Perception Visualization: Seeing Through the Eyes of a DNN

Loris Giulivi
Mark James Carman
Giacomo Boracchi

Politecnico Di Milano

Abstract

Artificial intelligence (AI) systems power the world we live in. Deep neural networks (DNNs) are able to solve tasks in an ever-expanding landscape of scenarios, but our eagerness to apply these powerful models leads us to focus on their performance and deprioritises our ability to understand them. Current research in the field of explainable AI tries to bridge this gap by developing various perturbation or gradient-based explanation techniques. For images, these techniques fail to fully capture and convey the semantic information needed to elucidate why the model makes the predictions it does. In this work, we develop a new form of explanation that is radically different in nature from current explanation methods, such as Grad-CAM. *Perception visualization* provides a visual representation of what the DNN perceives in the input image by depicting what visual patterns the latent representation corresponds to. Visualizations are obtained through a reconstruction model that inverts the encoded features, such that the parameters and predictions of the original models are not modified. Results of our user study demonstrate that humans can better understand and predict the system's decisions when perception visualizations are available, thus easing the debugging and deployment of deep models as trusted systems.

1 Introduction

Explaining a deep model is an intricate problem that requires balance between soundness and completeness to be effective in real-world scenarios such as model debugging [15], and is essential for building trusted systems. Explanations need to succinctly convey the model's reasoning and not overwhelm the user. Current explanation techniques such as Grad-CAM [27] have focused on generating pixel-wise measures of conspicuity, by analysing the effect each pixel has on the model's prediction using gradient or perturbation-based analysis [7]. However, these are often uninformative [25], as exemplified in Figure 1. We believe that explanations should instead carry semantic meaning at a higher level. Thus, we introduce *Perception Visualization* (PV), a novel technique to explain the latent semantics of a deep convolutional neural network (CNN).

PV consists of two components: *i*) a gradient-based saliency map and *ii*) a reconstruction obtained through network inversion. This combination allows PV to show both *where the model is looking* and *what the model is seeing*, in contrast to the vast majority of previous

Figure 1: Based on saliency maps it is unclear why this image is labelled as a *cat* rather than a *laundry basket*. Grad-CAM [27] explanations are essentially the same for both classes.

techniques that only show *where* the network is focusing its attention when making a decision. To the best of our knowledge, ours is the first work producing image explanations by using a neural network to invert latent representations. Moreover, our work aims at providing explanations for the diagnostic situation in which a data scientist performs error analysis on an image classifier, manually inspecting images on which the model performs poorly. *Useful explanations should inform the direction for resolving the fault*, such as procuring more training data from a particular domain. In Figure 2 we see examples of such misclassified images, and note that the PV for the first image shows that the model is confusing a neon sign for a television, which indicates a possible lack of training images containing neon signs. Such a realization could *not* have been obtained without the help of the perception visualization, since neon signs aren't even a class in this problem. In this case, we see that PV explanations can be employed in a *prescriptive manner* in order to improve model performance.

PV reveals itself empirically to be particularly effective at explaining incorrect predictions from the model, as shown in Figure 2. We validate PV through a user study, investigating the users' ability to guess the model's predictions when explanations are given. Results of a survey on circa 100 subjects show that PV is able to help respondents better determine the predicted class in cases where the model had made an error. We make our code publicly available at: https://github.com/loris2222/PerceptionVisualization

2 Related Work

Explainable artificial intelligence (XAI) is an emerging field that is experiencing a surge in research interest. We now discuss XAI techniques as they relate to image classification.

An explanation, in the context of XAI, is a way to present results to a human in understandable terms [8]. This definition is purposely vague and includes, for example: textual descriptions of the reasoning behind the prediction [14], heatmaps indicating the pixels that most contributed to the result [27], or graphs that match decisions with some knowledge base [31]. Our explanations are local to a single sample, similarly to what is done in Grad-CAM [27] and SHAP [17], and differently to works such as LIME [23] and SpRAy [16]. Moreover, PV differs from methodologies that make use of gradient information (*e.g.* Grad-CAM [27]) or perturbation analysis (*e.g.* RISE [22], Score-CAM [30]), in that we use a deep neural network to reconstruct the semantics of the latent space.

