Revealing Hidden Context Bias in Segmentation and Object Detection through Concept-specific Explanations — Supplementary Material —

Maximilian Dreyer[1], Reduan Achtibat[1], Thomas Wiegand[1,2,3],
Wojciech Samek[1,2,3,†], Sebastian Lapuschkin[1,†]
[1] Fraunhofer Heinrich-Hertz-Institute, [2] Technical University of Berlin,
[3] BIFOLD – Berlin Institute for the Foundations of Learning and Data
[†]corresponding authors: {wojciech.samek | sebastian.lapuschkin}@hhi.fraunhofer.de

A. Appendix

In the appendix, we provide additional information and results complementing the main manuscript. Beginning with Section A.1, details about the models, datasets, as well as technical minutae on the computation of Layer-wise Relevance Propagation (LRP) attributions are presented. This is followed by Sections A.4 and A.5, where more results on the evaluation of explanations are given regarding faithfulness and complexity, respectively. Finally, in Section A.6, additional results for the measurement of context scores of concepts are shown and discussed.

A.1. Technical Details

In the following, the models and datasets are presented, as well as details about the used LRP rules and their implementation.

Models and Datasets The UNet [21] model is implemented using the `segmentation_models_pytorch` framework [14] and consists of a VGG-13 [25] encoder with BatchNorm layers, and weights pre-trained on ImageNet [22]. The model is trained for 100 epochs with a batch size of 40 on the CityScapes [7] dataset with an initial learning rate of 10^{-3} using the Adam optimizer. The learning rate is reduced to 10^{-4} after 50 and $5 \cdot 10^{-5}$ after 75 epochs. Images are resized to a height of 256 and width of 512 pixels and normalized using a mean of (0.485, 0.456, 0.406) and standard deviation of (0.229, 0.224, 0.225) over the three RGB color dimensions[1]. During training, we further apply random cropping to 256×256 pixels, a random horizontal flip (50 % probability), brightness, saturation, hue and contrast perturbation (20 % strength, 50 % probability), as well as adding random Gaussian noise (zero mean, variance between 10 and 50, 50 % probability). The final training results in a pixel-wise accuracy of 74.7 % and a mean intersection over union score of 35.1 %.

The DeepLabV3+ [6] model is based on the PyTorch implementation [10] with a ResNet-50 [12] backbone and pre-trained on the Pascal VOC 2012 dataset.

The YOLOv6 model is based on the PyTorch implementation of the authors [16] and corresponds to the "small" variant named YOLOv6s trained on the MS COCO 2017 dataset [17].

Similarly, the YOLOv5 model is based on the PyTorch implementation of [9] and corresponds to the "medium" variant named YOLOv5m trained on MS COCO 2017.

LRP Canonization Canonization procedures restructure a model into a functionally equivalent architecture to which established attribution LRP-rules can be applied [20]. Canonization efforts typically concentrate on replacing the BatchNorm layer [11, 13] or handling skip connections [5] as in the ResNet architecture [12].

Regarding canonization, BatchNorm layers are merged into the preceding linear layer for all models. This way, significant improvements in terms of explanations can be achieved [19].

Further, in the `Bottleneck` modules of DeepLabV3+ (based on the ResNet) and YOLOv5, as well as the `RepVGGBlock` module (based on [8]) of the YOLOv6, activations of different parallel operations are merged using a sum operation. Here, a `Sum` module is inserted to handle the summation, and to which a `Norm`-rule can be applied to normalize relevances during the relevance backpropagation step, as available in [4] and discussed in [5].

The authors of [8] also present a canonization procedure for the `RepVGGBlock` module of YOLOv6, which merges all linear operations of the module into a single convolutional layer, which we apply to layer

[1]as proposed at https://pytorch.org/vision/stable/models.html

`model.backbone.stem`.

LRP Rules Multiple rules for LRP were defined in the literature [15]. The LRP-ε rule is based on the basic decomposition rule (see Equation (1)) and is given as

$$R_{i\leftarrow j}^{(l,\,l+1)} = \frac{z_{ij}}{z_j + \varepsilon \cdot \text{sign}(z_j)} R_j^{l+1} \tag{A.1}$$

with z_{ij} describing the contribution of neuron i to the activation of neuron j, and aggregated pre-activations $z_j = \sum_i z_{ij}$. The LRP-ε rule ensures that each neuron receives the attribution value (fraction), that it contributed to the output. The added parameter $\varepsilon \in \mathbb{R}^+$ stabilizes the division of the denominator, and "dampens" contradicting contributions for $|z_{ij}| \gg |z_j|$, as discussed in [18]. However, ε is usually set to a small value, *e.g.* 10^{-6} in [4], resulting in noisy attributions for deep networks due to large relevance values from contradicting contributions $|z_{ij}| \gg |z_j|$. In classification networks, the LRP-ε rule is often chosen for the dense layers, whereas increasingly stabilizing rules (presented in the following) are chosen for convolutional layers. In the YOLOv6 model implementation, single BatchNorm layers with label `rbr_identity` in `RepVGGBlock` modules exist without neighbouring linear layers. We apply the LRP-ε rule for these individual BatchNorm layers.

Alternatively, the LRP-γ rule is defined as

$$R_{i\leftarrow j}^{(l,\,l+1)} = \frac{z_{ij} + \gamma z_{ij}^+}{z_j + \gamma \sum_k z_{kj}^+} R_j^{l+1} \tag{A.2}$$

with positive parameter $\gamma \in \mathbb{R}^+$ and $(\cdot)^+ = \max(0, \cdot)$. The function of the LRP-γ rule is to favor positive contributions and at the same limit the unbounded growth potential of positive and negative relevance in the backpropagation step. Note, that the term $\gamma \sum_i z_{ij}^+$ (for strictly positive z_j) effectively corresponds to the ε in Equation (1), but scaled to the magnitude of contributions z_{ij}^+. The LRP-γ rule has thus been found to be effective in reducing noisy attributions, and the amount of "filtering" can be controlled by one parameter γ, which is by default set to $\gamma = 0.25$ in the zennit framework [4]. However, if $z_j < 0$, the denominator can become numerically unstable if $|z_j| \approx \gamma \sum_i z_{ij}^+$. Therefore, we use the generalized version presented in [3], and either favor positive contributions for $z_j > 0$ or negative contributions for $z_j < 0$, depending on the sign of z_j:

$$R_{i\leftarrow j}^{(l,\,l+1)} = \begin{cases} \frac{z_{ij} + \gamma z_{ij}^+}{z_j + \gamma \sum_k z_{kj}^+} R_j^{l+1} & \text{if } z_j > 0 \\ \frac{z_{ij} + \gamma z_{ij}^-}{z_j + \gamma \sum_k z_{kj}^-} R_j^{l+1} & \text{else} \end{cases} . \tag{A.3}$$

