ARL-TN-1325 ● AUG 2026

# Survey of Test Methods for Convolutional Neural Network Object Detection in New Environments

by Craig Lennon

## NOTICES

### Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TN-1325 ● AUG 2026


# Survey of Test Methods for Convolutional Neural Network Object Detection in New Environments

**Craig Lennon**
*DEVCOM Army Research Laboratory*

# REPORT DOCUMENTATION PAGE

| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED | |
|---|---|---|---|
| | | START DATE | END DATE |
| August 2026 | Technical Note | 3/1/2026 | 6/15/2026 |

**4. TITLE AND SUBTITLE**
Survey of Test Methods for Convolutional Neural Network Object Detection in New Environments

| 5a. CONTRACT NUMBER | 5b. GRANT NUMBER | 5c. PROGRAM ELEMENT NUMBER |
|---|---|---|
| **5d. PROJECT NUMBER** | **5e. TASK NUMBER** | **5f. WORK UNIT NUMBER** |

**6. AUTHOR(S)**
Craig Lennon

| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) | 8. PERFORMING ORGANIZATION REPORT NUMBER |
|---|---|
| DEVCOM Army Research Laboratory<br>ATTN: FCDD-RLA-IK<br>Aberdeen Proving Ground, MD 21005 | ARL-TN-1325 |

| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |
|---|---|---|
| | | |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**
DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

**13. SUPPLEMENTARY NOTES**
ORCID: Craig Lennon, 0000-0002-2229-2919

**14. ABSTRACT**
This report surveys and provides demonstrations of techniques that could be applied to test a convolutional neural network for application in a new environment using the Ultralytics YOLOv8n model. Army programs using such a network for object detection in an environment for which they lack sufficient training and test data can consider this survey to see if any of the methods demonstrated here have potential value to their program.

**15. SUBJECT TERMS**
Military Information Sciences, autonomy, artificial intelligence, convolutional neural network, test and evaluation

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES |
|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | C. THIS PAGE | | |
| UNCLASSIFIED | UNCLASSIFIED | UNCLASSIFIED | UU | 23 |

| 19a. NAME OF RESPONSIBLE PERSON | 19b. PHONE NUMBER *(Include area code)* |
|---|---|
| Craig Lennon | (520) 691-6117 |

**STANDARD FORM 298 (REV. 5/2020)**
*Prescribed by ANSI Std. Z39.18*

## Contents

**List of Figures** iv

**1. Introduction** 1

**2. Methods of Predicting Performance for YOLOv8n** 1

2.1 Activation Maps 1

2.2 Randomized Input Sampling 4

2.3 Concept Activation Vectors 5

2.4 Representational Influence 6

2.5 Natural Adversarial Perturbation 7

2.6 Synthetic Images 8

2.6.1 Stable Diffusion Inpainting 8

2.6.2 ControlNet 9

**3. Approaches to Validation** 12

**4. References** 14

**List of Symbols, Abbreviations, and Acronyms** 16

**Distribution List** 17

## List of Figures

Figure 1. Activation example: top-left original, top-right Ablation-CAM, bottom row Eigen-CAM components 1 and 2. ..... 2
Figure 2. D-RISE heatmaps with YOLOv8n bounding box around detected cats. ..... 4
Figure 3. Cat ears (positive example of concept) and dog (negative example). .. 5
Figure 4. Base image (upper left) and five images with high similarity. ..... 6
Figure 5. Original image (upper left) with added glare (upper right), rain (lower left), and distractor (lower right)..... 7
Figure 6. Background image (upper left) with two in-painted variants. ..... 9
Figure 7. Baseline image and blueprints from ControlNet. ..... 10
Figure 8. Blueprint (left) and Semantic Segmentation ControlNet output (right). ..... 10
Figure 9. Blueprint (left) and Canny Edge ControlNet output (right)..... 11
Figure 10. Blueprint (left) and Soft Edge ControlNet output (right). ..... 11
Figure 11. Blueprint (left) and Soft Edge output (right). ..... 11
Figure 12. Saliency of real (lower left) and synthetic (upper right) images using D-RISE..... 12

# 1. Introduction

Convolutional neural networks (CNNs) provide a means of object detection for robotic systems. Programs using such networks will need methods of predicting how the network will adapt to new environments not included in the original training and test sets. This report uses a You Only Look Once (YOLOv8n) object detector from Ultralytics (2026a) to demonstrate these methods. Section 2 briefly describes each method, and Section 3 discusses how such methods might be validated. This report does not consider networks with alternative architectures, such as visual transformers (Dosovitskiy et al. 2021), nor does it consider open world detectors with a CNN architecture such as YOLO world (Ultralytics 2026b). Blackbox methods such as randomized input sampling (Section 2.2) would likely work for such networks, as would generation of synthetic data (Section 2.6) and natural adversarial perturbation (Section 2.5). Other methods, such as activation maps (Section 2.1), may need modification to work, and concept activation vectors may not work for open world detectors (Section 2.3).

