scientific reports

Check for updates

OPEN

Explainable machine learning by SEE-Net: closing the gap between interpretable models and DNNs

Beomseok Seo[1]✉ & Jia Li[2]

Deep Neural Networks (DNNs) have achieved remarkable accuracy for numerous applications, yet their complexity often renders the explanation of predictions a challenging task. This complexity contrasts with easily interpretable statistical models, which, however, often suffer from lower accuracy. Our work suggests that this underperformance may stem more from inadequate training methods than from the inherent limitations of model structures. We hereby introduce the Synced Explanation-Enhanced Neural Network (SEE-Net), a novel architecture integrating a guiding DNN with a shallow neural network, functionally equivalent to a two-layer mixture of linear models. This shallow network is trained under the guidance of the DNN, effectively bridging the gap between the prediction power of deep learning and the need for explainable models. Experiments on image and tabular data demonstrate that SEE-Net can leverage the advantage of DNNs while providing an interpretable prediction framework. Critically, SEE-Net embodies a new paradigm in machine learning: it achieves high-level explainability with minimal compromise on prediction accuracy by training an almost "white-box" model under the co-supervision of a "black-box" model, which can be tailored for diverse applications.

Keywords Explainable neural networks, XAI, Generative mixture of linear models

Research interest in explainable artificial intelligence (XAI) has increased rapidly in recent years. The vast majority of explainable machine learning methods focus on explaining deep neural networks (DNN)[1]. There are two apparent reasons for the observed emphasis on DNN. First, being phenomenally popular, DNN often provides state-of-the-art accuracy for highly complex data, e.g., images[2,3] and biomedical data[4]. Secondly, DNN models are usually too complex to explain. Because of the great difficulty to understand the prediction, researchers and practitioners have not embraced neural network models in various medical, business, and industry areas[5]. One specific concern is that it is hard to foresee when the model will work. The prediction can fail surprisingly for seemingly easy samples; and when this happens, it is impossible to pinpoint the problem due to the model complexity.

Many surveys on XAI have emerged in the past few years[1,6–9]. The term "explanation" is inevitably subjective, and it seems that the machine learning (ML) community is converging to a definition[1]. Roughly speaking, a user would consider a ML system explainable if its prediction decision can be easily understood, either in a general sense, e.g., the decision being described by relatively simple mathematical formula, or for a particular purpose, e.g., the identification of most relevant input features. There is no consensus on whether the words "explainability" and "interpretability" should be distinguished. Following the suggestion in the survey[1], we use the two terms interchangeably.

What kind of explanation is needed depends strongly on the application, which in turn influences the type of methods developed to gain explainability and the measures used to evaluate the explanation. Nauta et al. [1] have proposed the categorization of explainable ML methods into three types: post-hoc methods for explaining already trained black-box models, models with built-in interpretability, and supervised explanation training. Our work here belongs to the second type, under which there is a wide spectrum of approaches and models. We illustrate the spectrum in Fig. 1a. At one extreme of the spectrum, we have white-box models, also referred to as self-explaining models[1], which require no accompanying method to explain; while at the other extreme, we have black-box models such as DNN, for which post-hoc explanation methods have been developed[10–15]. Starting from the black-box end of the spectrum, much effort has been devoted to enhancing the explainability of DNN from various aspects. For example, attention methods have been developed for text data[16,17], and causal discovery has been used to design DNNs that can readily evaluate the importance of variables[18]. Moreover,

[1]Department of Statistics, Sookmyung Women's University, Seoul 04310, Korea. [2]Department of Statistics, The Pennsylvania State University, University Park, PA 16802, USA. ✉email: bsseo@sookmyung.ac.kr

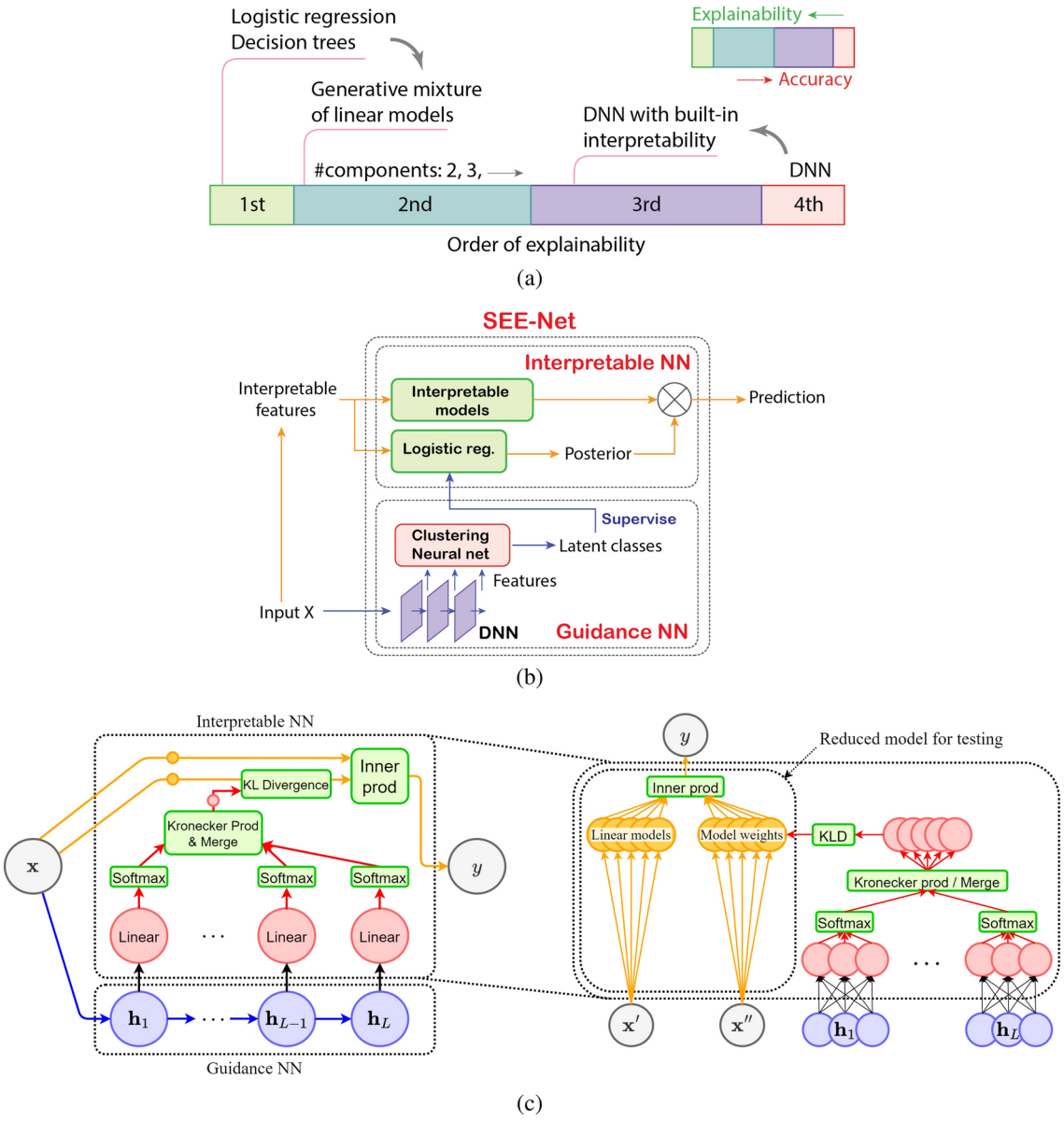


Fig. 1. (**a**) The spectrum of explainability; (**b**) The high-level design of SEE-Net; (**c**) The architecture of SEE-Net.

penalization methods have been developed to reduce network complexity and improve interpretation by finding interaction effects[19] or applying interpretable structures[20–22].

We divide the spectrum of explainability in Fig. 1a into four orders, with the first and the fourth order corresponding, respectively, to fully explainable models, e.g., logistic regression and decision trees, and hard-to-explainable models such DNN and random forest (RF). We put methods that enhance the interpretability of DNN into the third order, which dominates the existing work. According to[1], 75% of the over 300 reviewed papers are about methods for explaining DNN. Although many aspects of explainability have been addressed by the third-order methods, it is still substantially harder to understand how or why a prediction is made compared with the first-order methods, which can offer globally coherent and simple explanation. The second-order explainability, with no clear-cut separation from the first-order, is much less explored. Methods with second-order explainability should be self-explaining or at least nearly so although the explanation in general requires more steps than classic models such as linear models. It is also desirable that such methods allow us to control the complexity of explanation. The primary characteristic that distinguishes second-order explainability from third-order explainability is whether the model is directly explainable or requires auxiliary tools, such as local proxy models, for explanation. Following conventional terminology, we also call first-order methods white-box methods. For distinction, we call second-order methods nearly-white-box methods although whether a model is a "white-box" is subjective and context-dependent. The explainability of second-order models spans over a range such that relatively simple models in this category may be well justified as "white-box" models.

Second-order methods are interesting because they expand our choices where explainability is paramount. Presently, our options often lean towards traditional white-box models, which, despite their transparency, tend to lag in accuracy. Because second-order methods are self-explanatory in nature, there is no need for elaborate or expensive techniques to assess the quality of the explanation. In alignment with the taxonomy proposed by[1], these methods enable the direct adoption of white-box model explanations. Furthermore, conventional metrics of model complexity can effectively evaluate the succinctness of these explanations.

We aim at developing a second-order method that bridges the current gap in the spectrum of explainability. As will be shown in later sections, the final prediction model developed through our method is inherently explainable. Although a DNN is incorporated within our proposed framework, its role is solely to enhance the training of the explainable model and it does not constitute a component of the ultimate model. Several approaches, such as mixture-of-experts (MoE) methods and hierarchical statistical models, fall under the umbrella of second-order methods[23–26]. Second-order methods share a kinship with classical white-box models but are equally faced with the challenge of attaining prediction accuracy on par with black-box models.

Recent literature on Mixture of Experts (MoE)[26–29] has explored algorithms that enhance explainability. However, these studies focus primarily on tabular data, whereas image data pose unique challenges. Our work addresses both tabular and image data. Additionally, the types of explanations and application scenarios explored in previous studies differ from ours. Our method aims to provide an inherently explainable prediction function. In contrast, other methods emphasize various aspects of the prediction model. For example, the FEAMOE system by[27] focuses on the efficient computation of Shapley values[11], a method used to determine feature importance. In the next section, we will compare the explanations provided by our method with those obtained using Shapley values. The MoE Trees by[29] target the reinforcement learning setting. Another noteworthy study by[30] focuses on understanding why MoE is effective. Insights gained by[30] could potentially lead to the development of new network architectures. Revealing the general advantages of a network structure differs from our objective, which is to interpret the decision process based on a trained model from any given data. The work by[28] is most similar to ours. However, while their method divides samples by adopting additional rules to ensure the experts are balanced and their assignments are smooth for consecutive inputs, ours uses stochastic gates to divide samples, assigning the experts purely to minimize the loss.

