

# Trends and Overview of Evaluation Techniques in Image Recognition AI

Ryu Kidokoro, Lawrence Earl Carandang

## [Summary]

With the rapid evolution of artificial intelligence (AI) and its expanding application in fields where accuracy is critical, such as healthcare and quality inspection, comprehensive evaluation of AI models and their development processes is essential. This article overviews practical methods for evaluating image recognition AI and demonstrates the role of eXplainable AI (XAI) within an evaluation framework that encompasses not only accuracy but also interpretability and reliability. We demonstrate the need for evaluation at the pipeline level and introduce application of representative XAI methods, including Local Interpretable Model-Agnostic Explanations (LIME) and Class Activation Maps (CAMs), to inference interpretation and model improvement. We also discuss robustness evaluation through uncertainty quantification using Test-Time Augmentation and Bayesian networks. Furthermore, we discuss the outlook for challenges related to applying these evaluation methods to diverse tasks and real-world data, as well as integration of evaluation metrics with development objectives.

## 1 Introduction

Recent advances in deep learning have led to rapid proliferation of AI across a wide range of fields, including business and healthcare. In particular, generative AI and large language models (LLMs) have become central to corporate IT strategies. At the same time, the opacity of AI decision-making poses a significant challenge for ensuring reliability, making it urgent to establish evaluation techniques that guarantee fairness and accountability. Against this backdrop, the importance of a comprehensive evaluation framework centered on eXplainable AI (XAI) is growing.

XAI has drawn attention as a technology that resolves the black-box problem of AI and makes the basis for its decisions understandable. In critical sectors such as healthcare and finance, visualizing the rationale behind AI decisions is indispensable from both a regulatory and ethical standpoint. However, evaluation of current AI models tends to focus on accuracy and speed, making it difficult to identify the causes of misjudgments and bias. XAI methods such as LIME (Local Interpretable Model-agnostic Explanations)<sup>1)</sup> and SHAP (SHapley Additive exPlanations)<sup>2)</sup> have enabled contribution analysis, but the evaluation of explanation fidelity and stability remains in its early development stages. Trustworthy AI requires not only performance evaluation but also comprehensive assessment techniques that integrate explanation quality and uncertainty.

This article introduces novel evaluation metrics and

methods for quantifying interpretability, fairness, and explanation reliability through XAI.

The Microsoft COCO dataset (© 2014 Microsoft. Licensed under the Creative Commons Attribution 4.0 International License.) was used as training data for the detection model<sup>3)</sup>.

## 2 AI Model Evaluation

Building on the AI reliability challenges outlined in Section 1, this section introduces evaluation techniques for AI models.

There are various quantitative and qualitative perspectives in evaluation; for example, accuracy quantitatively assesses model performance on tasks such as classification. Training time indicates how quickly an AI model can be created and is an important factor in time-constrained operations. Furthermore, the validity of the decision rationale is a qualitative perspective that reflects the reliability and explainability of the results produced by an AI. These perspectives are indispensable for making a comprehensive assessment of AI performance.

### 2.1 AI Development Pipeline Evaluation

While Section 2 addresses the evaluation of “AI models,” the evaluation scope is not limited to the model itself. This section introduces the “AI development pipeline” as an additional subject of evaluation. A development pipeline (or simply an “AI pipeline”) refers to the entire sequence of steps from AI model development to just before deployment, and consists of four components: developers; datasets; AI models;

and inference results, along with six processes: data collection; pre- and post-processing; training; inference; and performance calculation. Figure 1 shows how the components and processes are interconnected.

A reactive development approach is common in traditional machine learning where a model is built first, its accuracy is then evaluated, and improvements are made based on the results. However, this approach can lead to unnecessary data collection and preprocessing, especially when feature selection during the data preparation stage is insufficient, resulting in reduced development efficiency. To address this challenge, it is important to identify features and noisy data that either do not contribute to or negatively impact performance through model evaluation. This enables optimization of data design, including feature selection, and makes it possible to build high-accuracy models even with limited data.

In this way, model evaluation serves not merely as a performance check, but as an important guide for data design and model improvement.