The LRP-z^+ rule represents a third rule, defined as

$$R_{i\leftarrow j}^{(l,\,l+1)} = \frac{z_{ij}^+}{\sum_k z_{kj}^+} R_j^{l+1} \tag{A.4}$$

by only taking into account positive contributions z_{ij}^+. The LRP-z^+ rule can be seen as the most stable or least noisy attribution method, representing Equation A.2 with $\gamma \to \infty$. To receive explanations with high human interpretability and low amounts of noise, explanations are thus computed with the LRP-z^+ rule applied to all convolutional layers of the models. For the YOLOv6 an exception is made for visual examples, as the LRP-γ rule also provides stable heatmaps, showing higher faithfulness at the same time, as discussed in 4.1.

Finally, it is established practice to apply the LRP-$\flat$ rule to the first layer [15, 18] in order for the attributions to become invariant against normalizations applied in inpu space and making them more human-readable by minimally reducing heatmap fidelity. The LRP-$\flat$ rule is hereby defined as

$$R_{i\leftarrow j}^{(l,\,l+1)} = \frac{1}{\sum_k 1} R_j^{l+1} . \tag{A.5}$$

Using the LRP-$\flat$ rule, the relevance of upper-level neuron j is equally distributed to all connected lower-level neurons disregarding any influence of learned weights or input features. For all models, the LRP-$\flat$ rule is applied to the first convolutional layer to smooth the attribution map in input space and yield robust concept localization.

Software CRP for Localization Models (L-CRP) as an extension to Concept Relevance Propagation (CRP) is implemented based on the open-source CRP framework `zennit-crp`[2] [1] toolbox for PyTorch and LRP framework `zennit` [4].

A.2. Explanation Examples Using L-CRP

In the following, two additional L-CRP explanation examples are shown, which extend the LRP heatmaps shown in Figure 2.

The first example shown in Figure A.1 corresponds to the sheep detection of the YOLOv5 model in Figure 2. Here, the traditional LRP heatmap indicates that parts of the sheep are relevant, such as the head or the fur part in the center of the bounding box. By applying L-CRP to layer `layer4.0.conv3`, we achieve an understanding of what exactly is the model using in terms of concepts here. By analyzing the top-2 most relevant concepts, we learn that the model perceives, *e.g.*, the white fur pattern of the sheep (concept 450) or the frontal face with ears protruding on both sides (concept 173).

The second example shown in Figure A.2 corresponds to the bus segmentation of the DeepLabV3+ model in Figure 2. Here, the traditional LRP heatmap indicates that all

[2]Available at the GitHub repository https://github.com/rachtibat/zennit-crp.

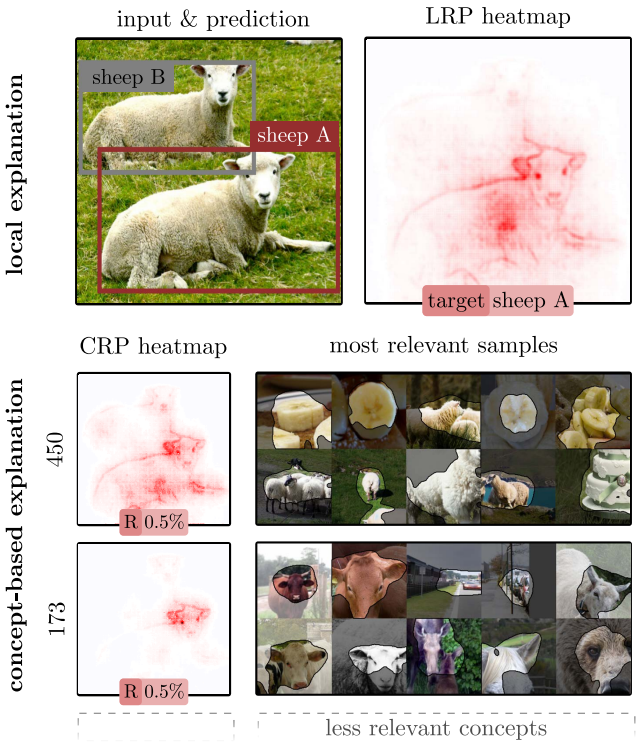


Figure A.1. Concept-based explanation with L-CRP for object detection using the YOLOv5 model. (*Top*): LRP heatmap for detection of a sheep. The heatmap marks the center of the body as well as head. (*Bottom*): Inspecting the two most relevance concepts in layer `7.conv` using L-CRP shows, that the model perceives the bright fur texture (concept 450) as well as the frontal face with ears protruding to either side (concept 173).