# 2. Methods of Predicting Performance for YOLOv8n

If one has data covering the objects and expected conditions from the environment in which the object detector will be deployed, the best option is to test against real data from that dataset. This report considers test methods that may be useful when data representing some (or all) classes are not available for the intended operational environment. All data used to demonstrate these techniques in this document is from publicly available datasets, most commonly from the Common Objects in Context (COCO) dataset (COCO 2017). Throughout the rest of this section, we will briefly describe a method and then demonstrate it. Readers who find a technique of interest for Army applications can follow the references provided for the technique or contact the author of this report.

## 2.1 Activation Maps

We start with two visual methods for understanding what regions of an image contribute to object detection. Class activation maps (CAMs), such as Eigen-CAM (Muhammad and Yeasin 2020) and Ablation-CAM (Desai and Ramaswamy 2020), can be applied to YOLOv8n to create heatmaps that provide a visual representation of which regions of the image were relevant to the network's detections. Figure 1 shows an example with an image and activation maps produced by Eigen-CAM and Ablation-CAM. Both algorithms are white box testing methods, requiring access to the network, but they do not require retraining or other modification of

the network. For each, the tester selects a layer of the network at which to apply the algorithm. In our examples, the algorithms were applied at layer 21 of YOLOv8n, at which point the image has been changed from a 640 × 640 image with three channels to a 20 × 20 image with 256 channels, which can be represented as a 400 × 256 matrix in which every row is a pixel with 256 channels. This far into the network, each pixel will have gone through convolutions in previous layers, each a weighted average of the features of neighboring pixels. Thus, the presence of heat on a pixel does not mean that the pixel of the original image was relevant, but rather that the relation of that pixel and its neighboring pixels was relevant. Still, one would expect that hot points of an image would be near objects detected in the image, and if the heatmap shows no relation to the detected object, then this would be cause for concern. Thus, the heatmaps based on activations deep within the network can serve as a tool for establishing that the network is paying attention to the region around the object it is detecting, but one should not assume that the hot spot on the image corresponds exactly to the pixels most relevant to the detection. In Section 2.2 we will look at a method that does aim to establish this type of pixel-specific correspondence to the original image as opposed to the region-specific correspondence of the methods in this section.

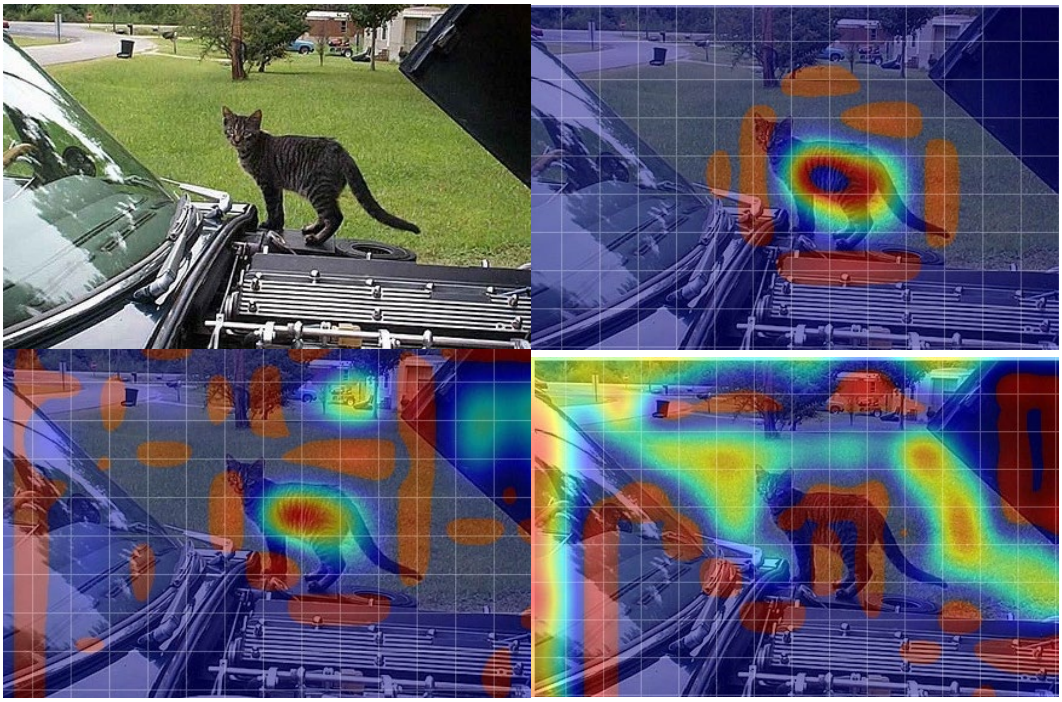

**Figure 1. Activation example: top-left original, top-right Ablation-CAM, bottom row Eigen-CAM components 1 and 2.**

The Eigen-CAM algorithm takes the output of a layer (e.g., layer 21) and finds the principal components (orthogonal eigenvectors, see Hastie et al. [2009]) of the previously mentioned 400 × 256 matrix. When a principal component is multiplied