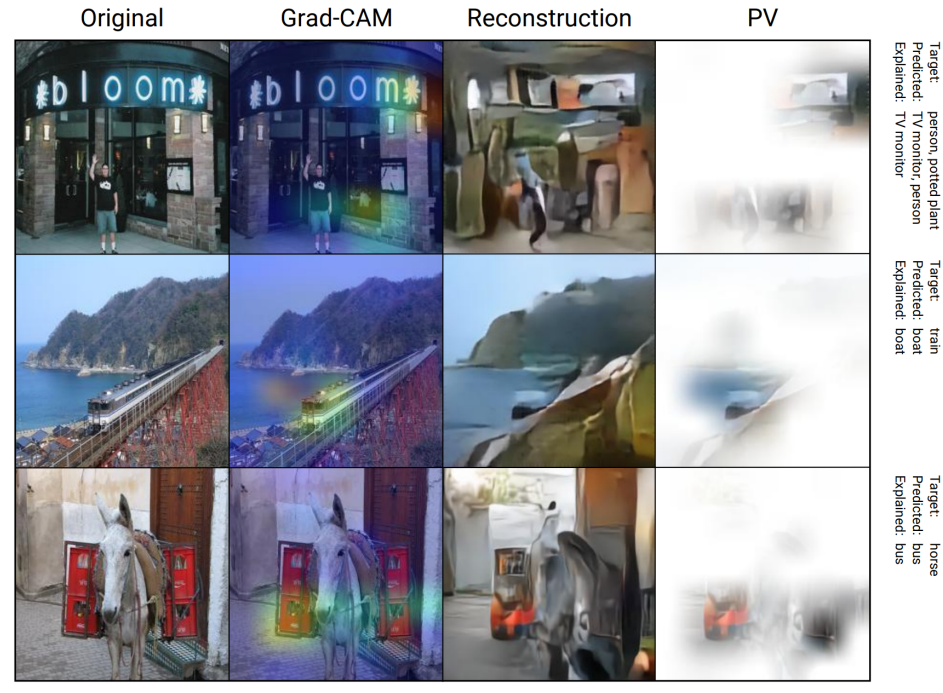

Figure 2: Misclassified examples and their explanations using Grad-CAM and PV. Explanations given by PV are much more informative and are able to describe the network's error by depicting semantic features of the network's perception. In particular, the three PVs depict: a rectangular object recalling a TV screen, a black and white object in water recalling a boat, and an orange shape recalling a bus.

The most prominent techniques to explain images in XAI literature are saliency maps, which attempt to visualize *where* the model places its attention in the image. These techniques include class-activation maps [21, 34] and their subsequent improvements [4, 19, 27]. A saliency map is itself an image, usually of the same size as the input, highlighting where the model focuses its attention. Given a saliency map, an explanation can be constructed either by superimposition or by masking. In the former case, the saliency map is displayed with different hues depending on the importance of the region. For the latter, the image is covered and only regions deemed relevant by the saliency map are shown. A more detailed description of these practices is provided in the supplementary material.

Other works, which are closer to ours in philosophy, have attempted to directly visualize *what the model sees* in the input image rather than simply *where it focuses its attention*. Google's *Inceptionism* [20] and Simonyan et. al. [28] optimize an input image to maximize the network's response to some desired class, and in turn give an intuition of how the model represents such class. Our work, instead, provides visualizations of the model's perception for each image sample. Perhaps the most relevant, HOGgles [29] inverts visual features, most notably allowing to view images through *HOG glasses (HOGgles)*. This work, however, relies on feature dictionaries to solve an optimization problem to reconstruct small image windows. Our work, on the other hand, uses a neural network to decode full images, only requires optimization during the learning phase, and works on much more complex models.