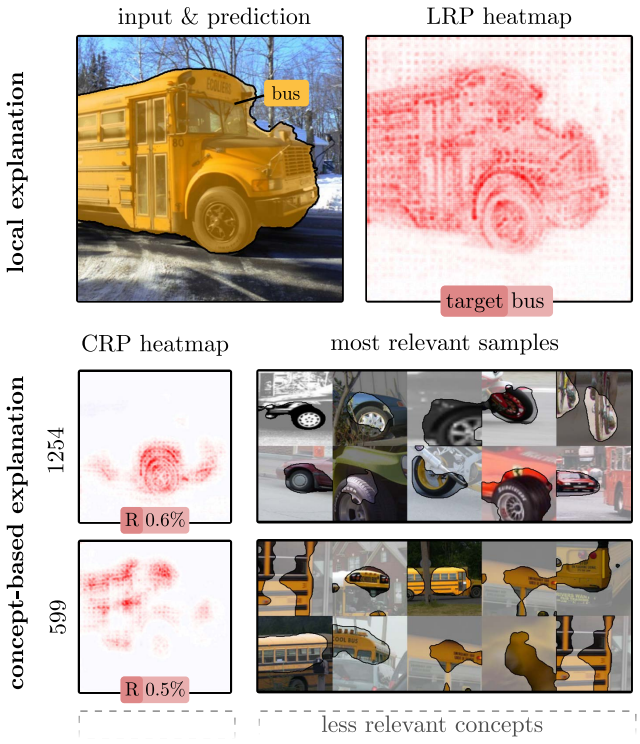


Figure A.2. Concept-based explanation with L-CRP for semantic segmentation using the DeepLabV3+ model. (*Top*): LRP heatmap for segmentation of a bus. The heatmap strongly resembles the segmentation output and marks the whole bus. (*Bottom*): Inspecting the two most relevance concepts in layer `layer4.0.conv3` using L-CRP shows, that the model perceives the bus wheels (concept 1254) as well as the typical school bus color combination of yellow and black (concept 599).

parts of the bus are relevant, strongly resembling the predicted segmentation itself. By applying L-CRP to layer `layer4.0.conv3`, we achieve an understanding of what exactly is the model using in terms of concepts. Investigating the top-2 most relevant concepts, shows that the model perceives, *e.g.*, the wheels of the bus (concept 1254) or the school bus color combination of yellow with black (concept 599).

A.3. Ablation Study: Relevance Initialization

As discussed in Section 3.3, for the explanation of segmentation models, the high dimensionality of the output tensor allows for different relevance initilization schemes. In literature, the predominantly used approach is to simply explain the output logits as they are [2]. However, in principle, the softmax probabilities or even a uniform initialization with ones is possible. In order to achieve a better understanding, we compare in the following different initializations in terms of the faithfulness evaluation metric applied in Section 4.1.

Specifically, for softmax initialization, we modify Equation (4) to

$$R^L_{(p,q,c)}(\mathbf{x}|\theta) = \delta_{cy}\sigma_{(p,q,c)}\left(f(\mathbf{x})\right)\mathbf{1}_{(p,q)}(\mathbf{x}|y) \qquad \text{(A.6)}$$

with softmax function σ applied over the class dimension c. Please note, that the indicator function $\mathbf{1}_{(p,q)}(\mathbf{x}|y)$ only keeps values corresponding to the highest class value. Analogously, uniform initilization is given by

$$R^L_{(p,q,c)}(\mathbf{x}|\theta) = \delta_{cy}\mathbf{1}_{(p,q)}(\mathbf{x}|y). \qquad \text{(A.7)}$$

The faithfulness results for the UNet model trained on CityScapes are shown in Table 1, and indicate, that for most methods, logit initialization leads to the most faithful explantions, followed by softmax initialization.

Changing the initialization ultimately changes the contribution of each output pixel to the resulting input feature attribution score. In an extreme case, an initially negative logit could be initialized as a positive softmax probability or a value of one. Then, for this particular output pixel, all attributions are (falsely) flipped in sign. Further, the contribution of an output pixel with a logit close to zero could

Table 1. Evaluation of faithfulness for different relevance initialization schemes (logits|softmax|uniform) with various concept attribution approaches (higher scores are better)).

	concept flipping	concept insertion
LRP-z$^+$	**4.24**| 4.21 | 4.20	**4.64**| 4.60 | 4.60
LRP-γ	**4.51**| 4.29 | 4.24	**5.16**| 4.94 | 4.88
LRP-ε	**4.54**| 3.73 |3.57	**5.38**| 4.57 | 4.40
GradCAM	**4.41**| 3.51 | 3.41	**5.27**| 4.34 | 4.22
gradient	**4.45**| 3.53 | 3.42	**5.25**| 4.32 | 4.20

be amplified very strongly, when initialized with a value of one. Thus, most methods perform best with logit initialzation followed by softmax initialization (that is a compromise between logit and uniform weighting). In the UNet model, no background class exists, which usually receives a high value when the model is uncertain. Therefore, especially the absence of the background class in the UNet model leads to strong fluctuations of all other output logits. Interestingly, different relevance initialization schemes have no significant effect for the DeepLabV3+ model, since a background class exists for the VOC2012 dataset that absorbs the model uncertainty leading to logits that do not vary strongly in magnitude.

A.4. Faithfulness

In Section 4.1.1, we propose to measure the faithfulness of concept attributions using various approaches based on relevance or activation. In this section, more detailed results for the discussion in Section 4.1.1 are presented. Specifically, we show additional results for the concept flipping and concept insertion experiment for different layers of the UNet, DeepLabV3+, YOLOv5 and YOLOv6 model in Figure A.3. Here, every convolutional layer of the UNet, every second convolutional layer of the DeepLabV3+, and every fourth convolutional layer of the YOLO architectures is analyzed, resulting in approximately 20 layers for each model. Here, the AOC or AUC over the resulting curves are measured for 100 randomly chosen predictions, and the mean values are plotted with the Standard Error of Mean (SEM) in semi-transparent color. The overall AOC or AUC scores (of all layers) are given in parenthesis.

Figure A.3 visualizes that the AOC or AUC scores can vary strongly between layers of an attribution method. This is due to shortcut connections in the network, illustrating that some layers are used more strongly than others. Taking the UNet as an example, it shows that the further the layer in the encoder (up to layer 10), the lower the overall relevance. This indicates, that a large part of features is detected using the lower-level layers.

A.5. Complexity

In Section 4.1.2, we measure and discuss the complexity of concept attribution scores for relevance and activation-based approaches by two means. Firstly, the standard deviation of class concept attribution scores is computed, and secondly, the amount of concepts needed to form 80 % of all attributions is measured. The lower the variation and the smaller the number of concepts to study, the easier it is to understand concept-based explanations.

In the following, we present in Figure A.5 more detailed results of the complexity analysis discussed in Section 4.1.2. Here, the same layers of the models as in previous Section A.4 are investigated. Regarding the analysis, concept attribution scores are collected over all predictions of the test dataset. Thereafter, in order to be interpretable as percentage scores, concept attributions are normalized to an absolute sum of one.

