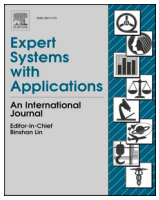
Expert Systems With Applications 272 (2025) 126720

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Illuminating the black box: An interpretable machine learning based on ensemble trees

Yue-Shi Lee [a], Show-Jane Yen [a,*], Wendong Jiang [b], Jiyuan Chen [c], Chih-Yung Chang [b]

[a] Department of Computer Science and Information Engineering, Ming Chuan University, Taoyuan City 333, Taiwan
[b] Department of Computer Science and Information Engineering, Tamkang University, New Taipei 25137, Taiwan
[c] The Faculty of Engineering and Information Technology, The University of Melbourne, Parkville, VIC 3052, Australia

ARTICLE INFO

Keywords:
Interpretable machine learning
Machine learning
Explanation

ABSTRACT

Deep learning has achieved significant success in the analysis of unstructured data, but its inherent black-box nature has led to numerous limitations in security-sensitive domains. Although many existing interpretable machine learning methods can partially address this issue, they often face challenges such as model limitations, interpretability randomness, and a lack of global interpretability. To address these challenges, this paper introduces an innovative interpretable ensemble tree method, EnEXP. This method generates a sample set by applying fixed masking perturbation to individual samples, then constructs multiple decision trees using bagging and boosting techniques and interprets them based on the importance outputs of these trees, thereby achieving a global interpretation of the entire dataset through the aggregation of all sample insights. Experimental results demonstrate that EnEXP possesses superior explanatory power compared to other interpretable methods. In text processing experiments, the bag-of-words model optimized by EnEXP outperformed the GPT-3 Ada fine-tuned model.

1. Introduction

With the rapid development of computer hardware, deep learning has achieved unprecedented success in unstructured fields. In image processing, convolutional neural networks and their variant models (Liu et al., 2024) have made significant breakthroughs in behavior recognition and target detection. In voice and text processing, Transformer architecture models based on multi-head self-attention mechanisms (Xiao et al., 2023; Hold et al., 2024) have achieved great success in natural language processing with generative networks. However, these end-to-end black box models are severely limited in their widespread application in security-sensitive areas due to a lack of interpretability (Shi et al., 2022; Bibal et al., 2023; Dandolo et al., 2023). To understand black-box decisions and enhance model transparency, some interpretable methods have been proposed (Cao et al., 2021; Liu et al., 2021; Dandolo et al., 2023; Liu et al., 2024) to enable people to understand the logic behind them, thereby ensuring trust and transparency. Fig. 1 presents a case that aims to demonstrate the use of interpretable machine learning to explain the Deep Learning (Wu et al., 2023) model. As shown in Fig. 1, although this model has high performance, its discrimination between violent and non-violent behavior is unreasonable. In explaining violent behavior, the model still tends to recognize color and environment rather than the behavior itself, which contradicts common sense. This proves that a successful deep learning model requires not only good model performance but also accurate interpretability so that people can understand and trust it.

Despite the rapid development of interpretable machine learning technologies, three main issues persist. Firstly, there are limitations to interpretable models (Iqbal et al., 2024; Habib et al., 2023). Fig. 2(a) presents an example where existing interpretable methods use heatmaps for interpretative analysis of image data, but this method cannot be directly applied to structured data, text, or speech data.

The second issue is the inability to provide a comprehensive and accurate global explanation for the entire dataset (Singh and Sharma, 2022; Mahapatra et al., 2022). As shown in Fig. 2(b), the model's interpretability in identifying whether an image depicts a cat is illustrated through explanatory heatmaps. For different data inputs, the explanations for identifying cats focus not only on the animal's body and head but also on some environmental features. This indicates that the model's decision-making might be influenced by factors other than the target object in the image. These localized focal points cannot provide a global explanation for the entire dataset. Even if the input image is

* Corresponding author.
E-mail address: sjyen@mail.mcu.edu.tw (S.-J. Yen).

https://doi.org/10.1016/j.eswa.2025.126720
Received 25 May 2024; Received in revised form 17 December 2024; Accepted 28 January 2025
Available online 6 February 2025

'correctly' classified, it does not guarantee that the model uses the same reasoning across the whole dataset.

The third issue is the inconsistency of explanation results (Ribeiro et al., 2016). Fig. 3 demonstrates an attempt to use interpretable machine learning (Ribeiro et al., 2016) for the interpretative analysis of images. This method identifies important regions by applying random masking perturbation to the images and calculating the resulting prediction differences in the deep learning model. However, due to the randomness of these masking perturbation, the explanation results may be unstable, as illustrated in Fig. 3.

This paper addresses the three aforementioned issues by introducing a novel interpretable machine learning method called EnEXP (Ensemble Explanation). EnEXP is an interpretable approach applicable to any model. The method functions by applying fixed masking to each data point in a dataset and employing ensemble tree models (Bagging tree and Boosting tree) to generate importance metrics for individual data point explanations. Furthermore, it achieves a global explanation by weighting each data point in the dataset to determine overall attribute importance. To validate the effectiveness of the proposed method, experiments were conducted on various unstructured datasets using different deep learning models.

This study extends previous research (Jiang et al., 2023) by exploring interpretable machine learning for attribute filtering in structured data. The primary contributions of this paper can be summarized in three key aspects:

1) EnEXP provides a framework that combines both global and local perspectives, aiming to balance understanding of the model's overall behavior with detailed explanations for individual cases. Additionally, this approach relies on domain experts to map the generated visual results into meaningful domain concepts, thereby enhancing the practical utility of the explanations. Global interpretability inherently has limitations when applied to individual prediction cases. By integrating both global and local explanations, this framework aims to offer a more comprehensive interpretive capability for diverse application scenarios.
2) EnEXP enhances model interpretability while simultaneously supporting reliability and transparency. By providing comprehensive explanations of data feature contributions, EnEXP enables researchers and developers to identify critical factors in the model's decision-making process, thereby fostering trust in model behavior. Moreover, it facilitates the calibration and refinement of model predictions, ensuring that the model excels not only in performance but also in employing more reliable and rational decision logic.
3) EnEXP demonstrates enhanced interpretability compared to commonly used explanatory methods in the field, such as LIME (Ribeiro et al., 2016), Kernel SHAP (Lundberg & Lee, 2017), and AD-CAM (Iqbal et al., 2024). This superior performance stems from EnEXP's utilization of the non-linear interpretability inherent in ensemble tree models, enabling more effective analysis of the black-box challenges present in deep learning. Experimental results further corroborate that EnEXP achieves superior performance relative to state-of-the-art baseline methods.

The remainder of the paper is organized as follows: Section 2 reviews and analyzes relevant previous studies. Section 3 details the proposed

Fig. 1. Interpretable analysis of Deep Learning model using Interpretable machine learning.

method and model. Section 4 outlines the assumptions and problem formulation. Section 5 presents the experimental results and performance evaluation. Section 6 concludes the paper with a discussion of findings and implications.

2. Related work

This section reviews research related to interpretable machine learning. Previous studies in this field can be broadly categorized into three main approaches: masking perturbation-based methods, gradient-based methods, and embedding-based methods.

2.1. Masking perturbation-based methods

These methods worked by randomly deleting parts of a single test data entry and then feeding the partially deleted content back into the model to obtain new prediction values. The importance of features in the test data was estimated by comparing the changes in the prediction values. (Zeiler & Fergus, 2014) observed changes in model prediction values by sliding a window with a gray square over images to identify the regions deemed most important by the model; (Ribeiro et al., 2016) obtained new prediction values by randomly masking certain areas in images or texts and used the slope of a linear regression model built on these values to explain the decision-making process of the model; (Petsiuk et al., 2018) used randomly masked versions of input images to estimate the importance of features; (Ancona et al., 2019) proposed a method using Shapley values to approximate explanations for deep learning models; (Lundberg & Lee, 2017) suggested using the probability derivatives of Shapley values as a way to measure the differences between random samples and original samples, exploring the model's working principles. (Zafar and Khan, 2021) utilized clustering methods to generate perturbed samples for interpretation. (Liu et al., 2024) proposed a method using interpretable dual trees combined with SHAP values for interpretability analysis. (David et al., 2024) proposed AcME, a method to accelerate SHAP value calculation for rapid interpretability. Although these methods applied to any black-box model, they had two issues: first, the sampling method typically used random or sampling approaches to obtain perturbed samples, which led to unstable interpretability results; second, when masking perturbation-based methods relied on surrogate models such as linear regression for explanations, they might be affected by the limitations of these models themselves, thus impacting the accuracy and applicability of the explanations. Firstly, linear models, such as linear regression, assume a linear relationship between the input features and the output. This linear assumption may fail to capture the true complexity of the underlying relationships, thereby reducing the reliability of the explanations provided by such models. Secondly, collinearity between input features can also affect linear models, making it difficult to distinguish the

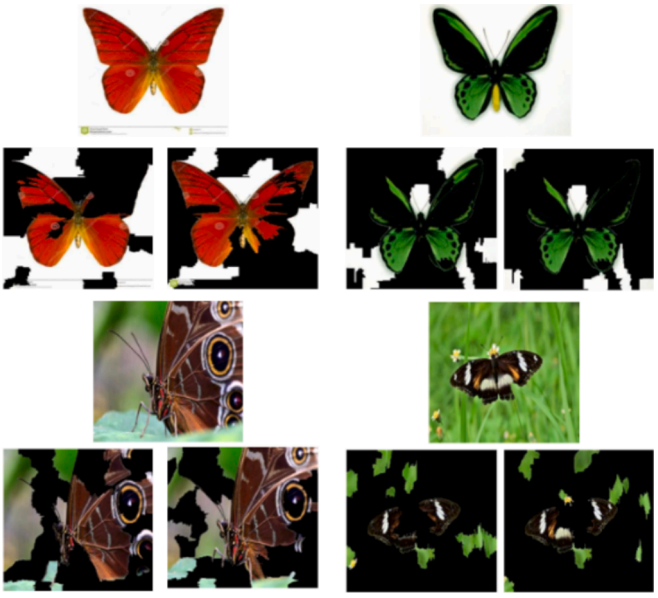


Fig. 3. The inconsistency of explanation results.

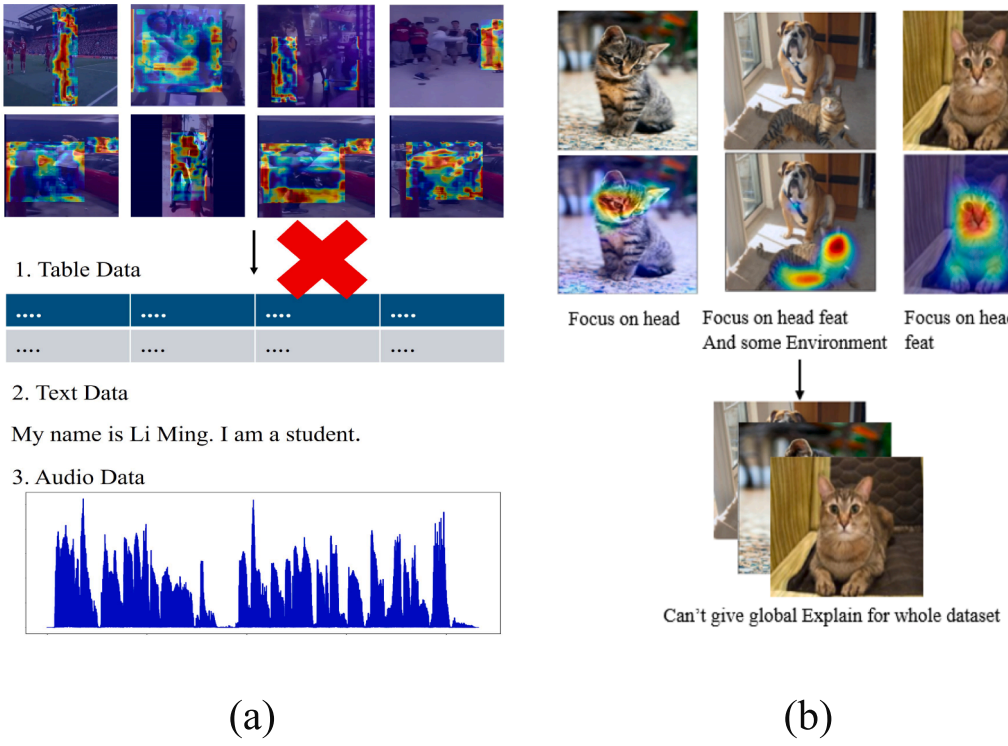


Fig. 2. Interpretable machine Learning issue.

independent contributions of each feature. This limitation can reduce the interpretability and accuracy of the surrogate model's explanations.

2.2. Gradients- based methods

The core purpose of these methods was to identify which input features significantly impacted the model's output through one or several backward propagation processes. For example, (Zhang et al., 2018) employed backward propagation to determine which input pixels activated specific nodes or categories within the network, thereby explaining the model's decision-making process; (Selvaraju et al.,2017) used gradient information from convolutional layers to highlight areas crucial to model predictions, allowing users to visually see what the model focused on; (Binder et al., 2016) identified the most influential features on predictions by proportionally backpropagating the output error to each input feature; (Shrikumar et al., 2017) assigned an "importance score" to each input feature by comparing the contribution of each feature against a reference input, thus explaining the model's predictions; (Smilkov et al., 2017) smoothed gradient maps by adding random noise to input images and calculating the average of gradients, aiming to enhance the stability and clarity of visual explanations; (Zahavy et al., 2016) generated Jacobian saliency maps and presented them over the input state to identify which pixels had the most significant impact on value or action predictions. (Bakchy et al., 2024) proposed a lightweight CNN-based model and used Grad-CAM technique for interpretability.