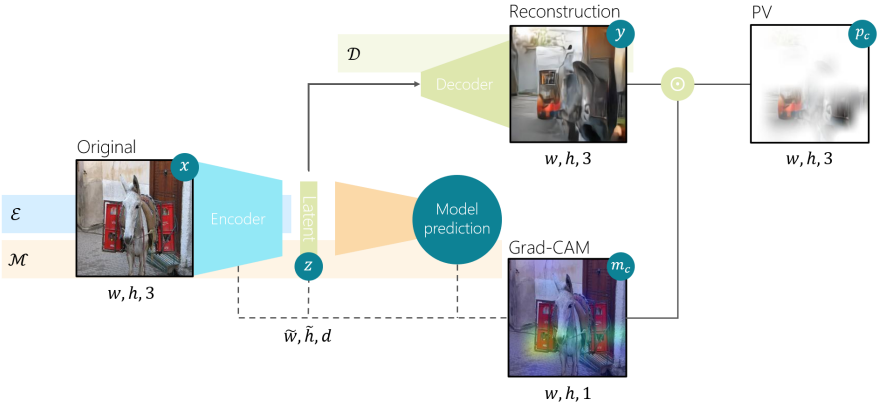

Figure 3: An overview of our method and interactions between the models involved. Encoder $\mathcal{E}$ is a truncation of the model $\mathcal{M}$ which we want to explain, decoder $\mathcal{D}$ is trained to reconstruct the encoder's latent representations. From these, we compute Grad-CAM saliency maps and reconstructions, which are then combined to obtain PV.

3 Perception Visualization

Our core contribution is *perception visualization* (PV): a novel explainability technique that makes use of a deep neural network to explain a pre-trained deep neural network. PV visually displays the input of the model *from the model's point of view*, which is that of its latent representation. PV leverages CNN inversion techniques and displays the resulting reconstructions in a way that can help users understand whether the model correctly perceived the input or not. Several works, including [9, 13, 33], propose techniques for reconstructing the original image from its latent representation. These take ideas from research in autoencoders [3] and generative adversarial networks (GAN) [11] to improve reconstruction quality. In our work, we show how network inversion can be used to generate novel explanations, providing a better insight into the model's functioning without modifying its performance.

Method: In what follows we describe Perception Visualization for CNN classifiers (Figure 3), which are the most widely used deep neural models for images. Let us denote the trained CNN we want to explain as:

$$\mathcal{M} : \mathbb{R}^{w,h,3} \to \mathbb{R}^n, \tag{1}$$

which maps RGB images of size (w,h) into a vector of n posterior probabilities. PV is a class-discriminative explanation that we define, for a class $c \in \{1,\dots,n\}$ and for an input image $x \in \mathbb{R}^{w,h,3}$, as a second image $p_c \in \mathbb{R}^{w,h,3}$ constructed such that each of its pixels $p_c(i,j,k)$ in position (i,j) of the k-th color channel is defined as:

$$p_c(i,j,k) \doteq (1 - m_c(i,j)) + m_c(i,j)y(i,j,k)\,, \tag{2}$$

where $m_c \in [0,1]^{w,h,1}$ is a saliency map for class c and $y \in \mathbb{R}^{w,h,3}$ is the reconstruction obtained through network inversion. In the remainder of this section, we will describe these two components in detail.

Saliency Map: The saliency map $m_c \in [0,1]^{w,h,1}$ is computed from x with respect to a specific class c. While any saliency map algorithm could in principle be used, we adopt Grad-CAM

[27] in our experiments. The map depicts *where* the model is looking, namely which portions of x have influenced the prediction for class c. The saliency map is used to determine which regions of the reconstruction to show in the PV. By definition (2), all the pixels where $m_c(i,j) = 0$ are mapped to white pixels in the explanation, while pixels where $m_c(i,j) = 1$ return the corresponding values in the reconstruction, namely $y(i,j,k)$.

Reconstruction: The reconstruction component $y \in \mathbb{R}^{w,h,3}$ of PV is responsible for displaying *what* the model is seeing, and it is obtained by training a network to invert the feature extraction portion of $\mathcal{M}$. Starting from $\mathcal{M}$, we define its submodel $\mathcal{E}$, namely the *encoder*:

$$\mathcal{E} : \mathbb{R}^{w,h,3} \rightarrow \mathbb{R}^{\widetilde{w},\widetilde{h},d}, \tag{3}$$

which, given an image x, computes its latent representation $z = \mathcal{E}(x)$, $z \in \mathbb{R}^{\widetilde{w},\widetilde{h},d}$ having spatial dimensions $(\widetilde{w},\widetilde{h})$ and depth d. The encoder is merely a truncation of the original model which we define to ease the description of our methods. Thus, *no network re-training is required*, and the latent representation z computed by $\mathcal{E}$ is the same for PV and for inference.