## 2.2 Explainable AI (XAI)

Recent worldwide research has aimed at interpreting black-box AI models to improve reliability and performance. One outcome of such research is eXplainable Artificial Intelligence (XAI). XAI refers to techniques and methods that present the basis for AI decisions and predictions in a form that people can understand. It serves as one approach to AI evaluation by comparing features that AI focuses on with human perspectives. This article introduces several widely used XAI techniques.

### 2.2.1 Global Explanation and Local Explanation

XAI methods are generally classified into two categories based on their scope of explanation: global explanation and local explanation<sup>1</sup>. Global explanation analyzes the overall behavior and dominant internal tendencies of an AI model, evaluating model behavior using multiple test data points. A representative example is the decision tree. Conversely, local explanation aims to understand the reasoning behind individual prediction results and to identify which components of the input contributed most. Unlike global explanation, local explanation evaluates model behavior using only a single test data point. A representative example is the saliency map<sup>1</sup>. This article focuses on local explanation and provides a detailed discussion of the saliency map as a representative method in Section 3.1 What Is a Saliency Map?

### 2.2.2 Model-Specific and Model-Agnostic Methods

XAI methods can also be classified according to whether they have access to a model’s internal parameters (such as weights and gradients).

Model-specific methods use internal parameters to analyze outputs between network layers, and can therefore only be applied to specific model architectures. For example, XAI techniques for recurrent neural networks (RNN) cannot be applied to convolutional neural networks (CNN) without adjustment, as their architectures and outputs are different.

On the other hand, model-agnostic methods only analyze changes in input and output, allowing them to be applied to a wide range of models, regardless of their internal structure.

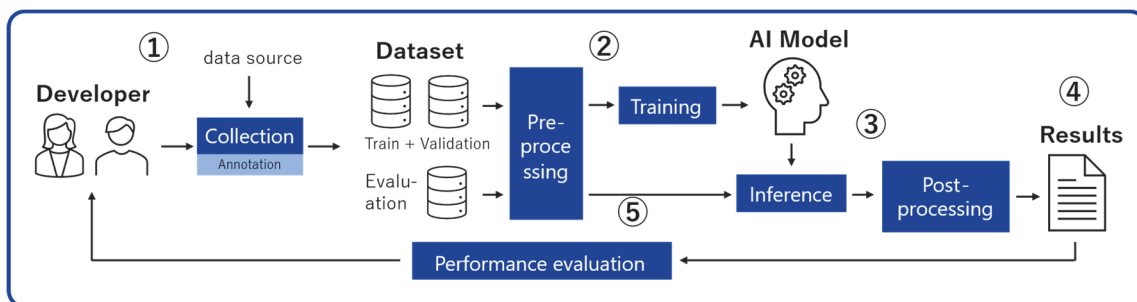


Figure 1 AI Pipeline. ① Developers acquire data from various sources and, where necessary, label (annotate) data to create the dataset. The dataset is then divided into three subsets: training data, validation data, and evaluation/test data. ② Training and validation data are fed through preprocessing to train the AI model (AI training). ③ Evaluation data is fed through preprocessing and is used for inference by model trained in ②. ④ Where necessary, results from ③ are fed through postprocessing to obtain the final results and measure model performance. ⑤ Where necessary, the experiment is repeated with different conditions based on model performance.

<sup>1</sup> Heat maps are often used to visualize saliency maps. In fact, a saliency map is often referred to as a “heat map” rather than a “saliency map.”

Since model-specific methods make use of internal parameters, they enable more fine-grained explanation and evaluation by examining specific parts of the AI. For example, by examining a specific layer of an image recognition neural network, one can verify whether the AI has appropriately learned features such as the shape, color, and size of objects. Model-agnostic methods, on the other hand, offer the advantage of being applicable to any model regardless of internal structure. It is therefore necessary to select the appropriate method depending on whether the model's internal mechanisms and parameters are accessible. Specific quantification methods for uncertainty are discussed in detail in Section 4 Evaluation of Model Robustness Using Uncertainty Calculation.

### 2.3 Evaluation Metrics

Accuracy is the most important and primary metric for measuring performance, but it is not the sole evaluation criterion. AI models and their development pipelines can also be assessed from the perspectives of transparency, consistency, data quality, and computational efficiency. These evaluation perspectives are worth considering because they are directly or indirectly related to model accuracy.