by the output matrix, the resulting vector can be converted to a 20 × 20 grid that can cover the (reduced size) image as a heatmap. Figure 1 shows this heatmap scaled up to the size of the original image, with the bottom-left image representing the first principal component and the bottom-right representing the second principal component. Red regions were high value in the heatmap, corresponding to regions in which the values of the product of the matrix and principal component were high, so they can be considered regions of the image relevant to that component. It is worth noting that Eigen-CAM's maps do not depend on the class detected. Grid lines have been superimposed on the heatmaps to inform the reader's intuition as to what a decomposition of the image into 20 × 20 would look like, but the cells should not be imagined as corresponding position-wise with the representation at layer 21, but only as affording an understanding of the scale of the image representation at that layer. The image has also been cropped to focus on the cat.

Object class is relevant to our other example of a CAM-based algorithm, Ablation-CAM. While Eigen-CAM finds the patches of the image most salient to the 21st layer of the CNN, Ablation-CAM finds patches most salient to the 21st layer for the purpose of detecting a cat. The image is passed through the CNN to the 21st layer (or any alternative layer you choose) at which point it has been reduced to a 20 × 20 image with 256 channels. Then the algorithm proceeds to execute a loop which, in each iteration, zeros out (ablates) exactly one of the 256 channels, and then passes this modified image through the rest of the CNN, calculating the difference between the confidence score of the cat class between the original and modified image. Each of the 256 channels is weighted, with higher weights for channels that caused greater drops in confidence. Finally, in the original image each of the 256 channels is multiplied by its weight and summed to create the heatmap, which then has higher values in those regions contributing to confidence in the cat class. The resulting image is shown in the top right of Figure 1.

The top-right and bottom-left images of Figure 1 suggest the region around the cat is relevant to its detection, as one would expect. Examination of many class instances across environments would improve our confidence that the detector does in fact pay attention to the objects it is detecting rather than base its detection on other properties of the image. While it is easy to see how one might apply a class-specific CAM method like Ablation-CAM to develop confidence that a class is being appropriately detected, the application of Eigen-CAM is more difficult to see. We have presented it here as a complement to Ablation-CAM because it has the advantage of requiring less computation. Eigen-CAM can be run in real time with minimal additional cost, while Ablation-CAM required 256 iterations to establish its heatmap, so one could conceivably run Eigen-CAM along with the inference

algorithm, while Ablation-CAM took over 30 s to run per image. This tradeoff may make a class-agnostic method preferable for some applications.

## 2.2 Randomized Input Sampling

While the previous methods required access to the weights at a specified layer of the model, our next method is black-box, requiring access only to the input and output of the model. Detector Randomized Input Sampling for Explanation (D-RISE) (Petsiuk et al. 2021) masks random parts of the input image to determine which parts are most relevant to detection of the object and then produces a heatmap highlighting those regions. The output is visually similar to the class activation maps of Section 2.1 but has the following differences:

- It does not require access to the internal model as did the CAM methods that relied on access to activations at a specified layer.
- The pixels highlighted in the heatmap are the pixels relevant to detection, while in the CAM methods there was only correspondence by region.
- D-RISE requires thousands of iterations of masking and processing by the CNN to produce results while Eigen-CAM required one and Ablation-CAM required 256.

The D-RISE method proceeds by applying the detector to obtain a baseline and then generating random masks to cover parts of the image. Each masked image is then processed by the model and scored by how it changed classification confidence and placement of the bounding box. Masks that change confidence or bounding box placement are highly weighted, and a heatmap is produced by adding together the weighted masks. Three examples of D-RISE heatmaps are shown in Figure 2 where red pixels indicate high saliency and blue pixels indicate low saliency.

