Tensor Polynomial Additive Model

Yang Chen, Ce Zhu, *Fellow, IEEE*, Jiani Liu, Yipeng Liu, *Senior Member, IEEE*

Abstract—Additive models can be used for interpretable machine learning for their clarity and simplicity. However, In the classical models for high-order data, the vectorization operation disrupts the data structure, which may lead to degenerated accuracy and increased computational complexity. To deal with these problems, we propose the tensor polynomial addition model (TPAM). It retains the multidimensional structure information of high-order inputs with tensor representation. The model parameter compression is achieved using a hierarchical and low-order symmetric tensor approximation. In this way, complex high-order feature interactions can be captured with fewer parameters. Moreover, The TPAM preserves the inherent interpretability of additive models, facilitating transparent decision-making and the extraction of meaningful feature values. Additionally, leveraging TPAM's transparency and ability to handle higher-order features, it is used as a post-processing module for other interpretation models by introducing two variants for class activation maps. Experimental results on a series of datasets demonstrate that TPAM can enhance accuracy by up to 30%, and compression rate by up to 5 times, while maintaining a good interpretability.

Index Terms—additive models, interpretable machine learning, model compression, class activation map.

◆

1 INTRODUCTION

Interpretability is crucial for understanding and validating the internal workings of machine learning models, facilitating trust, fairness, and accountability in decision-making processes. Many interpretable methods have been proposed to unveil the decision-making process of deep networks [1]–[3]. The post-hoc interpretability techniques approximate the decisions made by an already trained model to gain insights into its behaviors [4]–[9], while the self-interpreting models perform transparent and understandable decision-making processes during inference without the need for additional post hoc interpretability techniques [10]–[14].

The generalized additive model (GAM) is a well-established approach for its straightforward interpretability [10]. To handle nonlinearly transformed input features, various GAM extensions have been developed, such as NAM [13] and GA^{2}Ms [12]. Although additive models are interpretable, capturing higher-order feature interactions requires significant computational resources and leads to an exponential growth in weight space. The scalable polynomial additive model (SPAM) [14] utilizes polynomial approximation to improve model fitting performance, and employs tensor decomposition [14]–[16] to reduce model complexity, thereby achieves scalable additive models.

However, when it comes to higher-order tensor inputs, additive models based on vector representation may suffer from data structure loss and higher computational burden due to the vectorization operations, thus degenerating the system performance. Consequently, it is necessary to develop efficient self-interpretation models suitable for higher-order tensor data inputs.

As a transparent "white-box" model, the additive model can not only inherently explain its own internal decisions, but also frequently embedded in other neural network methods to gain insights into their behaviors. For instance, GAMs are frequently employed as the last layer in a convolutional neural network (CNN), functioning as a component of post-hoc interpretative method. The class activation maps (CAM) [17]–[25] generate saliency maps by applying the weights from the classifier header to the feature maps, thereby visualizing important regions in the input image for model classification decisions. However, existing additive model classifier heads provide only a single weight, which may result in poorly fine-grained and inaccurate saliency maps [26]–[29].

In this paper, we propose the tensor polynomial addition model (TPAM), which is characterized by its better accuracy, compression, and interpretation. For accuracy, TPAM directly employs high-order tensors as inputs to avoid performance loss due to vectorization. This is particularly beneficial for visual data, where TPAM enhances accuracy by integrating the a priori knowledge of CNNs with both local and global information. For compression, TPAM utilizes a hierarchical low-order symmetric tensor approximation in the weight space. It reduces computational complexity and enables the capturing of higher-order feature interactions with fewer parameters. As shown in Fig. 1a, TPAM achieves high performance with a minimal number of parameters. Regarding interpretability, as depicted in Fig. 1b, TPAM's clear decision-making process allows for extracting important values for each feature, which demonstrates its excellent self-explanation capability. Additionally, TPAM can serve as a post-processing module for other interpretative models. By integrating TPAM as a classification header in CAM, we develop two variants, i.e., P-CAM and PI-CAM. They directly input the feature map and apply a factor weighting tensor, matching the feature map's size, to achieve a more comprehensive interpretation. As shown in Fig. 1c, the structured weighting of feature maps in TPAM-based P-CAM and PI-CAM significantly refines the

(a) Interpretable model performance.   (b) Explanation of TPAM.   (c) P-CAM and PI-CAM Performance.
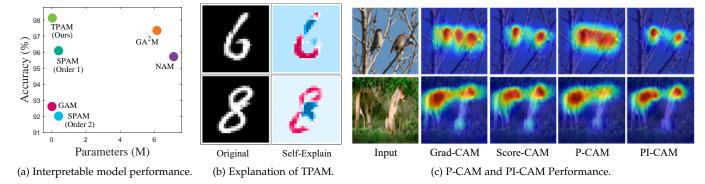
Fig. 1. (a) Classification performance of the self-interpretation model on the MNIST dataset. TPAM has the highest performance with a low number of parameters. (b) Explanation of the generation of the TPAM. Red represents the positive contribution and blue represents the negative contribution, just like in subjective human judgment. (c) P-CAM and PI-CAM Performance. The resulting saliency maps are more fine-grained.

granularity of the resulting saliency maps.

Our main contributions can be summarized as follows:

- We propose an interpretable polynomial model, referred to as TPAM, which takes tensor representation for higher-order input and utilizes low-rank symmetric tensor approximation to avoid the computational complexity of high-dimensional feature interactions.
- In the content of CAM, we propose two new methods, namely P-CAM and PI-CAM, to improve the fine-grainness of saliency maps. P-CAM and PI-CAM replace the classification head with TPAM in CAM, enabling a structured weighting of the feature maps and thereby benefiting the generation of detailed and precise explanations.
- As shown in Fig. 1, the proposed TPAM achieves a high level of accuracy with a reduced number of parameters. Its capability to handle high-order inputs and transparency benefits its interpretability and the application as a post-hoc interpretation technique.

2 NOTATIONS AND RELATED WORKS

2.1 Notations

Vectors, matrices, and tensors are denoted as bold lowercase, bold uppercase, and bold Eulerian letters, e.g., $\mathbf{x}$, $\mathbf{X}$, $\mathcal{X}$. x_{ij} and x_i represent the i,j-th element of a matrix $\mathbf{X}$ and i-th element of a vector x, respectively. A tensor of D-th order is represented as $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_D}$, whose $i_1, \cdots, i_D$-th element is denoted as $x_{i_1, \cdots, i_D}$. The Hadamard product and outer product are denoted as $\otimes$ and $\circ$, respectively. $\odot_k$ stands for the inner product of the repeated pattern along the k-th mode.

2.2 Interpretable Additive Modeling Machine Learning

The generalized additive modes (GAMs) [10] are favored in academia for their straightforward structure and transparent decision-making. However, they face limitations with complex data. The GA^{2}M model addresses this by incorporating second-order interactions into GAMs, thereby enhancing their capability. Despite its improvements, GA^{2}M struggles with large datasets due to the high computational demands during retraining. An alternative approach involves using interpretable transformations in neural networks to handle nonlinear data more effectively. For instance, neural additive models (NAMs) [13] employ neural network-based nonlinear transformations on individual features, improving model fit without the need for complex high-order interactions among features. Additionally, the NODE-GAM [3] model merges the nonlinear processing of NAMs with the advantages of decision trees, resulting in superior performance across various datasets.

scalable polynomial additive model (SPAM) [14] considers the interactions among features in a model of arbitrary order. It utilizes polynomials to capture and fit these features, resulting in improved model performance, and employs low-rank tensor decomposition to effectively reduce the number of parameters in the model. However, the above interpretable model is limited to one-dimensional inputs. When the input data consists of matrices or higher-order tensors, data need to be vectorized, resulting in the loss of spatial structure and a subsequent decrease in model performance. Our proposed model takes higher-order tensors as direct inputs and utilizes tensor decomposition techniques to greatly reduce the number of parameters. Consequently, this approach does not introduce any performance loss.