Interpretation of reasoning logic (transparency) evaluates the possibility that even a correct prediction may be based on erroneous information. A typical example is the case where an image classification model predicts “polar bear” based on snow in the background—a phenomenon called Clever Hans effect<sup>5)</sup>. As noted earlier, methods such as XAI can be used to identify the features a model is focusing on and verify whether they align with the features a human would notice. This assesses the effectiveness of the training method while also enhancing user trust.

Quantifying model robustness (consistency) evaluates variability caused by unavoidable factors, such as lighting and angle in input data such as camera images. While humans readily adapt to such changes, AI models' decisions are susceptible to even minor differences in pixel values. A good model must be capable of stable detection regardless of changes in conditions. Robustness is a metric for measuring this stability, and serves as a basis for evaluating under what conditions a model can be considered reliable.

Training Data Attribution (TDA) analyzes the impact of mislabeled or redundant data to evaluate data quality. For example, in an image classification task, if a photo of a cat is

incorrectly labeled as ‘dog’ and the model is trained on that image, it may cause degraded performance. To address such issues, TDA is used to calculate the importance (weight) of each training data point when updating model parameters. Based on this importance, it is possible to identify any data that either does not contribute to or may hinder model training and exclude it from the dataset.

Measurement of a model's computational efficiency (efficiency) evaluates whether the model's size can be optimized while maintaining accuracy. The higher the model's complexity and performance, the larger its file size and the greater the computational resources required to run it. Techniques such as quantization<sup>6)</sup> and distillation<sup>7)</sup> are means of simplifying a model while preserving its accuracy; computational efficiency can then be evaluated by comparing model size before and after applying these methods.

### 2.4 Application of XAI Techniques to Performance Improvement

In addition to providing developers with ideas for hyperparameter tuning, this article introduces several ways to directly leverage evaluation results as a form of feedback loop within the AI pipeline. For example, during development of an object detection model, a saliency map (discussed in detail in Section 3) can be computed for specific training images and compared against the ground truth (reference data) (Figure 2). This comparison result can be used as an additional loss value for use in backpropagation, serving as an example of loss/feature/gradient augmentation through XAI methods. Saliency maps can also be used to refine inference results. For example, it is possible to apply processing such as lowering the confidence score for images wherein the detected bounding box does not align with the high-saliency region on the saliency map.

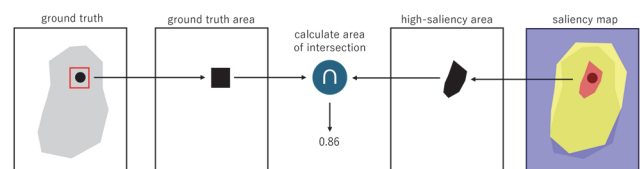


Figure 2 Example of a method for comparing ground truth with a saliency map. The intersection area (Intersection over Union, IoU) is computed between the bounding box area of the ground truth and the high-saliency area extracted by the saliency map.

Meanwhile, as discussed in the previous section, TDA can be applied to dataset design. It is widely known that selection of training data significantly influences model

performance, and dataset design using TDA is a means of quantitatively evaluating the quality and informativeness of training samples. This makes it possible to clearly identify the data that is most beneficial for the model.

### 3 Interpretation of Reasoning Logic Using Saliency Maps

This section introduces saliency maps for evaluating AI in object detection.

#### 3.1 What Is a Saliency Map?

A saliency map is an image showing the data features that are most attended to during inference. ‘Attended’ here refers to the features that contribute to the model’s output of an inference result. Figure 3 shows a saliency map visualized using Eigen-CAM (Eigenvalue-based Class Activation Mapping)<sup>8)</sup> applied to an image of a dog. As shown, the basis for the ‘dog’ classification was determined from the features at the center of the dog.

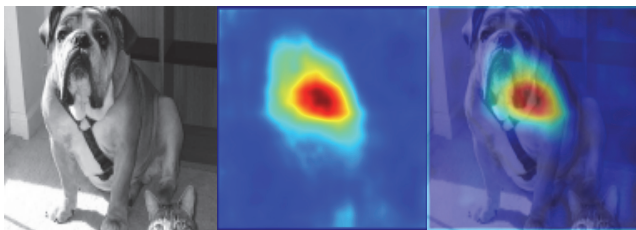


Figure 3 (Left to right) Input image, a saliency map, and input image overlaid with a saliency map

#### 3.2 Saliency Map Generation Techniques

While there are many methods for generating a saliency map, this article introduces three: the model-agnostic LIME, the model-specific Eigen-CAM, and D-CLOSE (Detector-Cascading multiple Levels of Segments to Explain)<sup>4)</sup>.