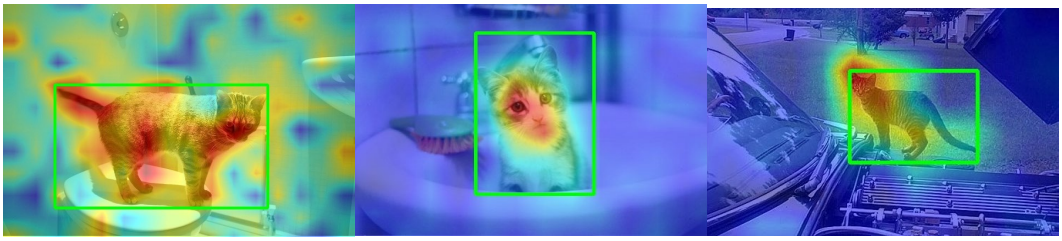

**Figure 2. D-RISE heatmaps with YOLOv8n bounding box around detected cats.**

This method is easier to interpret than those of Section 2.1 because it is based on the pixels in the original image rather than on the representation of the image in the network, but the computational cost is greater.

## 2.3 Concept Activation Vectors

We turn to a nonvisual approach of determining what properties of the input image contribute to the detection of a class in the image. Concept activation vectors (Kim et al. 2018) are an attempt to quantify the contribution of a human-interpretable property of an image to the CNN's detection of that image class. First, define a concept such as "cat ears" and select a group of images representing the concept and a set of images in which the concept does not appear, as shown in Figure 3. Next, select a layer at which the concept may be represented, or try a variety of layers. Layer 8 of the CNN was used for our cat ear example. At layer 8, the image has been reduced to an 80 × 80 image with 256 channels. For each channel, all pixels are averaged so that the result is a vector which is intended to represent the presence or absence of the concept. Then a linear classifier (e.g., logistic regression) separates the positive and negative examples resulting in a (linear) decision boundary, and the vector orthogonal to this boundary in the direction of the positive class is the concept activation vector (CAV). Finally, images including the class (e.g., a full cat and not just ears) are provided as input to the CNN, which calculates a confidence value for the class and the gradient vector representing the direction that the layer 8 output would be changed that would most increase the confidence (i.e., how the image would change to be more "cat-like"). The image outside the class bounding box is masked so that only cat-related activations are nonzero, and then the pixel values are averaged for each channel to obtain a class gradient, with which we take the dot product on the CAV to produce a directional derivative. The larger this directional derivative is, the more the concept contributes to the detection in that image; this can be averaged over a test set of images to estimate the dependence of the detection on the concept.

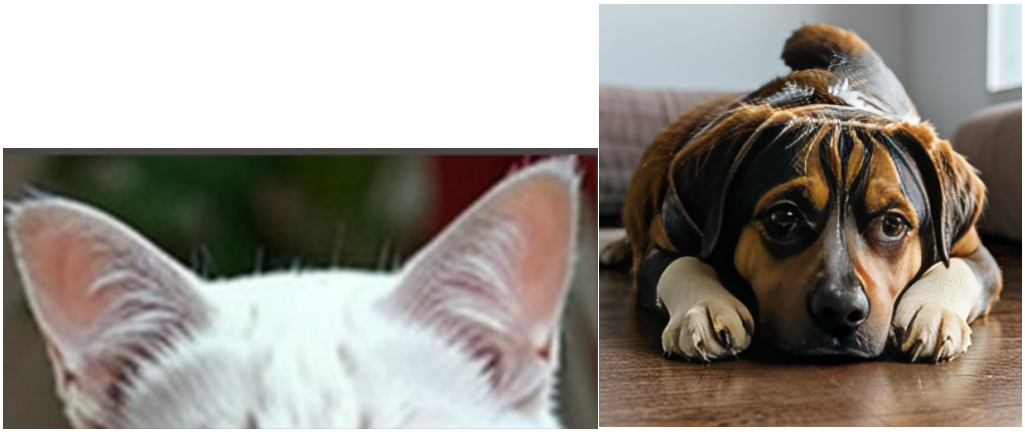

**Figure 3. Cat ears (positive example of concept) and dog (negative example).**

Tests with cat ears (Figure 3) found a CAV score of 0.298, indicating that the model is not strongly reliant on ears to detect cats, which accords with our D-RISE experiments as shown in Figure 2. The CAV approach requires a reasonable conjecture as to what features of the image contribute to detection, as well as representative positive and negative examples of the concept to provide for training. Like D-RISE and the CAM algorithms, it informs a tester as to what image properties were relevant to detection in environments for which data exists, which provides evidence as to how the algorithm may perform in new environments.

## 2.4 Representational Influence

Activation maps and randomized input sampling provide visual evidence as to what properties of the class are relevant to classification at a specified layer of the network, while the CAV method provides a numerical measurement as to how relevant a selected feature is. A different approach is to take an image as input and identify which members of our dataset are the most similar. Representational influence applies a cosine similarity metric to the input image features at a specified layer of the CNN (layer 21 in our example) to determine which of a specified set of images are most similar to the input image.