2.3 Saliency Map Generation

Additive models are known for their superior transparency and are often used as part of an auxiliary interpretation tool for convolutional neural network (CNN) classifiers. In this realm, the class activation map (CAM) [30] technique stands out as a distinctive post-hoc interpretation method. It efficiently creates saliency maps by utilizing the weights of the additive classifier to linearly weight the feature maps. The primary objective of this approach is to underscore the image regions most relevant to the target classification. There are two principal types of CAM techniques. The first relies on gradient backpropagation, employing the gradient of the additive model to weight the feature maps. Examples include Grad-CAM [17], Grad-CAM++ [19], XGrad-CAM [31], and Layer-CAM [32]. The second type depends on network forward reasoning, using the weights of the additive model to generate confidence scores, which in turn determine the weights for creating the saliency maps. Notable methods here are Score-CAM [28], IS-CAM [33], and SS-

CAM [24]. However, current CAM techniques, regardless of type, can only assign a single weight to the feature maps, which constrains the precision of the resulting saliency maps.

2.4 Polynomial Neural Network Learning

Recent studies have demonstrated the effectiveness of polynomial interactions in approximating simple neural networks and various functions. Among these, multiplicative interactions stand out as an effective polynomial interaction model. The studies in [34]–[38] offer theoretical insights into the benefits of polynomial connections, particularly for second- or third-order interactions. Moreover, [39]–[41] expands the use of the polynomial model to higher-order multiplicative interactions. Concurrently, polynomial addition [39]–[41] emerged as an innovative approach in interaction modeling. While additive models are more resource-intensive and require careful development, they are superior in improving decision-making interpretability and transparency.

3 METHODS

3.1 Scalable Polynomial Additive Models (SPAM)

Polynomial models characterize different levels of interactions between features by adjusting the degree of the polynomial. For a vector feature input $\mathbf{x} \in \mathbb{R}^I$, the learning of a polynomial of order K is as follows:

$$P(\mathbf{x}) = b + \sum_{k=1}^{K} \mathcal{W}^{(k)} \odot_k \mathbf{x} \qquad (1)$$

where $\mathcal{W}^{(k)} \in \mathbb{R}^{I \times \cdots \times I}$ is a k-th order tensor, which denotes the weights of the k-th order interaction of features in $\mathbf{x}$. Large input dimension I and order K result in a sharp increase in the weight space's dimension $\mathcal{O}\left(I^K\right)$, which makes the computation and model optimization challenging.

To simplify the model, SPAM [14] uses CP decomposition to factorize the symmetric weight tensor $\mathcal{W}^{(k)}$ as

$$\mathcal{W}^{(k)} = \sum_{r=1}^{R} \lambda_{kr} \cdot \underbrace{(\mathbf{u}_{kr} \circ \cdots \circ \mathbf{u}_{kr})}_{k \text{ times}} \qquad (2)$$

where $\{\mathbf{u}_{kr}\}_{r=1}^{R} \in \mathbb{R}^I$ are factor vectors, R represents the CP rank, and $\{\lambda_{kr}\}_{r=1}^{R}$ represents the wights of each rank one component. Then the polynomial (1) boils down to

$$P(\mathbf{x}) = b + \sum_{k=1}^{K} \sum_{r=1}^{R_k} \lambda_{kr} \cdot \langle \mathbf{u}_{kr}, \mathbf{x} \rangle^k \qquad (3)$$

which reduces the number of parameters required for estimation from $\mathcal{O}(I^K)$ to $\mathcal{O}(IKR)$. To simplify the presentation, we assume that $R_k = R$.

However, when the input data is a high-order tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, the model needs to transform the tensor $\mathcal{X}$ as a vector input to match the modeling. To simplify the presentation, we assume that $I_n = I, n = 1, \cdots, N$. As a result, the complexity of the model will reach $\mathcal{O}(KR \cdot I^N)$. Moreover, the vectorization operation destroys the spatial information of the original data and thus degrades the model's performance.

3.2 Interpretable Tensor Polynomial Additive Model

To reduce the number of model parameters and prevent the computational explosion problem after vectorization, a workable strategy is to maintain the tensor's original structure while discovering a new factorization rule for the weight space of feature interactions.

Specifically, we take the tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ directly as the feature input, and try to learn a polynomial as follows:

$$P(\mathcal{X}) = b + \sum_{k=1}^{K} \langle \mathcal{W}^{(k)}, \underbrace{\mathcal{X} \circ \mathcal{X} \ldots \circ \mathcal{X}}_{k \text{ times}} \rangle \qquad (4)$$

where $\mathcal{W}^{(k)} \in \mathbb{R}^{I_1 \times \cdots \times I_N \times \cdots \times I_1 \times \cdots \times I_N}$ is a kN-th order weight tensor that encapsulates how feature tensors interact. We refer to this model as the Tensor Polynomial Additive Model (TPAM). For accuracy, we decompose $\mathcal{W}^{(k)}$ into combinations of lower order factor tensors/vectors in a hierarchical way, resulting in T-TPAM and V-TPAM, respectively. Then we employ patch splitting to capture fine-grained details and further consider training on multiple scales for global dependencies capturing. As illustrated in Fig 2, we present the overall structure of the TPAM.

3.2.1 T-TPAM

First, we decompose $\mathcal{W}^{(k)}$ into a combination of several basis factor tensors of the same size as $\mathcal{X}$ as follows:

$$\mathcal{W}^{(k)} = \sum_{r=1}^{R_k} \lambda_{kr} \cdot \underbrace{(\mathcal{U}_{kr} \circ \cdots \circ \mathcal{U}_{kr})}_{k \text{ times}}, \qquad (5)$$

where $\left\{ \mathcal{U}_{kr} \in \mathbb{R}^{I_1 \times \cdots \times I_N} \right\}_{r=1}^{R_k}$ is the factor tensor of $\mathcal{W}^{(k)}$, $\{\lambda_{kr}\}_{r=1}^{R_k}$ is the corresponding weight of each component. The polynomial model (4) can be expressed as

$$P(\mathcal{X}) = b + \sum_{k=1}^{K} \sum_{r=1}^{R_k} \lambda_{kr} \cdot \langle \mathcal{U}_{kr}, \mathcal{X} \rangle^k. \qquad (6)$$

We refer to the aforementioned model as T-TPAM, which decreases the model complexity of TPAM from $\mathcal{O}(I^{NK})$ to $\mathcal{O}(KR \cdot I^N)$. As it shows in Fig. 2a, T-TPAM first applies a geometric scaling to $\mathcal{X}$, followed by K polynomial modules which perform multiplication and inner product operations over $\mathcal{X}$ and $\mathcal{U}_{kr}$. Then a linear layer is used to combine polynomials with different degrees.

3.2.2 V-TPAM

While the TPAM effectively reduces model complexity, the size of $\mathcal{X}$ and the factor tensors $\mathcal{U}_{kr}$ can be large in practical applications, making the computational effort of T-TPAM unacceptable. Further decomposition of $\mathcal{U}_{kr}$ seems promising. We consider the CP decomposition as

$$\mathcal{U}_{kr} = \sum_{r}^{R} \mathbf{u}_{kr}^{(1)} \circ \ldots \circ \mathbf{u}_{kr}^{(N)}, \qquad (7)$$