##### 3.2.1 LIME

LIME is a method that generates a set of samples by randomly perturbing parts of the input data and obtaining the model’s confidence scores for those samples, thereby constructing a local explanation model. Specifically, it performs linear regression using the perturbed data features as explanatory variables and the model’s predicted confidence scores as the objective variable in order to estimate the importance of each feature. If the perturbed region is important for the prediction, the confidence score will decrease; otherwise, we can expect that the confidence score will be the same. For object detection models, a method that masks the target region by replacing all pixel values with 0 can be

adopted as the perturbation when generating images visualized with LIME. An example of the result is shown in Figure 4(b), which suggests that the model determines the ‘dog’ object based on features such as the face, head, and legs.

##### 3.2.2 CAM/Grad-CAM/Eigen-CAM

CAM (Class Activation Mapping) is a technique that leverages the structure of CNNs (Convolutional Neural Networks)<sup>9)</sup> to visualize which regions in an image contributed to the classification. The algorithm generates a map (activation map) that highlights important regions in the image by multiplying the output of the final convolutional layer of the CNN by the weights of the classification layer.

A weakness of CAM is that it is constrained by model architecture. Specifically, it can only be applied to CNNs that have a classification layer that uses Global Average Pooling (GAP), which limits its flexible application to many existing models. This led to the proposal of Grad-CAM (Gradient-weighted Class Activation Mapping). Grad-CAM is a method that uses gradients with respect to the model’s output to weigh the feature map of the final convolutional layer, thereby visualizing the regions of attention within the image<sup>10)</sup>. This makes it applicable to a wider range of CNN-based models without dependence on a specific model architecture.

For tasks such as object detection, Eigen-CAM is particularly effective. It is a method that performs principal component analysis (PCA) or singular value decomposition (SVD) on a feature map to extract and visualize eigenvectors. This enables extraction of regions to which the model pays attention in a class-independent manner.

In addition to Eigen-CAM, there have been various derivatives and improvements proposed for CAM, such as Cluster-CAM<sup>11)</sup> and Expected Grad-CAM<sup>12)</sup>. These methods aim to improve processing speed and accuracy by clustering feature maps and refining gradient calculations. CAM-based methods are an important means of presenting the basis for a model’s decisions in a form that people can understand, serving as a technology within eXplainable AI (XAI) that continues to attract significant attention.

Figure 4(c) shows the image visualized by Eigen-CAM, showing that the model determines the ‘dog’ object based on features around the face.

##### 3.2.3 D-CLOSE

The D-CLOSE method achieves visualization by

repeatedly obtaining model predictions for a target image using multiple perturbation regions of different sizes, and then integrating those results. While LIME applies perturbations at a single segmentation granularity and generates explanations using a local linear model, D-CLOSE applies perturbations at multiple scales ranging from coarse to fine segmentation, making it possible to simultaneously capture the influence of both broad and detailed features. Figure 4(d) shows the image visualized by D-CLOSE, suggesting that the model determines the ‘dog’ object based on the face and head features.

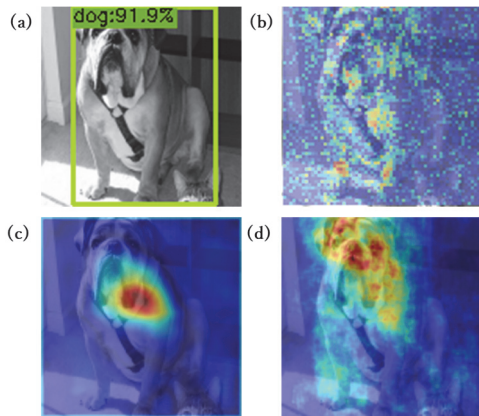


Figure 4 (a) Input image with inference result (b) Saliency map generated by LIME (c) Saliency map generated by Eigen-CAM (d) Saliency map generated by D-CLOSE

## 4 Evaluation of Model Robustness Using Uncertainty Calculation

This section introduces a method for quantifying uncertainty, a metric that indicates the reliability of model predictions.

