Explaining Defect Detection with Saliency Maps [⋆] [⋆⋆]

Joe Lorentz[1,2][0000−0003−0786−4488], Djamila Aouada[2][0000−0002−7576−2064], Thomas Hartmann[1][0000−0002−0465−6280], Assaad Moawad[1][0000−0002−1561−8992], and Francois Fouquet[1][0000−0001−9028−768X]

[1] DataThings, 5 rue de l'industrie, 1811 Luxembourg, Luxembourg
`first.last@datathings.com`
[2] University of Luxembourg, 29 avenue John F. Kennedy, 1855 Luxembourg, Luxembourg
`djamila.aouada@uni.lu`

Abstract. The rising quality and throughput demands of the manufacturing domain require flexible, accurate and explainable computer-vision solutions for defect detection. Deep Neural Networks (DNNs) reach state-of-the-art performance on various computer-vision tasks but widespread application in the industrial domain is blocked by the lacking explainability of DNN decisions. A promising, human-readable solution is given by so-called saliency maps, heatmaps highlighting the image areas that influence the classifier's decision. This work evaluates a selection of saliency methods in the area of industrial quality assurance. To this end we propose the distance pointing game, a new metric to quantify the meaningfulness of saliency maps for defect detection. We provide steps to prepare a publicly available dataset on defective steel plates for the proposed metric. Additionally, the computational complexity is investigated to determine which methods could be integrated on industrial edge devices. Our results show that DeepLift, GradCAM and GradCAM++ outperform the alternatives while the computational cost is feasible for real time applications even on edge devices. This indicates that the respective methods could be used as an additional, autonomous post-classification step to explain decisions taken by intelligent quality assurance systems.

Keywords: XAI · Saliency · Defect detection · Edge AI

1 Introduction

The ever-growing throughput and quality demands of modern manufacturing make it increasingly difficult to rely on the human eye for a rising number of quality assessment procedures. This development led to the introduction of computer

[⋆] The present project is supported by the National Research Fund, Luxembourg

[⋆⋆] The final authenticated version is available online at https://doi.org/10.1007/978-3-030-79463-7_43.

vision algorithms, which are now widely used in different fields such as the food industry [4] or the production of printed board-circuits [21]. Most of these approaches rely on handcrafted algorithms to recognize domain specific faults. The downside of these algorithms is that the development requires domain knowledge. Furthermore, the specific nature of handcrafted solutions makes them susceptible to changes on the production side and unsuitable for varying product types. The current evolution of the manufacturing domain towards the so-called Industry 4.0 demands for more flexible solutions, which can be introduced without extensive prior study of domain characteristics. Deep Neural Networks (DNNs) provide this by automatically learning high level features [9]. DNNs reach state-of-the-art performance on various computer vision tasks like object recognition or segmentation [10, 14]. A major blocking point for the wide-spread application of this emerging technology in industry is the lacking explainability of classification decisions [23]. This drawback is a direct consequence of the end-to-end feature learning. Neural networks tend to rely heavily on features which are unintuitive for human perception [17, 23]. This makes it difficult to justify decisions without profound knowledge of the technology. Additionally, DNN-based defect detection requires considerable computational effort and outsourcing costly computations to cloud services is not always an option. Therefore, any additional effort linked to providing explanations should be as low as possible.

The research field of explainable artificial intelligence (XAI) aims to provide human-understandable explanations to the decision-making process of machine learning models such as DNNs. The field is split in two main directions, explainable modeling and post-modeling explainability. Explainable modeling investigates ways to develop inherently more interpretable machine learning models. This methodology involves a trade-off between explainability and model performance [13]. Post-modeling explainability proposes additional algorithms to map existing models in a way that is easier to understand for humans. The drawback is the additional computational effort linked to the post processing. The main goal of this work is to investigate suitable XAI methods to explain DNN decisions in the context of industrial defect detection and how to optimize them for this task. We consider this task as an image classification problem where each class represents either a defect-free product or a product with a defined defect type. We argue that XAI methods for this task should:

1. be compatible with existing models for maximum performance and flexibility
2. provide meaningful insights to human operators
3. have low computational cost to enable computation on the edge