Although these methods provided intuitive visualization means to reveal the decision-making process of models, they faced four main issues: first, these methods struggled to offer clear explanations for models with large parameters; second, they were primarily applicable to convolutional neural networks or their variants and limited to image data; third, these methods could not provide a global interpretation of the data set; lastly, because the results were presented visually, the effectiveness of these explanatory methods was difficult to assess, not primarily because of visual presentation, but because raw pixel blocks used as explanations are not inherently meaningful concepts in the application domain.

2.3. Embedding-based methods

These methods enhanced the interpretability of data through nonlinear dimensionality reduction techniques. In short, these techniques mapped data points to proximate locations in a lower-dimensional space based on their similarity in high-dimensional space. For instance, (Mnih et al., 2015) demonstrated how to use t-SNE, a dimensionality reduction tool. It aimed to visualize complex high-dimensional data as points on a two-dimensional plane, where the distribution of these points reflected the intrinsic structure of the original data. The color of each point was set according to the features of the data, whether they were extracted automatically or specified manually, to provide additional information. On the other hand, (Engel & Mannor, 2001) proposed a method using embedding maps, which measured the similarity between data points by calculating the transition probabilities between them. (Adrien et al., 2023) proposed DT-SNE, a decision tree model-based improvement on t-SNE, to enhance the interpretability of traditional SNE. These methods effectively utilized statistical means for interpretation, but they faced challenges when dealing with large datasets, mainly due to the difficulty in designing and selecting appropriate features. Moreover, embedding methods often only partially preserve the properties of the original n-dimensional data, which makes interpretation difficult. Furthermore, t-SNE is an unsupervised method and thus fundamentally lacks the capability to ensure that instances belonging to the same class remain similar in the resulting visualization.

3. Interpretable machine learning method

In this section, we introduce the EnEXP algorithm, an interpretable machine learning method capable of explaining any unstructured model and supporting data interpretation.

Fig. 4 illustrates the interpretability process of EnEXP applied to an image classifier trained with a deep learning model $\mathbb{M}$. As shown in Fig. 4, EnEXP aims to explain which regions in the image the model relies on to classify the image as a Basenji. The goal of explainable models is to reveal the logic and basis behind the predictions made by the original model for specific inputs. Furthermore, to simulate and analyze the behavior of the original model, the feature dataset of inputs is usually perturbed and adjusted in weight to make these data more similar to the original images. The output is either the prediction results of the original model or the variations of these prediction results. By establishing such an input–output relationship, explainable models can provide an intuitive understanding and explanation of the model's decision-making process. This is crucial for enhancing the model's transparency and verifying its reliability.

The explanation process is divided into four steps: Generate Perturbed Images, Generate Interpretable Model Input and Output, Interpretable Model Contribution (including the details of Ensemble Tree − Boosting and Bagging Tree Explanation), and Interpretable Visualization. These steps are described in detail as follows:

3.1. Generate Perturb image

The first aspect is the generation of perturbed samples. Unlike previous works (Ribeiro et al., 2016; Liu et al., 2024) that adopted random methods, this study designed a fixed-proportion perturbation approach to achieve greater stability. The specific details are as follows:

Let I denote an image that need to be interpreted, and W represents the watershed image segmentation algorithm. After segmentation, image I' will be obtained. The segmentation process can be expressed by $I' = W(I)$. The segmented image I' can be represented as a set of segmented blocks, expressed as $I' = \{B_1, B_2, \cdots B_n\}$, where each block B_j, $1 \le j \le n$, represent the index of the block. Next, define a set of masks $M = \{M_1, M_2, \cdots M_K\}$ to be applied to I'. Each mask M_k, $1 \le k \le K$ contains a set of Boolean values $\{m_1, m_2, \cdots m_n\}$. Each m_j, $1 \le j \le n$, corresponds to block B_j indicating whether it is kept or masked (0 for masked and 1 for kept). For each specific mask application, it is as follows m_j: $\begin{cases} 0 if m_j = 0 \\ B_j if m_j = 1, \end{cases}$ the m_j of each group ensures that at most 30 % of the blocks are covered according to the rule of thumb. The calculation is as shown in $\sum_{j=1}^{n} (1 - m_j) \le 0.3n$.

Applying each mask M_K to I' produces a series of new images with selectively retained blocks $\{I_1^n = M_1(I'), I_2^n = M_2(I'), \cdots, I_K^n = M_N(I')\}$, forming the final set of perturbed images $I'' = \{I_1^n, I_1^n, \cdots, I_N^n\}$, where each image I_k^n represent the result of a specific block retention model determined by mask M_K.

Proof 1

Let $f : \mathbb{R}^d \to [0, 1]$ denote the model to be explained, and $x \in \mathbb{R}^d$ be the ample to be explained. Let $z \in \{0, 1\}^d$ denote masking perturbation vector and define the perturbed sample as $x'(z) = x \odot z$, where $\odot$ denote the elementwise Hadamard product.

The core idea of LIME is to approximate the target model f locally around the sample x using a linear model $g(z) = w_0 + \sum_{i=1}^{d} w_i z_i$. This approximation process is based on weighted least squares, where the weights are determined by the Euclidean distance between samples.

Let S define the stability of the variance of the coefficients w obtained from multiple LIME runs. Specifically, the stability metric can be expressed in Exp. (1):

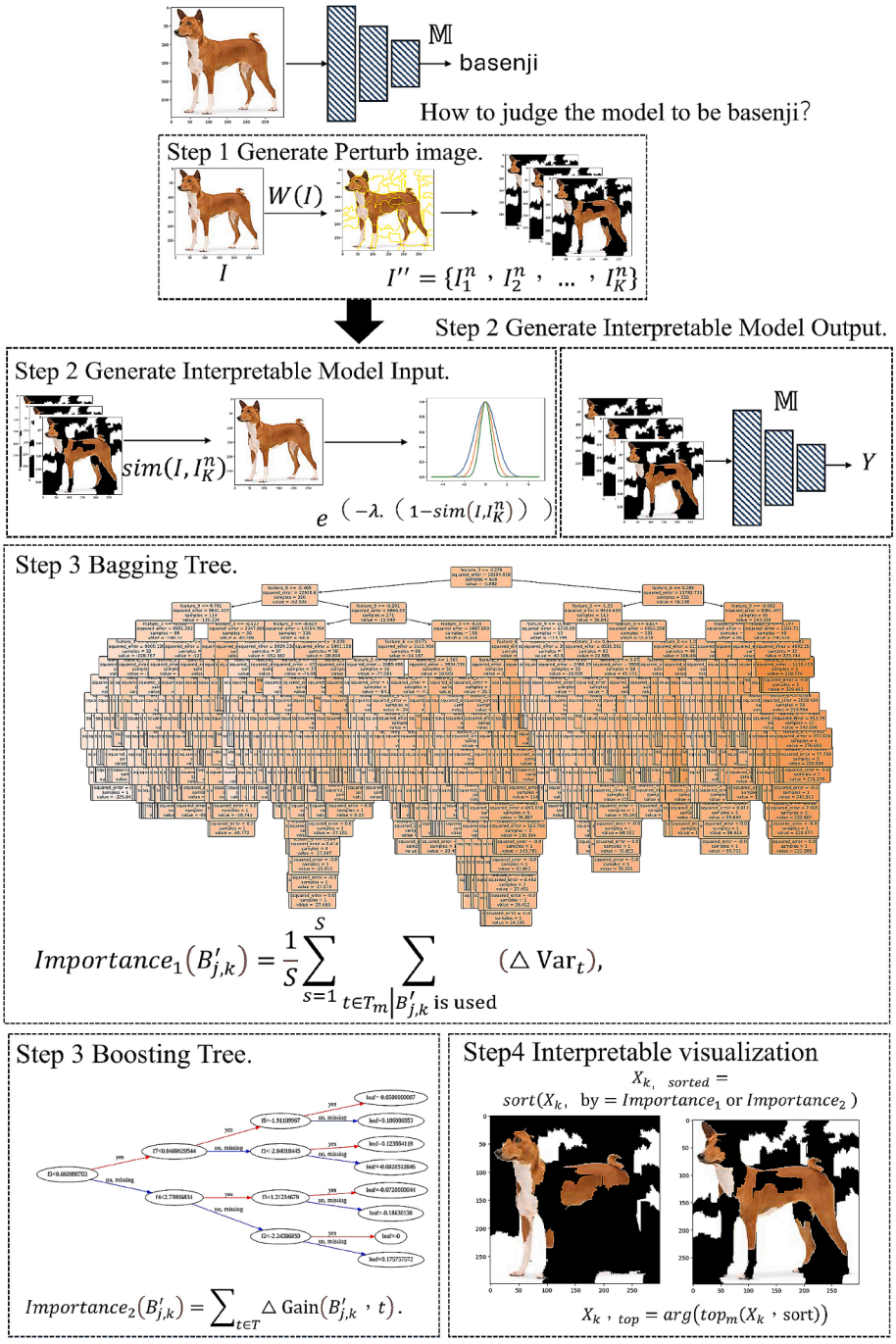


Fig. 4. EnEXP Workflow.

$$S = \frac{1}{d}\sum_{i=1}^{d} Var(w_i). \tag{1}$$

where $Var(w_i)$ represents the variance of the coefficient w_i.

For Fixed Ratio Generation, for each feature i, the masking perturbation vector z_i is set to 1 with a fixed probability p as follows $P = (z_i = 1) = p$ and $P = (z_i = 0) = 1 - p$. Under this setting, the expectation and variance of z_i are $\mathbb{E}[z_i] = p$, $Var(z_i) = p(1-p)$.

For random generation strategy, let M Uniform$(0, d)$ represent the number of retained features. In this case, the conditional probability of z_i is $P(z_i = 1 | M = m) = \frac{m}{d}$. Thus, the expectation and variance of z_i are $P(z_i = 1) = \mathbb{E}\left[\frac{M}{d}\right] = \frac{1}{2}$, $\mathrm{Var}(z_i) = \frac{1}{4}$.

Using least squares, the coefficients w of the linear model are obtained as $w = (Z^T WZ)^{-1} Z^T W_y$. Where Z is the matrix composed of masking perturbation vectors, W is the weight matrix, and y is the model output. The covariance matrix of the coefficients w can be be expressed by Exp. (2):

$$Cov(w) = (Z^T WZ)^{-1} Z^T \mathrm{Var}(y) WZ (Z^T WZ)^{-1}. \tag{2}$$

For fixed ratio generation, the covariance matrix Z_{fixed} is given by $Cov(Z_{fixed}) = p(1-p)I_d$. Given that the weight matrix $W = I_d$, and the output variance $Var(y) = \sigma^2 I_d$, the covariance matrix of the coefficients w_{fixed} can be expressed by Exp. (3):

$$Cov(w_{fixed}) = \sigma^2\left(Z_{fixed}^T Z_{fixed}\right)^{-1}. \quad (3)$$

Since $Z_{fixed}^T Z_{fixed}$ is a diagonal matrix with diagonal elements approximately equal to $np(1-p)$ (assuming n is large enough), we have $Z_{fixed}^T Z_{fixed} = np(1-p)I_d$. Taking the inverse, that $(Z_{fixed}^T Z_{fixed})^{-1} = \frac{1}{np(1-p)}I_d$. Substituting this into the expression for $Cov(w_{fixed})$, we obtain $Cov(w_{fixed}) = \frac{\sigma^2}{np(1-p)}I_d$. This shows the variance of each coefficient $w_{fixed,i}$ by Exp. (4):

$$Var(w_{fixed,i}) = \frac{\sigma^2}{np(1-p)}. \quad (4)$$

For the random generation strategy, the columns of the masking perturbation matrix Z_{random} are not independent, leading to a more complex covariance structure. The covariance matrix for Z_{random} can be described in Exp. (5):

$$Cov(Z_{random})_{i,j} = \begin{cases} \frac{1}{4}, if i = j \\ \frac{1}{12}, if i \neq j \end{cases}, \quad (5)$$

Herein, the covariance matrix of the coefficients w_{random} then becomes $Cov(w_{random}) = \sigma^2(Z_{random}^T Z_{random})^{-1}$. Since $Z_{random}^T Z_{random}$ is not a diagonal matrix, calculating its inverse is more complex. However, it is generally known that the diagonal elements of the inverse matrix are larger than those in the fixed ratio case, leading to $\left((Z_{random}^T Z_{random})^{-1}\right)_i > \frac{4}{n}$, Thus we can approximate in Exp. (6):

$$Var(w_{random,i}) \geq \frac{\sigma^2}{n \times \frac{1}{4}} = \frac{4\sigma^2}{n}. \quad (6)$$

By comparing the variances from both strategies, we observe $Var(w_{fixed,i}) = \frac{\sigma^2}{np(1-p)}$ and $Var(w_{random,i}) \geq \frac{\sigma^2}{n\times\frac{1}{4}} = \frac{4\sigma^2}{n}$. When $p = \frac{1}{2}$, $Var(w_{fixed,i}) = \frac{4\sigma^2}{n}$, indicating that the variances are equal. However, for $p \neq \frac{1}{2}$, we have $p(1-p) < \frac{1}{4}$, leading to Exp. (7):

$$Var(w_{fixed,i}) < Var(w_{random,i}). \quad (7)$$

3.2. Generate interpretable model input and output.

Because the image I' can be regarded as a vector of 1′s, and the perturbed image I_K^n is represented by a combination of 1 and 0, the similarity calculation measures the proportion of the retained block, the similarity can be shown in $sim(I, I_K^n) = \frac{\sum_{j=1}^{n} m_{j,k}}{n}$. Herein, $m_{j,k}$ represent the j-th element in mask M_K, and n is the total number of blocks. The calculated similarity changes linearly, which does not conform to the conventional nonlinear changes. Then the corrected similarity formula can be expressed as Exp. (8)

$$adj_k = e^{\left(-\lambda.\left(1-sim\left(I,I_K^n\right)\right)\right)}. \quad (8)$$

Herein, λ is a hyperparameter, representing the intensity of adjustment. $1-sim(I, I_K^n)$ represents the Hamming distance. Through this adjustment, if the proportion of coverage is small, the similarity will be high, and if the proportion of coverage is large, the similarity will be lower. The adjusted similarity adj_k is applied to each block of the perturbed image I_K^n. The calculation $B'_{j,k} = adj_k \times B_{j,k}$. Among them, $B'_{j,k}$ represents the block in the updated masking perturbation image, and $B_{j,k}$ represents the block in the pre-update masking perturbation image. The following proof elaborates on the rationale behind Exp. (8) as a similar measure. It is structured into three steps: first, Necessity of a Nonlinear Similarity Measure; second, Selection of a Specific Nonlinear Measure; and finally, Superiority over Other Nonlinear Measures. The detailed process is as follows:

Proof 2:

(1) Necessity of a Nonlinear Similarity Measure

Task Motivation: Let $sim(I, I_K^n) \in [0,1]$ represent a similarity measure where sim = 1 denote perfect similarity. We aim to construct an adjacency measure adj_k to quantify the strength of connection between I and I_K^n. This adjacent measure will influence tasks such as clustering, graph-based learning, or kernel-based analysis.