Our primary contribution is not the structure of the prediction model, which can be viewed as an instance of hierarchical mixture models in statistics or an MoE with white-box experts and a white-box allocator. Instead, it lies in the innovative training approach for an explainable model, achieved by harnessing the predictive power of a DNN. This strategy, known as co-supervision, plays a central role in our methodology. While co-supervision has been previously used by[26], a distinguishing aspect of our work is the construction of a new neural network architecture that enables end-to-end training with co-supervision from a DNN. This network contains the final network or model as one of its components but, taken as a whole, it is not equivalent to the final prediction model. The incorporation of a DNN in the training process is critical for boosting the accuracy of the explainable model, effectively addressing a common limitation associated with nearly-white-box models.

Figure 2 illustrates the organization of the remainder of the paper. The novelty of our work lies in the development of SEE-Net, which achieves both explainability and high accuracy. The figure outlines the main content of the subsequent sections and demonstrates how they support the key contributions of our research.

Methods

Synced explanation-enhanced neural network (SEE-Net)

We develop a model called *Relational-Class Logistic Regression* (Rec-LR) which achieves high accuracy by exploiting the power of DNN in training an explainable model. In the case of classification, Rec-LR contains two layers of linear logistic regression. In the case of regression, the first layer is a logistic regression model while the second layer contains linear models. For brevity of terminology, we do not explicitly distinguish the two types of linear models used at the second layer. LR is easy to explain in a global sense because the class decision boundary

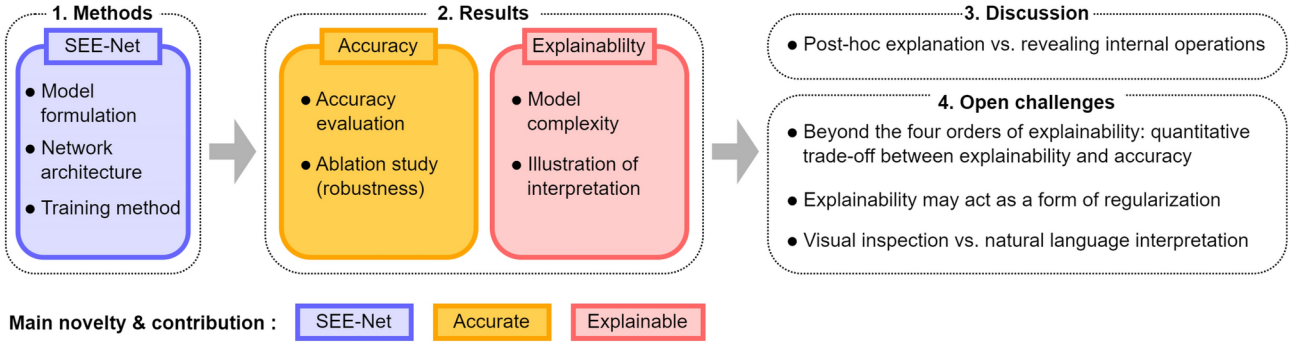


Fig. 2. An outline of the remaining sections of the paper, detailing how the content in these sections supports the stated novelty and contributions of the work. In summary, we introduce SEE-Net, a novel neural network that combines an interpretable NN with a guidance NN, enabling end-to-end training of a highly accurate and explainable predictive model.

is given by a linear function of the input variables and the role of each variable can be directly read off from their corresponding coefficients.

In a Rec-LR model, the first layer LR predicts a *latent class* (LC) label, which is to be distinguished from what we ultimately predict—the class of instances/objects. The latent class is called *relational class* because it determines which linear model will be used to predict the output (instead of the direct value of the output), thus reflecting a coherent relationship between the input and output. Since the LR model can be implemented as one layer of a neural network, Rec-LR is implemented as a shallow neural network (SNN). Rec-LR model boosts the flexibility of fitting data with little reduction in explainability provided that the division into the relational classes is performed by an explainable model, which is true in our case since we employ an LR model in the first layer. Rec-LR may seem deceivingly easy to train as a usual SNN. However, because relational class labels are unknown, we cannot train Rec-LR in a usual stand-alone fashion. The main novelty of our work is to form relational classes based on a DNN so that the prediction power of the DNN can be leveraged.

Next, we introduce the mathematical formulation of Rec-LR. Consider a predictor vector $X \in \mathcal{X} \subset \mathbb{R}^p$, where $\mathbb{R}^p$ is the p-dimensional Euclidean space, and a response $Y \in \mathcal{Y}$. For regression, Y takes a continuum of values, $\mathcal{Y} \subset \mathbb{R}$. For classification, Y is categorical. Denote the number of classes by $M = |\mathcal{Y}|$. Denote a realization of X by $\mathbf{x}$ and that of Y by y. The input data $\{\mathbf{x}_1, ..., \mathbf{x}_n\}$ are i.i.d. samples of X. Let the input data matrix $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n)^t \in \mathbb{R}^{n \times p}$, where each row corresponds to one instance. Let $\mathbf{x}' \in \mathbb{R}^{p'}$ be the *intra-latent-class (intra-LC) explainable features* that are input to the linear model within a particular relational class (also called a *regional linear model*). Definitions of intra-LC features are consistent within each relational class but differ across different relational classes. The degree of locality for these features is influenced by the number of relational classes, which in our experiments was no more than six for every dataset. Importantly, the locality is not defined at the level of individual data points, but rather at a broader class level.

Let $\mathbf{x}'' \in \mathbb{R}^{p''}$ be the *inter-latent-class (inter-LC) explainable features* used to predict the relational classes. Depending on the application, $\mathbf{x}'$ and $\mathbf{x}''$ can simply be the original input $\mathbf{x}$, or they can be processed data from $\mathbf{x}$ that are still interpretable. The Rec-LR model is given by:

$$\begin{aligned} \widetilde{y}_j =& \mathbf{b}_j^T \mathbf{x}' + a_j, \quad j = 1, \cdots, J, \\ \mathbf{v} =& \mathrm{softmax}(\mathbf{V}\mathbf{x}'' + \mathbf{c}), \\ \hat{y} =& < \mathbf{v}, \widetilde{\mathbf{y}} >, \end{aligned}$$

where $< \cdot, \cdot >$ is the inner product, and $\hat{y}$ is the final predicted value, and $\widetilde{\mathbf{y}} = (\widetilde{y}_1, \cdots, \widetilde{y}_J)^T$ is the vector of predicted values by the J regional linear models, and $\mathbf{v} \in [0, 1]^J$ is the weights estimated from the inter-LC explainable features. In the case of classification, the linear predictions $\widetilde{y}_j$, $j = 1, ..., J$, will go through softmax. Note that $\mathbf{v}$ contains the posterior probabilities of the relational classes. We need to estimate $\mathbf{b}_j \in \mathbb{R}^{p'}$, $a_j \in \mathbb{R}$, for $j = 1, \cdots, J$, and $\mathbf{V} \in \mathbb{R}^{J \times p''}$, $\mathbf{c} \in \mathbb{R}^J$. In particular, to estimate $\mathbf{V}$, we need the relational class labels.

To generate the latent relational class labels, we utilize a DNN. Subsequently, the classification of these latent classes is performed using linear logistic regression to preserve explainability. To facilitate end-to-end training and optimize prediction performance, we propose a new structure called *Synced Explanation-Enhanced* neural network (SEE-Net), shown in Fig. 1b-c, to train Rec-LR. SEE-Net contains two connected neural networks called *interpretable NN* and *guidance NN*. The interpretable NN, functionally a Rec-LR model, is the final prediction model to be used on test data, while the guidance NN, which has no structural restriction, is trained for the ultimate purpose of training the interpretable NN. Specifically, the guidance NN is used to generate latent classes, while the interpretable NN attempts to form relational classes that match those latent classes using linear models. If these linear models in the interpretable NN are bypassed and the latent classes produced by the guidance NN are used directly, we call the resulting network *pre-interpretable NN*, which does not offer the same high level of explainability as the interpretable NN. However, classification performance of the pre-interpretable NN will also be reported to demonstrate that the interpretable NN can achieve similar accuracy without accessing the guidance NN.

When a DNN uses the ReLU function to threshold the output of neurons, the final outcome is a linear function within any of the numerous regions created by the outputs of the intermediate layers of the DNN. We are thus motivated to create relational classes in Rec-LR by clustering intermediate DNN outputs across layers, enabling the DNN to aid in training Rec-LR. However, clustering is not readily realized by neural network modules, a technical challenge that we tackle using stochastic gates so that the interpretable NN in SEE-Net can be trained in an end-to-end fashion. Next, we describe how relational classes are formed in SEE-Net.

Generate relational classes by Guidance NN

The effectiveness of the latent relational classes is crucial for the accuracy of the final Rec-LR model. We utilize the guidance NN which is a DNN to form these latent classes. The DNN is embedded within the pre-interpretable NN to facilitate end-to-end training. We first present the basic notations for the guidance NN. Consider a network with L hidden layers and one output layer. Let p_l be the dimension of output at the l-th hidden layer, $l = 0, ..., L$, and $p_0 = p$. Let $\mathbf{z}_l \in \mathbb{R}^{p_l}$ be the output of the l-th hidden layer. Set $\mathbf{z}_0 = \mathbf{x}$. The mapping at the l-th hidden layer is denoted by $h_l(\cdot)$, which can be any type of hidden layer, e.g., dense, convolutional, or recurrent. We have $\mathbf{z}_l = h_l(\mathbf{z}_{l-1})$, $l = 1, \cdots, L$. The output layer, $o(\mathbf{z}_L)$, is defined as either a linear or softmax function depending on whether the purpose is regression or classification.

We cluster the outputs of the hidden layers of the guidance NN to generate relational classes for the input $\mathbf{x}$. This approach is motivated by the following consideration. In a DNN, since each of the hidden layers is the composite function of an affine transformation and an activation function, when the output at each layer

is clustered such that the non-linear effect of the activation function can be neglected within a cluster, the composite function within the cluster is linear. In particular, if the activation function is ReLU, the hidden layers become piecewise linear and the entire network is also piecewise linear. Consequently, within each cluster of $\mathbf{x}$, it becomes logical to predict y using linear models. The clustering process includes the following two major steps.