T-TPAM can be then transformed into

$$P(\mathcal{X}) = b + \sum_{k=1}^{K} \sum_{r=1}^{R_k} \lambda_{kr} \cdot \langle \mathbf{u}_{kr}^{(1)} \circ \ldots \circ \mathbf{u}_{kr}^{(N)}, \mathcal{X} \rangle^k, \qquad (8)$$
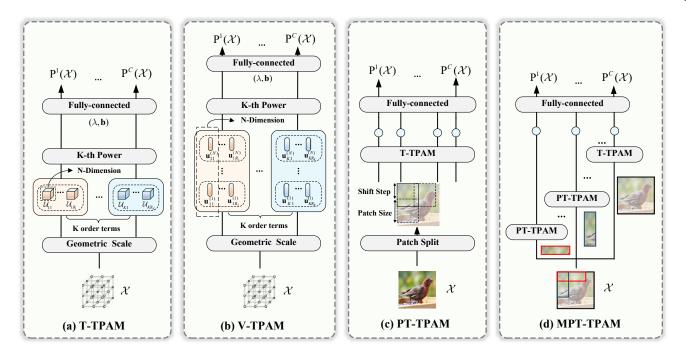
Fig. 2. (a) For a K-order polynomial, the K modules depicted in the figure correspond to different orders in the polynomial. The weight tensor $\mathcal{W}$ is decomposed into a factor tensor $\mathcal{U}$, which has the same size as the input $\mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$. Following inner-product and multiplication operations, $\mathcal{W}$ and $\mathcal{U}$ are then passed into the fully connected layer to obtain the final C classification result. (b) The training process of V-TPAM remains unchanged, except for the further decomposition of $\mathcal{W}$ into factor vectors $\mathbf{u}$. (c) PT-TPAM initially determines the patch size and shift step through patch slicing, and subsequently transfers this information to T-TPAM for further learning. (d) MPT-TPAM undergoes co-training with PT-TPAM, employing various patch sizes and shift steps. Using different patch sizes can further capture process and remote dependencies and increase the performance of the model.

where $\{\{\mathbf{u}_{kr}^{(n)}\}_{r=1}^{R_k}\}_{n=1}^{N}$ and $\{\lambda_{kr}\}_{r=1}^{R_k}$ represent the learnable basic factor vectors and weights of each component for $\mathcal{W}^{(k)}$.

We refer to this variant as V-TPAM, which reduces the model complexity from $\mathcal{O}(KR \cdot I^N)$ to $\mathcal{O}(KR \cdot NI)$. As shown in Fig. 2b, the training process is similar to that of T-TPAM. However, in V-TPAM, the factor tensor $\mathcal{U}_{kr}$ is further decomposed into factor vectors $\{\mathbf{u}_{kr}^{(n)}\}_{n=1}^{N}$, which yields a more structured model with fewer parameters.

3.2.3 PT-TPAM

Weight sharing is a commonly used strategy in neural networks to promote parameter sharing and reduce the overall number of parameters in the model. Weight sharing can reduce the number of parameters in T-TPAM. To reduce model complexity, the T-TPAM is constructed solely for the local space of $\mathcal{X}$, resembling a convolution kernel, rather than modeling the entire input feature space. We denote this model using localized patches as PT-TPAM. The overall block diagram of PT-TPAM is shown in Fig. 2c. Factor tensor $\mathcal{U}$ with a fixed patch size and shift step is employed. During training, the tensor $\mathcal{U}$ is shifted through the entire $\mathcal{X}$ space, capturing local relationships. The linear layer is responsible for generating the final output. PT-TPAM and T-TPAM are equivalent when the patch size matches that of $\mathcal{X}$.

3.2.4 MPT-TPAM

Although it is with decreased model complexity, PT-TPAM has a limited receptive field, similar to that of a convolutional kernel. This limits its ability to effectively capture

global information. To incorporate both local and global information, we propose to add larger patch sizes into the PT-TPAM model for a multi-training approach. The combined method is referred to as MT-TPAM, as depicted in Fig. 2d. In the specific training stage, we utilize a T-TPAM with a patch size identical to $\mathcal{X}$, along with multiple PT-TPAMs of smaller patch sizes for co-training. For instance, in the case of a 32×32 image size, we utilize patches of sizes 4×4, 8×8, and 32×32 during the training.

3.3 Learning and Discussion for TPAM

In this section, the discussion is focused on optimizing the learning process of TPAM models, encompassing aspects such as model initialization, geometric scaling, and approaches to multi-classification challenges. Besides, this section delves into the benefits of TPAM in model compression, highlighting its efficacy and potential advantages.

3.3.1 Initialization scheme

Initialization is very important for neural networks. However, the current neural network regularization techniques [42], [43] are primarily designed for convolutional and fully connected networks. Additionally, the initialization used for polynomial networks mainly considers depth [41]. TPAM stands out as a distinctive model due to its shallow and wide polynomial structure. Consequently, the previous initialization methods are not suitable for TPAM. Based on practical experiments, it has been observed that the magnitude of higher-order weights decreases as the number of terms increases. Additionally, the weights of higher-order terms

in polynomials are significantly smaller in comparison to the weights of lower-order terms. Here we propose a simple initialization scheme

$$\mathcal{U}_{kr}, \mathbf{u}_{knr} \sim \mathcal{N}\left(0, \sigma^2 \mathbf{I}\right), \text{ where } \sigma = Qk\sqrt{\frac{1}{M_k}}, \quad (9)$$

M_k is the number of parameters in the k-th order term, k is the order in which the polynomial is expanded, and Q is the initialization factor. We call this Poly-mean initialization.

3.3.2 Multi-classification Issues

For multiclassification problems with C classes, applying a polynomial model requires learning multiple polynomials simultaneously. It could lead to performance degradation and overfitting, especially when dealing with numerous classes.

To simplify the model, a shared factor tensor $\mathcal{U}_{kr}$ across all classes is introduced, similar to that of SPAM. This means that the same factor tensor is utilized for each class. Additionally, we use class-specific eigenvalues $\{\lambda_{kr}^c\}_{c=1}^{C}$ and bias values $\{b^c\}_{c=1}^{C}$ to learn the unique characteristics of each class. Taking T-TPAM as an example, we investigate the following polynomials for class c as follows:

$$\mathrm{P}^c(\mathcal{X}) = b^c + \sum_{k=1}^{K}\sum_{r=1}^{R_k} \lambda_{kr}^c \cdot \langle \mathcal{U}_{kr}, \mathcal{X}\rangle^k \quad (10)$$

3.3.3 Geometric rescaling

As the order of the interactions increases, training models to learn higher-order interactions becomes challenging. Moreover, the values obtained from such interactions become disproportionately larger than the original feature values. This presents a significant challenge to both the fitting ability of TPAM and its interpretability. Similar to SPAM, to mitigate the disproportionately negative impact of higher-order interactions, we employ a geometric scaling approach to adjust the TPAM model. For K-order interactions with input $\mathcal{X}$, scale scaling is defined as $\tilde{\mathcal{X}}_K = \mathrm{sign}(\mathcal{X}) \cdot |\mathcal{X}|^{1/K}$.

3.3.4 Model Compression

Here we consider the input tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ and a polynomial function with order K. Table 1 illustrates the number of model parameters required by GAM, SPAM, T-SPAM, and V-TPAM. All four models utilize the same linear function. TPAM maintains the multiway structure, whereas, for SPAM, the input tensor must be vectorized.

Table 1 demonstrates that particularly for high-order tensor data, both TPAM and SPAM have considerably fewer parameters compared with GAM. Furthermore, the accumulation of dimensions is significantly smaller than the cumulative multiplication of dimensions, i.e., $I^N >> NI$, which makes V-TPAM require far fewer parameters than both SPAM and T-TPAM and thus highly scalable and suitable for large datasets.

TABLE 1
Complexity of GAM, SPAM, TPAM.

Model	Parameters for input $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$
GAM	I^{NK}
SPAM	$KR \cdot I^N$
T-TPAM	$KR \cdot I^N$
V-TPAM	$KR \cdot NI$

4 MODEL INTERPRETATION

TPAM demonstrates transparency in its internal decision-making processes and exhibits a high level of interpretability. In this section, our focus lies on examining the interpretability of the TPAM model itself, as well as its application as a post-hoc module in the context of CAM.