Reasoning for Nonlinearity:

Linear measures (e.g., $adj_k = sim$) are insufficient due to the following limitations:

1. Insensitive to High Similarity: When $sim \to 1$, linear measures fail to emphasize strong connections. For example, $sim = 0.95$ and $sim = 0.9$ yield nearly identical weights under linear mapping, whereas tasks often require a sharper distinction.
2. Excessive Retention of Weak Similarity: Linear mappings poorly downweight low-similarity pairs. A linear adj_k would assign moderate weights to $sim \approx 0.5$, which could introduce noise in downstream tasks by overemphasizing weak connections.
3. Task-Specific Requirements: Many tasks, such as kernel methods, demand exponential-like sensitivity in feature space. This motivates nonlinear measures that naturally decay for moderate or low similarities.

Linear similarity measures are insufficient for capturing the desired sensitivity across different similarity intervals. Therefore, nonlinear similarity measures are hypothesized to better suit these requirements.

(2) Selection of a Specific Nonlinear Measure

To satisfy the demands outlined above, the adjacent measure adj_k must fulfill the following conditions:

(i): $adj_k : [0,1] \to (0,1]$ is continuous and differentiable.

(ii): $adj_k(0) = 1$. (ensuring maximum adjacency for perfect dissimilarity).

(iii): adj_k is strictly decreasing: for all $0 \leq x_1 < x_2 \leq 1$, $adj_k(x_1) > adj_k(x_2)$.

(iv): Exponential decay property: $\frac{d}{dx}adj_k = -\lambda adj_k$, ensuring rapid decay as similarity decreases.

Derivation of Exponential Measure: The exponential decay condition (iv) defines the adjacency function as the solution to the ODE: $\frac{d}{dx}adj_k(x) = -\lambda adj_k(x)$.

By solving:

$$adj_k(x) = e^{-\lambda x + C},$$

And using (ii) $adj_k(0) = 1$, we find $C = 0$, yielding:

$$adj_k(x) = e^{-\lambda x}$$

Substituting $x = 1 - sim(I, I_K^n)$, the final adjacency measure becomes:

$$adj_k = e^{\left(-\lambda.\left(1-sim\left(I,I_K^n\right)\right)\right)}.$$

This exponential form ensures rapid decay of adjacency as similarity decreases, aligning with the task's requirements for sharper discrimination at high similarity and faster suppression of weak connections.

(3) Superiority over Other Nonlinear Measures

To justify the use of $adj_k = e^{-\lambda.(1-sim)}$, we compare it against alternative nonlinear measures:

1. **Polynomials:** Polynomial functions (e.g., $adj_k = 1 - \alpha x^2$), may satisfy monotonicity but fail to provide exponential sensitivity. Their decay rates are either too steep (e.g., penalizing moderately low similarity excessively) or too shallow (e.g., failing to suppress weak connections effectively).
2. **Sigmoids Measures:** Sigmoid functions (e.g., $adj_k = \frac{1}{1+e^{\alpha(x-\beta)}}$) introduce unnecessary complexity with additional parameters (α, β). Moreover, sigmoid functions lack the proportional decay property

$\frac{d}{dx}adj_k = -\lambda adj_k$, making them less interpretable and less aligned with task-specific requirements.

3. **Logarithmic Measures:** Logarithmic mappings, such as e.g., $adj_k = \ln(1+\alpha(1-x))$ fail to maintain smooth decay over the full range $x \in [0,1]$. They can diverge near endpoints, making them unsuitable for tasks requiring robust sensitivity.

Uniqueness of the Exponential Form: The exponential measure uniquely satisfies: 1. Smooth monotonicity: Ensures consistent decay across the entire similarity range. 2. Task-driven sensitivity: Balances sharp discrimination at high similarity and rapid suppression at low similarity. 3. Simplicity and interpretability require only one parameter λ and directly align with the task-derived ODE.

Let $\mathbb{M}$ denote the model to be explained, The generated perturbed images are represented as $I'' = \{I_1^n, I_1^n, \cdots, I_N^n\}$, where each I_K^n is passed through the model $\mathbb{M}$ for prediction. Let $Y = \{Y_1, Y_2, \cdots, Y_N\}$ represent the set of prediction values obtained for all perturbed images by model $\mathbb{M}$.

3.3. Interpretable model contribute

This section introduces the interpretability of surrogate models and further clarifies the advantages of ensemble tree models over linear regression models. In this study, interpretability is defined as the ability of a model to provide clear and intuitive information to understand the specific contributions of features to the prediction outcomes. Although linear regression models are widely considered interpretable due to their simple mathematical form and explicit feature weights, their interpretability is limited in the following scenarios: first, when multicollinearity exists among features, the regression coefficients may fail to accurately reflect the importance of individual features; second, linear models are incapable of providing reliable explanations for nonlinear feature relationships.

Against this background, ensemble tree models offer an alternative form of interpretability through their built-in feature importance evaluation mechanisms (e.g., split gain and information gain). The interpretability of ensemble tree models can also be enhanced through visualization techniques, such as feature importance rankings and decision paths, mitigating the interpretability challenges caused by their complexity. Additionally, ensemble tree models can capture nonlinear relationships and interactions between features, making them more effective in explaining predictions in complex data scenarios compared to traditional linear models. Therefore, while the forms of interpretability differ between the two approaches, ensemble tree models demonstrate significant advantages in addressing challenges related to high-dimensional data, nonlinear relationships, and feature interactions. The specific implementation process is as follows:

Having established the perturbed sample set $I'' = \{I_1^n, I_1^n, \cdots, I_N^n\}$, where each I_K^n represents the k-th perturbed sample and there is a corresponding target value set $Y = \{Y_1, Y_2, \cdots, Y_N\}$. Each perturbed sample I_K^n contains n processed blocks $\left\{B'_{1,k}, B'_{2,k}, \cdots, B'_{n,k}\right\}$, these blocks serve as the feature set $X_k = \left\{B'_{1,k}, B'_{2,k}, \cdots, B'_{n,k}\right\}$. Define multiple regression tree models T_s, where $, s = 1, 2, \cdots., S$, and S represents the total number of trees.

3.3.1. Bagging tree explain

Bootstrap sampling is performed on the feature set X_k to generate a new training set D_m. This process is represented as randomly drawing N times with replacement from N samples to form the training set D_m, which includes both inputs and outputs.

For each dataset D_m, recursively build a regression tree T_s. For each node in the tree, find the optimal split block $B'_{j,k}$ and split point L_{best}, by minimizing the weighted total variance after the split, expressed as Exp. (9):

$$\left(B'_{j,best}, s_{best}\right) = argmin_{B'_{j,k,s}}\left[w_{left} \times Var\left(D_{left}\right) + w_{right} Var\left(D_{right}\right)\right] \tag{9}$$

where w_{left} and w_{right} respectively represent the weights of the datasets to the left and right of the split point, and $Var(.)$ denotes variance. For each leaf node in tree T_m calculate the mean of all target values of samples in that node as the prediction value for that node is $\hat{y}_{leaf} = \frac{1}{|D_{leaf}|}\sum_{i \in D_{leaf}} y_i$. Finally, the importance of block $\left(B'_{j,k}\right)$ can be measured by calculating the average reduction in variance caused by that block across all trees by Exp. (10):

$$Importance_1\left(B'_{j,k}\right) = \frac{1}{S}\sum_{s=1}^{S}\sum_{t \in T_m | B'_{j,k} \text{is used}} (\Delta \text{Var}_t), \tag{10}$$

where ΔVar_t represents the reduction in variance brought about by splitting at node t in tree T_s using feature $B'_{j,k}$.

3.3.2. Boosting tree explain

The boosting tree builds the model through the iteration of trees. The model's objective function $\text{Obj}(\Theta)$ consists of two parts: the training data loss L and regularization Ω, as shown in Exp. (11):

$$\text{Obj}(\Theta) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \sum_{i=1}^{N} \Omega(f_k). \tag{11}$$

Herein, $L(y_i, \hat{y}_i) = \frac{1}{2}(y_i, \hat{y}_i)^2$ is the squared error loss for the i-th sample; $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|w\|^2$ is the regularization term for the k-th tree, where γ is the fixed cost of each tree, T is the number of leaves in the tree, and λ is the L2 regularization coefficient for the leaf weights. Unlike previous studies (Zafar and Khan, 2021), which only applied L2 regularization, this method can directly control the number of leaf nodes to optimize model performance. The detailed proof is as follows:

Proof 3:

Aims to demonstrate that the heuristic regularization term $\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|w\|^2$ in Exp. (11) effectively controls the number of leaf nodes T and the leaf weights w. This approach not only ensures the model's convergence but also directly limits model complexity and generalization error. The selection of this regularization method is based on the following three considerations:

1. **Direct Control of Leaf Node Count:** Compared to other regularization methods (e.g., using only L2 regularization, as in Zafar and Khan, 2021), the introduced penalty term γT is directly linked to the number of leaf nodes, enabling explicit control of the split gain at each split. This effectively constrains the tree structure from overgrowing.

2. **Compatibility with the Tree model splitting Mechanism**: Although this regularization term has a heuristic design aspect, it reduces the split gain in a way that aligns with the splitting principles of gradient boosting trees. While other regularization methods can also achieve convergence, our approach is more compatible with the structural characteristics and splitting rules of tree models

3. **Significant effect on controlling generalization Error:** We observed that this regularization method more effectively limits the number of leaf nodes and reduces the complexity term, which consequently lowers the generalization error bound. Detailed reasoning is as follows

Traditional tree model regularization typically employs only L2 regularization on leaf weights, defined as follows:

$$\Omega(f_k) = \frac{1}{2}\lambda\|w\|^2, \tag{12}$$

where w represents the vector of leaf weights, and λ is the hyperparameter controlling the regularization strength.

In contrast, the proposed regularization method adds a heuristic penalty term, expressed as:

$$\Omega(f_k) = \gamma T + \frac{1}{2}\lambda\|w\|^2, \tag{13}$$

where T denotes the number of leaf nodes, and γ is the penalty coefficient associated with the number of leaf nodes. We demonstrate the effectiveness of this method from the perspective of controlling model complexity.

In traditional tree models, the objective function's regularization term is expressed as:

$$\mathrm{Obj}(\Theta) = \sum_{i=1}^{N} L(y_i, \hat{y}_i) + \frac{1}{2}\lambda\|w\|^2, \tag{14}$$

where $L(y_i, \hat{y}_i)$ represents the loss function between the predicted and actual values.

For a fixed tree structure, optimizing the leaf weights $w$is a convex optimization problem, ensuring solvability of the objective function. For a given split node A, the split gain $Gain_A$ under traditional regularization is given by:

$$Gain_A = \frac{1}{2}\left(\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right), \tag{15}$$

where G_L and G_R are the accumulated gradients for the left and right child nodes, and H_L and H_R are the corresponding accumulated second-order derivatives. According to the definition of split gain, a split occurs only when the gain is positive: $Gain_A > 0$. However, this condition lacks constraints on the number of splits, which can result in overly complex tree structures. To address this, we introduce a penalty term γ with each new leaf node, so the split gain becomes: $Gain_B = Gain_A - \gamma$, thereby effectively limiting the number of leaf nodes.

Let T' denote the denote the number of leaf nodes after applying the penalty term, where $T' \le T$.

Assuming the generalization error upper bound is given by $R(f) \le \widehat{R}(f) + C(H)$ where $R(f)$ is the expected risk of model f, $\widehat{R}(f)$ is the empirical risk of f on the training set, and $C(H)$ is the complexity term of space.