Explainable modeling contradicts our first proposition by design, as the development of inherently interpretable models limits the usage of existing models. Hence, we focus on post-modeling explainability in our work and investigate the computational cost of available methods to find suitable candidates that provide the best trade-off between propositions 2 and 3. To measure the *meaningfulness* of explanations, we propose to train a model for classification of a defect dataset with available segmentation of the defects in question. We use a dataset[3] of flat steel plates that provides annotated areas of four defect classes. We then compute the correlation between areas of the image deemed important by XAI methods and the ground-truth defect location. We suggest that an explainable quality assurance AI should be able to focus attention on the image areas that actually contain a defect. Furthermore, XAI methods that succeed to do so could be used as weakly supervised defect segmentation systems with drastically reduced effort in data acquisition. It is easier, less time-consuming and less error-prone for human experts to provide a single label per image instead of precisely extracting defect areas. Benchmarks on explainable AI exist already but only for the analysis of natural images or handwritten digits [1]. To the best of our knowledge, we are the first to provide a benchmark of XAI for the domain of industrial defect detection. The computational cost of post-modeling XAI is an important factor, especially when cloud computing is unwanted or not realizable. A powerful GPU is used to enable the extensive benchmark. We argue that the computational costs, reported with this setup, can also guide industry practitioners to decide which methods are suitable with potentially lower hardware capacity. The contributions of this work can be summarized as follows:

1. extensive benchmark of XAI for industrial defect detection
2. new metric to evaluate the meaningfulness of saliency maps
3. tools to prepare a datset for the benchmark and metric

The remainder of this work is organized as follows. Section 2 introduces the dataset as well as preprocessing steps necessary for our experiments. The investigated methods are presented in Section 3. Section 4 introduces the experiments conducted and presents our results. Conclusions and proposed future work are given in Section 5. The code used for data preprocessing and our experiments is publicly available[4].

2 Steel Patch Dataset

Our study requires a dataset linked to industrial defect detection with ground truth segmentation labels available. Furthermore, as to train a model on defect classification, samples should clearly belong to one class only. Datasets satisfying our requirements as well as quality demands are scarce, presumably due to the high effort linked to labeling. We use a dataset of flat steel sheets, provided by Severstal[5], a steel mining and manufacturing company. The set was originally used for a defect localization competition launched by Severstal and is now publicly available[6]. The set provides 12.6k grayscale training images and a file indicating defect locations for roughly half of them. Each image can contain a multitude of defects of any class. As we require samples that can be labeled as one class only, we split the original 1600x256 pixel images into 6 patches of 256x256

[3] https://www.kaggle.com/c/severstal-steel-defect-detection/overview
[4] link removed due to double blind review
[5] https://www.severstal.com
[6] https://www.kaggle.com/c/severstal-steel-defect-detection/overview

pixels. With no overlap among patches, we have a surplus of 64 pixels horizontally which we remove by cropping 34 pixels at the left and right borders each. The original pictures seem to have been taken on a running production line with a fixed camera. Hence, some images feature large black background areas. We remove image patches from our data which contain more than 95% background pixels to avoid classifying pure background patches while still capturing defects on the edges of sample steel plates. We end up with a total of 35k image patches. Table 1 gives an overview of how many images per class the resulting steel patch dataset contains. A large portion (54%) of the patches show no defects, while a negligible amount (0.8%) features more than one defect class. We use the defect-free images during training as we argue that learning the patterns of defect-free steel plates will help the model to spot defects. We exclude the patches with multiple classes to avoid confusing the classifier. Note that the remaining patches can show more than one defect area, however, all areas belong to the same class. Fig. 2 shows some examples of the steel patch datatset, with defect locations indicated by a boundary.

Table 1: Steel patch sample composition

Defect Class	# Samples	%
1	1473	4.2
2	221	0.6
3	12245	34.9
4	1871	5.3
no defect	18942	54.0
multiple	289	0.8

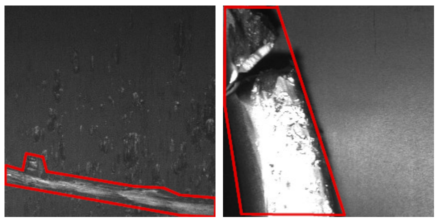

(a) Defect class 3 **(b)** Defect class 4

Fig. 2: Steel patch examples of the two most common classes. Defect areas indicated by red boundary.