4.1 Self-Explanation

The additive model boasts a straightforward structure where the contributions of individual features combine linearly to form the final output. The interaction between features and weights determines the importance of each feature in the model. Positive values indicate positive contributions, while negative values have negative effects. For example, for the GAM (Linear) model [10], its importance for the input $\boldsymbol{x} = \{x_1, ..., x_N\}$ can be expressed as $w_i \cdot x_i, i \in \{1, ..., N\}$, w_i is the weight value corresponding to x_i. Extracting the importance of each feature and organizing them in the same spatial location as the input, we can get a corresponding important map.

For a K-th order SPAM [14] model, the feature important vector is

$$\mathrm{IM}_{\mathrm{SPAM}}(\mathbf{x}) = \sum_{k=1}^{K}\sum_{r=1}^{R_k} \lambda_{kr} \cdot (\mathbf{u}_{kr} \otimes \mathbf{x}) \cdot \langle \mathbf{u}_{kr}, \mathbf{x}\rangle^{k-1} \quad (11)$$

where $\mathbf{x} = \mathrm{vec}(\mathcal{X})$, $\mathbf{u}_{kr}$ is the weight factor vector capturing the interactions of $\mathbf{x}$. For TPAM, the feature important tensor, denoted as $\mathrm{IM}_{\mathrm{TPAM}}$, can be defined as

$$\mathrm{IM}_{\mathrm{TPAM}}(\mathcal{X}) = \sum_{k=1}^{K}\sum_{r=1}^{R_k} \lambda_{kr} \cdot (\mathcal{U}_{kr} \otimes \mathcal{X}) \cdot \langle \mathcal{U}_{kr}, \mathcal{X}\rangle^{k-1}$$
$$(12)$$

where interactions of $\mathcal{X}$ are represented by the weight factors $\mathcal{U}_{kr}$.

This important vector/tensor is then visualized to generate intuitive interpretations. The details of this process are illustrated in Fig. 4. In the case of image data, we often sum the important tensor along the channel dimension, resulting in an important matrix commonly referred to as an interpretation map. For instance, the region that TPAM concentrates on is visible in the explanation map produced on MNIST [44], which is essentially the same as human subjective judgment, as shown in Fig. 4. The model performance improvement can also be partially explained by the generated explanation map because it is a direct result of the weights within the model.
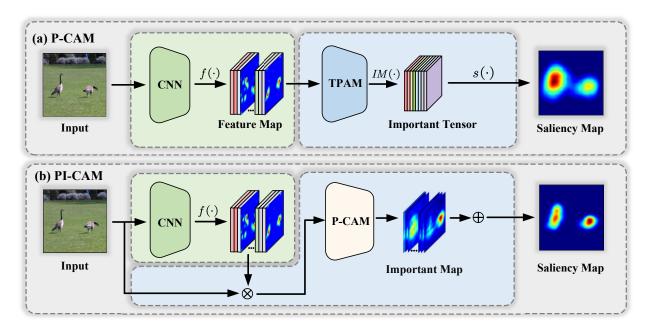
Fig. 3. (a) In P-CAM, the input $\mathcal{X}$ undergoes processing through CNN model $f(\cdot)$, resulting in the acquisition of feature map $\mathcal{A}_l$. Subsequently, the TPAM model classifies the feature map, and the trained TPAM is employed to extract the model's important tensor. The saliency map is then obtained by downscaling the important tensor using the function $s(\cdot)$. (b) PI-CAM acquires the feature map $\mathcal{A}_l$ through the CNN model $f(\cdot)$. Feature map is then utilized as input to P-CAM after performing a dot product with the input $\mathcal{X}$. Subsequently, P-CAM is trained to obtain the important map (the saliency map of each $\mathbf{A}_l^{i_c}$). The saliency map is derived by adding the important maps.
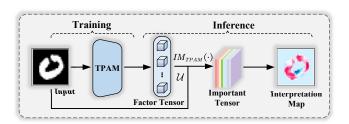

Fig. 4. General process of self-interpretation in TPAM.

4.2 P-CAM & PI-CAM

TPAM offers superior performance and internal decision-making transparency, making it suitable as a classifier instead of a conventional linear layer in various neural networks. For instance, in the case of CNNs, TPAM can be post-positioned on the last feature layer, similar to CAM [30], to extract the importance of that layer and generate reliable saliency maps without compromising performance. Furthermore, CAM and Score-CAM [28] often produce fine-grained maps with subpar accuracy due to the single weight of the feature layers. In contrast, TPAM can weigh the feature maps using weights of the same size, resulting in more accurate and finer-grained saliency maps. To achieve this, we propose the P-CAM and PI-CAM algorithms for CNNs and image inputs.

4.2.1 P-CAM

Given input data $\mathcal{X}$, we use the network action $f(\cdot)$ to interpret the CNN network. We pick an internal convolutional layer l in f and represent the corresponding feature map as

$\mathcal{A}_l = f(\mathcal{X}) \in \mathbb{R}^{I_C \times I_H \times I_W}$. Then the P-CAM saliency map for class c is specified as $L_{\text{P-CAM}}^c$

$$L_{\text{P-CAM}}^c = \text{ReLU}\left(\text{norm}\left(\text{s}\left(\alpha_l^c\right)\right)\right) \tag{13}$$

where

$$\alpha_l^c = \text{IM}^c\left(\mathcal{A}_l\right) \in \mathbb{R}^{I_C \times I_H \times I_W} \tag{14}$$

is the important tensor of $\mathcal{A}_l$, $\text{s}(\cdot) : \mathbb{R}^{I_C \times I_H \times I_W} \rightarrow \mathbb{R}^{I_H \times I_W}$ calculates the total sum over the channel dimension, $\text{norm}(\cdot)$ represents the normalization operation, and $\text{IM}(\cdot)$ illustrates the way to extract important tensor/vector as defined in (11) and (12). In Fig. 3a, the overall flow of P-CAM is depicted. After obtaining the saliency map, the ReLU operation is applied due to its experimental effectiveness.

4.2.2 PI-CAM

Standard CAM methods typically apply the weights from additive models directly onto feature maps. However, this approach overlooks the interactions between feature maps, which can compromise the integrity of the weight extraction process in these models. Score-CAM attempted to rectify this issue by incorporating confidence scores. Yet, its reliance on a singular weight for feature map weighting tends to produce saliency maps that are both unclear and imprecise.

In response, we introduce the PI-CAM algorithm, an enhancement of P-CAM and Score-CAM methodologies. It achieves this by placing the TPAM after the final convolutional layer in the CNN and employing pixel-level weights to weight the feature maps. This method ensures a more accurate and visually fine-grained of the saliency maps.
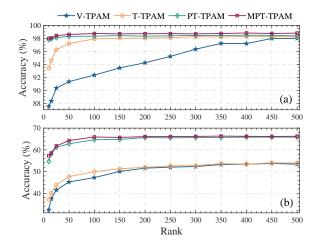
Fig. 5. The effect of different TPAM models on model performance is tested on different datasets for Ranks. (a) MNIST. (b) CIFAR-10

Specifically, for a class c, PI-CAM generates a saliency map as

$$\mathrm{L}^c_{\text{PI-CAM}} = \mathrm{ReLU}\left(\text{norm}\left(\sum_{i_c}^{I_C} s\left(\alpha^c_{i_c=1}\right)\right)\right) \quad (15)$$

where

$$\alpha^c_{i_e} = \mathrm{IM}^c\left(\mathrm{f}\left(\mathcal{X} \otimes \mathcal{H}^{i_c}_l\right)\right) - \mathrm{IM}^c\left(\mathrm{f}\left(\mathcal{X}_b\right)\right) \quad (16)$$

with

$$\mathcal{H}^{i_c}_l = \text{norm}\left(\phi\left(\mathbf{A}^{i_c}_l\right)\right) \quad (17)$$

$\mathbf{A}^{i_c}_l \in \mathbb{R}^{I_H \times I_W}$ is the i_c-th channel of $\mathcal{A}_l$, and $\phi(\cdot) : \mathbb{R}^{I_H \times I_W} \rightarrow \mathbb{R}^{I_C \times I_H \times I_W}$ denotes the operation that up-samples $\mathbf{A}^{i_c}_l$ into the same size of the input. $\mathcal{X}$ represents the image requiring interpretation, while $\mathcal{X}_b$ serves as the baseline image, typically consisting of either all white or all black.