In Figure 4, the upper-left image was compared to images of the class' cat, dog, and bear from the COCO 2017 validation dataset (COCO 2017), and the five most similar images are shown in Figure 4. The images span a variety of environments, but all clearly show the cat's face, and, more importantly, all are cats with none being dogs or bears.

**Figure 4. Base image (upper left) and five images with high similarity.**

In this instance, the resulting set of most similar images does not have an obvious interpretation. One can imagine applying this technique by taking an image of a class from a new environment and observing that the most similar images all have a property in common. For example, they might all be from one environment in the training set or might show the object in a similar pose. In such a case, the most similar images might provide valuable information about the new environment. However, given the increasing complexity of CNN, it is likely that the images seen as most similar by the CNN will have no common properties discernable to a human.

## 2.5 Natural Adversarial Perturbation

Some natural environmental effects, such as motion blur, glare, and others, may be reproduced by modifying the image to introduce surrogates for such effects. If one only needs to modify the image (and no other sensor data), one can manipulate image effects via a variety of commercially available products, although here we illustrate the method with the freely available open computer vision python package (Python Software Foundation 2026a). The OpenCV effects for glare and rain shown in Figure 5 do not faithfully reproduce natural conditions but illustrate how perturbations of the image may change detections.

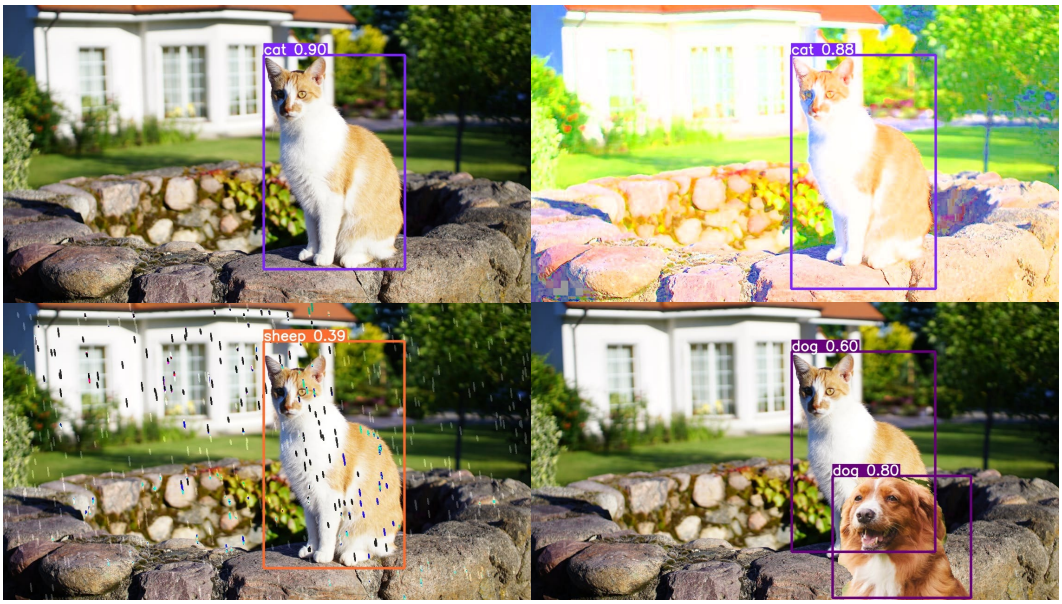


**Figure 5. Original image (upper left) with added glare (upper right), rain (lower left), and distractor (lower right).**

The lower-right image in Figure 5 contains a distractor object (a dog) added to the image and shows the effect of adding this distractor on the detection of the cat. These methods might be applied to testing against changing conditions in an environment

from which one has test data, but they would not work for an environment in which one has no example images of a class. The methods of the following section could be applied when one has no examples of a class in the new environment.

## 2.6 Synthetic Images

Creating an instance of a class in an environment from which you have no instances of that class requires some method of synthesizing the image. One could use a visually realistic simulation for this purpose, if one is available. In this section we assume that no such simulation is available and examine methods of generating images with Stable Diffusion (Rombach et al. 2022).

### 2.6.1 Stable Diffusion Inpainting

These methods all apply a python package for Stable Diffusion (Python Software Foundation 2026b). Figure 6 shows examples of a background image (upper left) that was modified to introduce a polar bear via inpainting, in which part of an image was removed and a generative model was prompted to fill in the missing region, for example, with an object. In the image on the upper right, a rectangular region was masked in the base image (upper left), and a prompt was provided to create "a photorealistic polar bear" in the missing space. Notice that the polar bear has a deformed head, and that buildings were removed from the image due to the rectangular mask. In the lower-left image, a real image of a polar bear was provided as input, with the bear masked, and the bear-shaped mask placed on the base image from the target environment. There was no masking of the background image except for the bear, so the buildings are not altered, but the bear is still deformed. The prompt provided was, "A realistic white polar bear walking naturally on gravel in front of grey buildings, 4k." A more detailed prompt was provided for the second image to ensure that the prompt would conform to the mask provided, which was a walking polar bear. A polar bear was used in this example because a bear is an object detectable by YOLOv8n, and a polar bear is not typically seen in environments like the one pictured here, so it serves as an example of an object in a new environment.