In the first experiment, the mean standard deviation σ_t of concept attribution scores $R_j(\mathbf{x}_i)$ for each class t is measured as

$$\sigma_t = \frac{1}{n_c} \sum_{j}^{n_c} \sqrt{\frac{1}{n_s - 1} \sum_{i}^{n_s} \left(R_j(\mathbf{x}_i) - \bar{R}_j\right)^2} \qquad \text{(A.8)}$$

with mean attribution $\bar{R}_j = \frac{1}{n_s} \sum_i^{n_s} R_j(\mathbf{x}_i)$ over n_s class samples and n_c concepts. To form a final deviation score σ, the mean over all n_t classes is computed as $\sigma = \frac{1}{n_t} \sum_t^{n_t} \sigma_t$.

As shown in Figure A.5, it is visible, that especially in lower-level layers gradient and LRP-ε attributions tend to be noisy, which is expected, as this has also been observed for input-level heatmaps [23]. However, gradient and LRP-ε can also show noisy attributions in higher-level layers, *e.g.*, in layer 20 of the UNet model. This indicates, that the commonly applied heuristic [18] for LRP heatmaps to use more faithful methods such as LRP-ε in higher-level layers, and more stable, but less faithful methods (*e.g.* LRP-z$^+$) in lower-level layer, might not always lead to stable concept-based explanations.

In the second experiment, the fraction of concepts forming 80 % of attributions is calculated. Here, attributions are firstly sorted according to their magnitude in descending order. Thereafter, the cumulative distribution is computed, and the smallest number of channels computed for which the cumulative value is smaller than 80 %. Finally, the fraction is computed via division by the total number of channels. This value depends, *e.g.*, on the feature specificity of concepts. For example, relevances are rather focused on a small set of concepts if they have a very specific function, compared to when all concepts are rather generic, leading to a uniform relevance distribution as all concepts are being used.

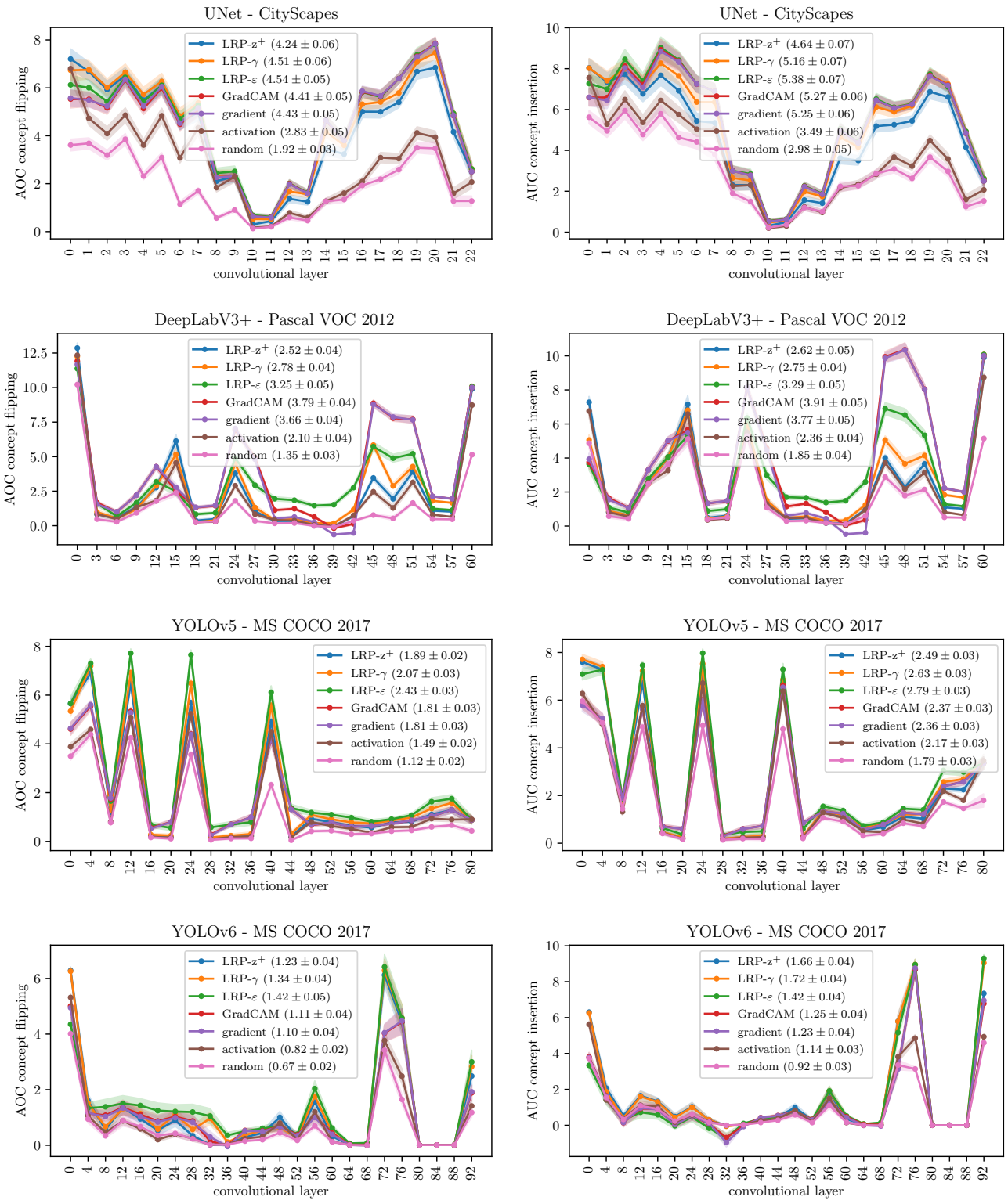


Figure A.3. Evaluating the faithfulness of concept attributions. (*Left*): Concept flipping experiment. (*Right*): Concept insertion experiment. The higher the Area Over Curve (AOC) or Area Under Curve (AUC), the more faithful the attribution method. The faithfulness scores over all layers is given in parenthesis.