Unlike humans, AI models cannot answer ‘I don’t know’ when presented with inputs they have not been trained on, and will always return some prediction. For example, if a lion image is given to an image classification model that has only been trained on images of cats and dogs, there is a high possibility that it will misclassify the image as ‘cat’ with high confidence, despite never having been trained on lions.

One metric for evaluating the reliability of predictions is ‘uncertainty.’ In object detection, while confidence indicates the probability of an object’s existence, uncertainty represents the reliability of the prediction itself. Following the earlier example, an inference without a proper basis (i.e., a prediction of ‘cat’ for a lion image) may have high uncertainty, even if the model’s confidence score is high.

The simplest method for measuring uncertainty is to have the model make multiple predictions on the same input under conditions that do not alter the essential nature of the input data. For example, uncertainty can be quantified by observing the variance in predictions obtained by modifying the input image’s color or size. Common metrics for uncertainty include confidence intervals, range of variation, variance, and standard deviation.

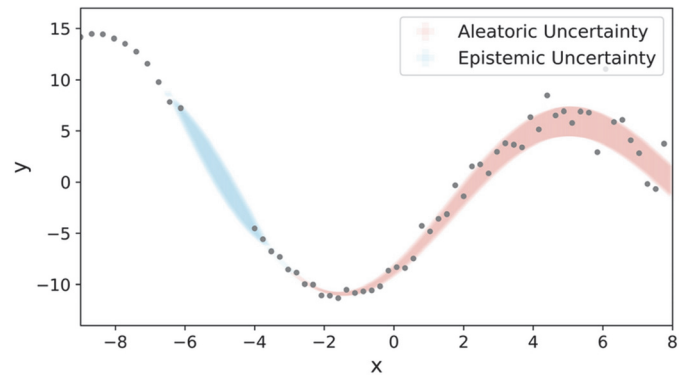


Figure 5 Epistemic uncertainty and aleatoric uncertainty<sup>13)</sup>. Due to insufficient data, the behavior between  $x=-6$  and  $x=-4$  cannot be modeled, giving rise to epistemic uncertainty. Meanwhile, due to variance in  $x=0$  to  $x=8$ , the actual behavior becomes unknown. As a result, the model’s prediction becomes a range rather than a fixed value, giving rise to aleatoric uncertainty.

Uncertainty can be broadly divided into two types: epistemic uncertainty and aleatoric uncertainty<sup>14)</sup>. The former arises from data outside the training distribution or from inappropriate model selection, and can be mitigated by adding data or improving the model. The latter stems from noise and randomness inherent in the environment, and is difficult to mitigate unless the environment itself is changed. Figure 5 visualizes both types in the context of a regression task.

As mentioned previously, a common method for quantifying uncertainty is to obtain multiple inference results for the same test input. The following sections introduce several methods classified according to whether or not modification of the network’s internals is required (model-specific or model-agnostic).

### 4.1 Model-Agnostic Methods

A common method for estimating uncertainty is to train multiple models on the same dataset with different initial values, hyperparameters, or architectures—a technique known as Deep Ensembles<sup>15)</sup>. By varying the training conditions, each model follows a different training trajectory, and

if the models converge to different points, their inference results may also differ. Assuming high-quality training data was used, high uncertainty with respect to unknown data indicates that the model has failed to correctly capture the important features—that is, it reflects epistemic uncertainty.

A related technique is bootstrap aggregating (bagging), which involves training multiple models using different subsets drawn by sampling with replacement from the training data (bootstrap sets), without changing the model architecture<sup>16)</sup>.

However, creating multiple models is not always practical. As a single-model approach, Test-Time Augmentation (TTA) allows evaluation of an existing model without requiring any additional training processes<sup>17)</sup>. In TTA, multiple transformations, such as affine transformations (e.g., translation and rotation) or recoloring operations (e.g., contrast enhancement), are applied to the input being evaluated, and each transformed version is passed through the model to obtain multiple inference results. Figure 6 shows several examples of such transformations.

