



An object-oriented neural representation and its implication towards explainable AI

Enoch Arulprakash¹ · A. Martin¹

Received: 3 February 2023 / Accepted: 18 August 2023 / Published online: 14 September 2023

© The Author(s), under exclusive licence to Bharati Vidyapeeth's Institute of Computer Applications and Management 2023

Abstract Rapid dissemination of Artificial Intelligence (AI) and machine learning in real-world problems has raised a concern about the reliability aspects of the model. A separate sub-branch of AI solicitudes on reliability is known as Explainable AI (XAI). XAI analyses the cause and impact of the decision made by AI systems. The neural network plays a significant part in AI's ability to upgrade with more recent data. However, the learning capability left the model obscure for most of its decisions. Breaking the black-box nature of the neural network model and giving understanding and insight into the functionality of the model will mitigate the uncertainty. Here, in this research, we have designed Object oriented neural representation to devise a “*Feature importance*” technique from the correlation between Loss and Weight distribution devised method is effective in interpreting the decision to the end user and AI practitioners with optimal time complexity ($TC = (L-1) \times (E \times C)$). The proposed neural representation also extended to incorporate domain/business rules from which we introduced a new Loss, known as business loss. From getting the impact of business loss, we obtained an earlier decline in the overall Loss and improved performance from the ablation study.

Keywords Explainable AI · Black Box · Domain Knowledge · Feature importance · Interpretability

1 Introduction

Artificial Intelligence is driving technological innovation to new dimensions that were once unimaginable, and various sectors have proven its supremacy [1, 2]. Neural Networks (NNs), especially, are a predominant part of AI due to their ability to infer complex patterns from data by passing through the layers of Artificial Neural Networks (ANNs) [3–5]. Hidden layers are responsible for learning patterns from the data, allowing ANNs to acquire experience with more data [6]. In contrast to rule-based systems, ANNs are known as Universal Function Approximators [7]. However, this raises significant concerns about the mapping learned in the hidden layers, as it lacks proper causes and consequences behind the decision. Therefore, the internal working mechanism of ANNs is generally known as a black box. The black box indicates a model whose inner workings are uninterruptible and unexplainable, which is a significant issue for end-users, AI practitioners, and researchers.

Even though renowned frameworks like TensorFlow, PyTorch, etc., suffice the requirements of practitioners and researchers using abstract methods (i.e., `model.fit()`), it is a concern for Explainable AI (XAI) since these abstract methods outline the parameters to be passed and mask the implementation details. XAI attempts to disclose the concrete nature of neural network systems by reasoning the causes and reasons for decisions using various XAI methods [8, 9]. XAI is a sub-branch of AI that is new and evolving.

The main objective and contribution of our work:

- We have used an object-oriented neural representation to study feature importance through a cyclic learning sequence and to enable the easy incorporation of business parameters into a neural network.

✉ Enoch Arulprakash
enocharulprakash03@gmail.com

A. Martin
martin@cutn.ac.in

¹ Department of Computer Science, Central University of Tamil Nadu, Thiruvavur 610104, India

- We utilized the extensibility property of Object Oriented Modelling (OOM) to incorporate business parameters using object-oriented design and standard design patterns. This enabled us to introduce a new loss function called the 'business loss' and analyze its impact during training.
- We avoided using abstract methods and instead used primitive types and custom-built methods based on differential calculus to train the neural network
- We evaluate the proposed Object Oriented model using a nonlinear XOR and XNOR function.
- Finally, with our proposed model, we attempted to answer one of the most transformative questions of the decade: from how machines work to how machines learn?

The paper is organized as follows: Sect. 2 describes the previous work. Section 3 reviews the literature. Section 4 covers the significance of feature importance using cyclic learning sequence. Section 5 focuses on the design of the proposed Object-Oriented Neural Network model. Section 6 provides the derivation for training the neural network. Section 7 describes the design of the algorithms. Section 8 describes implementation details. Section 9 conducts experiment. Section 10 presents the results and discussion. Finally, Sect. 11 draws the conclusions.

2 Previous work

In our earlier work, we reviewed deep neural networks (DNNs). DNNs automatically extract features for prediction without any hand-crafted feature engineering methods. More complex problems can be represented in DNNs. Increasing the number of layers and nodes substantially increase the potential for representing complex problems. We also reviewed challenges in different circumstances and security concerns. Techniques and defensive strategies have evolved to compensate for those challenges, maintain performance, and provide a secure environment [10, 11]. DNNs have grown exponentially in the past decade. However, the automatic learning capability has left the NN system uninterpretable and unexplainable to end users/domain experts, AI practitioners, and researchers [12, 13].

3 Literature review

Various techniques have been developed to explain the internal functioning of opaque NN models for decision making. Methods such as surrogate models, gradient-based techniques, undiscovered smooth functions, and model-specific methods have evolved. AI developers and researchers can gain insight into the inner workings of NNs using feature

importance from XAI methods. Visual Analytics (VA) researchers adopt XAI methods in visual analytics to give a far-enhanced interpretation and understanding of the neural network system to end-users [13, 14].