The pipeline of the proposed PI-CAM is illustrated in Fig. 3b. In PI-CAM, images undergo CNN processing to generate feature maps. These feature maps from various channels are then element-wise multiplied with the input image. The multiplication outcomes, along with the input image, act as inputs to the P-CAM module. P-CAM produces important maps for each channel, and these maps are summed to produce the final saliency map.

5 EXPERIMENTS

In the forthcoming sections, we conduct a thorough evaluation of TPAM's performance, focusing on its superior outcomes and notable interpretability, through various experiments. Initially, the section 5.1 examines the impact of parameter settings and model initialization on TPAM's accuracy. We then compare TPAM with various established, interpretable baseline models to showcase its enhanced performance across datasets of different sizes. Following this, In section 5.2 TPAM is compared with several other post hoc interpretable models. This comparison aims to confirm TPAM's self-interpretation abilities and to investigate the reasons for its exceptional classification task performance.
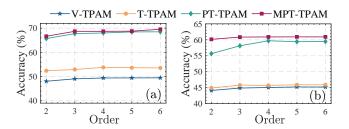

Fig. 6. The effect of different TPAM models on model performance is tested on different datasets for Orders. (a) CIFAR-10. (b) STL-10.

The paper also assesses the effectiveness of TPAM-derived PI-CAM and P-CAM algorithms, further supporting our conclusions.

5.1 Performance Evaluation of TPAM Classification

In section 5.1.1, we assess the performance of TPAM by conducting an ablation study on its hyperparameters. Then we compare the proposed TPAM with existing interpretable models on several real datasets for image classification in section 5.1.2. As model inputs, we use MNIST [45], CIFAR-10 [46], and STL-10 standard datasets and crop them to $28 \times 28 \times 1$, $32 \times 32 \times 3$, and $96 \times 96 \times 3$ sizes without using data enhancement operations. For all datasets with no defined train-val-test split, we use a fixed random sample of 70% of the data for training, 20% for validation, and 10% for testing. Each model undergoes 10 training iterations on Pytorch using the Adam [47] optimization algorithm, with only the optimal results being reported.

5.1.1 Ablation Studies
Building upon the foundational experiments outlined previously, this section delves into a detailed analysis of TPAM's hyperparameters. Specifically, we scrutinize the influence of factors such as the order and rank, initialization scheme, patch size, and shift step on the classification performance of TPAM.

Order and Rank. Earlier research on polynomial approximation has shown that both the order of polynomial expansion and the required rank for weight space reconstruction significantly influence TPAM's performance. To delve deeper into this relationship, we conducted ablation studies on the MNIST, CIFAR-10, and STL-10 datasets, specifically focusing on variations in adopted rank and order. Initially, we set the order to 2 and varied the rank from 50 to 500. Results for MNIST and CIFAR-10 are illustrated in Fig. 5(a) and Fig. 5(b), respectively. Subsequently, the rank was fixed at 200, and the order varied within the range of $\{2, 3, 4, 5, 6\}$. Results for MNIST and STL-10 are depicted in Fig. 6(a) and Fig. 6(b), respectively.

As expected, TPAM's performance improves with increases in both rank and order, where PT-TPAM and MPT-TPAM consistently outperform others. Specifically, with a constant order, as shown in Fig. 6a, all four TPAM models show continual performance gains with increasing rank, although the rate of improvement slows over time. Notably, V-TPAM demonstrates rapid improvement on the MNIST dataset, indicating a need for a higher rank in accurate weight space reconstruction. Furthermore, with a fixed rank,

TABLE 2
Evaluation of TPAM on benchmarks against prior work. (↑): the higher the better; (↓): the lower the better. **Bold** indicates the best performance and the least parameters (excluding uninterpretable baselines). The best results were reported with optimal hyperparameters and 10 randomized trials.

Method	2D								3D			
	ArticularyWordRecognition		Heartbeat		MNIST		Fashion-MNIST		CIFAR-10		STL-10	
	Acc↑	Param↓	Acc↑	Param↓	Acc↑	Param↓	Acc↑	Param↓	Acc↑	Param↓	Acc↑	Param↓
Uninterpretable Black-Box Baselines												
DNN	97.67	5.30	72.68	53.23	98.02	1.60	90.28	2.67	59.62	4.21	35.81	62.50
SOTA	99.00	7.22	76.60	6.97	99.87	1.51	94.91	24.73	99.10	23.00	97.20	5.10
Interpretable Baselines												
NAM	96.21	0.06	69.74	1.13	95.72	7.28	89.48	6.73	41.79	27.90	—	251.54
GAM (Linear)	96.00	**0.03**	66.83	**0.05**	92.61	**0.01**	84.79	**0.01**	36.40	0.03	30.68	0.27
GA2M (Linear)	96.35	36.33	—	1220.69	97.35	6.15	85.72	6.15	47.83	94.40	—	7644.39
SPAM (Order 1)	96.00	0.26	70.73	4.95	92.01	0.40	84.12	0.32	40.20	1.54	30.90	11.06
SPAM(Order 2)	95.61	0.28	72.20	4.95	96.09	0.40	85.22	0.32	45.27	1.54	32.04	11.06
Our Interpretable Models												
V-TPAM (Order 1)	**96.67**	0.04	73.20	0.09	96.73	0.03	84.85	0.03	42.54	**0.03**	42.32	**0.10**
V-TPAM (Order 2)	95.42	0.04	**74.15**	0.09	98.03	0.03	90.68	0.03	52.03	0.03	46.06	0.10
T-TPAM (Order 2)	96.31	0.58	74.01	19.76	98.31	0.36	90.27	0.36	53.95	1.54	46.47	13.83
PT-TPAM (Order 2)	—				98.46	0.85	91.09	0.85	65.84	0.94	59.97	2.59
MPT-TPAM (Order 2)					**98.73**	1.25	**91.24**	1.25	**66.24**	2.48	**60.55**	16.42

TABLE 3
Performance of TPAM with different initializations.

Model	Initialization	Order 2	Order 4
T-TPAM	Xavier	53.54	53.87
	Orthogonal	53.59	54.10
	Kaiming normal	52.97	54.04
	Kaiming uniform	53.28	54.10
	Poly-mean	**53.65**	**54.59**
PT-TPAM	Xavier	64.25	68.58
	Orthogonal	64.21	68.09
	Kaiming normal	63.69	66.75
	Kaiming uniform	63.89	66.43
	Poly-mean	**65.36**	**68.95**

Fig. 7. The impact of patch size and shift step size in PT-TPAM on the classification performance of (a) CIFAR-10 and (b) STL-10 datasets.

elevating the order results in enhanced model performance, albeit marginally. Therefore, lower orders suffice to achieve satisfactory classification results.

Model Initialization. The initialization of a model plays a crucial role in determining its performance. In our study, we employ widely recognized initialization schemes, including xavier initialization [42], orthogonal initialization [48], and Kaiming initialization [43], and our proposed initialization scheme. To evaluate their performance, we applied these techniques as the initialization module of both T-TPAM and MPT-TPAM models and conducted experiments on the MNIST dataset. The rank is set as 200, order is chosen from $\{2, 4\}$. The experimental results are exhibited in Table 3. As anticipated, previously proposed initialization schemes are unable to align with the structure of TPAM, consequently resulting in suboptimal training outcomes. In contrast, the multi-means initialization scheme demonstrates remarkable efficacy for TPAM.