Under traditional regularization (Zafar and Khan, 2021), which uses only L2 regularization on leaf weights and does not restrict the number of leaf nodes T, the upper bound of the generalization error depends on T as follows:

$$R_A(f) \le \widehat{R}(f) + \mathscr{O}\left(\sqrt{\frac{T\log T}{N}}\right), \tag{16}$$

where N is the number of training samples. As T increases, the complexity term $\mathscr{O}\sqrt{\frac{T\log T}{N}}$ also increases, resulting in a higher generalization error bound and affecting the model's generalization ability.

After introducing γ, the number of leaf nodes T' is reduced, and the complexity term decreases accordingly. The new generalization error upper bound becomes:

$$R_B(f) \le \widehat{R}(f) + \mathscr{O}\left(\sqrt{\frac{T'\log T'}{N}}\right), \tag{17}$$

Since $T' \le T$, we have:

$$\mathscr{O}\left(\sqrt{\frac{T'\log T'}{N}}\right) \le \mathscr{O}\left(\sqrt{\frac{T\log T}{N}}\right). \tag{18}$$

Therefore, $R_B(f) \le R_A(f)$, indicating that introducing the penalty term γ reduces the upper bound of the generalization error, thereby effectively controlling model complexity.

After t rounds of iteration, the model's prediction is updated as shown in Exp. (19):

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(X_i). \tag{19}$$

Where $\hat{y}_i^{(t-1)}$ is the prediction from the $t-1$ round, $f_t(X_i)$ is the prediction contribution from the tree added in round t for the sample $B'_{j,k}$, and η is the learning rate. The new tree f_t is trained by minimizing Exp. (20):

$$\mathrm{Obj}^{(t)} = \sum_{i=1}^{N} L\left(y_i, \hat{y}_i^{(t-1)} + f_t\left(B'_{j,k}\right) + \Omega(f_t)\right). \tag{20}$$

In the completed boosting tree, the importance of each block $B'_{j,k}$ can be assessed by calculating the gain brought about by feature splits, as shown in Exp. (21):

$$Importance_2\left(B'_{j,k}\right) = \sum\nolimits_{t\in T} \Delta\mathrm{Gain}\left(B'_{j,k}, t\right). \tag{21}$$

where T includes the set of all split points across the trees, and $\Delta\mathrm{Gain}\left(B'_{j,k}, t\right)$ represents the gain brought about by the block $B'_{j,k}$ at split point t.

3.4. Interpretable visualization

The sorting of features $X_k = \left\{B'_{1,k}, B'_{2,k}, \cdots, B'_{n,k}\right\}$ by importance descending, obtained from two different tree models as $Importance_1$ or $Importance_2$ can be represented as Exp. (22):

$$X_{k,sorted} =$$

$$sort(X_k, \mathrm{by} = Importance_1 \mathrm{or}\, Importance_2) \tag{22}$$

Next, selecting the top m most important blocks for visualization as an interpretation result, where $m < n$, is represented as Exp. (23):

$$X_{k,top} = arg(top_m(X_k, \mathrm{sort})) \tag{23}$$

The following demonstrates that tree models have greater interpretability than traditional regression models when used as surrogate models.

Proof 4: Table 1 presents the symbols that were undefined in Proof 4. In Proof 4, the generalization error in high-dimensional data was demonstrated to show that the proposed ensemble tree model is more interpretable than the linear regression model. The generalization error bounds for these models are given by the following formulas. For LR, let E_{LR} denote the LR generalization bound; we have:

$$E_{LR} = \sqrt{\frac{2(p+1)\left(log\left(\frac{2en}{p+1}\right)+1\right) + 2\log\left(\frac{1}{\delta}\right)}{n}},$$

$$R(h_{LR}) \le R_{emp}(h_{LR}) + E_{LR.}$$

Herein, the term $\frac{2(p+1)\left(log\left(\frac{2en}{p+1}\right)+1\right)+2\log\left(\frac{1}{\delta}\right)}{n}$ represents the complexity term for linear regression, which increases with the dimensionality p.

Table 1
Glossary.

$R(h_{LR})$	The generalization error of the linear regression (LR) model
$R(f_{ET})$	The true generalization error of the Ensemble Tree (ET) model
$R_{emp}(h_{LR})$	The empirical error (training error) of the linear regression (LR) model
$R_{emp}(f_{ET})$	The empirical error (training error) of the Ensemble Tree (ET) model
n	The number of training samples.
p	The number of features (dimensionality of the input space) in the linear regression model.
B	The number of trees in the Ensemble Tree (ET) model.
M	The maximum number of nodes (or splits) in each tree of the Ensemble Tree (ET).
δ	The confidence level parameter, which controls the probability that the generalization bound holds.

The $2\log\left(\frac{1}{\delta}\right)$ is a confidence adjustment term that decreases with the number of sample n. For ET, let E_{ET} denote the ET generalization bound; we have:

$$E_{ET} = R_{emp}(f_{ET}) + \sqrt{\frac{2B(log(2M)+1) + 2\log\left(\frac{1}{\delta}\right)}{n}}.$$

$$R(f_{ET}) \le R_{emp}(f_{ET}) + E_{ET}$$

Herein, the term $\frac{2B(log(2M)+1)+2\log\left(\frac{1}{\delta}\right)}{n}$ represents the complexity term for Ensemble Tree, which depends on the number of trees B. and the maximum number of nodes M in each tree. The $2\log\left(\frac{1}{\delta}\right)$ again accounts for the confidence level. To compare the upper bounds E_{LR} and E_{ET}, we start by squaring both expressions to eliminate the square roots as shown in

$$E_{LR}^2 - E_{ET}^2 = \frac{2(p+1)\left(log\left(\frac{2en}{p+1}\right)+1\right) - 2B(log(2M)+1)}{n}$$
$$= \frac{2\left[(p+1)\left(log\left(\frac{2en}{p+1}\right)+1\right) - 2B(log(2M)+1)\right]}{n}.$$ For high-dimensional data where p is large, the term $(p+1)\left(log\left(\frac{2en}{p+1}\right)+1\right)$ tends to grow larger than $B(log(2M)+1)$, implying in Exp. (24):

$$(p+1)\left(log\left(\frac{2en}{p+1}\right)+1\right) > B(log(2M)+1) \quad (24)$$

Thus:

$$E_{LR} > E_{ET}$$

This indicates that for high-dimensional and complex problems, the generalization bound of linear regression is usually larger than that of the ensemble tree model. Therefore, random forests tend to have better generalization performance in high dimensions, which implies stronger interpretability.

4. Assumptions and problem formulation

This chapter will introduce the assumptions and problem formulation of this paper. This paper defines two different approaches to address various interpretability needs. The first approach is a coarse-grained interpretability assessment, which is used to understand the decision-making process of the model. The second approach is a fine-grained interpretability visualization pathway, which guides experts in further decision analysis and improvements by showing the ranking of importance within different parts of the model, thereby facilitating human–machine interaction.

4.1. Coarse-grained interpretability assessment

The coarse-grained interpretability assessment aims to achieve interpretability by matching the parts selected by experts with those selected by the model and evaluating the number of matches. This design is intended to help users understand whether the model's cognition is fundamentally consistent with that of the experts.

For the image classification task, given a model $\mathbb{M}$ to be explained and an image I to be interpreted, the output of model $\mathbb{M}$ is the region that it believes corresponds to the classification.

Using the segmentation technique g, let $I = \{I_1, I_2, I_3, \cdots I_n\}$ denote the image I is segmented by g. For each segmented block I_j, where $0 < j \le n$. Let $U = \{U_1, U_2, U_3, \cdots U_m\}$ denote the experts select a set of blocks that they believe belong to this classification. For the selected m blocks, let $K = \{K_1, K_2, K_3, \cdots K_m\}$ denote an interpretable network output a set of blocks by importance.

The aim of this paper is to develop a mechanism M that uses the expert-selected set U as the ground truth, compares it with the set K, and calculates the number of matching elements. Exp. (25) demonstrates how M is calculated.

$$M = |U \cap K|. \quad (25)$$

Herein, $|U \cap K|$ represents the size of the intersection of sets U and K, i, e., the number of elements that exist in both U and K.

The objective of this paper is to develop mechanism M^{best} that satisfies Exp. (26).

$$M^{best} = argmax(M) \quad (26)$$

Fig. 5 illustrates M an example of evaluating an image classification task. As shown in Fig. 5, the blocks selected by the experts U= {0, 3, 7, 13, 18, 25, 26, 29, 32, 33, 34, 37, 38, 40, 41, 42, 43, 44, 45, 48, 49, 54, 55} totaling 23 blocks. Similarly, the output model $\mathbb{M}$ selected the top 23 most important blocks as K= {38, 39, 28, 52, 14, 13, 50, 44, 12, 21, 53, 56, 9, 5, 61, 15, 47, 0, 30, 34, 25, 18, 26}. The resulting M is 0.34782.

For a text classification task, the concept behind the design of text masking in this paper is to observe the changes in prediction scores by reintroducing the text into the model after applying fixed masking methods to the segmented text. By analyzing these changes, the importance of individual words is then determined through machine learning modeling, thereby achieving interpretability.

Given a model $\mathbb{M}$ that needs to be explained and an input text T the output is the set of words in the text on which the model bases its decisions. Taking Chinese text as an example, let $\mathscr{g}$ denote the Jieba segmentation technique, and let $T = \{T_1, T_2, T_3, \cdots T_n\}$ denote the sentence after segmentation by $\mathscr{g}$. For the text $T = \{T_1, T_2, T_3, \cdots T_n\}$, masking perturbation techniques are applied, and let $T' = \{T'_1, T'_2, \cdots T'_N\}$ denote the resulting perturbed samples, where N represents the number of masking perturbations. Let $T'_j = \{T_1, T_2, T_3, \cdots T_p\}$ denote each perturbed data sample, where $0.3n < p \le n$. For the perturbed samples T', the model $\mathbb{M}$ is used again to obtain the prediction scores $Y = \{Y'_1, Y'_2, \cdots Y'_N\}$. Then, for each perturbed data T'_j and the original sample T, the similarity $sim(T, T')$ is calculated, and the weights $w' = \{w'_1, w'_2, \cdots w'_N\}$ are adjusted through Exp. (1). For each perturbed data T'_j, the corresponding weight w'_j is obtained. Let $p = \{T'_1 \times w'_1, T_2 \times w'_2, T_3 \times w'_3, \cdots T_n \times w'_n\}$ denote the input to the interpretable model, and $Y = \{Y'_1, Y'_2, \cdots Y'_N\}$ represent the corresponding labels. The output model importance $P = \{P_1, P_2, P_3, \cdots P_n\}$ corresponds to the importance of each word in $T = \{T_1, T_2, T_3, \cdots T_n\}$.

Similarly, let $U = \{U_1, U_2, U_3, \cdots U_m\}$ denote the set of words selected by experts that they believe belong to this classification. For the selected m words, let $K = \{K_1, K_2, K_3, \cdots K_m\}$ a set of words output by an interpretable network ordered by importance. This paper aims to develop a mechanism M that uses the expert-selected set U as the ground truth, compares it with the set K, and calculates the number of matching elements. Exp. (25) demonstrates how M is calculated. Fig. 6 illustrates M an example of evaluating an image classification task.

As shown in Fig. 6, the input T is "這部電影很有趣, 演員的表演令人印象深刻" (This movie is very interesting, and the performances by the actors are impressive.) and the model to be explained is SVM, which is to explain why SVM predicts this sentence as positive. The specific process and steps are as follows: First, using Jieba segmentation $\mathscr{g}$, we obtain "這部/電影/很/有趣" and "演員 /的/表演/令人/印象/深刻". Let the number of generated perturbed samples be N = 3, which are: T'_1 : "這部 電影 很", "演員 的 表演 令人 印象 深刻". T'_2 : "這部 電影 很 有趣", "的 表演 令人 印象". T'_3 :"電影 很 有趣", "演員 的 表演 令人 印象 深刻".Then, these 3 perturbed samples are fed into the SVM, yielding prediction scores $Y = \{0.3, 0.7, 0.8\}$. At the same time, the similarity between the perturbed

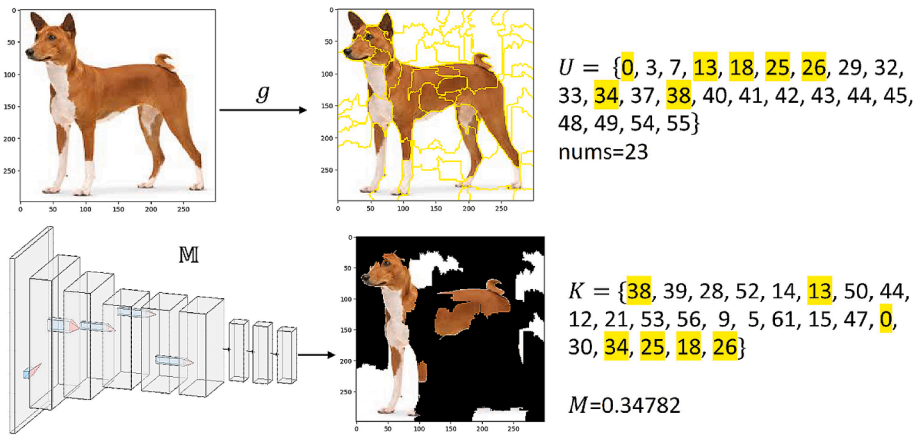


Fig. 5. An Example of *M* Evaluating an Image Classification Task.