**Figure 6. Background image (upper left) with two in-painted variants.**

Both bears were detected, but a desire to have more control over what the bear was doing prompted an examination of methods, which afford more control in the generation of the image.

### 2.6.2 ControlNet

ControlNet (Zhang et al. 2023) constrains the generation of synthetic images by using a baseline image to create a visual prior for the synthetic image. This baseline image determines the geometry of the synthetic image by creating a blueprint of outlines, general posture, or semantics of the environment. In Figure 7, the baseline image of a panda walking through a forest is shown in the upper left. The upper-right image shows the blueprint created by Canny Edge variant of ControlNet, and the general posture blueprint created by Soft Edge variant of ControlNet is shown in the lower right. The lower-left image is the blueprint created by the Semantic Segmentation variant ControlNet. A blue box was added to this image as an outline for the bear.

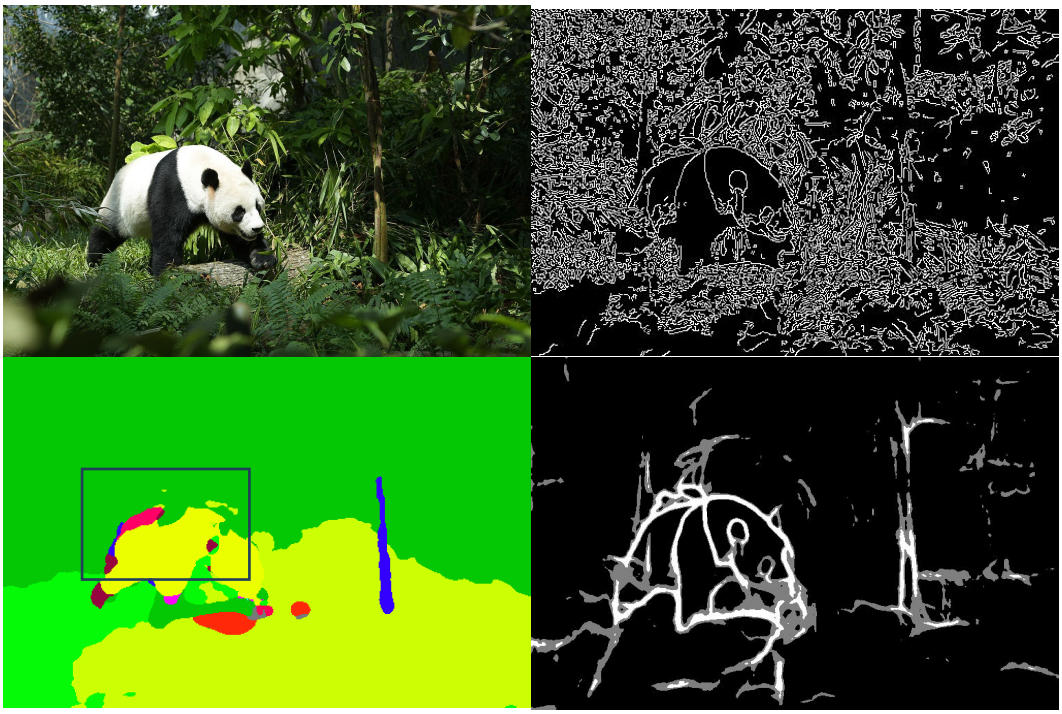

**Figure 7. Baseline image and blueprints from ControlNet.**

Figures 8–11 show the blueprint image next to the outputs provided by the Semantic Segmentation (Figure 8), Canny Edge (Figure 9), and Soft Edge (Figure 10) variants of ControlNet using the prompt "A photorealistic polar bear in a forest, high resolution, 8k, national geographic." For Soft Edge, a second prompt, "A photorealistic polar bear in a snowy arctic circle forest, high resolution, 8k, national geographic," was used in Figure 11 to compare whether the polar bear detection, which failed in Figure 10, would succeed in a different environment, which it did.