1. *Clustering at one hidden layer of a DNN*:

   Instead of using conventional clustering to generate the relational classes, which does not integrate well with modules of a neural network architecture and thus prevents end-to-end training, we opt for clustering through linear operations followed by softmax (essentially, logistic regression), as shown in Fig. 1c. Let k_l be a user-specified hyperparameter specifying the number of clusters to be generated from the l-th hidden layer outputs, $\mathbf{z}_l$, in the guidance NN. Let $\boldsymbol{\rho}_l$ be the posterior probabilities of the clusters at the l-th hidden layer (that is, to perform soft-clustering). Specifically, $\boldsymbol{\rho}_l = (\rho_1^{(l)}, \cdots, \rho_{k_l}^{(l)})^T$, $\rho_j^{(l)} \in [0,1]$ for $j = 1, \cdots, k_l$, and $\sum_{j=1}^{k_l} \rho_j^{(l)} = 1$. In SEE-net, ρ_l's are computed by an affine transformation of $\mathbf{z}_l$ followed by a softmax function:

$$(\tilde{z}_1^{(l)}, \cdots \tilde{z}_{k_l}^{(l)})^T = \mathbf{U}_l \mathbf{z}_l + \mathbf{d}_l \ , \quad \boldsymbol{\rho}_l = \mathrm{softmax}\left((\tilde{z}_1^{(l)}, \cdots, \tilde{z}_{k_l}^{(l)})^T\right) \ ,$$

   where $\mathbf{U}_l \in \mathbb{R}^{k_l \times p_l}$, $\mathbf{d}_l \in \mathbb{R}^{k_l}$, and $\tilde{z}_j^{(l)} \in \mathbb{R}$ for $j = 1 \cdots, k_l$. The softmax function $\mathrm{softmax}(\cdot)$ with a temperature parameter λ is given by the following equation. As λ decreases, the posterior probability converges to one for one class and to zero for others, achieving hard clustering.

$$\mathrm{softmax}\left((\tilde{z}_1^{(l)}, \cdots, \tilde{z}_{k_l}^{(l)})^T\right) = \left(\frac{e^{\tilde{z}_1^{(l)}/\lambda}}{\sum_{j=1}^{k_l} e^{\tilde{z}_j^{(l)}/\lambda}}, \cdots, \frac{e^{\tilde{z}_{k_l}^{(l)}/\lambda}}{\sum_{j=1}^{k_l} e^{\tilde{z}_j^{(l)}/\lambda}}\right)^T .$$

2. *Clustering across the layers of a DNN*:

   Since there are k_l clusters at the lth layer, the sequence of cluster labels across all the L layers has $K = \prod_{l=1}^{L} k_l$ configurations. We can take each configuration as one cluster, which we call a *guided cluster* since these clusters are generated with the help of the guidance NN. The guided clusters are the Cartesian product clusters of the clusters generated at each layer. The soft-clustering probabilities of all the guided clusters are computed by the Kronecker product of $\boldsymbol{\rho}_l$, $l = 1, \cdots, L$. Denote by $\mathbf{u}$ the output of the Kronecker product, $\mathbf{u} = \bigotimes_{l=1}^{L} \boldsymbol{\rho}_l$. Then, $\mathbf{u} = (u_1, \cdots, u_K) \in [0,1]^K$, and $\sum_{j=1}^{K} u_j = 1$. Each entry of $\mathbf{u}$ is the probability of a corresponding guided cluster. It is impractical to use the guided clusters directly as the relational classes to build a Rec-LR because the number of guided clusters grows exponentially with the number of layers. A Rec-LR with too many relational classes can become difficult to interpret. As shown in Fig. 1c, the guided clusters are merged into the relational classes using stochastic gates. Stochastic gates within SEE-Net are seamlessly optimized through end-to-end learning.

   Denote the number of merged clusters, that is, the final relational classes, by J, usually, $J \ll K$. Also refer to the soft-clustering probabilities of the relational classes as component weights $\mathbf{w} = (w_1, ..., w_J)^T$. $\mathbf{w} \in [0,1]^J$, and $\sum_{j=1}^{J} w_j = 1$. The merging of the guided clusters can be represented by a group-mask matrix $\mathbf{M}$ of size $J \times K$, $\mathbf{M} \in \{0,1\}^{J \times K}$ and $\sum_i \mathbf{M}_{ij} = 1$. The element of $\mathbf{M}$, $M_{ij} = 1$ if the j-th guided cluster belongs to the i-th merged cluster and 0 otherwise. Given $\mathbf{M}$, we obtain the component weights by $\mathbf{w} = \mathbf{M}\mathbf{u}$. However, given the discrete nature of $\mathbf{M}$, it cannot be trained in the framework of neural network. To overcome this hurdle, we are inspired to use the CONCRETE (CONtinuous relaxations of disCRETE) distribution[31] to generate a soft group-mask matrix $\mathbf{M}'$ that mimics the discrete $\mathbf{M}$ and is trainable in a unified framework of SEE-Net. The idea of using the CONCRETE distribution for regularization has been explored by [32]. Here, we extend the CONCRETE distribution to handle more than two classes and use the extension to emulate the process of merging clusters.

   To train the CONCRETE group-mask matrix $\mathbf{M}'$, we re-parametrize it using real-valued CONCRETE random variables $\boldsymbol{\Psi} = (\psi_{ij})_{i=1,\cdots J; j=1 \cdots K}$ of the same dimensionality as $\mathbf{M}'$. Every element of $\mathbf{M}'$ is ensured to be non-negative and no greater than 1, and can be controlled to approach either 0 or 1. Specifically, let $q(\boldsymbol{\Psi}|\boldsymbol{\phi})$ be a *hard CONCRETE multinoulli distiribution* with parameters $\boldsymbol{\phi}$. The function of $q(\boldsymbol{\Psi}|\boldsymbol{\phi})$ is called the *stochastic gate*. The parameter $\boldsymbol{\phi} = (\boldsymbol{\alpha}, \beta, \gamma, \zeta)$ contains a trainable location parameter $\boldsymbol{\alpha} = (\alpha_{ij})_{i=1,\cdots J; j=1 \cdots K} \in \mathbb{R}^{J \times K}$, pre-given real-valued hyperparameters β, γ, ζ, and an element-wise truncation function $\mathrm{trunc}(x)$ such that $\mathrm{trunc}(x) = \min(1, \max(0, x))$. The hyperparameter $\beta > 0$, called temperature, controls the extent to which

$\mathbf{M}'$ resembles a discrete group-mask matrix (more so at a smaller β), and $\gamma < 0$ and $\zeta > 1$ are stretch parameters that stretch a probability value to the interval (γ, ζ).
To estimate $\boldsymbol{\phi}$ and construct $\boldsymbol{\Psi}$, we first sample ϵ_{ij}, $i = 1, \cdots, J-1$, $j = 1, \cdots, K$, from $(J-1) \times K$ independent uniform distributions. Let $\epsilon_{(\cdot,j)}$ be the ordered statistics of $\epsilon_{\cdot,j}$. Set $\epsilon_{(0,j)} = 0$ and $\epsilon_{(J,j)} = 1$. Let $\delta_{(i,j)} = \epsilon_{(i,j)} - \epsilon_{(i-1,j)}$. The random variables ψ_{ij} for $i = 1, \cdots, J$, $j = 1, \cdots, K$, are computed by $\psi_{ij} = \text{trunc}\left(\widetilde{\psi}_{ij}(\zeta - \gamma) + \gamma\right)$ where

$$(\widetilde{\psi}_{1j}, \cdots, \widetilde{\psi}_{Jj})^T = \text{softmax}\left(\left(\frac{\log(\delta_{(1j)}) - \log(1-\delta_{(1j)}) + \log\alpha_{1j}}{\beta}, \cdots, \frac{\log(\delta_{(Jj)}) - \log(1-\delta_{(Jj)}) + \log\alpha_{Jj}}{\beta}\right)^T\right).$$

The location parameters α_{ij}'s are key for re-parameterizing $\mathbf{M}'$ and will be trained in an end-to-end fashion as part of the pre-interpretable NN. The soft group-mask matrix $\mathbf{M}'$ is obtained by the column-wise normalization of $\boldsymbol{\Psi}$ so that $\sum_i M'_{ij} = 1$ for $j = 1, \cdots, K$. Finally, we compute the relational class posterior probabilities (mixture component weights) by $\mathbf{w} = \mathbf{M}'\mathbf{u}$. As the number of non-empty relational classes is determined during the training of stochastic gates, it can become smaller than *J*, the pre-selected target number.

Although the guidance NN is used to produce the weights $\mathbf{w}$ for the relational classes, we do not use $\mathbf{w}$ when applying the final prediction model, because $\mathbf{w}$ is generated by a DNN, not explainable by the standard of the interpretable NN. In other words, $\mathbf{w}$ is computed only for training, and subsequently, an interpretable module in SEE-net is trained to infer $\mathbf{w}$. The interpretable NN is also trained in the end-to-end fashion with the capacity of selecting variables from $\mathbf{x}'$ and $\mathbf{x}''$ for the global and regional linear models respectively. For end-to-end training, it is difficult to apply directly the L_1 penalty on the linear coefficients, as is done in Lasso. Instead, we employ the technique of hard CONCRETE Bernoulli distribution as introduced by[31], which generates a mask of variable selection on the intra- and inter-LC features $\mathbf{x}'$ and $\mathbf{x}''$.
The possible candidates for $\mathbf{x}'$ and $\mathbf{x}''$ are different depending on the specific task and data types. For example, in image analysis, instead of raw pixel values, extracted features by low-level filtering, e.g., edge filters, wavelets, first convolution layer outputs, are often used. We use the first convolutional layer output or other relatively simple transformation of input features $\mathbf{x}$ to find reasonable $\mathbf{x}'$ and $\mathbf{x}''$ for each task.

Training and optimization criteria for SEE-Net

Several types of losses are considered in SEE-net, a weighted combination of which is defined as the overall loss. Depending on different purposes, a certain type of loss can be more prominent than the others.

Let $\hat{y}_i^I$ be the prediction of the *i*th response, $i = 1, ..., n$, by the interpretable NN, and $\hat{y}_i^G$ be the prediction by the pre-interpretable NN. Note that because the guidance NN does not include an output layer that predicts the response *Y*, the pre-interpretable NN is a combination of the guidance NN, the stochastic gates, and the linear models within each regional class. The difference between the processes that predict respectively $\hat{y}_i^G$ and $\hat{y}_i^I$ lies in whether the "original" or "predicted" relational classes are used. If we replace the module of the interpretable NN that contains linear models for predicting regional classes by the concatenation of the guidance NN and stochastic gates, we get $\hat{y}_i^G$ rather than $\hat{y}_i^I$. While both the interpretable and pre-interpretable NNs incorporate a module of regional linear models, as we will shortly elaborate, it is important to note that these models are trained at different stages. This distinction results in variations in parameters between the regional models in the two networks.