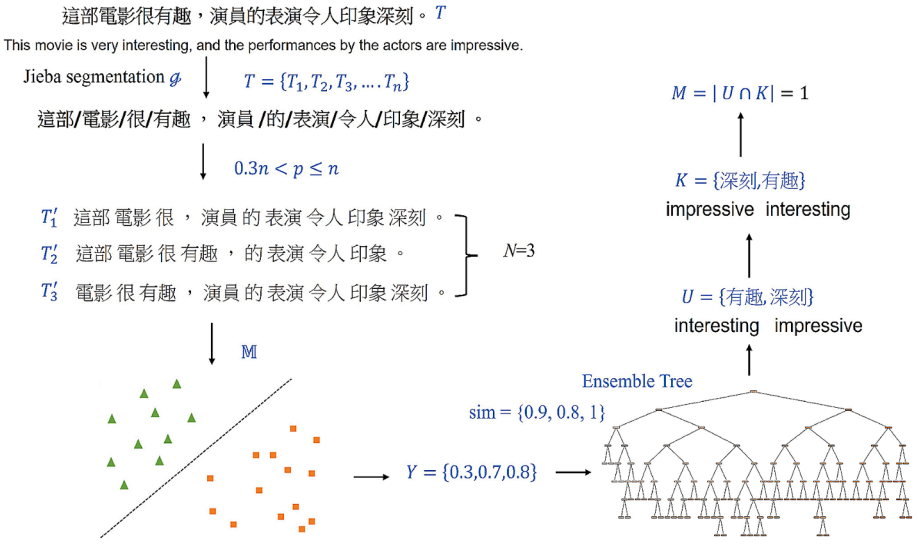


Fig. 6. An Example of *M* in a Chinese Text Classification Task.

samples and the original sample is calculated as sim = {0.9, 0.8, 1}. Next, an interpretable ensemble tree is constructed, where the prediction scores $Y = \{0.3, 0.7, 0.8\}$ are the targets, and the inputs are $p = \{T'_1 \times 0.9, T_2 \times 0.8, T_3 \times 1\}$.Experts consider the positive words in the sentence to be *U*={"有趣(interesting)", "深刻(impressive)" }. Similarly, the two words with the highest importance according to the model are *K*={"深刻(impressive),有趣(interesting)" }. Finally, the *M* score for this sentence is 1. Herein are the assumptions and problem formulation of this paper. The next section will introduce the experimental results and performance evaluation.

4.2. Fine-grained interpretability assessment

After completing the coarse-grained assessment, if the coarse-grained decision *M* score exceeds a certain threshold, decision visualization can be performed through an interpretability visualization pathway. This is applicable to some sensitive fields, such as motion behavior analysis and medical imaging. It should be noted that, in some practical applications, data annotated by experts is often challenging to directly apply at the decision-making level, primarily for the following reason. In real-life scenarios, such as determining whether a specific location contains a dog (i.e., a coarse-grained judgment), most annotators can reach consensus based on common-sense understanding. However, when it comes to finer-grained distinctions—such as whether to prioritize the eyes or nose as key regions—differences in individual perspectives may arise. Therefore, this paper proposes a fine-grained visualization method during the usage phase to output an interpretable model pathway, which shows experts the decision-making process of the model and provides further optimization suggestions. The specific steps are as follows:

let τ denote the threshold parameter. If the *M* score derived from the classification model, exceeds the threshold τ, the output set *U* is identified as a collection of salient regions or features. The importance of these regions is then quantified using the metric defined by Exp. (21).

Subsequently, let $\varnothing$ denote the mapping function, which is responsible for transforming the output set *U* into a corresponding visual representation. By applying the mapping function $\varnothing(U)$. a fine-grained visualization is produced.

Fig. 7 presents a fine-grained interpretable example. On the left side of the figure, the coarse-grained interpretable regions identified by the model as areas of violence are shown, consisting of five segments. On the right side of the figure, the model's identified fine-grained path for the violent regions is displayed, clearly indicating that the areas of interest are where physical conflicts occur, followed by certain parts of human subjects. The path is encoded from 0 to 4 and visualized using a red line. It can be clearly observed that the model's decision to classify the image

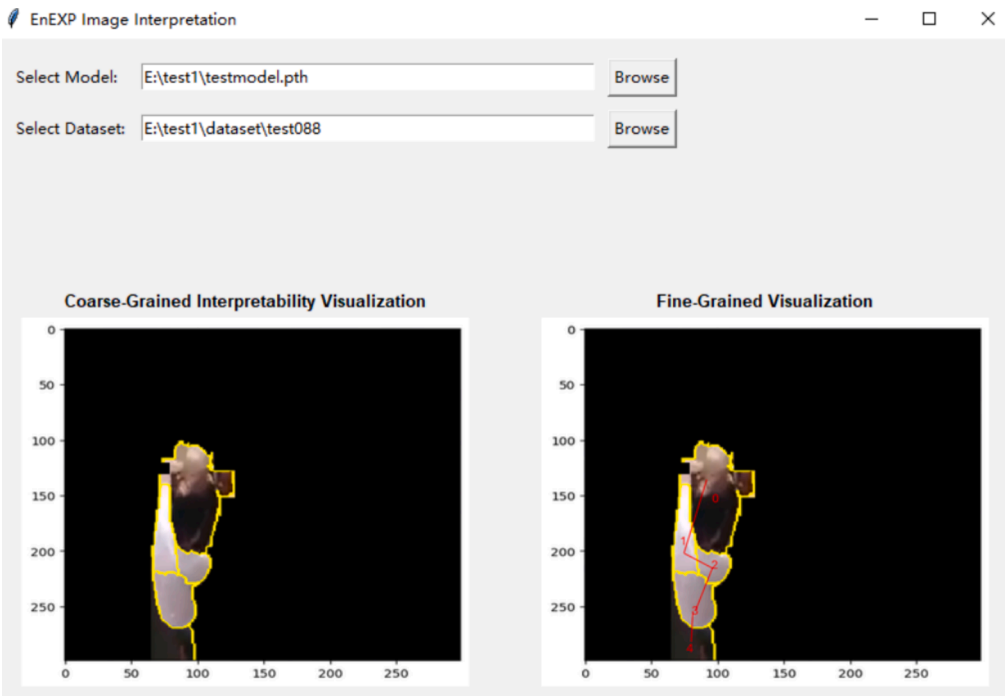


Fig. 7. Fine-Grained Interpretability Assessment.

as containing violence is primarily based on actions, followed by specific features of human subjects. This reasoning is almost identical to that of an expert.

5. Experiment

This section will introduce the model performance of the proposed EnExp interpretable method. All the information for the experiment is at the following the github URL.[1]

5.1. Image dataset and model selection

In the interpretable experiments with image data, this paper selects two datasets for study: Animal-10[2]and Animal-151.[3] The Animal-10 dataset contains 10 different types of animals, while the Animal-151 dataset includes 15 mainstream animal categories, each with 10 different rare species under each category. For each of these two data sets, 1000 images were selected, with a total of 2000 images for annotation and testing of the model. The model chosen for this research is the Inception-V3 model (Szegedy et al., 2016).

The inception-V3 model is an advanced deep convolutional neural network that has shown exceptional performance in image recognition and classification tasks. This paper aims to explore the rationality of the model's classification results in different tasks, whether at a coarse-grained or fine-grained level, through the method of interpretable machine learning.

5.2. Interpretable model performance in image classification

This section will present the performance of the proposed explainable machine learning method on image classification tasks. It is important to note that, except for the parameters specifically emphasized in Fig. 7, the λ in Exp. (1) is set to 0.25 in all experiments.

Table 2
The *M* score of each model under different numbers of random masking perturbation in Animal-10 and Animal-151.

number (Animal-10)	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
150	0.4883	0.4877	0.4972	**0.5077**
300	0.5327	0.5397	0.5453	**0.5520**
500	0.5448	0.5593	0.5739	**0.5916**
1000	0.5888	0.5984	0.6099	**0.6300**
number (Animal-151)	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
150	0.6332	0.6352	0.6424	**0.6447**
300	0.6826	0.6834	0.6946	**0.7074**
500	0.6983	0.7084	0.7202	**0.7388**
1000	0.7378	0.7387	0.7556	**0.7745**

Table 2 presents a comparative analysis of the explanatory performance of the EnEXP method proposed in this paper against other explanatory methods on different datasets for the Inception-V3 model. As shown in Table 2, the results on the Animal-10 and Animal-151 datasets indicate that, under random masking perturbation, the accuracy of the interpretable models increases with the number of masking perturbation. This trend is particularly evident when the number of masking perturbation reaches 1000, at which point the explanatory effect is most pronounced. This suggests that a larger volume of masking perturbation data allows explanatory methods to more effectively uncover the deep relationships between data features and outputs, thereby enhancing the *M* score. Compared to other explanatory methods, the experimental results demonstrate that the EnEXP method achieves higher explanatory accuracy under random masking perturbation. This is attributed to its use of an ensemble tree strategy, which optimizes overall explanatory performance by integrating the explanatory results of multiple tree models.

Tables 3 and 4 present the *M*−Score comparison under different masking perturbation counts (150, 300, and 1000) with a fixed masking

[1] https://github.com/1846659840/The-Research-on-a-new-LIME-based-interpretable-machine-learning-and-its-evaluation-method-imagepart.
[2] https://www.kaggle.com/datasets/alessiocorrado99/animals10.
[3] https://www.kaggle.com/datasets/sharansmenon/animals141.

Table 3
Each model's *M* score under the same quantity with different masking ratios in Animal-10 Dataset.

Num = 150	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
10 %	0.4894	0.4997	0.4994	**0.5211**
20 %	0.5021	0.5092	0.5248	**0.5320**
30 %	**0.5242**	**0.5310**	**0.5391**	**0.5455**
40 %	0.5192	0.5310	**0.5380**	0.5342
50 %	0.5082	0.5101	0.5180	**0.5322**
60 %	0.4925	0.4980	0.5123	**0.5232**
70 %	0.4563	0.4732	0.4935	**0.5012**
80 %	0.4012	0.4231	0.4431	**0.4673**
90 %	0.3896	0.4014	0.4237	**0.4368**
Num = 300	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
10 %	0.5116	0.5289	0.5206	**0.5451**
20 %	0.5240	0.5318	0.5535	**0.5619**
30 %	**0.5568**	**0.5585**	**0.5877**	**0.5934**
40 %	0.5423	0.5535	0.5592	**0.5625**
50 %	0.5316	0.5399	0.5405	**0.5534**
60 %	0.5185	0.5213	0.5360	**0.5470**
70 %	0.4785	0.4943	0.5152	**0.5258**
80 %	0.4302	0.4460	0.4715	**0.4931**
90 %	0.4306	0.4414	0.4643	**0.4773**
Num = 1000	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
10 %	0.5894	0.5996	0.5994	**0.6211**
20 %	0.6021	0.6092	0.6248	**0.6320**
30 %	**0.6242**	**0.6330**	**0.6538**	**0.7029**
40 %	0.6192	0.6299	**0.6380**	0.6342
50 %	0.6082	0.6099	0.6180	**0.6322**
60 %	0.5925	0.5981	0.6123	**0.6232**
70 %	0.5563	0.5732	0.5935	**0.6012**
80 %	0.5012	0.5231	0.5431	**0.5673**
90 %	0.4896	0.5014	0.5237	**0.5367**

perturbation ratio across two different datasets, Animal-10 and Animal-151. As shown in Tables 3 and 4, using a masking perturbation count of 1000 and varying the masking perturbations ratio across the two datasets, a 30 % masking perturbations ratio achieved the best explanation effect. This result indicates that a 30 % masking perturbation ratio significantly improves the accuracy of the model explanation. This is because, at a coverage rate of 0.3, the selected exponential kernel function provides optimal weights, while too high or too low a ratio results in inappropriate weights, affecting the explanation effectiveness. The experimental results further validate that, under the same masking perturbation data, the EnEXP method outperforms other methods at different fixed coverage rates, proving the effectiveness of EnEXP in enhancing the accuracy of model explanations. Compared to the random masking perturbation in Table 2, using fixed masking perturbation also showed superior performance. Moreover, the interpretable experimental results indicate that, although the Inception V3 deep learning model has achieved good performance for improvement in predicting certain species categories, especially in the Animal-10 dataset, where the explanation accuracy for 10 rare species is only 70 %.

Tables 5 and 6 aim to compare the proposed EnEXP method with previous works across three dimensions: M−Score, AED (Average Explanation Discrepancy), and Local Fidelity.

especially in the second dataset, where its Local Fidelity Score.

As shown in Tables 5 and 6, the EnEXP-Bagging Tree performed the best in the Animal-10 dataset, consistently demonstrating superior performance across all key metrics in the comparison of these two groups of explainable machine learning methods. In the first dataset, the EnEXP-Bagging Tree achieved an explanation accuracy (*M*−Score) of 0.7029, an average explanation discrepancy (AED) of 0.030, and a Local

Table 4
Each model's *M* score under the same quantity with different masking ratios in Animal-151 Dataset.