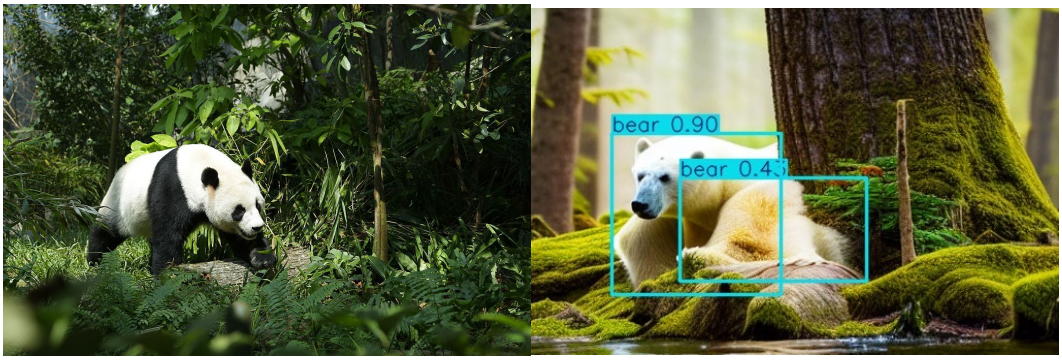


**Figure 8. Blueprint (left) and Semantic Segmentation ControlNet output (right).**


**Figure 9. Blueprint (left) and Canny Edge ControlNet output (right).**


**Figure 10. Blueprint (left) and Soft Edge ControlNet output (right).**


**Figure 11. Blueprint (left) and Soft Edge output (right).**

Each of the synthetically generated polar bears has some type of deformity, but each is recognizable as a polar bear. However, whether an image looks reasonable to a human may not be a suitable method of determining whether the image tests how the CNN will perform in a new environment. Figure 12 shows the D-RISE algorithm applied to images from Figures 10 and 11. The misdetection of the polar bear as a bird shows a saliency pattern similar in some respect to the correct detection of the panda, but substantially different from the correct detection of the polar bear. A comparison of saliency across real and synthetic images could provide evidence of the utility, or lack thereof, of the synthetic data generation process,

though this one example does not provide enough information for this purpose. It is worth noting, as a caveat, that the panda image and its synthetic variants may be detected differently if the image was processed differently prior to going into the model. Processing of an image prior to providing it as input to a model is not the subject of this report, but it is important, and any of the detections demonstrated in this report might have changed with different processing.

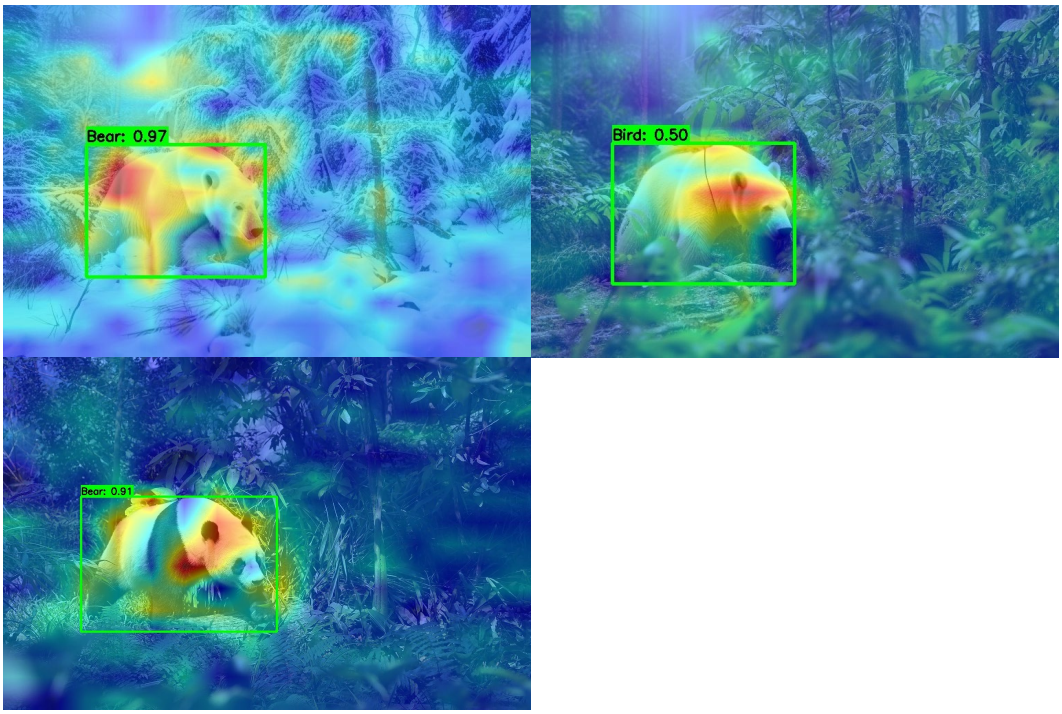


**Figure 12. Saliency of real (lower left) and synthetic (upper right) images using D-RISE.**

The next section addresses how one might validate the methods we have described to predict performance in a new environment.