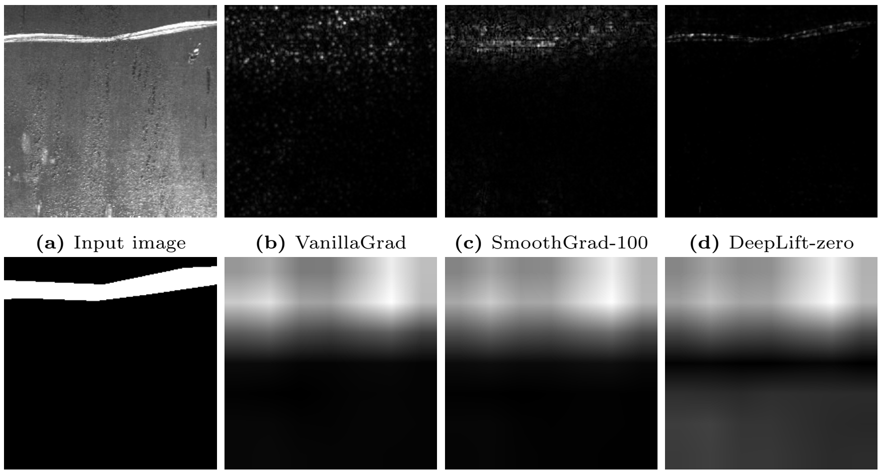
3 Review of Saliency Maps for Industrial Defect Detection

Saliency maps are heatmaps showing the salient (i.e. important) areas of a picture. In XAI they are used to visualize the importance of every input image pixel in relation to the output of a DNN. In the case of image classification, the output of a DNN is usually a vector of length n, where each value indicates the probability of the input belonging to each of the n classes the model knows.

Methods that compute saliency maps for DNNs can roughly be divided into four groups: pertubation-based [6, 7, 12], deconvolution [19, 23, 24], gradient-based [3, 16–18, 20] and class activation map based [5, 15, 22, 25]. Table 2 gives an overview of these categories and their features as exposed by the respective literature and existing reviews [1, 2]. Pertubation methods achieve impressive results at the cost of high computational cost which contradicts our third proposition (see Section 1) as it makes them unsuitable for applications on the edge.

Category	**Meaningfulness**	**Computational cost**	**Implementation**
Pertubation [6, 7, 12]	very high	very high	easy, flexible
Deconvolution [19, 23, 24]	low-high	low	challenging, unflexible
Gradient [3, 16–18, 20]	low-high	low-high	easy, flexible
CAM [5, 15, 22, 25]	high	low-high	easy, flexible

Table 2: Overview of saliency method categories and the features according to the respective literature and reviews [1, 2].



(a) Input image **(b)** VanillaGrad **(c)** SmoothGrad-100 **(d)** DeepLift-zero

(e) Defect mask **(f)** GradCAM **(g)** GradCAM++ **(h)** ScoreCAM

Fig. 3: (a) Input image; **(e)** Segmentation mask; saliency maps using gradient **(b-d)** and CAM **(f-h)** approaches

Simonyan et al.[17] show that gradient-based methods generalize deconvolution, while the implementation of the latter is more challenging and restrictive. Therefore, we choose to exclude pertubation and deconvolution based approaches from our study. The following sections introduce the gradient based and class activation based methods selected for investigation. Fig. 3 shows the saliency maps computed for one sample from the steel patch dataset according to the selected methods.

3.1 Gradient-based

Simonyan et al.[17] were among the first to introduce the idea of using the gradient of the model output w.r.t. the input as a measure of importance. For a given input image, the authors use the model's output class score vector (before softmax) and set all values to zero except for the value that corresponds to the class that should be visualized (e.g. the class with the highest score). Next they use the backpropagation algorithm to compute the partial derivatives of the one-hot vector w.r.t the input image pixels. Intuitively, the result indicates how small changes at any pixel would affect the confidence of classifying the image as the class in question. The method is fast as it can rely on efficient GPU-implementations of backpropagation which are widely available with most deep learning frameworks (e.g. PyTorch [11]). In our study we refer to this procedure as *VanillaGrad*. The outputs of *VanillaGrad* tend to be very noisy. Subsequent works tried to identify the reasons and counteract this behavior. In [18], meaningless local variations of the partial derivatives are framed as the main cause. The authors suggest *SmoothGrad* as a solution. They smooth the gradient by running several iterations of *VanillaGrad* while adding noise to the input image and calculating the average over the resulting saliency maps. Other works [16, 20] suggest that the noise is mainly caused by saturated nonlinearities for which *VanillaGrad* fails to backpropagate values. To counteract, Sundararajan et al. [20] let the user choose a reference input (e.g. all zero) and integrate the gradient while the input varies along a linear path from the reference to the original input image. The integral is approximated by taking the average over a predefined number of discrete steps. Shrikumar et al. [16] also make use of a reference input. They propose *DeepLift*, a method to compute the saliency in terms of difference from reference in a single backpropagation-like step. As a reference input, [16] suggests either an all zero image or blurring the original input. In [2], it is shown that *DeepLift* (without the RevealCance rule) can be implemented via the standard backpropagation algorithm and is most often a good approximation of integrated gradients. *Layer-wise Relevance Propagation (LRP)* [3], also shares some similarities with DeepLift, and is even identical to the latter under certain circumstances [2].