Num = 150	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
10 %	0.6362	0.6465	0.6462	**0.6680**
20 %	0.6489	0.6560	0.6716	**0.6788**
30 %	**0.6710**	**0.6778**	**0.6859**	**0.6923**
40 %	0.6660	0.6767	0.6848	**0.6810**
50 %	0.6550	0.6567	0.6647	**0.6789**
60 %	0.6402	0.6450	0.6591	**0.6701**
70 %	0.6031	0.6201	0.6403	**0.6480**
80 %	0.5480	0.5700	0.5899	**0.6141**
90 %	0.5364	0.5481	0.5705	**0.5835**
Num = 300	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
10 %	0.7013	0.7136	0.7125	**0.7360**
20 %	0.7147	0.7196	0.7319	**0.7455**
30 %	**0.7394**	**0.7340**	**0.7545**	**0.7618**
40 %	0.7263	0.7439	0.7493	**0.7487**
50 %	0.7226	0.7168	0.7283	**0.7451**
60 %	0.7044	0.7085	0.7272	**0.7319**
70 %	0.6673	0.6858	0.7101	**0.7103**
80 %	0.6181	0.6357	0.6594	**0.6751**
90 %	0.5967	0.6109	0.6350	**0.6458**
Num = 1000	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP-Bagging Tree
10 %	0.7362	0.7465	0.7462	**0.7680**
20 %	0.7489	0.7560	0.7716	**0.7788**
30 %	**0.7710**	**0.7798**	**0.8005**	**0.8496**
40 %	0.7669	0.7768	0.7848	**0.7810**
50 %	0.7549	0.7567	0.7647	**0.7789**
60 %	0.7492	0.7448	0.7591	**0.7700**
70 %	0.7031	0.7199	0.7403	**0.7480**
80 %	0.6480	0.6699	0.6899	**0.7141**
90 %	0.6364	0.6481	0.6704	**0.6835**

Table 5
Comparison with state-of-the-art works on the Animal-10 dataset.

Method	M−Score ↑	AED ↓	Local Fidelity Score ↑
LIME	0.6144	0.045	0.79
LIME-Decision Tree	0.6211	0.042	0.81
EnEXP-Boost Tree	0.6538	0.036	0.85
EnEXP-Bagging Tree	0.7029	0.030	0.90
Kernel-Shap	0.6243	0.038	0.82
Sliding Window	0.4764	0.065	0.72
AcME	0.6637	0.033	0.87

Table 6
Comparison with state-of-the-art works on the Animal-151 dataset.

Method	M−Score ↑	AED ↓	Local Fidelity Score ↑
LIME	0.7612	0.043	0.82
LIME-Decision Tree	0.7679	0.041	0.84
EnEXP-Boost Tree	0.8005	0.038	0.87
EnEXP-Bagging Tree	0.8496	0.032	0.92
Kernel-Shap	0.7842	0.039	0.85
Sliding Window	0.7612	0.046	0.71
AcME	0.6637	0.035	0.89

Fidelity Score of 0.90. In the second dataset, the M−Score of the EnEXP-Bagging Tree further increased to 0.8496, the AED decreased to 0.032, and the Local Fidelity Score improved to 0.92. The EnEXP-Bagging Tree consistently outperformed others in terms of explanation accuracy, robustness, and fidelity, demonstrating stability and reliability across different environments.

In comparison, the EnEXP-Boost Tree also performed well in both datasets, with M−Scores of 0.6538 and 0.8005, AEDs of 0.036 and 0.038, and Local Fidelity Scores of 0.85 and 0.87, indicating high explainability and stability. The Kernel-Shap and LIME series methods showed slightly weaker performance; although their explanation accuracy and fidelity were relatively close, their robustness was slightly lower. The Sliding Window method performed the worst in both datasets, was only 0.71, indicating instability in its explanatory performance across different environments. In summary, the EnEXP-Bagging Tree demonstrated the strongest explainability, robustness, and fidelity across different datasets and environments, making it the most recommended explanatory method. Other methods, while having their strengths and weaknesses, should be selected based on specific needs.

Table 7 discusses the changes in interpretability accuracy under different hyperparameter settings. The other experimental parameters were set with 1000 masking perturbations and a coverage ratio of 30 %. The experiment compared the effects of different λ settings in equation (1) on the performance of four interpretable machine learning models: LIME, LIME-Decision Tree, EnEXP-Boost Tree, and EnEXP-Bagging Tree. As shown in Table 7, with the variation of λ, the interpretability accuracy of each model exhibits significant patterns across both the Animal-10 and Animal-151 datasets. Overall, when λ is 0.05, the interpretability accuracy of the LIME model is 0.5582 on the Animal-10 dataset but only 0.3977 on the Animal-151 dataset. The performance of LIME-Decision Tree differs slightly between the two datasets, with an accuracy of 0.4175 on Animal-10 and 0.5582 on Animal-151. This indicates a significant difference in the performance of LIME and LIME-Decision Tree at low λ values. However, EnEXP methods remain relatively stable at low λ values. For instance, when λ is 0.10, EnEXP-Boost Tree achieves an accuracy of 0.7611 on the Animal-151 dataset, while EnEXP- Bagging Tree has an accuracy of 0.7268. This demonstrates that EnEXP methods can maintain high interpretability accuracy even at low λ values. As λ increases to 0.20, the performance of the LIME model improves across both datasets, with an accuracy of 0.6002 on the Animal-10 dataset. LIME-Decision Tree reaches its peak at a λ value of 0.25, achieving an accuracy of 0.7798 on the Animal-151 dataset. This suggests that moderate λ values help enhance the interpretability accuracy of models. EnEXP-Boost Tree achieves its highest accuracy of 0.8005 on the Animal-151 dataset at a λ value of 0.25, while EnEXP-Bagging Tree also performs excellently at the same λ value, reaching an accuracy of 0.8496. This indicates that EnEXP models perform best in the moderate λ range, making them the most suitable models for balancing complexity and interpretability.

As λ further increases, the interpretability accuracy of most models begins to decline. For example, when λ increases to 0.50, the accuracy of the LIME model drops to 0.4857 on the Animal-10 dataset and to 0.6054 on the Animal-151 dataset. LIME-Decision Tree and EnEXP models exhibit similar declining trends at high λ values, particularly when λ is 0.75, with EnEXP-Bagging Tree's accuracy dropping to 0.6552 on the Animal-151 dataset.

It is particularly noteworthy that when λ increases to 1.00, the interpretability accuracy of all models significantly decreases. For example, LIME's accuracy drops to 0.3862 on the Animal-10 dataset, and EnEXP-Boost Tree's accuracy fallsto 0.5888 on the Animal-151 dataset.

This phenomenon suggests that excessively large λ values may lead to imbalanced weight distribution in models, thereby weakening their interpretability. Therefore, selecting an appropriate λ value is crucial,

Table 8
Comparison with stability.

Animal-10/Random	LIME	LIME-Decision Tree	EnEXP-Boost	EnEXP-Bagging
1	0.5104	0.5144	0.5241	0.5271
2	0.4362	0.4769	0.4801	0.4937
3	0.5389	0.5401	0.5523	0.5738
4	0.5229	0.5331	0.5798	0.5904
5	0.3603	0.3628	0.3701	0.3742
standard deviation	0.0697	0.0668	0.0748	0.0821
Average value	0.4737	0.4854	0.5012	0.5118
Animal-10/30 % Ratio	LIME	LIME-Decision Tree	EnEXP-Boost	EnEXP-Bagging
30 %-1	0.5242	0.5311	0.5391	0.5456
30 %-2	0.4553	0.4767	0.5381	0.5407
30 %-3	0.4968	0.5081	0.5403	0.5507
30 %-4	0.5048	0.5618	0.5416	0.5436
30 %-5	0.4514	0.5282	0.5354	0.5474
standard deviation	0.0276	0.0293	0.0347	0.0389
Average value	0.4865	0.5212	0.5403	0.5405
Promote (Divide the standard deviation)	LIME 2.52	LIME-Decision Tree 2.28	EnEXP-Boost 2.16	EnEXP-Bagging **4.22**

Table 7
Experimental comparison on Animal-10 under different hyperparameters.

λ	Animal-10				Animal-151			
	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP Bagging Tree	LIME	LIME-Decision Tree	EnEXP Boost Tree	EnEXP-Bagging Tree
0.05	0.5582	0.3977	0.4175	0.5582	0.6107	0.6335	0.6928	0.7234
0.10	0.5742	0.4105	0.5227	0.5742	0.7156	0.6610	0.7611	0.7268
0.15	0.5922	0.5691	0.5916	0.5922	0.6423	0.6934	0.7745	0.7412
0.20	0.6002	0.6107	0.6351	0.6002	0.7439	0.7524	0.7810	0.8279
0.25	**0.6242**	**0.6330**	**0.6538**	**0.7029**	**0.7710**	**0.7798**	**0.8005**	**0.8496**
0.30	0.5522	0.6115	0.4612	0.5522	0.7381	0.6853	0.7769	0.8174
0.35	0.5302	0.5898	0.4279	0.5302	0.6847	0.5679	0.7745	0.8112
0.40	0.5152	0.6012	0.5734	0.5152	0.6262	0.7456	0.7481	0.7920
0.45	0.5006	0.5184	0.5026	0.5006	0.5915	0.6498	0.7264	0.7798
0.50	0.4857	0.4329	0.4128	0.4857	0.6054	0.6014	0.7258	0.7736
0.55	0.4706	0.5446	0.4251	0.4706	0.5330	0.7152	0.7153	0.7649
0.60	0.4629	0.4952	0.6292	0.4629	0.6758	0.6705	0.7075	0.7109
0.65	0.4453	0.4273	0.4829	0.4453	0.6735	0.5578	0.6902	0.6793
0.70	0.4301	0.3894	0.4485	0.4301	0.5484	0.6827	0.6352	0.6698
0.75	0.4262	0.5691	0.6187	0.4262	0.6802	0.7213	0.6119	0.6552
0.80	0.3972	0.4530	0.5042	0.3972	0.7501	0.7464	0.6199	0.6305
0.85	0.3846	0.5898	0.4373	0.3846	0.7332	0.7392	0.5893	0.5924
0.90	0.3694	0.4817	0.6003	0.3694	0.6259	0.6197	0.5698	0.6257
0.95	0.3692	0.4975	0.5374	0.3692	0.7285	0.5869	0.6475	0.5671
1	0.3862	0.3950	0.3916	0.3862	0.6714	0.6122	0.5888	0.5587

highlighting the importance of optimizing parameter selection in interpretable machine learning models.

Table 8 compares the improvement in interpretability stability between fixed-ratio masking perturbation and random masking perturbation. With a masking perturbation sample size of 150, stability enhancement was observed in the Animal-10 dataset using both random masking and a fixed 30 % ratio masking. As shown in Table 8, under fixed-ratio masking, the stability of the explanations improved across the board, with EnEXP-Bagging Tree exhibiting the most significant increase in interpretability. The rationale for using a fixed-ratio mask is that it ensures the decision trees generated by Bagging do not vary significantly from one another. When the masking ratio is fixed, each tree encounters consistently perturbed data, leading to similar decision-making behaviors across different trees. This consistency enhances the interpretability stability of the entire model. Consequently, when using the EnEXP-Bagging Tree model, this consistency results in the greatest improvement in interpretability.

Fig. 8. Visualization Example.

Fig. 8 presents three visual examples to validate the effectiveness of the proposed EnEXP method. The experiment was set to perform 1000 masking perturbations, with five sets of visual comparisons for each example. As shown in Fig. 8, in the interpretability analysis of the Shiba Inu, French Bulldog, and Holstein Cow, the results of LIME and LIME-DecisionTree exhibited randomness, while the proposed EnExp method provided consistent explanations across all five sets, demonstrating the stability of the EnExp method. Furthermore, the visual results clearly indicate that the proposed EnEXP method can identify more critical regions compared to LIME and LIME-DecisionTree, with EnEXP-Bagging tree offering the most accurate interpretations.

5.3. *Text data interpretable*

This section will introduce the experimental design for the interpretability of Chinese text. As shown in Fig. 9, the experiment starts with using Jieba for word segmentation of the dataset, followed by further cleansing the data with a stop-word list and a custom dictionary. Word vector transformation is then performed using TF-IDF, which is used to train the Support Vector Machine (SVM) model. After the model is trained, four different interpretable machine learning algorithms will be used to conduct an interpretative analysis of the SVM model and compare the interpretative accuracy of each algorithm. Based on the conclusions drawn from the best interpretive algorithm, the dataset and model will be reviewed and corrected, and the revised results will be compared with the GPT model. The purpose of conducting this experiment is to address real-life scenarios such as detecting bullying behavior in school settings (e.g., identifying abusive language) or filtering sensitive words in private online forums. These scenarios often involve edge computing and the deployment of localized models, requiring rapid response times. Although the existing Transformer-based LLM models have achieved significant success in text processing, their large parameter counts and need for extensive training corpora make them difficult to deploy effectively in these scenarios. Therefore, if improvements in interpretable machine learning could enable traditional BoW methods to provide good performance while remaining lightweight, it would facilitate successful deployment and application in these scenarios. More specifically, if traditional BoW models could achieve similar effectiveness to LLMs in detecting sensitive vocabulary and specific terms, this would further enhance their success rate in practical applications.

The dataset used in the experiment is ChnSentiCorp_htl_all[4], from which the first 1000 positive reviews and 1000 negative reviews were selected, totaling 2000 sentences. These sentences were annotated using a bag-of-words model for interpretability, identifying words considered positive or negative in each sentence. The interpretive model chosen was SVM, while the model used for comparison was a finetuned version of the GPT series (Radford et al., 2019) (Brown, 2020). Fig. 10 displays the version number of GPT-3 used in this experiment.

Table 9 compares the performance of various explainability methods across different metrics, focusing on M−Score, AED (Average Explanation Distance), and Local Fidelity Score. EnEXP-Boost Tree displayed the highest M−Score (0.7239), significantly outperforming other methods, indicating its superior accuracy in model explanation. In contrast, LIME's M−Score is 0.5147, while its variant, LIME-Decision Tree, scores 0.5468, with the latter improving explanation accuracy by integrating decision trees. On the AED metric, EnEXP-Boost Tree also performs best (0.027), suggesting that its explanations are more consistent with actual

[4] https://github.com/InsaneLife/ChineseNLPCorpus.