We call the *decoder* the model that computes the reconstruction, which is instead trained to restore the original image x from the latent representation z:

$$\mathcal{D} : \mathbb{R}^{\widetilde{w},\widetilde{h},d} \rightarrow \mathbb{R}^{w,h,3}. \tag{4}$$

In this setup, given an input sample x, we define its reconstruction y as:

$$y = \mathcal{D}(\mathcal{E}(x)). \tag{5}$$

In our experiments we have inverted a pre-trained ResNet-50 [13] model, using as latent representation the output of its deepest convolutional layer ($conv5\_block3$), which is a tensor of spatial dimensions $[7,7]$ and depth 2048. Our decoder is therefore a CNN made of blocks of convolutional layers and transposed convolutions to perform image up-sampling. The kernel size was set to $[3,3]$ for all layers, and all the layers have leaky ReLU activation with $\alpha = 0.2$, except for the output layer which uses a sigmoid activation and the transposed convolutional layers which are linearly activated. We also perform batch normalization after each transposed convolution. An overview of our model architecture and further discussion regarding our design choices are detailed in the supplementary material.

Lastly, we note that while we have defined PV for CNN-based classifiers, our choice was only guided by the popularity of this kind of models as a benchmark for explainability techniques. Indeed, PV is extensible to any architecture that allows latent space reconstruction and saliency map computation. This includes, but is not limited to, CNN-based models for captioning and segmentation. We discuss possible extensions of our work in Section 5.

Decoder Training: We train the decoder to invert latent representations generated by pre-trained models. The decoder, both during training and inference, has only access to the embeddings $\mathcal{E}$, which renders reconstructions dependent on the model $\mathcal{M}$ to be explained. Inspired by recent studies [9], we train $\mathcal{D}$ using different terms in the loss function:

$$\mathcal{L} = \alpha_1 \, \mathcal{L}_{MSE} \; + \; \alpha_2 \, \mathcal{L}_{SSIM} \; + \; \alpha_3 \, \mathcal{L}_{DSIM}, \tag{6}$$

where $\alpha_i \geq 0 \; i = 1,\ldots,3$ are tuning parameters that we force to sum to one to disentangle the norm of the loss from the learning rate. The components of the loss function are defined in an attempt to yield reconstructions that are faithful to the latent representation. For the same

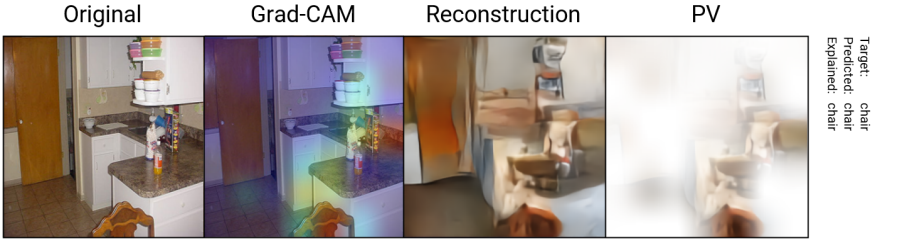

Figure 4: Grad-CAM for class "chair" shows that the model has correctly recognized the chair at the bottom of the image, but that also the table was part of the region of interest. PV explains how the network was fooled by reconstructing a second chair in that region.

reason, we do to allow skip connections (such as those of U-nets [24]), as doing so would enable the decoder to use information from layers further away from the latent representation. The following losses are for batches of input images $X = \{x_i\}$ of size b, their latent representations $Z = \mathcal{E}(X)$ and their reconstructions $Y = \mathcal{D}(\mathcal{E}(X))$.