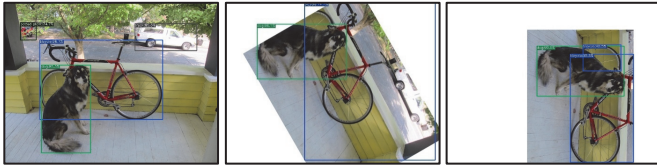


Figure 6 (Left to right) Original image, Transformed image #1, Transformed image #2

By exploring a range of transformation patterns and recording detection confidence with respect to the transformation, the degree of model uncertainty can be visualized as a graph. This makes it possible to pinpoint the model's weaknesses and appropriately adjust the training methodology.

Figure 7 shows how the detection confidence for the 'dog' object in the left image of Figure 6 varies with image rotation angle ( $-150^\circ$  to  $150^\circ$ ). At  $0^\circ$  (original image), the confidence is 0.95, but drops to 0.64 at a rotation of  $78^\circ$ , and detection fails entirely at approximately  $100^\circ$ . This demonstrates that the model is not invariant to rotation and that such transformations may affect its performance.

While this graph can itself be treated as a quantification of uncertainty, when a single metric is required, 'flatness' measures such as standard deviation ( $\sigma$ ) or maximum slope are effective. This article uses  $\sigma$ . Since confidence is bounded within the interval  $[0,1]$ , its  $\sigma$  is constrained to the interval

$[0,0.5]$ . Accordingly, Eq. (1) for expressing uncertainty as a percentage is defined as:

$$\text{Uncertainty} = 2\sigma \times 100\% \quad (1)$$

Computing the  $\sigma$  of the confidence values in Figure 7 yields  $\sigma = 0.2689$ , and from Eq. (1), the uncertainty is 53.78%. Therefore, with respect to the detection confidence of 0.95 for 'dog' in the left image of Figure 6, we can say that this prediction is approximately 53.8% uncertain.

The uncertainty metric reflects a model's stability and consistency rather than its performance. Accurate evaluation requires consideration of both. A model that achieves high performance but exhibits high uncertainty may be failing to capture essential features and instead relying on background cues; possible causes include insufficient diversity in the training data, among various other factors. Conversely, when both performance and uncertainty are low, this suggests that the model is overly simplistic, making a review of its architecture important. The ideal model achieves both high performance and low uncertainty.

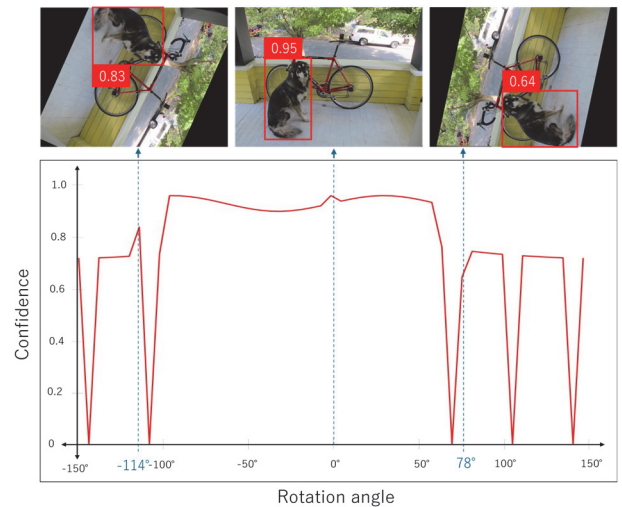


Figure 7 Image Rotation Angle vs. Detection Confidence for 'Dog'

## 4.2 Model-specific Methods

Compared to model-agnostic methods, model-specific methods require access to the model's source code, making their implementation more sophisticated and challenging. However, these methods enable more refined evaluation and allow developers to identify specific areas of the model in need of improvement. The following outlines two such methods.

In conventional neural networks, precise weights are computed for each neuron or node and adjusted during training.