Patch Size and Shift Step. As depicted in Fig. 5, P-TPAM exhibits exceptional performance. However, PT-TPAM is significantly influenced by the shift step and the patch size. To delve deeper into this correlation, we conducted experiments on both the CIFAR-10 and STL-10 datasets with PT-TPAM while maintaining a constant rank and order of 200 and 2, respectively. The results are shown in Fig. 7. On both datasets, PT-TPAM performs well with a patch size of 8 and a shift step size of 2. In addition, it can be seen
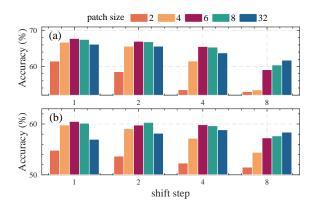
using excessively small patch size or large shift step size on real datasets can lead to incomplete feature acquisition, thus causing overfitting and excessively large models. Therefore, it is advisable to select a moderate patch size and shift step size for PT-TPAM for training.

5.1.2 Measuring Benchmark Performance

In this study, we evaluate TPAM in comparison to a set of benchmark algorithms for image classification. The baseline comprises black-box models, including DNNs and CNNs. For DNNs, we set the hidden layers to $[512, 2048]$. As for CNNs, we select Resnet18, which shares the same input pattern as TPAM. Furthermore, among the interpretable additive models, we choose NAM [13], SPAM [14], GAM [10], and GA^{2}M [12] as baselines. V-TPAM and SPAM are set to first and second-order polynomials, while T-TPAM, MPT-TPAM, and PT-TPAM are set to second-order. The rank for both TPAM and SPAM is chosen from $[1, 200]$. All self-interpretation models utilize linear functions. To evaluate the performance of TPAM on any dimensional datasets, we conducted experiments on both 2D and 3D datasets. For the 2D task, we use datasets a and b for time series classification task and MNIST and Fashion datasets for 2D

image classification. The CIFAR-10 and STL-10 datasets are utilized for 3D image classification tasks.

Table 2 summarizes the results we have obtained. Some conclusions can be drawn from the results. We have found that TPAM performs better than previous interpretable models on all datasets, with second-order TPAM performing significantly better than the baseline of all interpretable models. Next, the number of parameters in V-TPAM and TPAM is drastically reduced, resulting in model compression. It appears that the use of various patches during the training of PT-TPAM and MPT-TPAM has resulted in improved performance. However, this has also led to an increase in parameters on MPT-TPAM. For large datasets, TPAM can perform the same expansion as SPAM, while NAM and GA^{2}M cannot fit higher-order interactions due to increased computational effort. Compared to the uninterpretable baseline, TPAM demonstrates superior performance and parameter reduction compared to DNNs. However, TPAM still exhibits a significant disparity in classification accuracy when compared to CNNs.

5.2 Interpretation Evaluation

TPAM exhibits exceptional interpretability, which is another significant advantage. We proceed to assess the self-interpretability of TPAM and the viability of P-CAM and P-CAM methods derived from it.

5.2.1 Self-Explanatory Performance Evaluation

We extract the feature importance of each pixel after the input passes through the model to create an interpretation map. In this section, we introduce a series of established metrics used to evaluate the interpretability of models. Then, we evaluate the explanations produced by both the TPAM model and the post-hoc explanatory model. Besides, our evaluation highlights the exceptional explanatory performance of TPAM and analysis delves into the factors contributing to the enhanced performance of TPAM, providing comprehensive explanations.

Evaluation Metrics. Here are four commonly used evaluation metrics:

Average Drop (AD) [19] quantifies the impact on the predicted class probability when the input image is masked using an interpretation map. Let Y_n^c and O_n^c denote the predicted probability of class c when the n-th test image and its corresponding masked image are utilized as inputs to the model. Then, AD is expressed as $\text{AD}(\%) := \frac{1}{N} \sum_{n=1}^{N} \frac{\max(0, Y_n^c - O_n^c)}{Y_n^c} \times 100$. Here, N represents the total number of test images. AD calculates the degree of predictive power reduction solely caused by masking the images, where the predictive probability of the model decreases as important pixels are removed. Lower scores indicate superior results.

Average Increase (AI) [19] quantifies the percentage of images in which the masked image yields a higher class probability compared to the original image, where the predictive probability of the model gradually increases as the number of significant pixels increases. AI is expressed as $\text{AI}(\%) := \frac{1}{N} \sum_{n=1}^{N} \frac{\text{sign}(Y_n^c < O_n^c)}{N} \times 100$. For O_n^c, we add the top 25% of significant original images to the baseline image
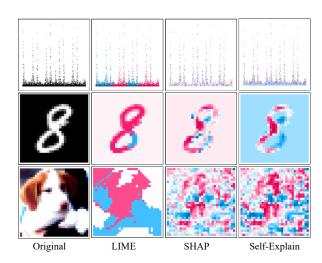

Fig. 8. Interpretation maps generated by the TPAM self-interpretation and post-hoc interpretation methods (LIME and SHAP) are created for Hearbeat, MNIST and CIFAR-10 datasets. Red represents positive contributions while blue represents negative contributions. The first row represents the model's decision against the healthy people. The second row represents the decision against the digit 8. The third row focuses on the decision against dogs.

by interpretation map. Higher values indicate preferable outcomes.

Deletion (Del) [6] and Insertion (Ins) [6] are used to analyze the impact of individual pixels on image classification. Del involves progressively removing pixels based on their importance and observing the effect on the class prediction probability. The method generates a probability curve, where the area under this curve (Del) becomes smaller as more significant pixels are removed, indicating a good explanation if there's a steep decrease. Conversely, Ins starts with a baseline image and adds the most significant pixels incrementally. This generates a probability curve where a sharp increase and a larger area under the curve indicate a more effective explanation, demonstrating the importance of the added pixels.

Interpretable Performance of TPAM. TPAM exhibits strong interpretability, particularly in image classification tasks and making model decisions. It calculates the contribution value of each pixel, which is then directly summed to generate the result. With the equation (11), an interpretation map of the same size as the image is created without the need for an additional interpretation model. In this section, we evaluate the interpretation maps generated by TPAM using its interpretations. For comparison, we employ additional post-hoc explanatory models, including LIME [7] and SHAP [5], to extract decisions within the TAPM without utilizing self-interpretation. Our assessment metrics for evaluating the reliability of the interpreted maps include AD, AI, Ins, and Del. To present the interpretation map graphs and report the assessment results, we randomly selected 2000 images from the MNIST and F-MNIST datasets, respectively. The results are depicted in Table 4.

Fig. 8 presents the interpretation maps generated by LIME, SHAP, and TPAM. From Table 4, it can be observed that TPAM-produced interpretation maps outperform those generated by SHAP in all evaluation metrics, while LIME achieves worse performance. This outcome is anticipated

TABLE 4
A comparison between TPAM model self-interpretation and other post-hoc explanation methods.

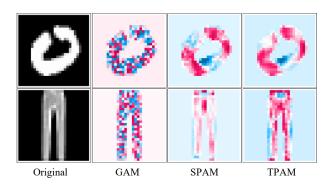| Methods	MNIST				Fashion-MNIST			
	AD↓	AI↑	Ins↑	Del↓	AD↓	AI↑	Ins↑	Del↓
LIME	9.97	23.04	0.78	0.15	48.13	10.90	0.54	0.35
SHAP	1.02	64.73	0.91	0.07	21.75	34.26	0.73	0.14
Self-Explain	**0.84**	**67.95**	**0.92**	**0.06**	**21.29**	**37.30**	**0.75**	**0.14**

Fig. 9. Interpretation maps generated by TPAM, GAM, and SPAM for MNIST and F-MNIST datasets.

because TPAM generates interpretation maps through self-interpretation, wherein each pixel's weight directly influences the outcome without any errors. On the other hand, SHAP employs gradient-based weight determination, which is prone to errors and may result in vanishing gradients. Furthermore, LIME relies on function approximation to obtain weight values, leading to significant approximation errors and unstable interpretation map results.

Interpretation Performance Evaluation of Additive Models. In study 5.1.1, we find that TPAM significantly outperforms the other baseline additive models. Here we leverage the interpretability of additive models to generate interpretation maps for TPAM, SPAM, and GAM models to ensure an understanding of the improvements achieved.