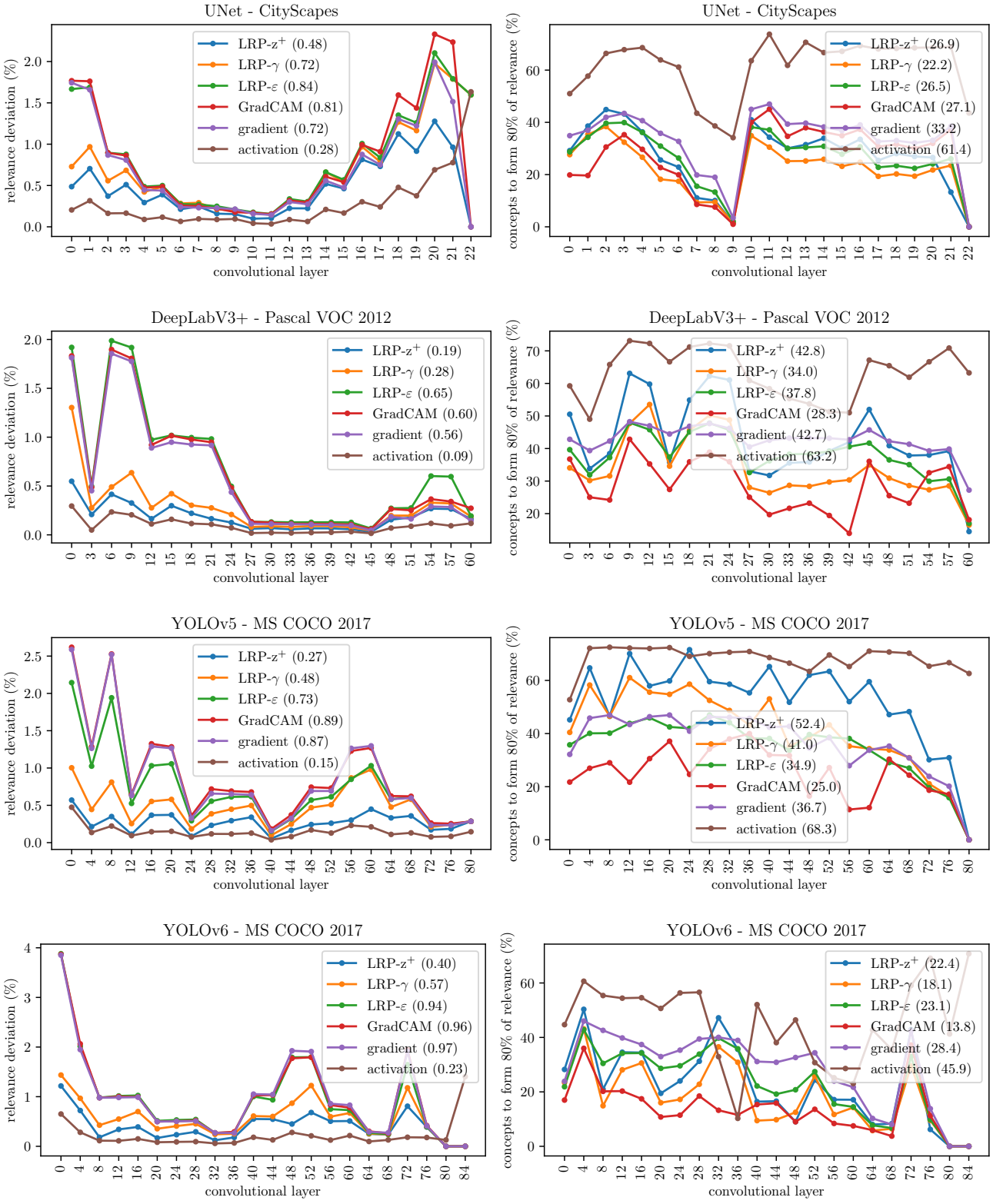


Figure A.4. Measuring the explanation complexity of concept attributions. (*Left*): Mean standard deviation of relevances/attributions per class. The deviation over all layers is given in parentheses. (*Right*): Number of concepts to form 80 % of all attributions. The concept number over all layers is given in parentheses. The lower the scores, the lower the complexity and the easier to read explanations.

A.6. Context

In Section 4.2, we compute context scores C of concepts by measuring the number of concept attributions that correspond to the background of detected objects in the spatial dimension, as described by Equation (7). We firstly evaluate the computed context scores comparing the use of latent activation maps, latent relevance maps (using LRP, GradCAM and Spatial Sensitive Grad-CAM (SS-GradCAM)), as well as input heatmaps (using Guided GradCAM and L-CRP), and thereafter show how to interact with the model and test its reliance on background features. For both parts, we in the following present additional information.

GradCAM Implementation The idea of GradCAM is to upscale latent attribution maps to visualize the importance of single input features [24]. These latent attributions maps are generated by computing a weighted average of the positive latent activation maps $a^+_{(p,q,i)}$ (of width W and height H) with channel dimension i and spatial dimensions p and q. Specifically, latent GradCAM feature maps are given as

$$R^{\text{GradCAM}}_{(p,q)}(\mathbf{x}_j) = \sum_i w_i(\mathbf{x}_j) a^+_{(p,q,i)}(\mathbf{x}_j) \tag{A.9}$$

with weights $w_i(\mathbf{x}_j) = \frac{1}{Z}\sum_{m,n} g_{(m,n,i)}(\mathbf{x}_j)$ representing the mean channel gradient over the spatial dimension with size $Z = H \times W$. To localize a single neuron l, all weights with $i \neq l$ are set to zero.

Extending GradCAM, SS-GradCAM takes into account the spatial gradient information [27], leading to

$$R^{\text{SS-GradCAM}}_{(p,q)}(\mathbf{x}_j) = \sum_i w_i(\mathbf{x}_j) a^+_{(p,q,i)}(\mathbf{x}_j) s_{(p,q,i)}(\mathbf{x}_j) \tag{A.10}$$

with $s_{(p,q,i)}(\mathbf{x}_j) = \frac{|g_{(p,q,i)}(\mathbf{x}_j)|}{\max_{m,n}|g_{(m,n,i)}(\mathbf{x}_j)|}$.

Further, [24] proposes Guided GradCAM achieving more fine-grained feature maps by multiplying the upscaled GradCAM map with input heatmaps computed through Guided Backpropagation [26].

Context Score Evaluation The context scores are evaluated by computing background sensitivity scores S for each concept via Equation (8). The background sensitivity is hereby measured by perturbing the background of objects and tracking the change in concept attribution.