We can measure the total loss between the prediction of the interpretable NN and the true response by $\sum_{i=1}^n \mathcal{L}(\hat{y}_i^I, y_i)$. Similarly, the total prediction loss for the pre-interpretable NN is $\sum_{i=1}^n \mathcal{L}(\hat{y}_i^G, y_i)$, which is the objective function used to train this network. The connection between the interpretable NN and the guidance NN is captured in the relational classes—the posterior $\mathbf{v}$ estimated by the interpretable NN aims at being aligned with the posterior $\mathbf{w}$ estimated by the guidance NN. We measure the distance between the posterior vectors by the relative entropy $\mathcal{K}(\mathbf{w}_i||\mathbf{v}_i)$ and define $\sum_{i=1}^n \mathcal{K}(\mathbf{w}_i||\mathbf{v}_i)$ as a loss.

Consider the objective of achieving high prediction accuracy through the interpretable NN. Specifically, predictions are provided by the Rec-LR, which is derived from the interpretable NN, while the guidance NN identifies the relational classes. To maximize prediction accuracy by effectively leveraging the relational classes, we incorporate two loss functions. The first is the empirical prediction loss, $\sum_{i=1}^n \mathcal{L}(\hat{y}_i^I, y_i)$, and the second is $\sum_{i=1}^n \mathcal{K}(\mathbf{w}_i||\mathbf{v}_i)$. Let $\mathcal{R}(\theta)$ be the overall loss of SEE-Net where θ is the collection of parameters. We define $\mathcal{R}(\theta)$ by

$$\mathcal{R}(\theta) = \frac{1}{n}\sum_{i=1}^n \left[\mathcal{L}(\hat{y}_i^I, y_i) + \lambda\mathcal{K}(\mathbf{w}_i||\mathbf{v}_i)\right], \tag{1}$$

where λ is a hyperparameter.

We conduct training by two stages to optimize SEE-Net. In the first stage, we train the pre-interpretable NN to produce $\mathbf{w}_i$'s by minimizing the average loss $\mathcal{L}(\hat{y}_i^G, y_i)$. After establishing the guidance NN and setting the stochastic gates, we proceed to the second stage, where the interpretable NN is trained. Here, we focus on minimizing the objective function in Eq. (1). During this stage, we train the linear models that predict the relational classes, and concurrently, we refit the regional models within each relational class for the optimal performance of the interpretable NN. Theoretically, it would be possible to train the entire SEE-Net in a single

stage. However, our experimental study indicates that SEE-Net may not train effectively if we attempt to optimize the linear models that predict the relational classes without first establishing a useful set of relational classes. By adopting a two-stage training strategy, we effectively circumvent this challenge.

Results

Datasets and model setups

Our experiments were conducted on the following five datasets: three image datasets and two clinical datasets.

1. MNIST handwritten digits: This dataset contains images of handwritten digits (0 to 9), each digit being one class.
2. Flower Recognition Dataset: Images classified into five flower types—daisy, dandelion, roses, sunflowers, and tulips. This dataset is accessible through the *Tensorflow* Python package at URL: http://download.tensorflow.org/example_images/flower_photos.tgz.
3. CIFAR-10 Dataset: This dataset consists of 32 by 32 colour images from 10 classes, with 5000 and 1000 images per class for training and testing sets. This dataset is accessible through the URL: https://www.cs.toronto.edu/~kriz/cifar.html.
4. TCGA Skin Cutaneous Melanoma (SKCM): This dataset includes binary overall survival (OS) status (1 for living, 0 for deceased) and 30 demographic and clinical variables (e.g., age, gender, tumor status), resulting in 73 features after dummy encoding. This dataset is accessible via the R package TCGAretriever.
5. Parkinson's Disease Dataset (PD)[33] : This dataset consists of data from 188 Parkinson's patients (class 1) and 64 healthy individuals (class 0), focusing on PD detection through vocal impairments. A total of 753 features are generated by various speech signal processing algorithms, e.g., time-frequency features and Mel Frequency Cepstral Coefficients (MFCCs).

We utilize the widely-used ADAM optimizer[34] to train SEE-Net. Data samples are iteratively used in training as follows: for MNIST, training occurs over 20 epochs with evaluations every 200 iterations; for PD, 50 epochs with evaluations every 500 iterations; for SKCM, 50 epochs with evaluations every 500 iterations; and for Flowers and CIFAR-10, 20 and 40 epochs respectively without periodic iterations. As we conducted 5-fold CV, for all datasets, 80% of the samples are used for training and the remaining 20% for testing in each fold. We now describe the detailed model setups for each dataset.

1. For MNIST digit classification, we use LeNet-4[35] as the guidance NN, which consists of 2 convolutional layers and 1 dense layers. The hyperparameters of SEE-Net are chosen as $k_l = 2$ for $l = 1, 2, 3$ and $J = 10$ based on CV results that yielded the highest training accuracy. SEE-Net divides samples into 6 non-empty relational classes. The test accuracy of SEE-Net is slightly lower than that of LeNet-4 (see Table 1). We assume that the shape and positioning of activated pixels are crucial for differentiating digits. For instance, the digit '1' typically has active pixels in the vertical center, whereas '0' often has a central inactive region. This location information of active pixels is captured by average pooling of the original image. We employ 5×5 pixel windows with 2×2 strides to determine the approximate locations of activated pixels, using these values as inter-LC explainable features (denoted by $\mathbf{x}''$). Quantifying the shapes of activated pixels presents a greater challenge. To address this, we utilize the outputs from the first convolutional layer of the guidance NN as the intra-LC explainable features (denoted by $\mathbf{x}'$).
2. For the Flower dataset, the guidance NN is chosen to be LeNet-4, used again for its relatively simple structure, but more filters are included in the network, specifically, 32 3×3 filters at each of the two convolutional layers. By setting the hyperparameters $k_l = 5, l = 1, 2$, and $J = 10$ in SEE-Net based on CV results that yielded the highest training accuracy, the flower images are categorized into 4 non-empty relational classes. The inter-LC features utilized for classifying these relational classes are derived from filters learned at the second convolutional layer of the guidance NN, i.e., a total of 32 filters applied to 3×3 pixel windows. Specifically, each inter-LC explainable feature represents the maximum value across an entire feature map. If we consider that each filter aims to capture a specific pattern, then each inter-LC feature quantifies the strength of one pattern across the entire image. To define intra-LC explainable features, we utilize another distinct convolutional layer with 128 filters, each with a 6×6 pixel window, applied individually to the RGB color channels. Thus, we obtain 384 feature maps in total, and once again, the maximum values of each feature map are used as the intra-LC explainable features.
3. For the CIFAR-10 dataset, the guidance NN and the explainable features are designed using the same structure as those used for the Flower dataset. However, because the images are small (32×32), it is difficult to define effective intra-LC explainable features.
4. For SKCM, we use three dense layers with $256 - 64 - 32$ units as the guidance NN. We set both inter- and intra-LC explainable features to be the 73 original inputs. The guidance NN identified 4 distinct relational classes when we set the maximum number of it as $J = 5$ and $k_l = 7$ for $l = 1, 2$ based on CV results that yielded the highest training accuracy. For MLP, we used three dense layers and $16 - 64 - 32$ units for each layer. For RF, we set the maximum depth as 7.
5. For PD data, we use two dense layers with $20 - 20$ units as the guidance NN. For interpretable NN, the original features are used for both inter- and intra-LC explainable features. $k_l = 5$ for $l = 1, 2$ and $J = 5$ based on CV results that yielded the highest training accuracy. Interpretation can be obtained through the two global and regional linear models that have the original features as covariates. For MLP, we used two dense layers and $20 - 20$ units for each layer. For RF, we set the maximum depth as 7.

Method	MNIST		Flowers		CIFAR-10		SKCM		PD	
	Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
LR	0.964 (0.000)	0.957 (0.001)					0.829 (0.005)	0.739 (0.026)	0.822 (0.011)	0.765 (0.039)
SVM	0.994 (0.000)	0.988 (0.001)								
RF							0.942 (0.037)	0.724 (0.046)	0.974 (0.008)	0.760 (0.038)
CNN			0.881 (0.013)	0.822 (0.011)	0.733 (0.005)	0.728 (0.004)				
MLP							0.986 (0.007)	0.715 (0.009)	0.935 (0.049)	0.801 (0.073)
LeNet-4	0.999 (0.000)	0.999 (0.000)								
MobileNet			0.779 (0.040)	0.750 (0.032)	0.910 (0.008)	0.849 (0.006)				
ConvNeXT			0.704 (0.045)	0.696 (0.037)	0.865 (0.183)	0.720 (0.110)				
SEE-Net (Pre-intp)	0.999 (0.000)	0.999 (0.000)	0.940 (0.004)	0.927 (0.005)	0.864 (0.006)	0.859 (0.007)	1.000 (0.000)	0.720 (0.020)	0.967 (0.040)	0.815 (0.053)
SEE-Net (Intp)	0.999 (0.000)	0.999 (0.000)	0.908 (0.010)	0.901 (0.011)	0.780 (0.020)	0.775 (0.020)	0.934 (0.006)	0.761 (0.039)	0.932 (0.049)	0.778 (0.045)

Table 1. AUC (Area under the ROC curve) scores and standard deviations based on 5-fold cross-validation (CV) for the pre-interpretable and interpretable NNs of SEE-Net. The prediction performance of SEE-Net is comparable to competitive complex neural networks. Comparisons are made with other methods including logistic regression (LR) implemented by Lasso, support vector machine (SVM), random forest (RF), and multilayer perceptions (MLP). Not all methods are applied to every dataset because some are not suitable to the nature of the data (e.g., LR and RF have difficulty with high dimensional images). Instead, SEE-Net is compared with three other most suitable methods for each dataset. For the image data, state-of-the-art DNNs such as LeNet-4[35], ConvNeXT[36], and MobileNet[37] with default settings from Tensorflow are used. For the two medical data, multi-layer perceptron (MLP) is used. SEE-Net (Intp) indicates the interpretable NN, the final model with explainability. SEE-Net (Pre-intp) represents the pre-interpretable NN that contains the guidance NN. One-sided t-tests for SEE-Net (Intp) being better than LR (for MNIST, SKCM, and PD) and for SEE-Net (Intp) being better than CNN (for Flowers and CIFAR-10) are conducted. The p-values obtained are <0.001, <0.001, 0.037, 0.342, and 0.425, for the five datasets in the order from left to right as shown in the table.