3.2 CAM-based

The basic idea of class activation maps (CAM) is to compute a weighted average over convolutional feature maps (i.e. output of hidden convolution layers). Convolutional feature maps give a notion of learned features being present at spatial locations. Zhou et al. [25] use a fully convolutional model, compute the spatial average of each feature map of the last layer and use these values as input for a fully connected layer that produces the desired output (e.g. class scores). The weights of this classifier layer are first trained on the target data and later used as weights for the corresponding feature maps to compute the averaged CAM. Lastly, the resulting CAM is up-sampled to the input image resolution which leads to smoother but less detailed saliency maps when compared to gradient-based approaches. *GradCAM* [15] generalizes this procedure. To avoid the need for changes on existing models and prior training of CAM-weights, the authors suggest to instead rely on the gradient similarly to the gradient-based methods (see Section 3.1). Gradients are backpropagated until the target layer and the spatial average is used as CAM-weight for the corresponding feature maps. The authors suggest targeting the last convolutional layer of a given model as it gives the best compromise between high-level semantics and detailed spatial information. *GradCAM++*[5] extends on *GradCAM* by computing the CAM-weights as a weighted average instead of a global average with identical weights. Wang et al. [22] propose *ScoreCAM* to compute CAM without alternation of the model and without dependence on gradients. The authors first perform a single forward computation with the input image to retrieve the feature maps of the last convolutional layer. Next, they up-sample each feature map to match the input image resolution and normalize the maps on the range of $[0, 1]$. The results are multiplied with the original input to rank pixel importance according to individual feature maps. Additional forward passes for each masked image are performed and the changes in the target class score observed. Lastly individual scores are used as weights for the average CAM computation using the up-sampled feature maps retrieved during the initial forward step. The downside of this method is that the computational complexity increases with the number of feature maps used at the target layer.

4 Experiments

For our investigations, we used a ResNet-18 model [8] pretrained on ImageNet [14], as available with PyTorch [11]. The fully connected layer and the later half of the convolution layers were unfrozen and fine-tuned for 50 epochs on the steel patch data presented in Section 2. We randomly selected one fourth of the images to validate the performance and reached an accuracy of 89%. As the objective is to compare saliency maps with human-provided defect areas, saliency was only computed of validation samples that feature defects. Additionally, we chose to only use samples classified with high confidence ($\geq$ 90%). Low confidence decisions lead to fuzzy explanations which would negatively impact our benchmark. This left us with 1984 samples for which saliency maps were computed using VanillaGrad [17], SmoothGrad [18], DeepLift [16], GradCAM [15], GradCAM++ [5] and ScoreCAM [22]. For DeepLift we experimented with two reference variants. We either used a zero-image or blurred the input image with a Gaussian kernel. The authors of SmoothGrad propose to smooth gradients over 50 iterations and setting the standard deviation of the added noise to 0.2 times the range of the input image. Initial tests proofed that this noise scale is not suitable for our data. Instead, the noise was scaled with 0.02 and smoothing over 10, 50 and 100 iterations was investigated.

The computation time of methods that require more than 1 forward and backward pass on the model can be optimized via batched execution, in case the memory capacity of the used hardware allows for it. We conducted our experiments on an *Nvidia GTX 1060* with 6GB memory which allowed us to investigate how far the speed of the more costly methods can be enhanced. The implementation of batched execution is straightforward for ScoreCAM and SmoothGrad and does not differ much from the unbatched variant; hence, the computation times of both variants was compared. For SmoothGrad, we set the batch size to the number of iterations (i.e 10, 50, 100) which we were able to fit on our GPU. For ScoreCAM we experimented with batch sizes of 1, 25 and 50. There was no need to further increase the batch size as the speedup between the latter two turned out to be negligible. In the case of DeepLift, PyTorch implementations, with and without batching differ substantially. We suggest that a batch size of 2, required for parallelized execution, is feasible even for edge computation. Hence, only the batched variant of DeepLift was investigated.