Ideally, we expect that concepts with a high context score C will also result in a high background sensitivity S. Therefore, we assume of faithful context scores to result in a high correlation ρ as well as a small Root Mean Square Deviation (RMSD) value between context and background sensitivity scores. The correlation ρ is calculated as

$$\rho = \sum_i \frac{(C_i - \bar{C})(S_i - \bar{S})}{\sqrt{\sum_j (C_j - \bar{C})^2}\sqrt{\sum_k (S_k - \bar{S})^2}} \tag{A.11}$$

Table 2. Comparing computed context scores with measured background sensitivity. Ideal is a low Root Mean Square Deviation (RMSD) and high correlation.

	RMSD (%)	correlation (%)
L-CRP (ours)	**17.5**	**69.4**
LRP	19.9	66.8
Guided GradCAM ($w_i = 1$)	22.6	65.0
Guided GradCAM	24.4	43.0
SS-GradCAM ($w_i = 1$)	24.9	64.1
SS-GradCAM	24.6	40.1
GradCAM ($w_i = 1$)	40.5	65.1
GradCAM	25.2	24.7
activation	40.5	65.1

with means $\bar{C} = \frac{1}{m}\sum_i C_i$ and $\bar{S} = \frac{1}{m}\sum_i S_i$ over m evaluated concepts. The RMSD is further given as

$$\text{RMSD} = \sqrt{\frac{1}{m}\sum_i (C_i - S_i)^2}\,. \tag{A.12}$$

The final correlation and RMSD values per model shown in Table 2 are computed by taking the mean over three layers.

For a visualization of the resulting similarity between context scores and background sensitivity values, distribution plots are shown in Figure A.5 and Figure A.6, for L-CRP, LRP, activation as well as L-CRP and all GradCAM-based methods, respectively. Here, it becomes apparent, that using latent activation leads to an over-estimation of context scores, as they are often significantly higher than the corresponding background sensitivity values, as, *e.g.*, for the three layers of the UNet shown in Figure A.5 (*1st row*). It can also be seen, that the higher-level layers show the best alignment between C and S values (*3rd column*), as features are more specialized, becoming better to characterize as either background or foreground concepts.

GradCAM Limitations GradCAM effectively simply rescales activation maps, as can be seen in Equation (A.9). The rescaling, however, depends on the unmodified gradient, which is very faithful, but noisy for Deep Neural Networks (DNNs). Therefore, although a concept might be used, the mean channel gradient (described by weight w_i in Equation (A.9)) can become negative, leading to a flip of sign in activations from positive to negative, and thus a context score of zero. Pleas note, such sign flips appear for all GradCAM variants.

For completeness, we evaluate context scores also with weights $w_i = 1$ set to one, in order to prevent the sign flips. As can be seen in Table 2, the correlation generally improves for CAM-based methods, if weights are set to $w_i = 1$. For GradCAM, the mean RMSD increases