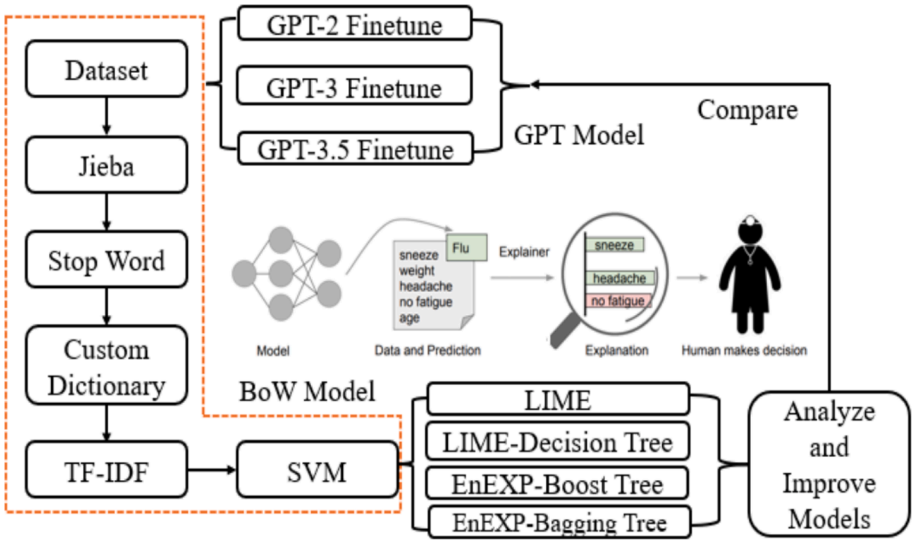


Fig. 9. Chinse Text Experiment Workflow.

davinci	Most capable GPT-3 model. Can do any task the other models can do, often with higher quality.	2,049 tokens	Up to Oct 2019
curie	Very capable, but faster and lower cost than Davinci.	2,049 tokens	Up to Oct 2019
babbage	Capable of straightforward tasks, very fast, and lower cost.	2,049 tokens	Up to Oct 2019
ada	Capable of very simple tasks, usually the fastest model in the GPT-3 series, and lowest cost.	2,049 tokens	Up to Oct 2019

Fig. 10. The version number of GPT-3 fine-tuning used in the experiment.

Table 9
Comparison with state-of-the-art works on the ChnSentiCorp_htl_all.

Method	M−Score ↑	AED ↓	Local Fidelity Score ↑
LIME	0.5147	0.048	0.52
LIME-Decision Tree	0.5468	0.045	0.55
EnEXP-Boost Tree	**0.7239**	**0.027**	**0.73**
EnEXP-Bagging Tree	0.6944	0.028	0.70
Kernel-Shap	0.6888	0.031	0.69
AcME	0.7058	0.029	0.71

model decisions. Among other methods, Kernel-Shap has a higher AED (0.031), possibly due to the more complex allocation formulas involved in SHAP computations. In terms of Local Fidelity Score, EnEXP-Boost Tree also ranks highest (0.73), further validating the reliability and consistency of its explanations. Lower scores, such as LIME (0.52), imply that while it is easy to understand, it may lack sufficient detail and precision. Experimental results indicate that methods based on ensemble learning, such as Boosting and Bagging, may be more effective in building interpretable models, especially in scenarios requiring high precision and consistency.

Fig. 11 displays the statistics of interpretable vocabulary using the EnEXP-Boost Tree, where subfigure a accounts for words not matched in negative vocabulary, and subfigure b for those not matched in positive vocabulary. From Fig. 11, subfigures (a) and (b) respectively show the vocabulary not matched in negative and positive sentiment texts. These data reflect specific contexts or biases in text classification when using the EnEXP-Boost Tree for interpretability analysis. In subfigure a, terms like “企業” (enterprise) and “反饋” (feedback) frequently appear in negative reviews. For instance, “企業” (enterprise) is often associated with recruitment ads, appearing in negative texts as “企業招聘,歡迎洽談” (enterprise recruitment, welcome to negotiate), but not found in positive texts. This may indicate that recruitment ads are linked to negative emotions of users, or that data cleaning was not effective in distinguishing between ads and substantive content. “反饋” (feedback) mainly relates to customer service and hotel feedback, typically mentioned in negative reviews, likely because customers are more inclined to leave feedback when dissatisfied. To address these phenomena, irrelevant ads associated with these terms were removed during data cleaning, and all text following customer feedback was deleted because such texts are often positive, potentially affecting the model’s judgment. For subfigure b, the absence of words like “服務” (service) and “滿意” (satisfaction) in positive sentiment analysis indicates that the trained SVM model did not omit these positive terms.

Table 10 aims to compare the re-calibrated SVM model after applying interpretable machine learning techniques with the original SVM model. As shown in Table 10, the model’s performance has improved following the application of interpretable machine learning, demonstrating that interpretable machine learning can elucidate the importance of features in machine learning models.

This experiment utilized GPT series models for evaluation and experimentation. Tables 11, 12, and 13 present the results of testing the revised datasets and the initial datasets using GPT-2 and GPT-3, respectively. As shown in Tables 11 and 12, the experimental results

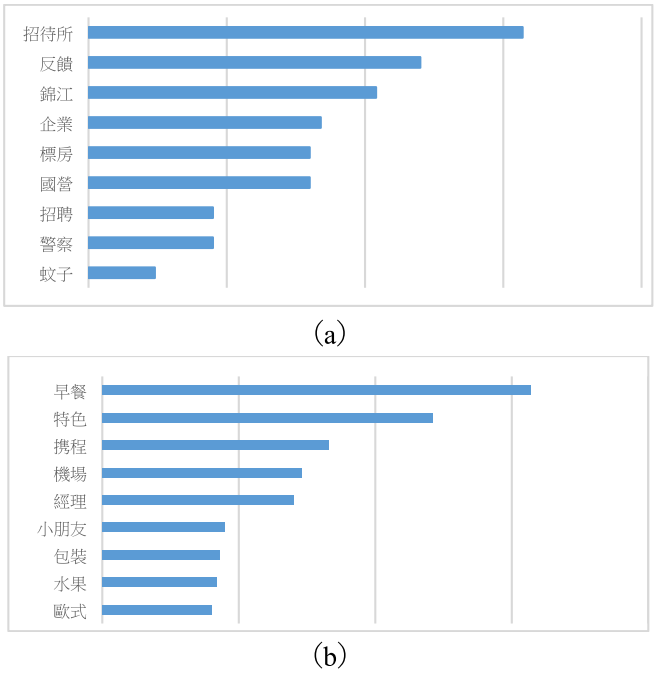


Fig. 11. (a) shows the emotional words in negative sentiments that were not tagged in the original data used for training the SVM statistics. Fig. 11(b) presents emotional words in positive sentiments that were not tagged in the original data used for training the SVM statistics.

Table 10
Comparison with SVM and SVM—Adjust.

SVM	Precision	Recall	F1-score	Accuracy
0	0.87	0.93	0.89	0.90
1	0.94	0.91	0.92	
SVM—Adjust	**Precision**	**Recall**	**F1-score**	**Accuracy**
0	0.91	0.93	0.92	0.94
1	0.93	0.92	0.92	

Table 11
Comparison with GPT-2 and GPT-2—Adjust.

GPT-2	Precision	Recall	F1-score	Accuracy
0	0.84	0.91	0.87	0.87
1	0.90	0.82	0.86	
GPT-2—Adjust	**Precision**	**Recall**	**F1-score**	**Accuracy**
0	0.88	0.91	0.90	0.90
1	0.91	0.88	0.89	

indicate that using data refined through interpretable machine learning improves the model's performance compared to the original data, highlighting the significance of interpretable machine learning. Furthermore, the performance of the bag-of-words model, refined through interpretable machine learning, surpassed that of the original GPT-3 Ada model, approaching the performance of GPT-3 Babbage.

Additionally, comparing Tables 12 and 13, the experiment used the modified data with explanations to fine-tune the GPT-3 model, but the results were not significantly different. This indicates that, despite the enhancement in data quality through explanatory modifications, large language models still exhibit considerable limitations in processing these modified data. This phenomenon reflects that the adaptability and generalization capabilities of current large language models for specific

Table 12
Using Original DATA FOR GPT-3 model.

Davinci	Precision	Recall	F1	Accuracy	Cost
0	0.98	0.94	0.96	0.97	1.87USD
1	0.99	0.97	0.98		
Curie	**Precision**	**Recall**	**F1**	**Accuracy**	**Cost**
0	0.97	0.94	0.96	0.95	1.34USD
1	0.95	0.96	0.95		
Babbage	**Precision**	**Recall**	**F1**	**Accuracy**	**Cost**
0	0.93	0.91	0.92	0.95	0.74USD
1	0.92	0.96	0.94		
Ada	**Precision**	**Recall**	**F1**	**Accuracy**	**Cost**
0	0.89	0.90	0.89	0.90	0.13USD
1	0.91	0.89	0.89		
SVM—Adjust	**Precision**	**Recall**	**F1**	**Accuracy**	
0	0.91	0.93	0.92	0.94	
1	0.93	0.92	0.92		

Considering that the performance of the GPT fine-tuned model improves with updates, it is declared that the GPT-3 used in this study was fine-tuned using the official model via API, with the testing date being Dec. 6, 2022.

Table 13
Using Adjust Data for GPT-3 model.

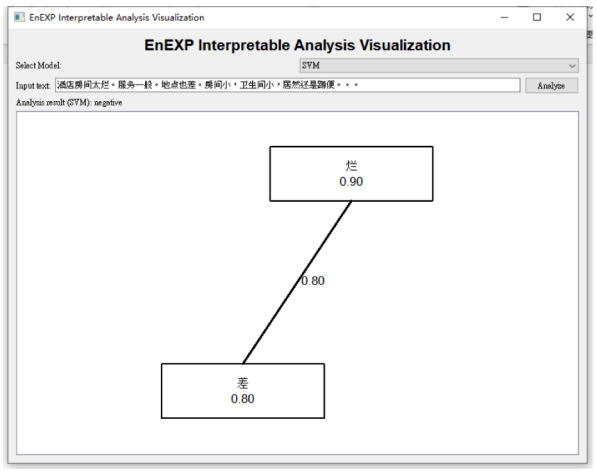
Davinci	Precision	Recall	F1	Accuracy	Cost
0	0.98	0.94	0.96	0.97	1.07USD
1	0.99	0.97	0.98		
Curie	**Precision**	**Recall**	**F1**	**Accuracy**	**Cost**
0	0.97	0.94	0.96	0.96	1.24USD
1	0.96	0.96	0.96		
Babbage	**Precision**	**Recall**	**F1**	**Accuracy**	**Cost**
0	0.93	0.91	0.92	0.95	0.48USD
1	0.92	0.96	0.94		
Ada	**Precision**	**Recall**	**F1**	**Accuracy**	**Cost**
0	0.91	0.91	0.91	0.91	0.13USD
1	0.91	0.89	0.90		
SVM—Adjust	**Precision**	**Recall**	**F1**	**Accuracy**	
0	0.91	0.93	0.92	0.94	
1	0.93	0.92	0.92		

Given the enhancement in the performance of GPT fine-tuned models through updates, this document specifies that the GPT-3 model employed here underwent fine-tuning via the official API, with the test conducted on April 6, 2023.

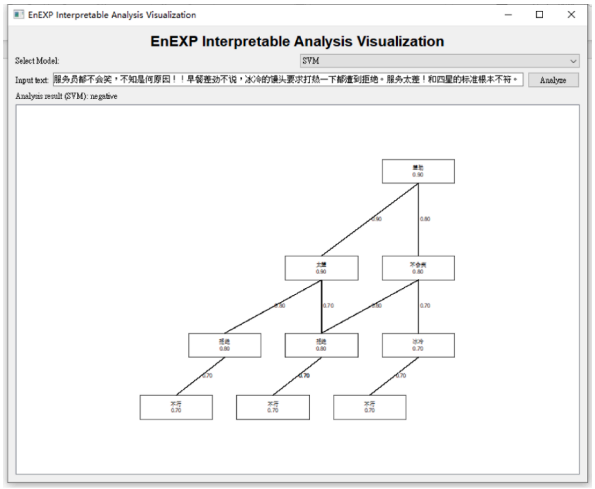
tasks may be insufficient, necessitating further research and improvement.

Fig. 12 show some Examples of EnExp Interpretable Analysis Visualization. As shown in Fig. 12, the proposed EnExp method, through the output of the tree model, can clearly explain the trained Support Vector Machine (SVM) model and provide intuitive visual results. This visualization technique effectively reveals the internal decision logic of the SVM model, offering valuable insights into feature importance and decision boundaries. The tree structure in the visualization reflects the hierarchical nature of the model's decision process, with each node corresponding to a key feature or decision point within the SVM's hyperplane. The branching patterns and associated quantifications illustrate the relative impact of different features on the model's predictions, enabling users to gain a deep understanding of the model's behaviour.