The raw saliency maps usually feature a wide spread with extreme outliers and negative values. The authors of the original methods propose various normalization schemes to produce more insightful explanations. Propositions from the original papers were gathered and an ablation study was conducted to determine the impact of normalization. We investigated taking the absolute values and using a rectified linear unit (ReLU) to reduce the spread potentially caused by large negative values. Afterwards a min-max-nomalization was performed to map the values on the range of $[0, 1]$:

$$\hat{x} = \frac{x - min(x)}{max(x) - min(x)}, \tag{1}$$

where x and $\hat{x}$ denote the saliency maps before and after normalization respectively. Additionally, we investigated capping the normalization at the 99th percentile instead of the maximum to remove outliers.

4.1 Metrics

We report two metrics for our benchmark. We timed the individual saliency map generation over our 2k samples to determine the computational complexity. The median frames per second that the GPU could handle is reported. This should allow practitioners to decide which method could be feasible to use on the hardware available to them. The quality of the saliency maps is investigated by comparing them with the ground-truth defect areas. Wang et al. [22] propose the *energy-based pointing game* in their work. For a given sample they first binarized the ground-truth, where pixels inside segmented areas were assigned a 1 and 0 otherwise. They computed the normalized saliency map, multiplied it point-wise with the binary mask and determined the sum of the saliency map values that fell inside the important areas. Lastly they divided this value by the sum over the full saliency map and reported the fraction. A score of 1 means that the whole saliency energy is displayed inside actually important regions and 0 indicates a map that fails to provide any meaningful highlights. We note one major flaw with this metric in that it does not matter how far away salient pixels are from the labeled areas. This is especially problematic for defect detection. Highlights in the near surroundings of defects are indeed helpful with the goal of focusing a human's attention. We therefore propose the following changes to the metric. In addition to the binary segmentation mask, a distance mask is determined. For every pixel the euclidean distance to the closest point labeled as a defect is computed. We min-max-normalized (see Equation 1) the mask by setting the maximum to the length of the diagonal of the input image and the minimum to 0 (i.e. inside defect areas). We did not change the computation of the fraction's numerator (inside-energy). For the denominator we took the sum of the inside-energy and the outside-energy. We define the latter as the sum over the result of multiplying the saliency map point-wise with the distance map. We denote this metric as the *distance pointing game* which allows to quantify the ability of saliency maps to focus attention on important image areas. Fig. 4 shows the inside- and outside-maps for a constructed example to illustrate the effect of point-wise multiplication of saliency maps and segmentation or distance masks.

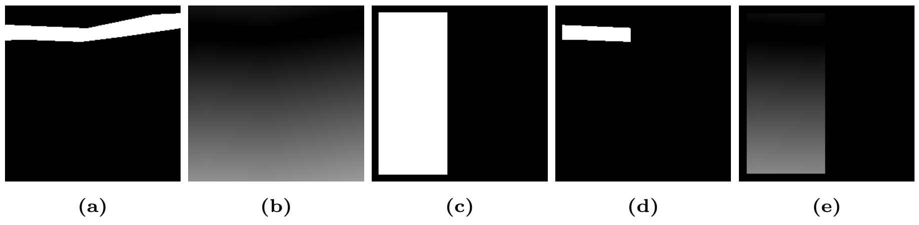
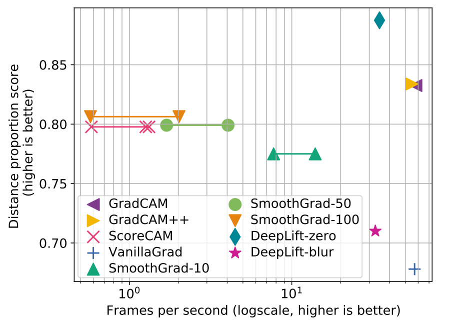
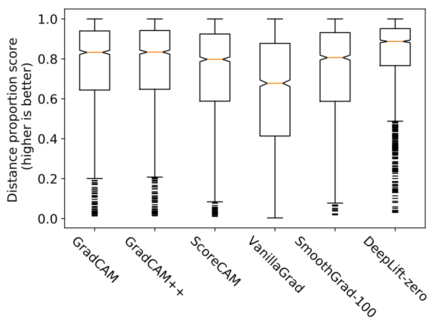