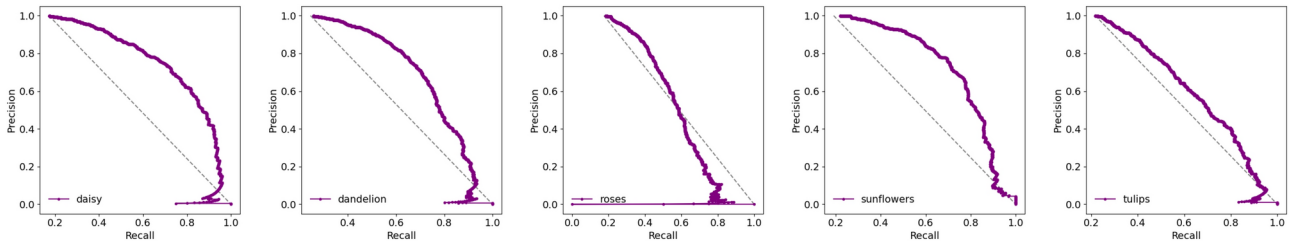


Fig. 3. Prediction-recall curves for the Flowers data obtained by the SEE-Net (Intp) model. The precision-recall curves are presented for every class.

Predictive accuracy of SEE-Net

We compared SEE-Net with the easy-to-explain logistic regression (LR) model and several black-box models. Depending on the datasets, different black-box models most suitable to the type of data were compared. For image data, the black-box models are mostly state-of-the-art DNNs. The shallow networks of interpretable NN under the co-supervision of the guidance NN are expected to achieve similar accuracy to the complex networks. Table 1 presents the predictive performance measured by AUC (Area under the ROC curve) for SEE-Net (Intp), i.e., the interpretable NN, and SEE-Net (Pre-intp), i.e., the pre-interpretable NN. In practical applications, SEE-Net (Intp) serves as the final model, while SEE-Net (Pre-intp) is assessed to glean insights into the network's efficiency. Furthermore, as an example of more detailed results than AUC obtained by SEE-Net (Intp), the precision-recall curves for each class in the Flower dataset are shown in Fig. 3. The average AUC and standard deviation in Table 1 are based on 5-fold cross-validation (CV). For all datasets, SEE-Net (Intp) performs better on average than LR or CNN. To formally test whether SEE-Net (Intp) achieves better accuracy than LR for the MNIST, SKCM, and PD datasets, and better accuracy than CNN for the Flowers and CIFAR-10 datasets, we conducted one-sided t-tests. For the MNIST, Flowers, and CIFAR-10 datasets, SEE-Net (Intp) is significantly better at the 5% significance level, but not for the SKCM and PD datasets. The p-values are listed in the caption of Table 1.

The hyperparameters of each model were determined by 5-fold CV. The candidate values of J and k_l vary around the number of final classes (i.e., the number of different values of Y) and are taken from the set $\{2, 5, 7, 10, 15, 20\}$. However, for computational effectiveness, only a subset of values are examined and the subset varies with the dataset. Finally, we selected hyperparameters that yielded the highest training accuracy based on CV. Table 2

	MNIST	Flowers	CIFAR-10	PD	SKCM
SEE-Net (Pre-intp)					
Num. of trainable params.	343,882	4,102,269	313,895	19,605	2374
Num. of merged cls., J	10	10	20	5	5
Num. of cls. at l's layer, k_l	10	10	20	5	2
SEE-Net (Intp)					
Num. of trainable params.	463,790	13,500	27,100	15,070	1470
Num. of merged cls., J	10	10	20	5	5
Num. of cls. at l's layer, k_l	2	5	20	5	7

Table 2. The number of merged clusters J and the number of clusters at l's layer k_l are selected based on training accuracy given by 5-fold cross-validation. Other DNNs have varying trainable parameter sizes: LeNet-4 for MNIST has 18,958 parameters, MobileNet for Flowers and CIFAR-10 has 3,212,101 and 3,217,226 parameters respectively, and MLP for PD and SKCM has 15,521 and 4,385 parameters respectively.

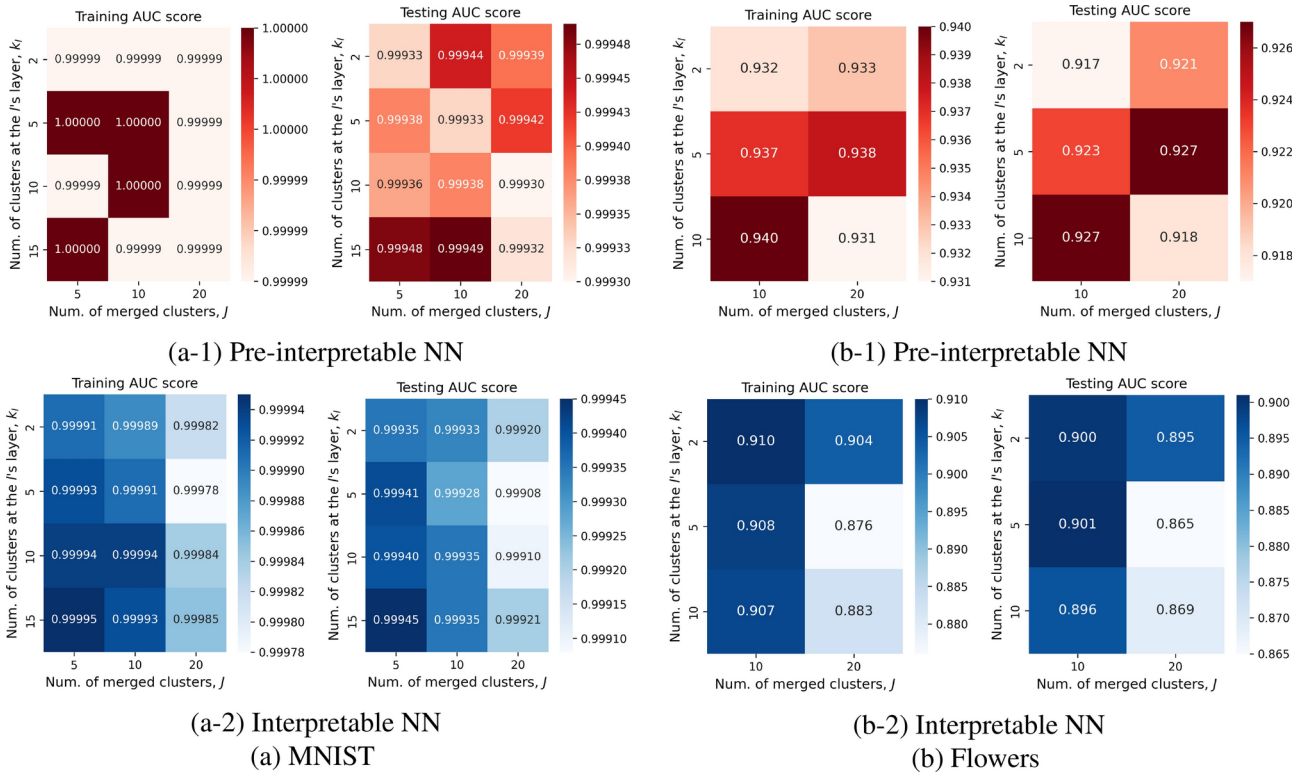


Fig. 4. Ablation study for the model hyperparameters k_l and J. For both the MNIST and Flowers datasets, the performance differs marginally across different settings. However, an excessive number of relational classes, J, can degrade the performance.

provides selected model parameters for the five datasets. To assess the sensitivity of prediction accuracy to the choices of hyperparameters, Fig. 4 shows the ablation study results for two image datasets. The performance of SEE-Net remains stable under different parameter settings. However, a noticeable decrease in accuracy occurs if J is too large. For brevity, we skip presenting results for the other datasets. However, we observe that SEE-Net is not highly sensitive to hyperparameters, achieving similar results under multiple settings.

As indicated in the Table 1, the interpretable NN exhibits only a negligible average difference in performance compared to its pre-interpretable counterpart. On the test datasets, SEE-Net (Intp) outperforms DNNs on two of the five datasets and ranks among the top-performing methods on another two datasets. The occasionally superior results of SEE-Net over the black-box models may be attributed to the regularization of model complexity imposed by the structure of the interpretable NN. The relatively small gap between the training and testing accuracy by SEE-Net indicates that overfitting is less serious with SEE-Net compared with the other models. Furthermore, SEE-Net exhibits notably higher accuracy on test data compared to DNN models when the latter display a significant disparity between training and test accuracy. This observation suggests that SEE-Net is less susceptible to overfitting due to its more restrictive architecture. Additionally, SEE-Net consistently

outperforms logistic regression by a notable margin, highlighting the benefits of creating multiple relational classes.

We compared the computation time for training SEE-Net versus other neural network models according to one of the 5-fold CV experiments on the Flower dataset (consisting of 2,936 images of size 180×180 and 3 color channels). Each neural network model was trained for 1 epoch with a batch size of 32 in a Google Colab T4 GPU environment. The CNN, comprising 3 convolutional layers and 2 feed-forward layers, took 6.033 seconds. MobileNet required 22.099 seconds, while ConvNeXt took 74.453 seconds. SEE-Net (Pre-intp) took 34.306 seconds, and SEE-Net (Intp) took 16.135 seconds.

Interpretation of SEE-Net models

The linear models within each relational class in the interpretable NN can pinpoint key variables in the decision-making process. The choice of these variables depends on the relational class of each sample. In Table 3, we present the estimated counts of relational classes for each dataset and the numbers of variables selected in various linear models. For all five datasets, both the number of inter-LC features identifying the largest relational class and the number of intra-LC features selected in the linear model for this class are significantly lower than the original variable dimension. In addition, across the datasets, the number of relational classes is small, suggesting that interpreting these SEE-Net models is nearly as straightforward as explaining a single logistic regression model.