## 3. Approaches to Validation

The CAM methods of Section 2.1 provide a basic test of what region of the image is relevant to object detection, while the methods of Sections 2.2 and 2.3 provide visual and quantitative information as to what parts of the object are important to its detection. Each of these methods provides insight into the tester from which inferences might be drawn as to how the detector will perform in a new environment. As these are methods of explainability, they may not require validation, especially since the validation would be achieved by obscuring areas of the image to see it if changes detection, which D-RISE already does.

Section 2.4 provided a method by which one might identify relationships between images from a new environment and existing datasets. This general method cannot

be validated, but specific relationships hypothesized based on representational similarity might be validated by a means suitable to the hypothesis.

The methods of Sections 2.5 and 2.6 relied on synthetic data generation. There are two possible approaches to validating them. First, if one has data from the new environment, but lacks data for one class (or more), one could attempt to validate synthetic data by using real data from other classes. Supposing one has data from class A in the new environment but not class C, one could generate synthetic data from both classes using the same method and then compare detection results on the real and synthetic data from class A. The more classes one can compare to validate the method, the more confidence one might have in it. Of course, one cannot be certain the generation of different classes will yield similar results. One could go further by also generating synthetic data from environments where one does have data for all classes for a comparison of success in generating each class, but one cannot be certain that generation across environments will yield similar results. By doing both comparisons, one might achieve greater confidence, but certainty will still be elusive.

If a visually realistic simulation is available, this could be used in place of, or to supplement, real data for the validation process. If one has no data from the new environment and no simulation, one could analyze the real and synthetic data with the methods shown in Sections 2.1–2.3 to see if the results are similar for real and synthetic data. Similarity of results when applying D-RISE to real and synthetic data, for example, would be evidence in favor of the belief that performance of the detector will be similar in the environment which the synthetic data represents. Finally, experience generating synthetic data from new environments and then testing detection in that environment would be valuable for identifying what methods of generating synthetic data provide the best results, and what process one should use in applying the method.

## 4. References


[COCO] Common Objects in Context. Common objects in context 2017 validation images. COCO Consortium; 2017. https://cocodataset.org/#download

Desai S, Ramaswamy HG. Ablation-CAM: visual explanations for deep convolutional network via gradient-free localization. 2020 IEEE Winter Conference on Applications of Computer Vision (WACV); 2020; Snowmass, CO. p. 972–980.

Dosovitskiy A et al. An image is worth 16x16 words: transformers for image recognition at scale. arXiv; 2021 June 3. arXiv:2010.11929v2. https://doi.org/10.48550/arXiv.2401.17270

Hastie T, Tibshirani R, Friedman J. The elements of statistical learning: data mining, inference, and prediction. 2nd ed. Springer; 2009 Feb.

Kim B et al. Interpretability beyond feature attribution: quantitative testing with concept activation vectors (TCAV). Proceedings of the 35th International Conference on Machine Learning, PMLR. 2018;80:2668–2677.

Muhammad MB, Yeasin M. Eigen-CAM: class activation map using principal components. 2020 International Joint Conference on Neural Networks (IJCNN); 2020; Glasgow, UK. p. 1–7.

Petsiuk V et al. Black-box explanation of object detectors via saliency maps. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition; 2021.

Python Software Foundation. opencv-python 5.0.0.93; 2026a [accessed 2026 June 24]. https://pypi.org/project/opencv-python/

Python Software Foundation. diffusers 0.39.0; 2026b [accessed 2026 June 20]. https://pypi.org/project/diffusers/

Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-resolution image synthesis with latent diffusion models. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2022. p. 10684–10695.

Ultralytics Inc. Explore Ultralytics YOLOv8, performance metrics. c2026a [accessed 2026 June 24]. http://docs.ultralytics.com/models/yolov8#performance-metrics

Ultralytics Inc. YOLO-World Model. c2026b [accessed 2026 Aug 7]. https://docs.ultralytics.com/models/yolo-world

Zhang L, Rao A, Agrawala M. Adding conditional control to text-to-image diffusion models. arXiv; 2023 Nov 26 [accessed 2026 June 24]. arXiv:2302.05543v3. https://doi.org/10.48550/arXiv.2302.05543

## List of Symbols, Abbreviations, and Acronyms

| | |
|---|---|
| CAM | Class Activation Map |
| CAV | Concept Activation Vector |
| CNN | Convolutional Neural Network |
| COCO | Common Objects in Context |
| D-RISE | Detector Randomized Input Sampling for Explanation |
| YOLO | You Only Look Once |

| | |
|---|---|
| 1 (PDF) | DEFENSE TECHNICAL INFORMATION CTR DTIC OCA |
| 1 (PDF) | DEVCOM ARL FCDD RLB CB TECH LIB |