Fig. 4: Illustration of distance metric, **(a)** defect mask, **(b)** distance mask, **(c)** saliency map, **(d)** defect mask * saliency , **(e)** distance mask * saliency



(a) Median score and median fps, lines connect batched and unbatched method variants



(b) Score distribution

Fig. 5: Distance pointing game results

4.2 Results

Fig. 5 shows the results of the *distance pointing game*. For each method we report the values with the normalization setup that resulted in the highest median score respectively. Fig. 5a plots the median score in relation to the median frames per second (fps) over the 2k samples. Lines connect points resulting from batched and unbatched variants of SmoothGrad and ScoreCAM. Additionally, the score distribution of the best hyperparameter setup per method, is reported at Fig. 5b. We record beyond 50fps on methods that only require a single forward and backward pass. The more advanced weight computations of GradCAM++ in relation to GradCAM lead to a drop in median fps and a small increase in distance score. The batched DeepLift computation reached beyond 30 fps. The additional cost for computing a blurred image has negligible impact. However, we report substantially higher distance score when using a zero reference instead. DeepLift-zero reached the highest median score in our experiment and also features a low spread when compared with the remaining methods. We report the lowest score for VanillaGrad. Smoothing the gradient over 10 iterations results in an increase of 0.097. Raising the number of iteration to 50 and 100 leads to a median score of 0.7991 and 0.8063 at the cost of drastically lowered fps. Computing all smoothing iterations within one batch increases the fps substantially for both SmoothGrad and ScoreCAM. The latter reaches lower distance scores compared to the other CAM methods while requiring substantially more computation time. Moreover, the batched implementations require memory capacities beyond 2GB on the target hardware. This, paired with the low fps for unbatched SmoothGrad and ScoreCAM may rule out these methods for application on the edge. Table 3 compares the median distance score of the various normalization schemes. We note that for all methods the scores are generally higher when capping at 100 percentile (true maximum) instead of the 99th percentile. Discarding negative values with a ReLU leads to slightly better results than using the absolute values. These observations indicate that saliency maps on the proposed dataset did not lead to extreme positive outliers and only a small amount of negative outliers. The score of GradCAM without ReLU or ABS before the normalization is only slightly lower. For ScoreCAM and GradCAM++ the scores do not vary at all, which indicates that no negative saliency values were computed. Allowing negative values has a more substantial impact on the gradient-based methods.

5 Conclusion

The rising quality and throughput demands of the manufacturing domain require flexible, accurate and explainable computer-vision solutions for defect detection. Deep Neural Networks coupled with saliency maps to highlight important image areas provide a promising alternative, suitable for computation on the edge. We provide an extensive benchmark of recent saliency methods for the use-case of defect detection on flat steel plates. For this we propose a new metric to quantify the ability of saliency maps to focus attention on defect areas. Our results show that DeepLift, GradCAM and GradCAM++ outperform the alternatives while the computational cost is feasible for real time applications even on edge devices. Furthermore, we show the importance of suitable normalization. This work is an important step in the direction of explainable, automated defect detection on the edge. Future work could extend our benchmark on additional DNN architectures and investigate the relation between saliency maps and model decision confidence. Additionally, the substantial performance difference between DeepLift-zero and DeepLift-blur expose the selection of good reference inputs as a potential research area.