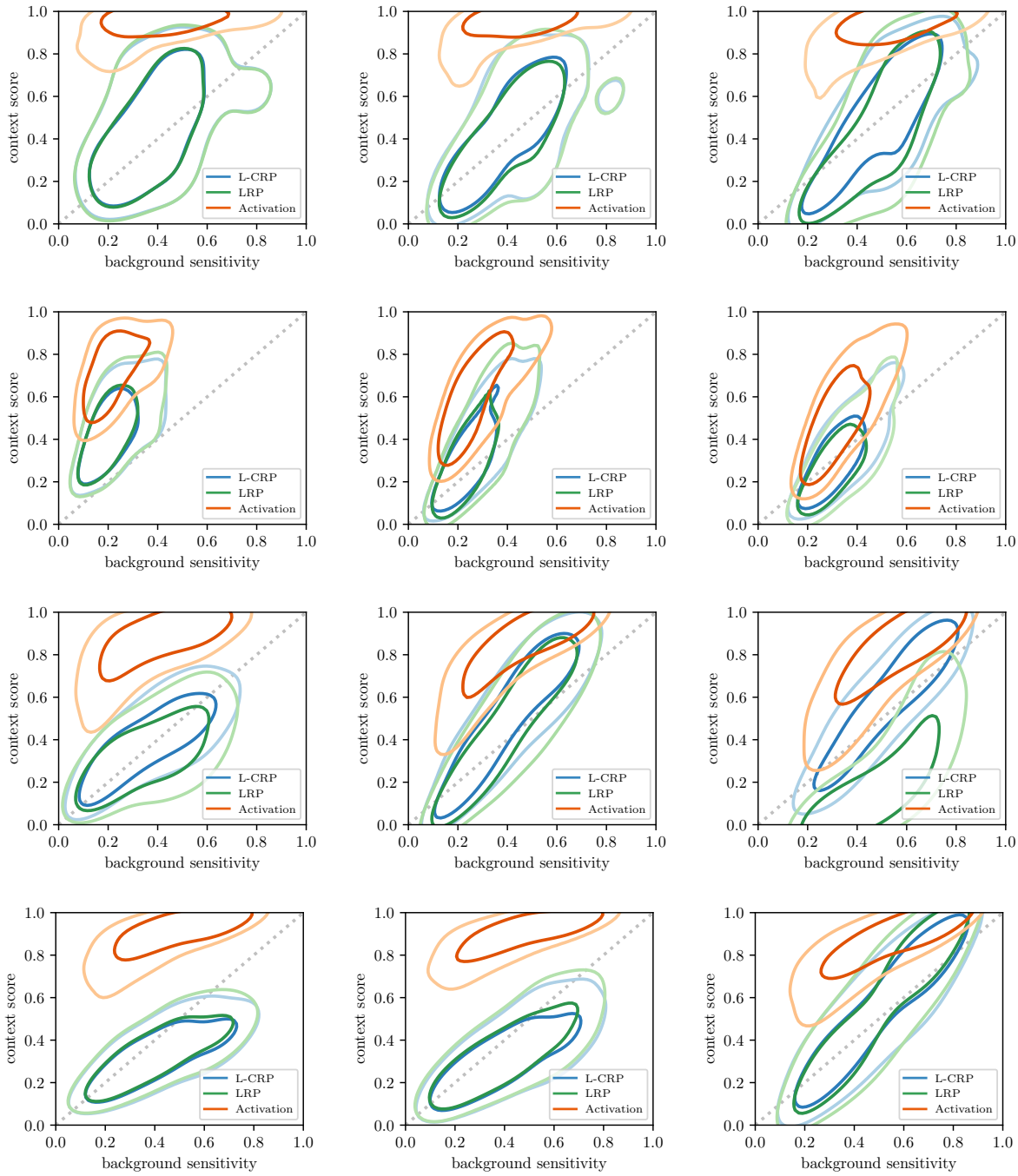


Figure A.5. Evaluation of context scores C by comparing these to background sensitivity scores S from latent activation, latent LRP attribution maps maps and L-CRP input attributions. (*1st row*): Layers `features.0`, `features.5` and `features.10` (from left to right) of the UNet model. (*2nd row*): Layers `layer3.0.conv1`, `layer4.0.conv3` and `layer4.2.conv2` (f.l.t.r.) of the DeepLabV3+ model. (*3rd row*): Layers `6.cv3`, `8.cv3` and `10` (f.l.t.r.) of the YOLOv5 model. (*4th row*): Layers `ERBlock_3.0.rbr_dense`, `ERBlock_4.0.rbr_dense` and `ERBlock_5.0.rbr_dense` (f.l.t.r.) of the YOLOv6 model. Contours correspond to 50 % and 80 % of values using estimated densities via Gaussian kernels of bandwidth 0.4. Ideally, a linear relationship exists as indicated by a dotted gray line.

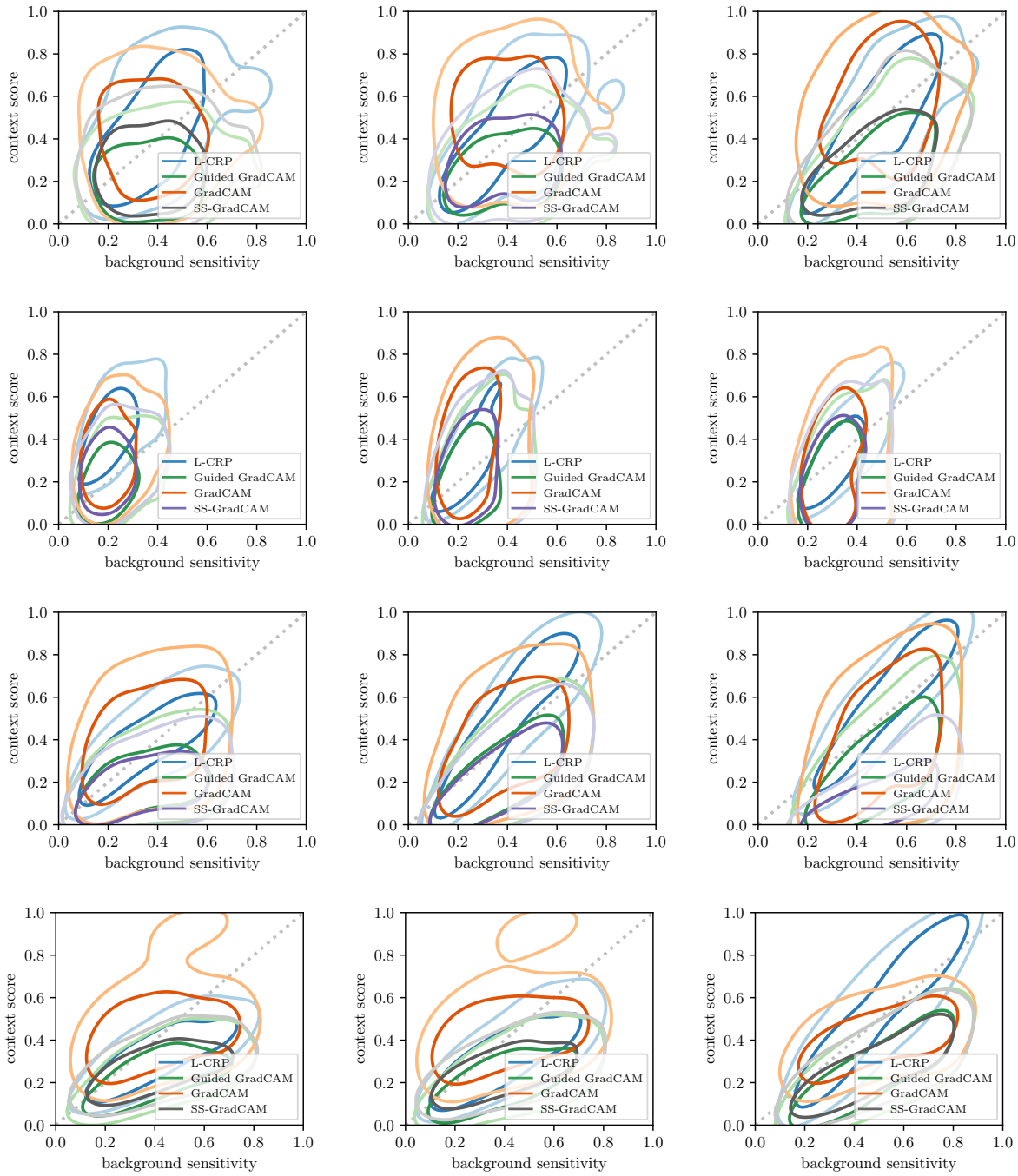


Figure A.6. Evaluation of context scores C by comparing these to background sensitivity scores S from latent GradCAM and SS-GradCAM attribution maps maps, as well as L-CRP and Guided GradCAM input attributions. (*1st row*): Layers `features.0`, `features.5` and `features.10` (from left to right) of the UNet model. (*2nd row*): Layers `layer3.0.conv1`, `layer4.0.conv3` and `layer4.2.conv2` (f.l.t.r.) of the DeepLabV3+ model. (*3rd row*): Layers `6.cv3`, `8.cv3` and `10` (f.l.t.r.) of the YOLOv5 model. (*4th row*): Layers `ERBlock_3.0.rbr_dense`, `ERBlock_4.0.rbr_dense` and `ERBlock_5.0.rbr_dense` (f.l.t.r.) of the YOLOv6 model. Contours correspond to 50 % and 80 % of values using estimated densities via Gaussian kernels of bandwidth 0.4. Ideally, a linear relationship exists as indicated by a dotted gray line.

strongly from 25.2 to 40.5 %. This follows from the fact, that context scores are usually over-estimated due to object-inspecific attributions, and no sign flips happen anymore acting against over-estimation by decreasing the context score values. All in all, L-CRP remains the method with the most faithful context scores.