In SEE-Net, the interpretable NN calculates class posterior probabilities as a weighted sum of probabilities given by linear models of each relational class, the weights being the data point's relational class posteriors. We assessed if the posteriors of relational classes are sharp, i.e., dominated by a single class, which would simplify explanations as decisions can be primarily attributed to one linear model at any data point. Conversely, a blend of multiple models would indicate more complex decision-making. For each data point, we calculate the posterior probabilities for its relational classes, recording the maximum value. Fig. 5 presents the histograms of these maximum values across all data points. The posteriors are computed by both the interpretable NN and the pre-interpretable NN, with the former producing sharper posterior values—those close to 0 or 1. Consequently, the histograms of maximum values often peak at 1, indicating that, for many data points, the classification decision can be attributed to a single linear model. This trend is noticeable in the MNIST, PD, and SKCM datasets but less pronounced in the Flower dataset. As mentioned earlier, despite the interpretable NN being more explainable than the pre-interpretable NN, their performance on test data is nearly identical for the MNIST, Flower, PD and SKCM datasets (see Table 1). For the CIFAR-10 dataset, which consists of 32 by 32 small colored images, there is a relatively large gap between interpretable and pre-interpretable NNs because the data cannot be easily explained with simple features.

Next, we present the insights obtained about how prediction decisions are made for individual datasets, facilitated by the explanatory capabilities of SEE-Net.

Image data

To interpret SEE-Net for image data, we first visualize the relational classes by displaying images assigned to different groups. Although we can also inspect the coefficients in the linear models that predict the relational class, when many spatially localized features are used, the interpretation may not clearly align with high-level patterns. In contrast, displaying example images from different relational classes can be more effective for understanding these patterns. Fig. 6 illustrates how MNIST digits are categorized into relational classes. For a given digit, the images in different relational classes display different writing styles.

We can visually understand how regional linear models make predictions for a given image by computing the feature maps using various convolution filters, each referred to as a channel. Take MNIST as an example. Fig. 7 illustrates four channels capturing distinct shape characteristics in the original images. For instance, the last channel focuses on horizontal marks, while the other three emphasize diagonal marks. These feature maps, multiplied by corresponding coefficients in the regional linear models, are further visualized in the rightmost column of the figure. Fig. 7 suggests that a '0' is identified by vertical marks on two sides and a horizontal mark at the bottom, while a '1' is primarily distinguished by its central vertical mark. In the figure, we showcase instances when a '3' is correctly identified, a '3' is incorrectly classified as '7', and a '2' is wrongly classified as '3'. Notably, in all cases, the evidence suggests that a clear horizontal mark at the center leads the trained classifier to identify a '3'. The required horizontal mark at the center is absent for the '3' classified as '7', whereas it is prominently

Data	# rel.cls.	# of reduced var. in inter-LC features			# of reduced var. in intra-LC features		
		Largest cls.	Across cls.	Total var.	Largest cls.	Across cls.	Total var.
MNIST	6	78	144	144	1204	2274	2304
Flowers	4	18	32	32	205	382	384
SKCM	4	42	70	73	38	72	73
PD	2	384	655	753	391	655	753

Table 3. Complexity of trained models for each dataset. The number of relational classes is shown in the first column. For each relational class, there is a linear model that uses inter-LC features to identify this relational class and another regional linear model to predict the final class using intra-LC features. For each type of linear models, the numbers of variables selected for the largest relational class, those selected across all relational classes, and the original number of variables are listed.

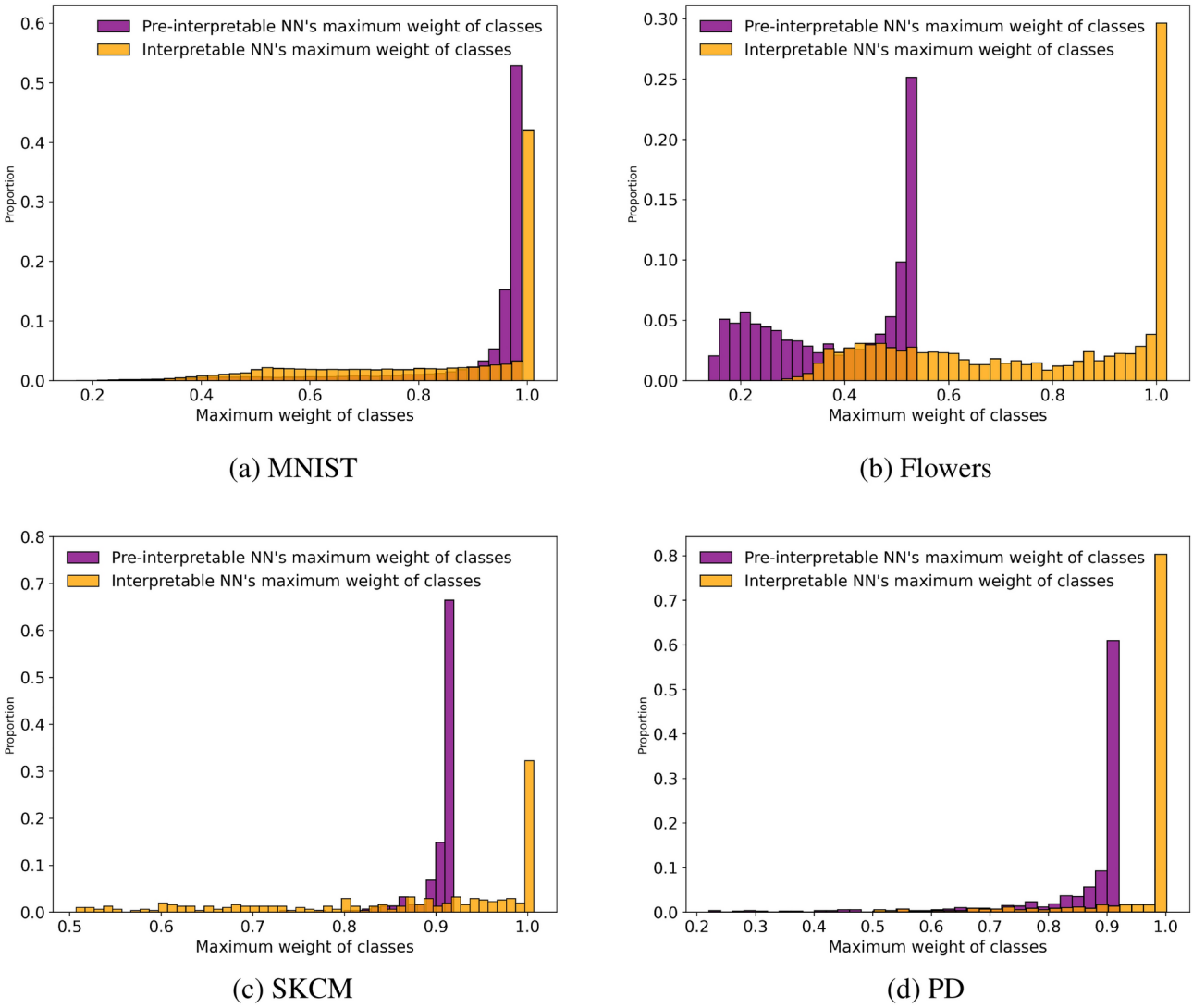


Fig. 5. Histograms of the maximum relational class posterior probabilities per data point. The posteriors are computed respectively from the pre-interpretable NN and the interpretable NN. The interpretable NN is designed to generate more extreme values of the posterior probabilities—those close to either 1 or 0.

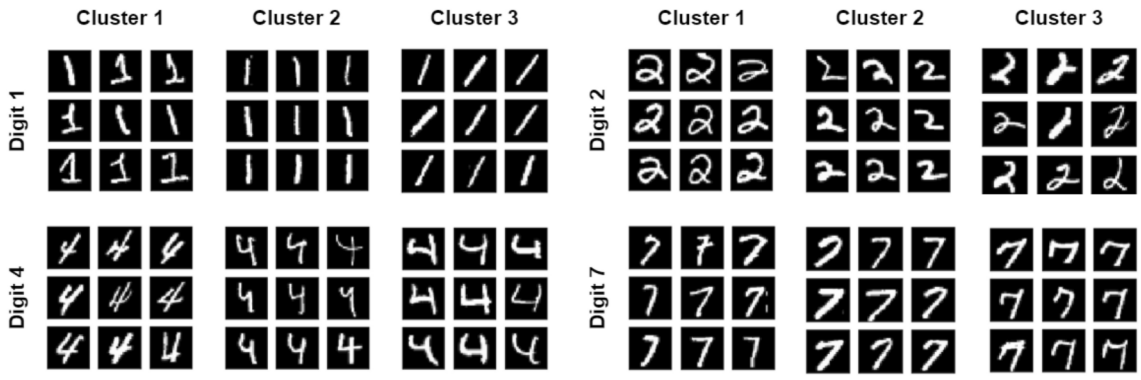


Fig. 6. MNIST digits 1, 2, 4, and 7 are grouped into three relational classes (labeled as clusters in the figure), indicating variations in writing styles among the same digit images in different clusters.