• **Reconstruction Error:** The MSE between the input images and the recovered images:

$$\mathcal{L}_{MSE} \doteq \sum_{i=1..b} \sum_{j,k,l} |y_i(j,k,l) - x_i(j,k,l)|^2. \tag{7}$$

• **Structural similarity loss:** SSIM [32] is a full reference metric that is widely used in image restoration, and that has also been used to train CNNs [33]. SSIM measures the similarity of images from the perspective of perceived change in structural information, and exploits measures of luminance and contrast. User studies show how SSIM correlates better than MSE with visual quality assessment [32]. The SSIM loss is obtained by negating the sum of the SSIM index over each channel of an RGB image:

$$\mathcal{L}_{SSIM} \doteq - \sum_{i=1..b} \sum_{c \in \{R,G,B\}} SSIM_c(x_i, y_i). \tag{8}$$

• **Deep perceptual SIMilarity loss (DSIM):** When training a decoder using only image-space losses, it is possible that the decoder learns to invert any embedding to the input image, independently of the correctness of the prediction. The DSIM component counters this effect by forcing reconstructions and input images to be similar *in latent space*. This translates in a similarity between the encoding $z = \mathcal{E}(x)$ of the input and the encoding $\mathcal{E}(y)$ of the output. Thus, following Dosovitskiy & Brox [9], we define the DSIM loss as the L_2 norm between the original latent representation and the latent representation of the reconstructed image:

$$\mathcal{L}_{DSIM} \doteq \sum_{i=1..b} \sum_{j,k,l} |\mathcal{E}(y_i)(j,k,l) - z_i(j,k,l)|^2. \tag{9}$$

We have also run experiments including a Wasserstein GAN [2] loss term, but found that, due to the very large and sparse nature of the latent space, these were not beneficial in our case. Our best model achieves similar reconstruction quality as in [9] (model without GAN) using the hyper-parameter combination $\alpha_1 = 0.2, \alpha_2 = 0.4, \alpha_3 = 0.4$. We discuss hyper-parameter tuning in the supplementary material, including the topic of GAN loss terms.

PV as an Explanation: The intuition behind PV is that whenever the model correctly predicts an output label, the latent features must be consistent with the input. Consequently, the

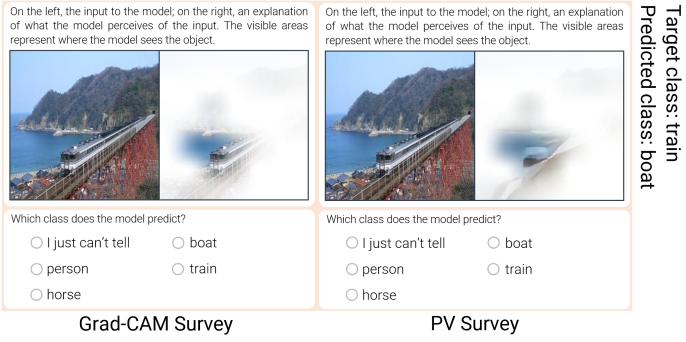

Figure 5: Example question from our surveys. Note how the possible answers are the same for PV and Grad-CAM surveys (the order is randomized in the user's view).

reconstruction will be similar to the input. Instead, whenever the model makes a mistake, activations in the latent space are not consistent with the input, and the reconstruction will semantically resemble those erroneous features, as observed in the PVs of Figure 2.

Another advantage of PV over techniques such as Grad-CAM is that PV can explain faults in saliency maps even when the prediction is correct, as shown in Figure 4. Indeed, there are cases where the localization seems to make little sense, despite the correctness of the model's prediction. Whenever this happens, PV will provide insight on the network's output. The user, looking at the reconstruction, can determine which features of the image were most difficult for the network to correctly perceive.

Lastly, we mention that, in contrast to saliency maps, PV requires to train an additional network ($\mathcal{D}$). We argue, however, that this is a small price to pay to overcome the inherent limitations of plug-and-play explanations, such as those discussed in [25]. Moreover, we stress that training only involves $\mathcal{D}$, while $\mathcal{M}$ *and its performance remain unaltered*.