Context-based Model Interaction In the background concept flipping experiments in Section 4.2, we interact with the model based on identified (by the model used) background concepts. Therefore, three background concepts are flipped successively and the change in the predicted object logit measured.

In Figure A.7 we show the identified background concepts in detail with reference samples conditioned on the respective target. All concepts correspond to layer `ERBlock_5.0.rbr_dense` of the YOLOv6 model. Please note, that concepts are flipped in the corresponding order as in Figure A.7 from top to bottom.

Interestingly, the perturbation of three concepts of in total 512 leads to missed surfing board predictions. In Figure A.8, we show four such examples with initial prediction confidence scores given.

References

[1] Reduan Achtibat, Maximilian Dreyer, Ilona Eisenbraun, Sebastian Bosse, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. From "where" to "what": Towards human-understandable explanations through concept relevance propagation. *arXiv preprint arXiv:2206.03208*, 2022.

[2] Awadelrahman MA Ahmed and Leen AM Ali. Explainable medical image segmentation via generative adversarial networks and layer-wise relevance propagation. *Nordic Machine Intelligence*, 1(1):20–22, 2021.

[3] Léo Andéol, Yusei Kawakami, Yuichiro Wada, Takafumi Kanamori, Klaus-Robert Müller, and Grégoire Montavon. Learning domain invariant representations by joint wasserstein distance minimization. *arXiv preprint arXiv:2106.04923*, 2021.

[4] Christopher J. Anders, David Neumann, Wojciech Samek, Klaus-Robert Müller, and Sebastian Lapuschkin. Software for dataset-wide XAI: from local explanations to global insights with Zennit, CoRelAy, and ViRelAy. *arXiv preprint arXiv:2106.13200*, 2021.

[5] Alexander Binder. Notes on canonization for resnets and densenets. https://github.com/AlexBinder/LRP_Pytorch_Resnets_Densenet/blob/master/canonization_doc.pdf, 2020.

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 801–818, 2018.

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[8] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13733–13742, 2021.

[9] Glenn Jocher et. al. YOLOv5n Nano models, Roboflow integration, TensorFlow export, OpenCV DNN support. *Zenodo*, 2021.

[10] Gongfan Fang. Deeplabv3plus-pytorch. https://github.com/VainF/DeepLabV3Plus-Pytorch, 2019.

[11] Mathilde Guillemot, Catherine Heusele, Rodolphe Korichi, Sylvianne Schnebert, and Liming Chen. Breaking batch normalization for better explainability of deep neural networks through Layer-Wise Relevance Propagation. *arXiv preprint arXiv:2002.11018*, 2020.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[13] Lucas Y. W. Hui and Alexander Binder. BatchNorm Decomposition for deep neural network interpretation. In *Advances in Computational Intelligence*, Lecture Notes in Computer Science, pages 280–291, Cham, 2019. Springer.

[14] Pavel Iakubovskii. Segmentation models pytorch. https://github.com/qubvel/segmentation_models.pytorch, 2019.

[15] Maximilian Kohlbrenner, Alexander Bauer, Shinichi Nakajima, Alexander Binder, Wojciech Samek, and Sebastian Lapuschkin. Towards best practice in explaining neural network decisions with LRP. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

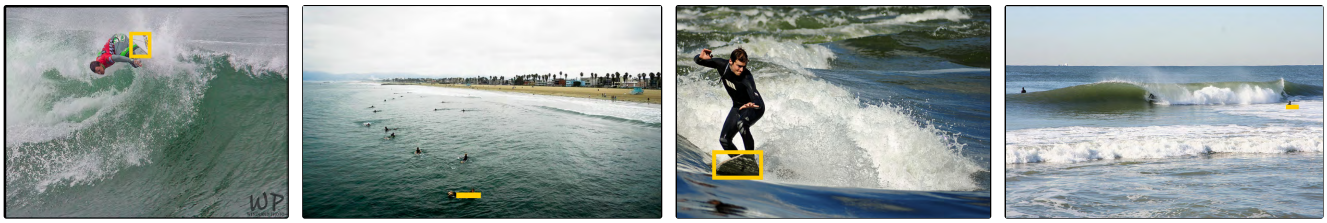
[16] Chuyi Li, Lulu Li, Hongliang Jiang, Kaiheng Weng, Yifei Geng, Liang Li, Zaidan Ke, Qingyuan Li, Meng Cheng, Weiqiang Nie, et al. Yolov6: a single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*, 2022.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[18] Grégoire Montavon, Alexander Binder, Sebastian Lapuschkin, Wojciech Samek, and Klaus-Robert Müller. Layer-Wise Relevance Propagation: An overview. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, volume 11700 of *Lecture Notes in Computer Science*, pages 193–209. Springer, Cham, 2019.

[19] Franz Motzkus, Leander Weber, and Sebastian Lapuschkin. Measurably stronger explanation reliability via model canonization. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 516–520. IEEE, 2022.

Figure A.7. Most relevant reference samples for background concepts being flipped for the model interaction experiment in Section 4.2. The shown reference samples are conditioned on the corresponding target class displayed above.



69.65% 49.47% 89.67% 49.26%

Figure A.8. Flipping three background concepts (177, 478, 445) of layer `ERBlock_5.0.rbr_dense` of the YOLOv6 model leads to missed surfing board predictions in the shown samples. Initial confidence scores are given below each image.

[20] Frederik Pahde, Galip Ümit Yolcu, Alexander Binder, Wojciech Samek, and Sebastian Lapuschkin. Optimizing explanations by network canonization and hyperparameter search. *arXiv preprint arXiv:2211.17174*, 2022.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[22] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[23] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

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

[25] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[26] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[27] Toshinori Yamauchi and Masayoshi Ishikawa. Spatial sensitive grad-cam: Visual explanations for object detection by incorporating spatial sensitivity. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 256–260. IEEE, 2022.