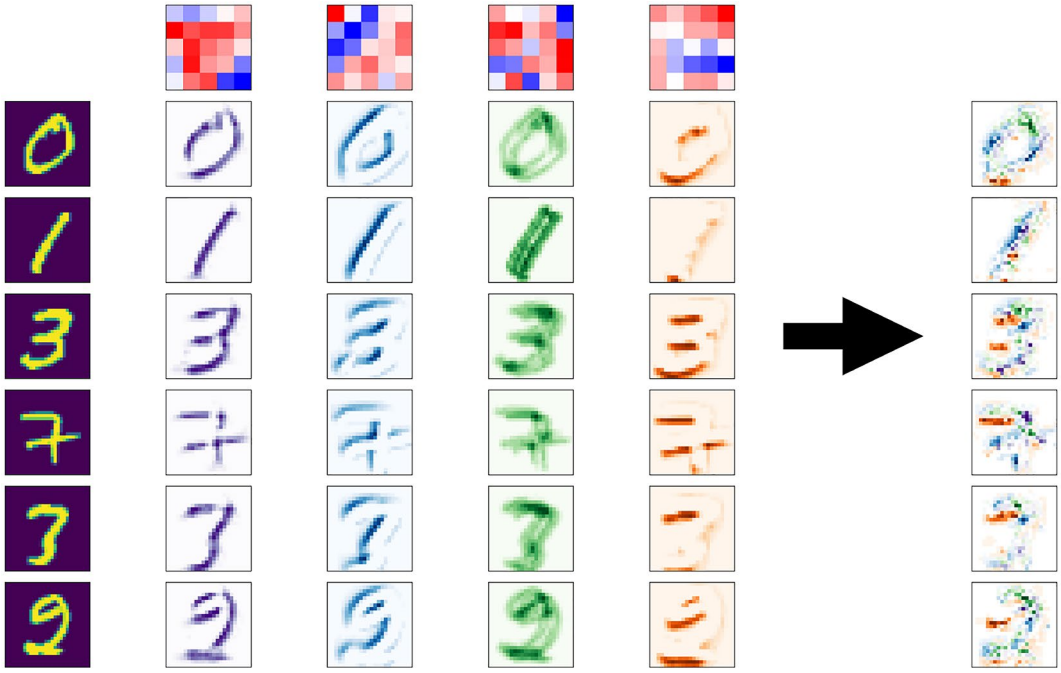


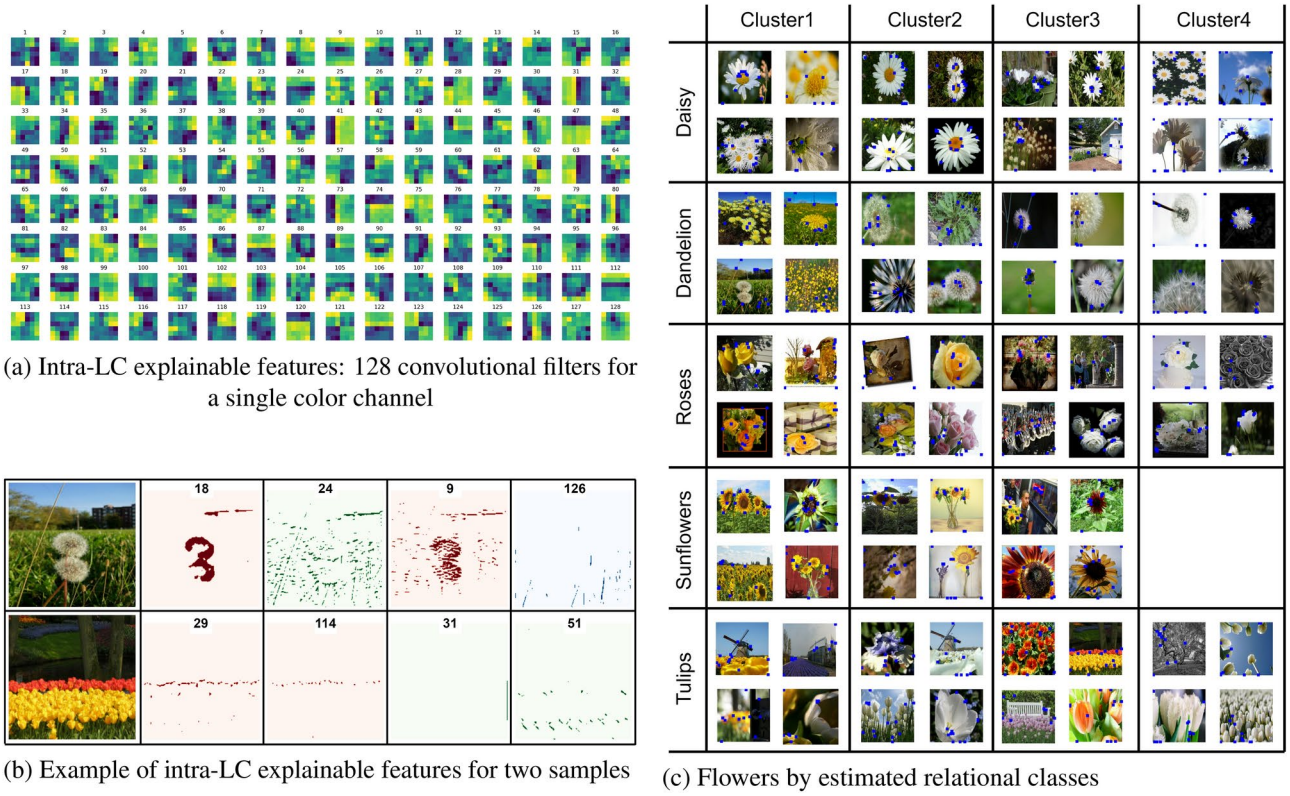
Fig. 7. The interpretable NN for MNIST data uses feature maps computed from four filter channels at the first convolution layer as intra-LC explainable features. The top row exhibits filters employed in the four channels. The leftmost column contains the original sample images. The following four columns correspond to feature maps produced by each filter. Finally, the rightmost column shows the weighted sum of the feature maps, with weights given by the linear coefficients of the regional linear models. To differentiate between channels, distinct colors are employed to illustrate the feature maps. The intensity of the feature map at each pixel is denoted by the level of darkness, with a darker shade indicating a larger magnitude. The '3' in the third row is correctly classified, the '3' in the fifth row is incorrectly classified as '7', and the '2' in the last row is incorrectly classifieid as '3'.

present for the '2' classified as '3'. The significance of the horizontal mark at the center is not entirely surprising in hindsight but is not readily apparent in foresight. This example highlights the SEE-Net's capacity to reveal the decision-making mechanism.

Fig. 8c presents sample images from each of the four relational classes established for the Flower data. In relational class 1, the majority of images feature yellow flower patterns, while relational class 4 predominantly displays white flower patterns. Interestingly, in the case of dandelions, the yellow flower heads only appear in relational class 1, while the white seed heads appear in the other classes. For sunflowers, there are no samples in relational class 4. The blue dots shown in the images indicate the pixels where non-zero maximums occur for any of the 32 patterns. At times, these blue dots are situated at the border of an image, signaling that some filters might not function optimally, as the borders do not offer valuable information about the characteristics of flowers.

Fig. 8a shows the 128 filters used for extracting intra-LC explainable features. Each feature indicates a specific pattern such as linear (e.g., 31st filter), mottled (e.g., 18th filter), and spotted (e.g., 9th filter) patterns. In Fig. 8b, pixel positions that yield large magnitudes when a certain filter is applied are shown for two sample images. Among the 128 filters, we only showcase the top four filters that contribute to an intra-LC feature with the strongest impact on class prediction. The dandelion image illustrates that red mottled patterns (18th filter) in the seed head and green diagonal patterns (24th filter) correspond to the top two features with the greatest impact on the classification decision for this image. On the other hand, the tulip image reveals that the edges of the flower field, identified by the diagonal (29th filter) and dotted (114th filter) patterns, have the most significant impact. The linear pattern (31st filter) on the edge is used to classify the tulips, even though this pattern contains no direct information about the tulips.

As a comparison, we present in Fig. 9 alternative explanations for example images using the methods Grad-CAM [38] and Shapley values [11]. Grad-CAM is a local explanation technique that visualizes sample-specific important pixels for image classification. Similarly, Shapley values indicate feature importance for a specific decision or the entire model. While these alternative methods offer insights into which areas of an image or which features are most crucial in a decision, they fail to explain why a particular decision is made-a capability our model specifically aims to provide. For instance, if a decision is incorrect, these methods cannot identify which specific traits of an image contributed to the erroneous decision.



(a) Intra-LC explainable features: 128 convolutional filters for a single color channel

(b) Example of intra-LC explainable features for two samples

(c) Flowers by estimated relational classes

Fig. 8. Model interpretation for the Flower dataset. (**a**) The 128 single-color-channel filters trained to compute intra-LC explainable features. (**b**) For two sample images, pixel positions that yield large magnitudes when a certain filter is applied are shown. Among the 128 filters, the top four filters are selected, each contributing to an intra-LC feature with the strongest impact on class prediction. The dandelion image illustrates that red mottled patterns (18th filter) in the seed head and green diagonal patterns (24th filter) correspond to the top two features with the greatest impact on the classification decision for this image. (**c**) Flowers in different relational classes show different characteristics. In relational class 1, the majority of images feature yellow patterns, while relational class 4 predominantly displays white flower patterns. The blue dots shown in the images indicate the pixels where non-zero maximums occur for any of the 32 patterns.

Tabular data

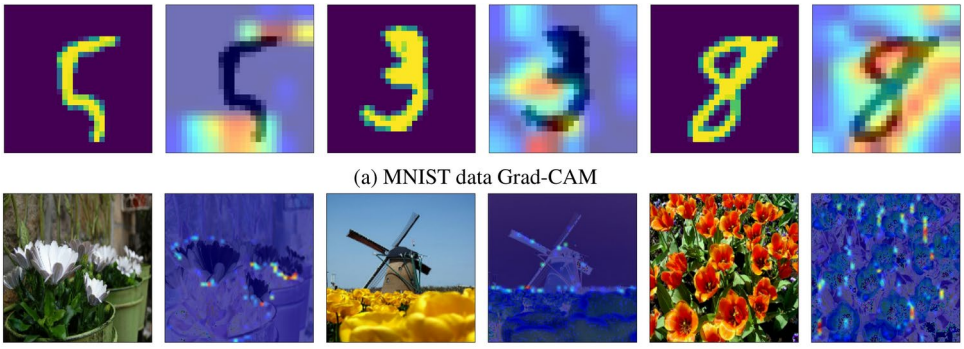
In the case of tabular datasets, SEE-Net's interpretability stems from the linear coefficients attributed to the intra- and inter-LC explainable features, x' and x''. If the original variables possess human-interpretable characteristics, SEE-Net provides straightforward dual-tier explanations by utilizing these variables for both x' and x''. It begins by initially partitioning samples into relational classes based on the original variables and subsequently predicts the target using relational-class-specific linear regressions.

Take SKCM data as an example. As the interpretable NN consists of two layers of linear models, we can first explore the global linear models to identify the most crucial features in determining the relational class assignment. Notably, key features influencing sample allocation into the relational classes include 'ICD O 3 HISTOLOGY,' 'SUBMITTED TUMOR DX DAYS TO,' and 'TISSUE SOURCE SITE0'. Subsequent predictions for the overall survival status of samples in the four relational classes are made using distinct linear models. The impact of the regional linear model can be comprehended by examining the associated regression coefficients.

Discussion and open challenges

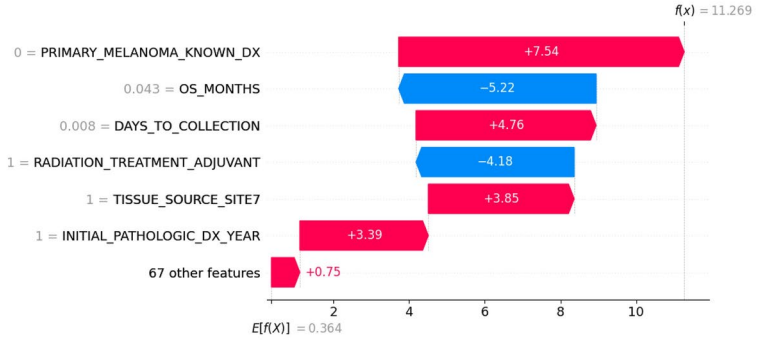
Current post-hoc methods primarily focus on providing explanations for individual data points without revealing the intricate internal workings of the model. In contrast, traditional linear models shed light on the model's internal operations, offering a more comprehensive understanding. In comparison to conventional linear models, SEE-Net introduces an additional explanatory layer for relational classes. This augmentation is straightforward to materialize since the latent relational classes are inferred through linear models, rendering their predictions inherently interpretable.