4 Experiments

The main objective of explainability techniques is to make the user understand why the model provided some output. A best practice to quantitatively assess the effectiveness of an explainable model is to measure *simulatability* [1, 8, 12], which means measuring how often users are able to guess the model's prediction when given the explanation to be evaluated (possibly paired with the input). The core of our experiments comes in the form of a survey, composed of a multiple choice quiz, following in the footsteps of [1]. During the survey, the participants are asked, given different explanations, to predict the model's output. To compare with other state-of-the-art explanations, two different surveys have been administered to different people, one showing PV and the other showing Grad-CAM explanations. Furthermore, we perform additional experiments demonstrating the resilience of PV's semantic value to changes in the decoder.

Survey Structure: The structure of the survey is identical for both PV and Grad-CAM, and is composed of a first portion introducing the explainability technique and of a series of questions. The first section is meant to help users understand what they are about to see. Then, a series of 30 input images are shown paired with their explanation, and for each

Table 1: Aggregated participants' accuracy at determining the model's predictions on the 14 questions where the model had predicted incorrectly and the 16 questions where it predicted correctly for the two explainers: Grad-CAM and PV.

	User Accuracy (± standard error)	
	Model's prediction is **incorrect**	Model's prediction is **correct**
Grad-CAM	8.3% (±1.3%)	**95.9**% (±0.7%)
PV	**35.0**% (±3.0%)	75.5% (±2.5%)

of them we ask: *"What is the model's prediction?"*. The user can then select one of five options, amongst which will be the class (or classes) that is actually present in the image (i.e. the true label) and the class that the model predicts, which will differ when the model misclassified the sample. The remaining options are chosen randomly. It is important to note that throughout the survey we do not inform the user whether or not the model has made a mistake. Figure 5 shows an example question from our surveys. Survey structure, question images and respondent's answers are detailed in the supplementary material.

Experiment Details: We run the experiment by applying PV and Grad-CAM on a pre-trained ResNet-50 [13] model performing transfer learning on the popular PASCAL VOC [10] dataset, due to its use in other similar works [1], the small number of classes (needed for non-dispersive surveys), and its multi-label nature (which renders the task more complicated). Question images are uniformly sampled between correctly classified (16 samples), and incorrectly classified (14 samples). Due to the multi-label classification, we have confined our selection to those images for which the prediction and label set coincided (correctly classified), and those for which none of the targets were in the prediction set (incorrectly classified). During the survey, we provide explanations and require answers only for the top predicted class. For each question, we avoid providing sets of options where the correct answer is very apparent and the others are clearly wrong. To do this, we select five candidate answers for each question, composed of: *i*) the model's prediction, *ii*) three other target labels for the sample (i.e. the objects actually present in the image), and *iii*) a "I just can't tell" option. In the rare case where there were an excess of true labels, some were dropped, and likewise when there were insufficient labels, random classes were chosen (from those in the dataset). The order of answer options was randomized for each question and user. *Additionally, to counteract possible response biases, choices are the same for both surveys.*

A web application assigned users to the surveys in a round-robin fashion. Survey participants were gathered via university mailing lists sent to graduate students enrolled in machine learning related subjects. We expect participants to be somewhat literate in artificial intelligence. No personally identifying information was collected from participants.

Survey Results: We gathered responses from a total of 98 participants. The two surveys were administered in a round-robin fashion, therefore, we expected the same number of responses from the Grad-CAM and the PV surveys. However, responses were uneven: 40 for the PV survey, and 58 for the Grad-CAM survey. As the survey was administered on-line, we do not know how many people dropped out before the end of the survey, however, this number is higher for the PV survey, and may be due to the poor reconstruction quality. As a baseline, we expect random guessing (excluding the "I just can't tell" option) to score 25% overall, both on the correct and incorrect subsets. Instead, if the explanations provided were to be uninformative, we expect users to gather all their information from the original image, and to predict the model's output well only when the model made a correct prediction.