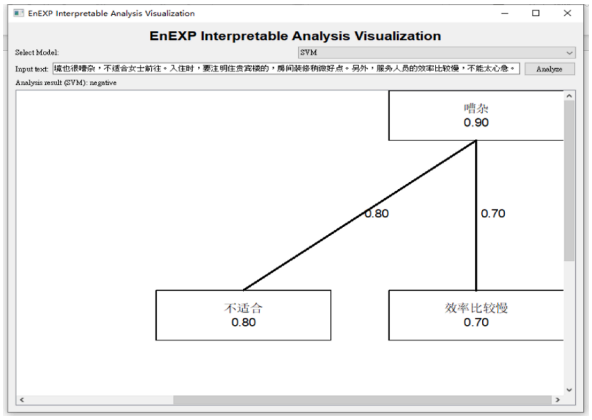
Table 14 compares the *M* score performance of various interpretable models on the ChnSentiCorp_htl_all dataset under different hyperparameters λ. As shown in Table 14, with the increase of the hyperparameter λ, all models exhibit a varying degree of performance decline. Among them, the EnEXP-Boost Tree performs the best when λ is small ($\lambda \leq 0.25$), with the performance indicator rising from 0.6098 to 0.7239,



Input：“The hotel room is terrible. The service is average. The location is also bad. The room is small, the bathroom is small, and it is a squat toilet... “



Input：“Although it is a four-star hotel, the waiters here don't smile. I don't know why!! The breakfast is terrible, and the cold steamed buns were refused when I asked to heat them up. The service is terrible! It doesn't meet the standards of a four-star hotel at all. “



Input: The hotel is located opposite the train station, with dense traffic and a noisy environment, which is not suitable for ladies. When checking in, please indicate that you are staying in the VIP building, the room decoration is slightly better. In addition, the service staff is relatively slow, so don't be too impatient.

Fig. 12. Some Examples of EnExp Interpretable Analysis Visualization.

Table 14
Experimental comparison on ChnSentiCorp_htl_all under different hyperparameters.

λ	ChnSentiCorp_htl_all			
	LIME	LIME-Decision Tree	EnEXP-Boost Tree	EnEXP Bagging Tree
0.05	0.4398	0.4519	0.6098	0.5745
0.10	0.4547	0.4812	0.6215	0.5867
0.15	0.4701	0.4921	0.6528	0.6493
0.20	0.4902	0.5327	0.7085	0.6745
0.25	**0.51147**	**0.5468**	**0.7239**	**0.6944**
0.30	0.4826	0.5184	0.6952	0.6639
0.35	0.4289	0.5039	0.6791	0.6319
0.40	0.4103	0.4657	0.6653	0.6194
0.45	0.3987	0.4395	0.5923	0.6012
0.50	0.3945	0.4238	0.6098	0.5634
0.55	0.3806	0.4123	0.5807	0.5745
0.60	0.3685	0.3971	0.5536	0.5378
0.65	0.3539	0.3845	0.5687	0.5511
0.70	0.3427	0.3698	0.5402	0.5102
0.75	0.3345	0.3584	0.5105	0.5254
0.80	0.3158	0.3325	0.5279	0.4991
0.85	0.3074	0.3458	0.4856	0.4847
0.90	0.2861	0.2987	0.4987	0.4618
0.95	0.2705	0.3125	0.4732	0.4476
1	0.2593	0.2864	0.4601	0.4303

and then gradually decreasing to 0.4601 as λ increases. In contrast, the performance of the EnEXP-Bagging Tree is slightly inferior, reaching a peak of 0.6944 at $\lambda = 0.25$, but overall, its performance remains lower than that of the Boost Tree. The performance of LIME and LIME-Decision Tree is relatively close, and they show a rapid decline when $\lambda > 0.60$, indicating weaker interpretability at high λ. Therefore, the EnEXP-Boost Tree is the best-performing model across different λ values, especially when a balance between interpretability and performance is required.

6. Limitation

The EnEXP method proposed in this study primarily focuses on image and text classification tasks and has been explored for specific types of models. Specifically, our experiments and analyses cover the following models:

1. InceptionV3 network
2. Ensemble learning models (such as Support Vector Machines, SVM)

It is important to note that the scope of this study has certain limitations.

This paper relies on raw pixels for image analysis. However, interpretation methods based on raw pixels can only provide implicit results, making it difficult to give meaningful and completely explicit explanations in terms of specific domain knowledge (e.g., medical applications or fine quality defect detection in manufacturing). This limitation arises because such methods lack the incorporation of domain-specific expertise in their interpretation processes, which restricts the transparency and contextual applicability of the explanations.

Furthermore, global interpretation of the boosting tree model also faces certain challenges, as the most critical segments may not align with the branches relevant to new cases. Compared to local explanations based on a single branch, this suggests that the view of global importance often lacks precision for individual cases. Therefore, we believe it is necessary to clearly state the limitations of “global interpretability,” including the impact of the weighting method proposed in this paper on such interpretations.

In terms of evaluation methods, comparing the segments identified by experts with those discovered by the model is a meaningful attempt. However, in some sensitive domains, such a comparison is still a relatively weak matching criterion, as it does not take into account the order

of feature importance. Although this paper proposes a fine-grained approach to visual output, it still lacks more accurate quantitative metrics for overall matching. For instance, using Kendall's rank correlation coefficient could be considered to evaluate the alignment between the segments identified by the model and the decisions made by domain experts. These improvements will be explored in future research.

This study has not yet explored the application of the EnEXP method to other tasks and models, such as object detection, speech recognition, reinforcement learning, and some emerging deep learning architectures. We recognize that with the rapid development of machine learning and artificial intelligence, new model architectures and application scenarios continue to emerge. Therefore, expanding the applicability of the EnEXP method and exploring its performance across a broader range of tasks and models will be one of our important research directions in the future.

7. Conclusion

This paper introduces EnEXP, an innovative interpretable machine learning method based on ensemble trees, designed to explain deep learning models in image and text data. By integrating multiple decision tree models, EnEXP enhances the precision and robustness of explanations. When dealing with complex image and text data, EnEXP can uncover key factors and patterns in the decision-making process of deep learning models. Experimental results show that this ensemble tree-based method not only provides intuitive explanations in qualitative analysis but also demonstrates higher interpretability accuracy in quantitative evaluations compared to traditional masking perturbation methods. Therefore, EnEXP serves as an effective tool for understanding and validating the decisions of deep learning models, particularly suitable for applications requiring high reliability and transparency. The experimental results also indicate that, despite the significant advantages of large language models (LLMs) in processing text data, the traditional bag-of-words model, when refined with interpretable machine learning methods and combined with SVM, can achieve performance surpassing some LLMs. Compared to the lack of interpretability and privacy security concerns in enterprise applications of LLMs, this approach offers better assurances. However, generative explanations based on LLMs remain challenging, and future research will focus on addressing this issue and further exploring other types of unstructured data.

CRediT authorship contribution statement

Yue-Shi Lee: Conceptualization, Supervision, Validation, Writing – review & editing, Project administration. **Show-Jane Yen:** Conceptualization, Supervision, Validation, Writing – review & editing, Project administration. **Wendong Jiang:** Methodology, Writing – original draft. **Jiyuan Chen:** Software, Writing – original draft. **Chih-Yung Chang:** Formal analysis, Resources.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All data used in the experiments can be found in the public domain.

References

Liu, Y., et al. (2024). Few-shot object detection in remote-sensing images via label-consistent classifier and gradual regression. *IEEE Transactions on Geoscience and Remote Sensing, 62*, 1–14. https://doi.org/10.1109/TGRS.2023.3325086

Xiao, R., et al. (2023). Towards energy-preserving natural language understanding with spiking neural networks. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 31*, 439–447. https://doi.org/10.1109/TASLP.2022.3229543

Hold, C., Pulkki, V., Politis, A., & McCormack, L. (2024). Compression of higher-order ambisonic signals using directional audio coding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing, 32*, 651–665. https://doi.org/10.1109/TASLP.2023.3328284

Shi, W., Huang, G., Song, S., Wang, Z., Lin, T., & Wu, C. (2022). Self-supervised discovering of interpretable features for reinforcement learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(5), 2712–2724. https://doi.org/10.1109/TPAMI.2021.3054886

Cao, Q., Liang, X., Li, B., & Lin, L. (2021). Interpretable visual question answering by reasoning on dependency trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(3), 887–901. https://doi.org/10.1109/TPAMI.2019.2943215

Liu, H., Wang, R., Shan, S., & Chen, X. (2021). What is a Tabby? Interpretable model decisions by learning attribute-based classification criteria. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 43*(5), 1791–1807. https://doi.org/10.1109/TPAMI.2019.2954820

Wu, P., Liu, X., & Liu, J. (2023). Weakly supervised audio-visual violence detection. *IEEE Transactions on Multimedia, 25*, 1674–1685. https://doi.org/10.1109/TMM.2022.3158975

Iqbal, S., Qureshi, A. N., Alhussein, M., Aurangzeb, K., & Anwar, M. S. (2024). AD-CAM: Enhancing interpretability of convolutional neural networks with a lightweight framework - From black box to glass box. *IEEE Journal of Biomedical and Health Informatics, 28*(1), 514–525. https://doi.org/10.1109/JBHI.2023.3321290

Habib, A., Karmakar, C., & Yearwood, J. (2023). Interpretability and optimisation of convolutional neural networks based on sinc-convolution. *IEEE Journal of Biomedical and Health Informatics, 27*(4), 1758–1769. https://doi.org/10.1109/JBHI.2022.3224707

Singh, P., & Sharma, A. (2022). Interpretation and classification of arrhythmia using deep convolutional network. *IEEE Transactions on Instrumentation and Measurement, 71*, 1–12. https://doi.org/10.1109/TIM.2022.3141611

Mahapatra, D., Ge, Z., & Reyes, M. (2022). Self-supervised generalized zero-shot learning for medical image classification using novel interpretable saliency maps. *IEEE Transactions on Medical Imaging, 41*(9), 2443–2456. https://doi.org/10.1109/TMI.2022.3164855

Ribeiro, M., Singh, S., & Guestrin, C. (2016). 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* (pp. 1135–1144). https://doi.org/10.1145/2939672.2939778

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of the European Conference on Computer Vision* (pp. 818–833). https://doi.org/10.1007/978-3-319-10590-1_53

Petsiuk, V., Das, A., & Saenko, K. (2018). RISE: Randomized input sampling for explanation of black-box models. In *Proceedings of the British Machine Vision Conference* (Art. no. 151).

Ancona, M., Oztireli, C., & Gross, M. (2019). Explaining deep neural networks with a polynomial time algorithm for Shapley values approximation. In *Proceedings of the International Conference on Machine Learning* (pp. 272–281).

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Neural Information Processing Systems* (pp. 4765–4774).

Zhang, J., Bargal, S. A., Lin, Z., Brandt, J., Shen, X., & Sclaroff, S. (2018). Top-down neural attention by excitation backprop. *International Journal of Computer Vision, 126*(10), 1084–1102. https://doi.org/10.1007/s11263-018-1078-2

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *IEEE International Conference on Computer Vision* (pp. 618–626). https://doi.org/10.1109/ICCV.2017.74

Binder, A., Montavon, G., Lapuschkin, S., Müller, K.-R., & Samek, W. (2016). Layer-wise relevance propagation for neural networks with local renormalization layers. In *Proceedings of the International Conference on Artificial Neural Networks* (pp. 63–71). https://doi.org/10.1007/978-3-319-44781-0_8

Shrikumar, A., Greenside, P., Shcherbina, A., & Kundaje, A. (2017). Learning important features through propagating activation differences. In *Proceedings of the International Conference on Machine Learning* (pp. 3145–3153).

Smilkov, D., Thorat, N., Kim, B., Viegas, F., & Wattenberg, M. (2017). SmoothGrad: Removing noise by adding noise. *arXiv:1706.03825*.

Zahavy, T., Ben-Zrihem, N., & Mannor, S. (2016). Graying the black box: Understanding DQNs. In *Proceedings of the International Conference on Machine Learning* (pp. 1899–1908).

Zafar, M. R., & Khan, N. (2021). Deterministic local interpretable model-agnostic explanations for stable explainability. *Machine Learning and Knowledge Extraction, 3*, 525–541. https://doi.org/10.3390/make3030027

Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529. https://doi.org/10.1038/nature14236

Engel, Y., & Mannor, S. (2001). Learning embedded maps of Markov processes. In *Proceedings of the International Conference on Machine Learning* (pp. 138–145).

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2818-2826). Las Vegas, NV, USA. 10.1109/CVPR.2016.308.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. Retrieved from https://openai.com.

Brown, T. (2020). Language models are few-shot learners. *Proceedings of the 33rd Neural Information Processing Systems*.

Jiang, W.-D., Lee, Y.-S., & Yen, S.-J. (2023). The study of explainable machine learning research on attribute selection. In Proceedings of the Symposium on Digital Life Technologies (DLT'2023).

Liu, Y., Wang, T., & Chu, F. (2024). Hybrid machine condition monitoring based on interpretable dual tree methods using Wasserstein metrics. *Expert Systems with Applications, 235*, Article 121104. https://doi.org/10.1016/j.eswa.2023.121104

Dandolo, D., Masiero, C., Carletti, M., Pezze, D. D., & Susto, G. A. (2023). AcME—Accelerated model-agnostic explanations: Fast whitening of the machine-learning black box. *Expert Systems with Applications, 214*, Article 119115. https://doi.org/10.1016/j.eswa.2022.119115

Bakchy, S. C., Peyal, H. I., Hoshen, M. R., et al. (2024). Colon cancer detection using a lightweight-CNN with Grad-CAM++ visualization. In *2024 3rd International Conference on Advancement in Electrical and Electronic Engineering (ICAEEE)* (pp. 1-6). IEEE.

Bibal, A., Delchevalerie, V., & Frénay, B. (2023). DT-SNE: T-SNE discrete visualizations as decision tree structures. *Neurocomputing, 529*, 101–112. https://doi.org/10.1016/j.neucom.2023.01.073