A key limitation of SEE-Net is its reliance on pre-defined features that are inherently interpretable. For tabular data, this is less of an issue, as original variables often have distinct physical meanings, lending themselves well to interpretation. Notably, SEE-Net is particularly promising for many biomedical datasets, which are typically tabular and demand high levels of explainability. In contrast, for image data, identifying natural explainable features is more challenging. In our research, we adopted small-window filters and image-wide pooling to ensure



(a) MNIST data Grad-CAM

(b) Flower data Grad-CAM



(c) SKCM data SHAP

Fig. 9. Explanation results using alternative methods are shown for sample images: heat maps for pixel importance provided by Grad-CAM[38] in (**a**) and (**b**), and Shapley values[11] for SKCM data in (**c**). In (**a**) and (**b**), all image samples are correctly classified; however, the Grad-CAM results do not offer deep insights into the models' internal workings. Specifically, in (**b**), although both the second and third images are classified as tulips, the Grad-CAM results do not indicate whether they are recognized using different filters or the same filters that capture varying local patterns.

the interpretability of these features. The selection of appropriate techniques for constructing such features can greatly depend on the application at hand.

The exploration of explainable ML via SEE-Net has led us to identify several open challenges that we believe are worth pursuing. We outline these challenges below and discuss potential technical advances that could be achieved if they are addressed.

1. We have not yet quantified the level of "explainability." Although we have proposed four orders of explainability—ranging from white-box models to black-box models—the categorization remains at a broad scale. Unlike prediction performance, which can be straightforwardly evaluated, the quality of an explanation is inherently subjective, application-dependent, and difficult to measure. Developing a numerical measure of explainability could allow us to integrate it into predictive model design and create methods to adjust the level of explainability based on specific application needs. However, attempting to measure explainability might introduce drawbacks, such as undermining the role of human judgment and potentially disrupting the feedback loop between automated learning and domain expert scrutiny.
2. We have observed that prediction accuracy sometimes improves when using more explainable models. We hypothesize that explainability may act as a form of model regularization. Can we develop learning methods that formally leverage explainability to enhance prediction models? Specifically for SEE-Net, could we create a feedback mechanism where the interpretable NN helps improve the guidance NN?
3. In image classification, interpreting predictions often involves visualizing feature maps due to the spatial locality of image features. However, there is a significant gap between visual inspection and natural language interpretation. Bridging this gap is crucial for improving the effectiveness of explanations.

Conclusions

In this paper, we present SEE-Net, a neural network designed to enhance explainability in machine learning models without substantially sacrificing prediction accuracy. SEE-Net supports end-to-end training and integrates an interpretable neural network with a black-box DNN. The latter co-supervises the former, ensuring that high accuracy accompanies explainability. Our experimental results confirm that the predictive power of DNNs can be harnessed to develop models that are both explainable and precise. This suggests a promising new direction for employing DNNs in fields where explainability is crucial. Even in cases where the complexity of a DNN precludes straightforward explanation and thus disqualifies the adoption of the model, it can still serve as a valuable tool for guiding the training of more interpretable models.

Data availibility

The datasets used for this study are publicly available, and their sources are listed below. MNIST handwritten digits (https://yann.lecun.com/exdb/mnist/), Flower Recognition Dataset (https://www.tensorflow.org/datasets/catalog/tf_flowers), CIFAR-10 Dataset (https://www.cs.toronto.edu/~kriz/cifar.html), TCGA Skin Cutaneous Melanoma (SKCM) (https://cran.r-project.org/web/packages/TCGAretriever/index.html), Parkinson's Disease Dataset (PD) (https://archive.ics.uci.edu/dataset/470/parkinson+s+disease+classification).

Received: 1 February 2024; Accepted: 23 October 2024
Published online: 01 November 2024

References

1. Nauta, M. et al. From anecdotal evidence to quantitative evaluation methods: A systematic review on evaluating explainable ai. Preprint at arXiv:2201.08164 (2022).
2. Perez, L. & Wang, J. The effectiveness of data augmentation in image classification using deep learning. Preprint at arXiv:1712.04621 (2017).
3. Wang, F. et al. Residual attention network for image classification. In: Proc. IEEE conference on computer vision and pattern recognition, 3156–3164 (2017).
4. Muzio, G., O'Bray, L. & Borgwardt, K. Biological network analysis with deep learning. *Brief. Bioinform.* **22**, 1515–1530 (2021).
5. Linardatos, P., Papastefanopoulos, V. & Kotsiantis, S. Explainable AI: A review of machine learning interpretability methods. *Entropy* **23**, 18 (2021).
6. Adadi, A. & Berrada, M. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access* **6**, 52138–52160 (2018).
7. Montavon, G., Samek, W. & Müller, K.-R. Methods for interpreting and understanding deep neural networks. *Digit. Signal Process.* **73**, 1–15 (2018).
8. Das, A. & Rad, P. Opportunities and challenges in explainable artificial intelligence (xai): A survey. Preprint at arXiv:2006.11371 (2020).
9. Burkart, N. & Huber, M. F. A survey on the explainability of supervised machine learning. *J. Artif. Intell. Res.* **70**, 245–317 (2021).
10. Ribeiro, M. T., Singh, S. & Guestrin, C. "why should i trust you? " explaining the predictions of any classifier. In: Proc. 22nd ACM SIGKDD international conference on knowledge discovery and data mining, 1135–1144 (2016).
11. Lundberg, S. M. & Lee, S.-I. A unified approach to interpreting model predictions. In: Proc. 31st international conference on neural information processing systems, 4768–4777 (2017).
12. Simonyan, K., Vedaldi, A. & Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. Preprint at arXiv:1312.6034 (2013).
13. Zhao, T. & Wu, X. Pyramid feature attention network for saliency detection. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 3085–3094 (2019).
14. Zhang, J. et al. Uc-net: Uncertainty inspired rgb-d saliency detection via conditional variational autoencoders. In: Proc. IEEE/CVF conference on computer vision and pattern recognition, 8582–8591 (2020).
15. Liu, N., Zhang, N. & Han, J. Learning selective self-mutual attention for rgb-d saliency detection. In: Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition, 13756–13765 (2020).
16. Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. Preprint at arXiv:1409.0473 (2014).
17. Vaswani, A. et al. Attention is all you need. In: Advances in neural information processing systems, 5998–6008 (2017).
18. Zhang, L., Lin, L. & Li, J. Vtnet: A neural network with variable importance assessment. *Stat* **10**, e325 (2021).
19. Tsang, M., Cheng, D. & Liu, Y. Detecting statistical interactions from neural network weights. Preprint at arXiv:1705.04977 (2017).
20. Wu, C., Gales, M. J., Ragni, A., Karanasou, P. & Sim, K. C. Improving interpretability and regularization in deep learning. *IEEE/ACM Trans. Audio Speech Lang. Process.* **26**, 256–265 (2017).
21. Burkart, N., Huber, M. & Faller, P. Forcing interpretability for deep neural networks through rule-based regularization. In 2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA), 700–705 (IEEE, 2019).
22. Wu, M., Parbhoo, S., Hughes, M. C., Roth, V. & Doshi-Velez, F. Optimizing for interpretability in deep neural networks with tree regularization. *J. Artif. Intell. Res.* **72**, 1–37 (2021).
23. Jacobs, R. A., Jordan, M. I., Nowlan, S. J. & Hinton, G. E. Adaptive mixtures of local experts. *Neural Comput.* **3**, 79–87 (1991).
24. Ebrahimpour, R., Kabir, E., Esteky, H. & Yousefi, M. R. A mixture of multilayer perceptron experts network for modeling face/nonface recognition in cortical face processing regions. *Intell. Autom. Soft Comput.* **14**, 151–162 (2008).
25. Tang, B., Heywood, M. I. & Shepherd, M. Input partitioning to mixture of experts. In Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290), vol. 1, 227–232 (IEEE, 2002).
26. Seo, B., Lin, L. & Li, J. Mixture of linear models co-supervised by deep neural networks. *J. Comput. Graph. Stat.* **31**, 1303–1317 (2022).
27. Sharma, S., Henderson, J. & Ghosh, J. Feamoe: fair, explainable and adaptive mixture of experts. Preprint at arXiv:2210.04995 (2022).
28. Ismail, A. A. et al. Interpretable mixture of experts. Preprint at arXiv:2206.02107 (2022).
29. Vasić, M. et al. Moët: Mixture of expert trees and its application to verifiable reinforcement learning. *Neural Netw.* **151**, 34–47 (2022).
30. Chen, Z., Deng, Y., Wu, Y., Gu, Q. & Li, Y. Towards understanding mixture of experts in deep learning. Preprint at arXiv:2208.02813 (2022).
31. Maddison, C. J., Mnih, A. & Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. Preprint at arXiv:1611.00712 (2016).
32. Louizos, C., Welling, M. & Kingma, D. P. Learning sparse neural networks through l_0 regularization. Preprint at arXiv:1712.01312 (2017).
33. Sakar, C. O. et al. A comparative analysis of speech signal processing algorithms for parkinson's disease classification and the use of the tunable q-factor wavelet transform. *Appl. Soft Comput.* **74**, 255–263 (2019).
34. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. Preprint at arXiv:1412.6980 (2014).
35. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **86**, 2278–2324 (1998).
36. Liu, Z. et al. A convnet for the 2020s. In: Proc. IEEE/CVF conference on computer vision and pattern recognition, 11976–11986 (2022).
37. Howard, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. Preprint at arXiv:1704.04861 (2017).
38. Selvaraju, R. R. et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proc. IEEE international conference on computer vision, 618–626 (2017).

Acknowledgements
Beomseok Seo's research is supported by the Sookmyung Women's University research grant (1-2403-2025). Jia Li's research is supported by the National Science Foundation under grant CCF-2205004.

Author contributions
Both authors conceptualized the framework, analyzed the results, and wrote parts of the manuscript. B.S. implemented the code and conducted the experiments.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Supplementary Information The online version contains supplementary material available at https://doi.org/10.1038/s41598-024-77507-2.

Correspondence and requests for materials should be addressed to B.S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.