In the results depicted in Fig. 9, the interpretation maps generated by TPAM and GAM exhibit smoother, whereas SPAM displays a significant amount of noise. This observation is further supported by the findings presented in Table 6, where the generated interpretation maps were normalized and their variances were measured. As expected, TPAM, with its ability to capture feature interactions in spatial locations, showcases lower variance and generates superior interpretation maps. In contrast, SPAM struggles to account for the spatial relationships between features due to its reliance on vector inputs.

5.2.2 Performance of P-CAM and PI-CAM

We evaluated the accuracy of P-CAM and PI-CAM in generating interpretation maps for an image classification task and assessed the effectiveness of the resulting saliency maps for image localization. The experiments utilize the VGG-16 model [36] and Inception-V3 model [49], modified as described in [30], to generate feature maps. Two publicly accessible object classification datasets, Mini-Imagenet [50] and CUB-200 [51], are used in our experiments. Prior to inputting the images into the VGG-16 network, we resize

them to $224 \times 224 \times 3$. Similarly, for Inception-V3, the images are resized to $299 \times 299 \times 3$. The images are normalized using the mean vector $[0.485, 0.456, 0.406]$ and the standard deviation vector $[0.229, 0.224, 0.225]$ to ensure they fall within the range of [0, 1]. Additionally, we set the initial input $\mathcal{X}_b$ for PI-CAM as an all-one tensor. Finally, we compare the saliency maps generated by Grad-CAM [17], Grad-CAM++ [19], XGrad-CAM [31], SGrad-CAM++ [18], Layer-CAM [32], and Score-CAM [28].

Faithfulness Evaluation. We assess the confidence of PI-CAM and P-CAM by conducting quantitative testing on 2000 images from each of the two datasets. We measured the performance using the AD, AI, Ins, and Del metrics.

The evaluation results of PI-CAM and P-CAM, along with other CAM methods, are presented in Table 5. The results demonstrate that PI-CAM and P-CAM exhibit superior performance on multiple metrics for both network structures and datasets. P-CAM performs well on the AD and AI metrics, which can be attributed to its emphasis on highlighting significant regions in the saliency maps through weight allocation. In contrast, PI-CAM outperforms other CAM methods on the Ins and Del metrics. This advantage can be attributed to PI-CAM's ability to generate saliency maps with reduced noise and enhanced focus on important network regions.

Notably, the interference of noise affects the evaluation of saliency maps by the network, rendering existing CAM methods sensitive to noise and unable to handle feature maps containing noise. Additionally, the performance of the remaining CAM methods is comparably close, with a noticeable gap separating their performance from that of P-CAM and PI-CAM in smaller interpretation task classes. Fig. 10 presents sample saliency maps generated by PI-CAM, P-CAM, and other CAM. Our method stands out by producing clearer and less noisy saliency maps, displaying more concentrated labeled regions.

Localization Evaluation. We conduct an evaluation of the localization capability, which is another important metric used to assess the performance of saliency maps. Instead of the conventional approach of locating the maximum value of the saliency map, we employ the Proportion to measure the energy of the saliency map. This metric evaluates the saliency map's ability to accurately identify the number of salient regions within the bounding box, thereby examining the distribution of noise in the saliency map. To perform quantitative analysis, we select 2000 images from the CUB-200 dataset. The results of this analysis are presented in Table 7.

Table 7 shows that PI-CAM concentrates over 80% of its energy within the true bounding box in the saliency maps, while P-CAM and other CAM methods allocate approximately 70% of the energy within the true bounding box. This observation suggests that the saliency map generated by PI-CAM is cleaner and exhibits a higher concentration of energy.

6 CONCLUSION

In this paper, we propose TPAM, an innovative interpretable polynomial model utilizing higher-order tensor inputs, bypassing potential performance losses associated with tra-

TABLE 5
Comparison of P-CAM and PI-CAM with other CAM for generating saliency maps. P-CAM and PI-CAM are close to or exceed existing methods in AD, AI, Ins, and Del metrics.

| Methods | VGG-16 | | | | | | | | Inception-v3 | | | | | | | |
| | Mini-ImageNet | | | | CUB-200 | | | | Mini-ImageNet | | | | CUB-200 | | | |
	AD↓	AI↑	Ins↑	Del↓	AD↓	AI↑	Ins↑	Del↓	AD↑↓	AI↑	Ins↑	Del↓	AD↓	AI↑	Ins↑	Del↓
Grad-CAM	44.57	10.93	68.44	13.32	20.63	**32.59**	68.79	3.77	48.03	8.77	72.07	17.28	30.50	24.97	64.22	3.50
Grad-CAM++	43.92	**11.53**	68.57	14.04	21.24	30.67	67.84	3.46	51.37	8.00	70.98	17.63	31.89	23.16	62.93	3.78
XGrad-CAM	45.32	11.00	68.92	13.16	20.31	31.44	68.03	3.47	46.87	**9.10**	**72.82**	17.22	**29.90**	24.92	64.53	3.50
SGrad-CAM++	58.75	6.87	60.46	21.48	27.00	24.86	66.02	4.63	54.73	5.97	68.48	21.24	32.56	21.99	63.60	4.01
Layer-CAM	48.01	9.97	66.74	14.00	21.44	29.73	68.27	3.45	51.33	7.17	70.90	17.97	30.50	23.76	64.02	3.57
Score-CAM	48.79	9.63	66.07	15.61	21.49	29.73	66.44	3.68	50.87	8.00	69.34	19.30	30.49	23.79	63.16	3.84
P-CAM	**43.86**	11.57	69.20	13.52	**19.45**	31.31	68.49	3.44	48.91	7.57	72.11	17.40	31.33	**25.39**	64.50	**3.40**
PI-CAM	49.41	10.83	**70.11**	**12.82**	21.62	29.30	**69.15**	**3.32**	**46.17**	8.45	72.68	**17.11**	30.20	24.42	**64.69**	3.41

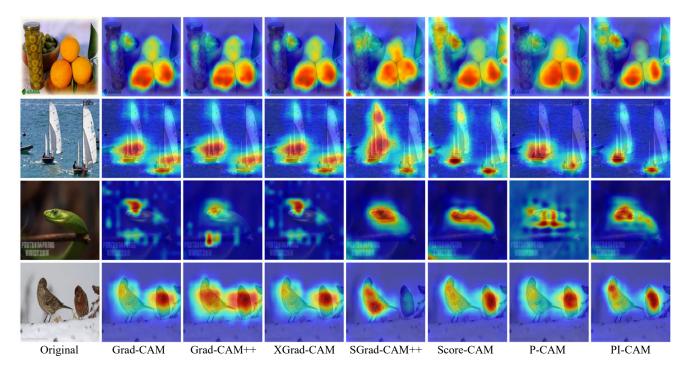| Original | Grad-CAM | Grad-CAM++ | XGrad-CAM | SGrad-CAM++ | Score-CAM | P-CAM | PI-CAM |

Fig. 10. Comparison of saliency maps generated by Grad-CAM, Grad-CAM++, XGrad-CAM, SGrad-CAM++, Layer-CAM, Score-CAM, P-CAM, and P-CAM on the VGG-16 model in Mini-Imagenet dataset.