Our results (Table 1, Figure 6) show that *respondents to the PV survey were better at*

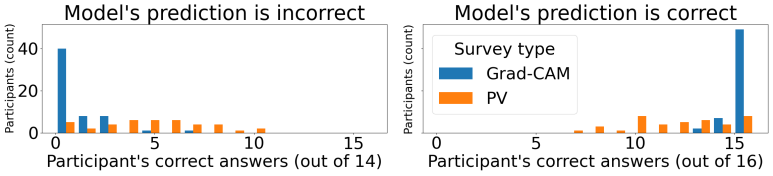

Figure 6: Histograms of participant's accuracy in terms of the number of times they determined the models prediction for cases where the model was *incorrect* in its prediction (left) and *correct* in its prediction (right).

determining which class the model was predicting when this made a mistake, case for which performance improved from 8.3% for Grad-CAM to 35.0% for PV. For samples that were instead correctly classified, due to the lower visual clarity of PV, Grad-CAM users were better at determining the predicted class by 95.9% compared to 75.5%.

We used the Mann-Whitney U test to check if there was a statistically significant difference in accuracy between users of the PV and of the Grad-CAM versions of the survey. For cases where the model was *incorrect*, the left tailed test (H_1: Acc(Grad-CAM)<Acc(PV)) confirms that the performance improvement achieved by PV is significant ($p = 2.3e^{-11}$, $Z = -6.58$). For cases where the model gave a *correct* prediction, the finding is in the opposite direction: a right tailed test (H_1: Acc(Grad-CAM)>Acc(PV)) suggests that, for this case, Grad-CAM was significantly better ($p = 2.45e^{-11}$, $Z = 6.57$).

Overall, the performance improvement seen in cases where the model is wrong is in line with the objective of providing insight for the purpose of debugging a model, that is, whenever its predictions are incorrect. For other use-cases, we recommend pairing PV and Grad-CAM explanations together to leverage their respective stregths. Lastly, these results suggest that the encoder and decoder are not separate entities, that is, that *the decoder has not merely learned to imitate the input*. Indeed, if this were the case, users would not have been able to correctly predict the model's output when the input was misclassified.

Invariance to decoder training: Since PVs depend on the result of the decoder training process, we investigate whether explanations remain consistent under different training conditions. On the one hand, this would verify that the decoder's output is strongly linked to $\mathcal{M}$'s latent representation, thus to the network's perception. On the other hand, this would ensure practical use of PV in different circumstances, such as that of using a different dataset.

The decoder $\mathcal{D}$ used in our previous experiment was trained using the same dataset used to also train the model $\mathcal{M}$ to be explained (PASCAL VOC). For this experiment, we train a new decoder by using a dataset composed of a random subset of 8000 ImageNet [26] images (a sample count very similar to the 8077 VOC images used for training $\mathcal{M}$). Since we are always using a ResNet-50 model to encode images, also ImageNet samples are cropped and resized to 224×224.

We provide qualitative evaluation of the results in Figure 7, showing that reconstructions are only marginally altered, and still possess all the key features necessary for explaining the model's mistakes. In particular, we note (Figure 7) how features resembling the train in the first sample remain (even though in a less clear way) also when the decoder is trained on ImageNet. In the second and fourth samples, the dog features are present also in the ImageNet trained decoder, and possibly even more clearly in the second sample. Finally, in the third sample, we see how features pertaining the TV monitor are visible in both decoders.

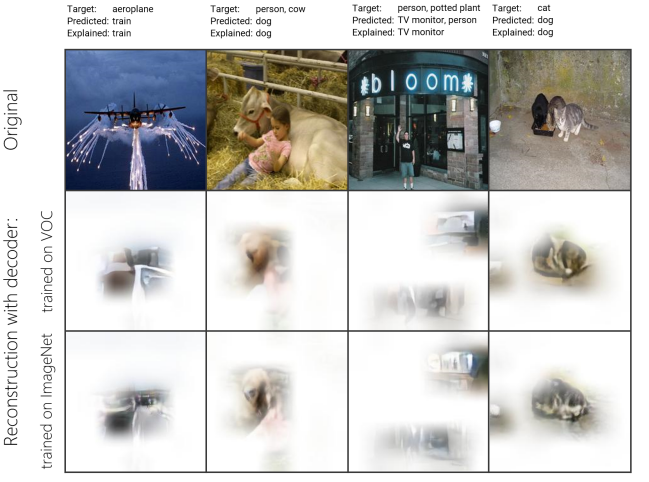

Figure 7: Reconstructions remain semantically similar between decoders, even when the decoder is trained using a different dataset than that used to train $\mathcal{M}$.