Method	Normalization	Median	Method	Normalization	Median
GradCAM	All-100	0.8240	SmoothGrad 100	All-100	0.5718
	Abs-100	0.8303		Abs-100	0.8055
	ReLU-100	**0.8325**		**ReLU-100**	**0.8063**
	All-99	0.8235		All-99	0.5680
	Abs-99	0.8298		Abs-99	0.7971
	ReLU-99	0.8318		ReLU-99	0.7971
GradCAM++	**All-100**	**0.8338**	SmoothGrad 10	All-100	0.5925
	Abs-100	**0.8338**		Abs-100	0.7752
	ReLU-100	**0.8338**		**ReLU-100**	**0.775**
	All-99	0.8333		All-99	0.5886
	Abs-99	0.8333		Abs-99	0.766
	ReLU-99	0.8333		ReLU-99	0.766
ScoreCAM	**All-100**	**0.7977**	DeepLift zero	All-100	0.4611
	Abs-100	**0.7977**		Abs-100	0.7658
	ReLU-100	**0.7977**		**ReLU-100**	**0.7660**
	All-99	0.7971		All-99	0.4176
	Abs-99	0.7971		Abs-99	0.7156
	ReLU-99	0.7971		ReLU-99	0.7156
VanillaGrad	All-100	0.6014	DeepLift blur	All-100	0.3255
	Abs-100	0.6778		**Abs-100**	**0.4555**
	ReLU-100	**0.6780**		**ReLU-100**	**0.4555**
	All-99	0.5971		All-99	0.3123
	Abs-99	0.6717		Abs-99	0.4198
	ReLU-99	0.6715		ReLU-99	0.4199

Table 3: Median distance score for various normalization, best setups in bold font.

References

1. Adebayo, J., Gilmer, J., Muelly, M., Goodfellow, I., Hardt, M., Kim, B.: Sanity checks for saliency maps. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31, pp. 9505–9515. Curran Associates, Inc.
2. Ancona, M., Ceolini, E., Öztireli, C., Gross, M.: Towards better understanding of gradient-based attribution methods for Deep Neural Networks
3. Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.R., Samek, W.: On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation **10**(7), e0130140. https://doi.org/10.1371/journal.pone.0130140
4. Brosnan, T., Sun, D.W.: Improving quality inspection of food products by computer vision—a review **61**(1), 3–16. https://doi.org/10.1016/S0260-8774(03)00183-3
5. Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 839–847. IEEE. https://doi.org/10.1109/WACV.2018.00097
6. Fong, R., Patrick, M., Vedaldi, A.: Understanding Deep Networks via Extremal Perturbations and Smooth Masks. pp. 2950–2958
7. Fong, R.C., Vedaldi, A.: Interpretable Explanations of Black Boxes by Meaningful Perturbation. In: 2017 IEEE International Conference on Computer Vision (ICCV). pp. 3449–3457. IEEE. https://doi.org/10.1109/ICCV.2017.371

8. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 770–778. IEEE. https://doi.org/10.1109/CVPR.2016.90
9. Li, H., Ota, K., Dong, M.: Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing **32**(1), 96–101. https://doi.org/10.1109/MNET.2018.1700202
10. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: Common Objects in Context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014, Lecture Notes in Computer Science, vol. 8693, pp. 740–755. Springer International Publishing. https://doi.org/10.1007/978-3-319-10602-1_48
11. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., dAlché Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc.
12. Petsiuk, V., Das, A., Saenko, K.: RISE: Randomized Input Sampling for Explanation of Black-box Models
13. Rai, A.: Explainable AI: From black box to glass box **48**(1), 137–141. https://doi.org/10.1007/s11747-019-00710-5
14. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge **115**(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y
15. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization. pp. 618–626
16. Shrikumar, A., Greenside, P., Kundaje, A.: Learning Important Features Through Propagating Activation Differences. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. pp. 3145–3153. ICML'17, JMLR.org
17. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
18. Smilkov, D., Thorat, N., Kim, B., Viégas, F., Wattenberg, M.: SmoothGrad: Removing noise by adding noise
19. Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for Simplicity: The All Convolutional Net
20. Sundararajan, M., Taly, A., Yan, Q.: Axiomatic Attribution for Deep Networks
21. Tsai, D.M., Lin, C.T.: Fast normalized cross correlation for defect detection **24**(15), 2625–2631. https://doi.org/10.1016/S0167-8655(03)00106-5
22. Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., Hu, X.: Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 111–119. IEEE. https://doi.org/10.1109/CVPRW50498.2020.00020
23. Zeiler, M.D., Fergus, R.: Visualizing and Understanding Convolutional Networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014. pp. 818–833. Lecture Notes in Computer Science, Springer International Publishing. https://doi.org/10.1007/978-3-319-10590-1_53

24. Zhang, J., Lin, Z., Brandt, J., Shen, X., Sclaroff, S.: Top-down Neural Attention by Excitation Backprop
25. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning Deep Features for Discriminative Localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2921–2929. IEEE. https://doi.org/10.1109/CVPR.2016.319