TABLE 6
Variance of interpretation maps produced by TPAM, GAM, and SPAM on MNIST and F-MNIST determines their smoothness and focus. Smaller variances indicate smoother and more focused interpretation maps.

| | Variance | |
Model	MNIST	F-MNIST
GAM	0.00907	0.00927
SPAM	0.02206	0.01097
TPAM	**0.00891**	**0.00803**

TABLE 7
Comparison of P-CAM and PI-CAM with other CAM on the localization task. Higher proportions are associated with better localization performance, and PI-CAM far outperforms existing methods on the proportions metric.

| Method | Proportion | |
	VGG-16	Inception-V3
Grad-CAM	71.247	72.304
Grad-CAM++	70.623	72.648
XGrad-CAM	71.281	72.491
SGrad-CAM++	71.907	70.848
Layer-CAM	72.067	75.122
Score-CAM	71.380	73.375
P-CAM	70.494	72.305
PI-CAM	**81.817**	**78.195**

ditional vectorization. To effectively learn interactions between higher-order features, we adopt a hierarchical low-order symmetric tensor approximation in weight space, reducing computational complexity. Compared to existing additive models, TPAM offers considerable advantages in parameter efficiency and accuracy performance. Additionally, its decision-making process is transparent, enabling the extraction of each feature's importance for credible

interpretations. Moreover, we apply TPAM to the classification header in CNNs and accordingly propose two new algorithms for CAM generation. These algorithms directly

process feature maps, applying a factorization weighting tensor matching the feature maps' size to create more comprehensive saliency maps.

REFERENCES

[1] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–42, 2018.

[2] F. Bodria, F. Giannotti, R. Guidotti, F. Naretto, D. Pedreschi, and S. Rinzivillo, "Benchmarking and survey of explanation methods for black box models," *Data Mining and Knowledge Discovery*, pp. 1–60, 2023.

[3] C.-H. Chang, R. Caruana, and A. Goldenberg, "Node-gam: Neural generalized additive model for interpretable deep learning," *arXiv preprint arXiv:2106.01613*, 2021.

[4] X. Li, H. Xiong, X. Li, X. Wu, X. Zhang, J. Liu, J. Bian, and D. Dou, "Interpretable deep learning: Interpretation, interpretability, trustworthiness, and beyond," *Knowledge and Information Systems*, vol. 64, no. 12, pp. 3197–3234, 2022.

[5] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," *Advances in neural information processing systems*, vol. 30, 2017.

[6] V. Petsiuk, A. Das, and K. Saenko, "Rise: Randomized input sampling for explanation of black-box models," *arXiv preprint arXiv:1806.07421*, 2018.

[7] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[8] I. Puri, A. Dhurandhar, T. Pedapati, K. Shanmugam, D. Wei, and K. R. Varshney, "Cofrnets: interpretable neural architecture inspired by continued fractions," *Advances in neural information processing systems*, vol. 34, pp. 21 668–21 680, 2021.

[9] T. Fel, M. Ducoffe, D. Vigouroux, R. Cadène, M. Capelle, C. Nicodème, and T. Serre, "Don't lie to me! robust and efficient explainability with verified perturbation analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 153–16 163.

[10] T. J. Hastie, "Generalized additive models," in *Statistical models in S*. Routledge, 2017, pp. 249–307.

[11] H. Zhuang, X. Wang, M. Bendersky, A. Grushetsky, Y. Wu, P. Mitrichev, E. Sterling, N. Bell, W. Ravina, and H. Qian, "Interpretable learning-to-rank with generalized additive models," *arXiv preprint arXiv:2005.02553*, 2020.

[12] Y. Lou, R. Caruana, J. Gehrke, and G. Hooker, "Accurate intelligible models with pairwise interactions," in *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013, pp. 623–631.

[13] R. Agarwal, L. Melnick, N. Frosst, X. Zhang, B. Lengerich, R. Caruana, and G. E. Hinton, "Neural additive models: Interpretable machine learning with neural nets," *Advances in neural information processing systems*, vol. 34, pp. 4699–4711, 2021.

[14] A. Dubey, F. Radenovic, and D. Mahajan, "Scalable interpretability via polynomials," *Advances in neural information processing systems*, vol. 35, pp. 36 748–36 761, 2022.

[15] J. Nie, "Generating polynomials and symmetric tensor decompositions," *Foundations of Computational Mathematics*, vol. 17, pp. 423–465, 2017.

[16] J. Brachat, P. Comon, B. Mourrain, and E. Tsigaridas, "Symmetric tensor decomposition," *Linear Algebra and its Applications*, vol. 433, no. 11-12, pp. 1851–1872, 2010.

[17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[18] D. Omeiza, S. Speakman, C. Cintas, and K. Weldermariam, "Smooth grad-cam++: An enhanced inference level visualization technique for deep convolutional neural network models," *arXiv preprint arXiv:1908.01224*, 2019.

[19] A. Chattopadhay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks," in *2018 IEEE winter conference on applications of computer vision (WACV)*. IEEE, 2018, pp. 839–847.

[20] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer, 2014, pp. 818–833.

[21] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[22] H. G. Ramaswamy *et al.*, "Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization," in *proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2020, pp. 983–991.

[23] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," *Advances in neural information processing systems*, vol. 30, 2017.

[24] H. Wang, R. Naidu, J. Michael, and S. S. Kundu, "Ss-cam: Smoothed score-cam for sharper visual feature localization," *arXiv preprint arXiv:2006.14255*, 2020.

[25] R. Fong, M. Patrick, and A. Vedaldi, "Understanding deep networks via extremal perturbations and smooth masks," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2950–2958.

[26] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3429–3437.

[27] K. Schulz, L. Sixt, F. Tombari, and T. Landgraf, "Restricting the flow: Information bottlenecks for attribution," *arXiv preprint arXiv:2001.00396*, 2020.

[28] H. Wang, Z. Wang, M. Du, F. Yang, Z. Zhang, S. Ding, P. Mardziel, and X. Hu, "Score-cam: Score-weighted visual explanations for convolutional neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 24–25.

[29] M. B. Muhammad and M. Yeasin, "Eigen-cam: Class activation map using principal components," in *2020 international joint conference on neural networks (IJCNN)*. IEEE, 2020, pp. 1–7.

[30] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2921–2929.

[31] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, and B. Li, "Axiom-based grad-cam: Towards accurate visualization and explanation of cnns," *arXiv preprint arXiv:2008.02312*, 2020.

[32] P.-T. Jiang, C.-B. Zhang, Q. Hou, M.-M. Cheng, and Y. Wei, "Layercam: Exploring hierarchical class activation maps for localization," *IEEE Transactions on Image Processing*, vol. 30, pp. 5875–5888, 2021.

[33] R. Naidu, A. Ghosh, Y. Maurya, S. S. Kundu *et al.*, "Is-cam: Integrated score-cam for axiomatic-based explanations," *arXiv preprint arXiv:2010.03023*, 2020.

[34] Y. Shin and J. Ghosh, "The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation," in *IJCNN-91-Seattle international joint conference on neural networks*, vol. 1. IEEE, 1991, pp. 13–18.

[35] C.-K. Li, "A sigma-pi-sigma neural network (spsnn)," *Neural Processing Letters*, vol. 17, pp. 1–19, 2003.

[36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[37] C. Voutriaridis, Y. S. Boutalis, and B. G. Mertzios, "Ridge polynomial networks in pattern recognition," in *Proceedings EC-VIP-MC 2003. 4th EURASIP Conference focused on Video/Image Processing and Multimedia Communications (IEEE Cat. No. 03EX667)*, vol. 2. IEEE, 2003, pp. 519–524.

[38] S.-K. Oh, W. Pedrycz, and B.-J. Park, "Polynomial neural networks architecture: analysis and design," *Computers & Electrical Engineering*, vol. 29, no. 6, pp. 703–725, 2003.

[39] G. G. Chrysos, S. Moschoglou, G. Bouritsas, Y. Panagakis, J. Deng, and S. Zafeiriou, "P-nets: Deep polynomial neural networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 7325–7335.

[40] G. G. Chrysos, S. Moschoglou, G. Bouritsas, J. Deng, Y. Panagakis, and S. Zafeiriou, "Deep polynomial neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 8, pp. 4021–4034, 2021.

[41] G. G. Chrysos, B. Wang, J. Deng, and V. Cevher, "Regularization of polynomial networks for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 16 123–16 132.

[42] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[43] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[45] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[46] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[48] J. Pennington, S. Schoenholz, and S. Ganguli, "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice," *Advances in neural information processing systems*, vol. 30, 2017.

[49] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[50] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[51] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.