By contrast, in Bayesian networks<sup>18)</sup> the network weights are assumed to be random variables, and the goal of training is to estimate the mean and variance of the probability distributions (typically Gaussian distributions) that describe these variables. During the forward pass when inputting data, sampling is performed from these probability distributions to obtain specific weight values. When computing uncertainty for a given input, multiple forward passes are executed, resulting in a different combination of weight values within the network for each pass. Equation (2) shows the mathematical formulation of a Bayesian neuron.

$$y = f(\sum_i w_i x_i + b), \text{ where } w_i \sim N(\mu_i, \sigma_i^2) \quad (2)$$

$y$ : Neuron output

$f(\cdot)$ : Activation function

$w_i$ : Weight

$x_i$ : Input features

$b$ : Bias value

$\mu_i, \sigma_i^2$ : mean and variance of distribution N

When a model includes Dropout layers<sup>2)</sup>, Monte Carlo Dropout<sup>19)</sup> is a technique that leverages them to compute uncertainty. While Dropout is originally only used during training to prevent overfitting, this technique keeps the Dropout layers active at inference time as well, assigning zeros to random weights in order to vary the weight patterns. This effectively disables the corresponding weights, producing an effect analogous to that of a Bayesian network. As a result, multiple inference outputs can be obtained.

## 5 Conclusions

In AI model development, it is important to evaluate accuracy after building a model and to make improvements based on the results. However, conventionally, improving a model requires addition of large amounts of data, entailing considerable cost. One reason for this is that the basis for AI decision-making is a black box, making it difficult for people to understand what data characteristics are effective. This article addresses these challenges by introducing XAI techniques and presenting implementation examples, thereby demonstrating the potential for a more effective approach to selecting additional training data. Furthermore, we propose that by incorporating these XAI techniques into the performance evaluation of an AI pipeline, it is possible to visualize

the presence of insufficient or excess data and facilitate better understanding. It should be noted that since this study evaluated an object detection model, different evaluation methods will be required for other tasks.

Going forward, we plan to pursue application of these methods to different AI models and real-world data, as well as further development and evaluation of XAI techniques using uncertainty-based functions. In addition, when incorporating these evaluation techniques into an AI pipeline, it will be necessary to design evaluation metrics suited to the specific objectives, and these too remain subjects for future research.

## References

- 1) M. T. Ribeiro *et al.*, ACM SIGKDD **22**, 1135–1144 (2016).
- 2) S. Lundberg *et al.*, Neural Inf. Process. Syst. **31**, 4768–4777 (2017).
- 3) T.-Y. Lin *et al.*, Eur. Conf. Comput. Vis. **13**, 740–755 (2014).
- 4) V. B. Truong *et al.*, Proc. Mach. Learn. Res. **222**, 1385–1400 (2023).
- 5) J. Kauffmann *et al.*, arXiv:2006.10609 (2020).
- 6) N. Wang *et al.*, Adv. Neural Inf. Process. Syst. **32**, 7686–7695 (2018).
- 7) G. Hinton *et al.*, NIPS Workshop (2014).  
<https://doi.org/10.48550/arXiv.1503.02531>
- 8) M. B. Muhammad *et al.*, arXiv:2008.00299 (2020).
- 9) B. Zhou *et al.*, IEEE Conf. Comput. Vis. Pattern Recognit., 2921–2929 (2016).
- 10) R. R. Selvaraju *et al.*, Int. J. Comput. Vis. **128**, 336–359 (2019).
- 11) Z. Feng *et al.*, arXiv:2302.01642 (2023).
- 12) V. Buono *et al.*, arXiv:2406.01274 (2024).
- 13) C. Yang *et al.*, J. Cheminform. **15**, 13 (2023).
- 14) A. Kendall *et al.*, Adv. Neural Inf. Process. Syst. **30**, 5574–5584 (2017).
- 15) B. Lakshminarayanan *et al.*, Adv. Neural Inf. Process. Syst. **30**, 6402–6413 (2017).
- 16) L. Breiman, Mach. Learn. **24**, 123–140 (1996).
- 17) G. Wang *et al.*, Med. Image Anal. **54**, 154–162 (2019).
- 18) D.J.C. MacKay, Neural Computation, **4**, 448–472 (1992).
- 19) Y. Gal *et al.*, Proc. Mach. Learn. Res. **48**, 1050–1059 (2016).

<sup>2</sup> Dropout layer: A layer that randomly disables some neurons (computational units) during training.

Authors



Ryu Kidokoro  
3rd laboratory  
Advanced Research Laboratory



Lawrence Earl Carandang  
3rd laboratory  
Advanced Research Laboratory

Publicly available