used, we evaluate the performance of the MLP output score. The most notable observation is that for predictions containing less than 100 points, combining the XC scores with top class score can often result in better performance in distinguishing TP vs. FP predictions, especially among the pedestrian predictions. The AUPR for pedestrian increased by (0.540 - 0.492) / 0.492 = 9.8% after combining the 4 XC scores with top class score. The improvement in AUROC for pedestrian predictions and in AUPR for cyclist predictions also exceed 1%. Among the predictions with more than 100 points, the benefit of incorporating the XC scores is less observable. Note that the top class score alone is already performing very well for these predictions with more points. This might be why it is more difficult to get additional benefit from the XC scores on these predictions.

Table 4: Ablation study on the features used to help classify TP vs. FP predictions on KITTI.

Object Class	Features Used					Points < 100			Points >= 100		
	Top class score	XC_c^-	XC_c^+	XC_s^-	XC_s^+	AUROC	AUPR	AUPR_op	AUROC	AUPR	AUPR_op
Car	✓					0.958	0.926	0.977	0.956	0.980	0.914
	✓	✓				0.958	0.927	0.977	0.950	0.978	0.903
	✓	✓	✓	✓	✓	0.960	0.927	0.979	0.957	0.980	0.918
Pedestrian	✓					0.932	0.492	0.997	0.969	0.911	0.989
	✓	✓				0.938	0.527	0.997	0.973	0.919	0.991
	✓	✓	✓	✓	✓	0.944	0.540	0.997	0.973	0.920	0.992
Cyclist	✓					0.958	0.765	0.996	0.982	0.947	0.995
	✓	✓				0.958	0.777	0.996	0.972	0.931	0.993
	✓	✓	✓	✓	✓	0.958	0.779	0.996	0.980	0.946	0.994

6 Conclusion and future work

To use the explanations quantitatively, we proposed four XC scores to measure the concentration of the attribution values generated for individual predictions. Applying the four XC scores on the task of classifying TP vs. FP predictions led to over 100% improvement in AUPR on the pedestrian class on both the KITTI and the Waymo datasets compared to simple heuristics such as distance to sensor and number of LiDAR points in bounding box. Although the XC scores alone could not outperform class score in the TP vs. FP classification task, combining class score with the XC scores using an MLP led to notable improvement compared to using class score alone on the pedestrian predictions. Thus, it is worthwhile to explore the XC scores further for more use cases such as using it in loss functions to improve model performance, or use it as a tool for adversarial or out-of-distribution sample detection.

We also discovered that the XC metrics derived from backprop attributions often outperform those derived from IG attributions on TP vs. FP classification for the pedestrian and cyclist objects. This indicates that explanations that are more understandable to a human observer, such as IG, may not offer superior value when analyzed quantitatively and researchers should not discard simpler explanation methods when exploring quantitative use cases for XAI.

Acknowledgments and Disclosure of Funding

We would like to thank Huawei Noah's Ark Lab, Canada, for sponsoring our work. We also thank our colleagues Sean Sedwards, Rick Salay, Chengjie Huang, Harry Nguyen, and Matthew Pitropov for their support and feedback.

References

[1] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 4

[2] Weijie Chen, Yilu Guo, Shicai Yang, Zhaoyang Li, Zhenxin Ma, Binbin Chen, Long Zhao, Di Xie, Shiliang Pu, and Yueting Zhuang. Box re-ranking: Unsupervised false positive suppression for domain adaptive pedestrian detection. *arXiv preprint arXiv:2102.00595*, 2021. 3

[3] Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *ICML*, 2006. 4

[4] Yadolah Dodge. *The Concise Encyclopedia of Statistics*. Springer, 2008. 5, 7

[5] Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. Learning credible deep neural networks with rationale regularization. *arXiv preprint arXiv:1908.05601*, 2019. 2

[6] Gabriel Erion, Joseph D. Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Improving performance of deep learning models with axiomatic attribution priors and expected gradients. *arXiv preprint arXiv:1906.10670*, 2020. 2, 8

[7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 4, 5, 6, 7, 8

[8] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. 1

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 1

[10] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, Feb 2017. 3

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2017. 9

[12] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, June 2019. 1, 2, 3, 5, 6, 8

[13] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, Dec 2018. 3

[14] Xiao-Hui Li, Caleb Chen Cao, Yuhan Shi, Wei Bai, Han Gao, Luyu Qiu, Cong Wang, Yuanyuan Gao, Shenjia Zhang, Xue Xue, and Lei Chen. A survey of data-driven and knowledge-aware explainable AI. *IEEE Transactions on Knowledge and Data Engineering*, 2020. 1

[15] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, April 2018. 3

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *ECCV*, volume 9905, pages 21–37. Springer, 2016. 1, 3

[17] Julia Lust and Alexandru P. Condurache. Gran: An efficient gradient-norm based detector for adversarial and misclassified examples. In *ESANN*, Oct 2020. 3

[18] Xingjun Ma, Bo Li, Yisen Wang, Sarah Erfani, Sudanthi Wijewickrema, Michael Houle, Grant Schoenebeck, Dawn Song, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *ICLR*, April 2018. 3

[19] Chris Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. 4

[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*, pages 8024–8035. 2019. 5, 8

[21] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 1

[22] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28, 2015. 1

[23] Laura Rieger, Chandan Singh, W. James Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. *arXiv preprint arXiv:1909.13584*, 2020. 2

[24] Andrew Slavin Ross, Michael C. Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2662–2670, 2017. 2

[25] 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 *ICCV*, Oct 2017. 1

[26] Rakshith Shetty, Bernt Schiele, and Mario Fritz. Not using the car to see the sidewalk — quantifying and controlling the effects of context in classification and segmentation. *arXiv preprint arXiv:1812.06707*, 2018. 4

[27] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3d object proposal generation and detection from point cloud. In *CVPR*, June 2019. 1, 4

[28] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2014. 1, 2, 3, 5

[29] Facebook Open Source. Captum: Model interpretability for pytorch. `https://github.com/pytorch/captum`, 2020. 5

[30] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2015. 1, 2

[31] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Sheng Zhao, Shuyang Cheng, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. *CoRR*, abs/1912.04838, 2020. 5, 7

[32] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *ICML*, 2017. 1, 2, 3, 5, 7, 8

[33] OpenPCDet Development Team. OpenPCDet: An open-source toolbox for 3d object detection from point clouds. `https://github.com/open-mmlab/OpenPCDet`, 2020. 5, 6

[34] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. In *NDSS*, Feb 2018. 3

[35] Yan Yan, Yuxin Mao, and Bo Li. Second: Sparsely embedded convolutional detection. In *Sensors*, 2018. 2

[36] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *arXiv preprint arXiv:1311.2901*, 2013. 1, 2

[37] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3d object detection. In *CVPR*, June 2018. 1, 2