These results suggest that the semantics of the decoder's reconstructions are preserved in two successful training runs in different conditions. This, in turn, indicates that PVs are heavily determined by $\mathcal{M}$'s latent representation, rather than the specific decoder training. Furthermore, this experiments shows that it is possible to train $\mathcal{D}$ using a different dataset than that used to train $\mathcal{M}$, and that the two datasets might not even have the same class sets (as is the case for ImageNet and PASCAL VOC).

5 Conclusions and Future Work

We have introduced Perception Visualization (PV): the first method to provide explanations by exploiting a neural network to invert latent representations. PV provides semantically relevant information by learning to visualize the model's perception. As shown in our experiments, this allows PV to increase the user's performance in predicting the model's output in cases where the model misclassifies a sample, hence giving better insight on the model's functioning than what was previously achievable using only saliency maps.

An important future direction consists in improving reconstructions, firstly in terms of visual clarity, but also in terms of faithfulness to the latent representation. So far, we were not able to improve reconstruction quality with GANs due to the nature of the space that needs to be inverted, however, further studies regarding the properties of the latent representations may allow to overcome this problem. We also plan to investigate class-discriminative decoding to provide a broader insight on the model, and exemplar losses to generate prototype-based reconstructions [5]. We believe that better reconstructions could enable the application of PV also in medical imaging and in other critical domains.

Other promising directions regard the extension of PV to different tasks and architectures other than classification. For example, class-discriminative decoding could be used to explain recurrent/transformer networks used for image captioning [6]. In this context, we would generate sequences of PVs, one for each output token.

References

[1] Ahmed Alqaraawi, Martin Schuessler, Philipp Weiß, Enrico Costanza, and Nadia Berthouze. Evaluating saliency map explanations for convolutional neural networks: a user study. In *Proceedings of the 25th International Conference on Intelligent User Interfaces*, pages 275–285, 2020.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[4] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 839–847. IEEE, 2018.

[5] Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: deep learning for interpretable image recognition. *arXiv preprint arXiv:1806.10574*, 2018.

[6] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-Memory Transformer for Image Captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[7] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

[8] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.

[9] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *arXiv preprint arXiv:1602.02644*, 2016.

[10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[11] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

[12] Peter Hase and Mohit Bansal. Evaluating explainable ai: Which algorithmic explanations help users predict model behavior? *arXiv preprint arXiv:2005.01831*, 2020.

[13] 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.

[14] Jinkyu Kim, Anna Rohrbach, Trevor Darrell, John Canny, and Zeynep Akata. Textual explanations for self-driving vehicles. In *Proceedings of the European conference on computer vision (ECCV)*, pages 563–578, 2018.

[15] Todd Kulesza, Simone Stumpf, Margaret Burnett, Sherry Yang, Irwin Kwan, and Weng-Keen Wong. Too much, too little, or just right? ways explanations impact end users' mental models. In *2013 IEEE Symposium on visual languages and human centric computing*, pages 3–10. IEEE, 2013.

[16] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8, 2019.

[17] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[18] Aravindh Mahendran and Andrea Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3): 233–255, 2016.

[19] Pietro Morbidelli, Diego Carrera, Beatrice Rossi, Pasqualina Fragneto, and Giacomo Boracchi. Augmented grad-cam: Heat-maps super resolution through augmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4067–4071. IEEE, 2020.

[20] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. 2015.

[21] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 685–694, 2015.

[22] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv preprint arXiv:1806.07421*, 2018.

[23] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[24] 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.

[25] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215, 2019.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C.

Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[27] 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.

[28] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[29] Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz, and Antonio Torralba. Hoggles: Visualizing object detection features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, 2013.

[30] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 24–25, 2020.

[31] Jiaxuan Wang, Jenna Wiens, and Scott Lundberg. Shapley flow: A graph-based approach to interpreting model predictions. In *International Conference on Artificial Intelligence and Statistics*, pages 721–729. PMLR, 2021.

[32] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[33] Hang Zhao, Orazio Gallo, Iuri Frosio, and Jan Kautz. Loss functions for image restoration with neural networks. *IEEE Transactions on computational imaging*, 3(1):47–57, 2016.

[34] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.