Commonly in the surrogate technique, a local instance is selected for perturbation to frame similar instances from the impact of the perturbation. The surrogate model is simple and easy compared to the actual model, which can be easily discerned with linear relations, decision trees, feature scores, and decision boundaries. Representative methods include LIME [15], SHAP [16] [17], and ANCHOR [18]. Each method uses a different perturbation technique to determine the feature's importance. LIME explains the linear model by discerning the positive and negative impact of the feature on the result. SHAP considers similar features as a team that contributes to the decision process, similar to game theory. On the other hand, the ANCHOR method selects the ANCHOR region from the feature space such that changes outside the ANCHOR will not affect the anchor region, using IF-THEN rules. ANCHOR creates the decision boundary for all the decisions in the feature space.

The gradient-based technique calculates feature importance based on gradients. Representative methods include Saliency Map, Class Activation Mapping (CAM) [19], GRAD-CAM [20], Deep LIFT [21], Integrated Gradients, and Layer-Wise Relevance Propagation (LRP) [22]. Saliency Map produces a detailed heat map to show the feature's importance to end-users, indicating which part of the data has been referred to for the decision. CAM uses average global pooling in the deep layer to produce feature importance. However, the CAM technique is not scalable. Grad-CAM addressed the drawback of CAM by using a gradient of any class technique to highlight the particular contribution. Integrated Gradients is similar to DeepLIFT but computes the gradient differently by averaging a linear path between the referred and selected input. LRP works based on conservation of relevance; propagated relevance for the decision is preserved and transmitted back to the lower layer along with the prediction to study the feature importance with the magnitude of neuron activation in each layer. Bayesian Rule List uses static pre-defined rules to reduce the feature space and makes it more interpretable with those rules by using simple IF-THEN rules.

Model-specific methods include GAM [23] and Mean Decrease Impurity (MDI) [24]. GAM provides feature importance by inferring the undiscovered smooth function. The inferred smooth functions are aggregated to demonstrate the reason behind the decision. MDI uses a decision tree-based approach to obtain feature importance; it tries to find a point to split the Node to form a tree-like hierarchy. However, GAM and MDI are model-specific. Significant methods are depicted with a timeline in Fig. 1. Furthermore, feature distributions are analyzed from various



Fig. 1 Evolution of the XAI method with a timeline

central data tendencies, such as mean, median, mode, sorting, binning, and a combination of two or more methods, to interpret the weight distribution to the end-users and AI practitioners [25]. Moreover, researchers and experts suggest incorporating domain/business rules into neural network systems [26–28].

We have reviewed significant contributions from XAI methods. Our focus is to devise a feature importance method that is easily interpretable by the end-user and adaptable to Visual Analytics, as depicted in Fig. 2. Along with the feature importance method, we have extended the study to incorporate domain/business rules into a neural network system based on suggestions from experts and researchers in recent studies and analyses [26–28].

4 Significance of the method (methodology)

We have reviewed significant XAI techniques, such as gradient implication, undiscovered smooth function, model-specific surrogating, and others [15–25]. These methods focus on weight distribution alone for feature importance. We approach feature importance from a novel perspective. We take a cyclic learning sequence that starts from the prediction, based on how close the prediction is to the expected outcome. The Loss is impacted (the Loss is a derivative of the L2 function for the output nodes, and for non-output nodes, it is a propagated gradient to the node level). The Loss, in turn, affects the weights, as illustrated in Fig. 3a. Every neuron and connection weight must undergo these cyclic steps repeatedly to extract the necessary features for

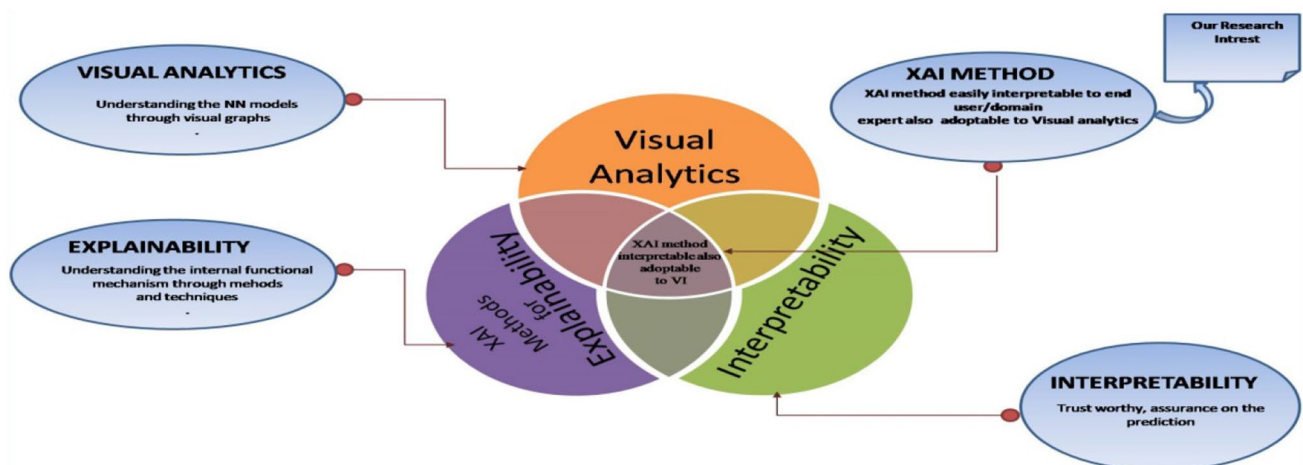


Fig. 2 The focus of our research

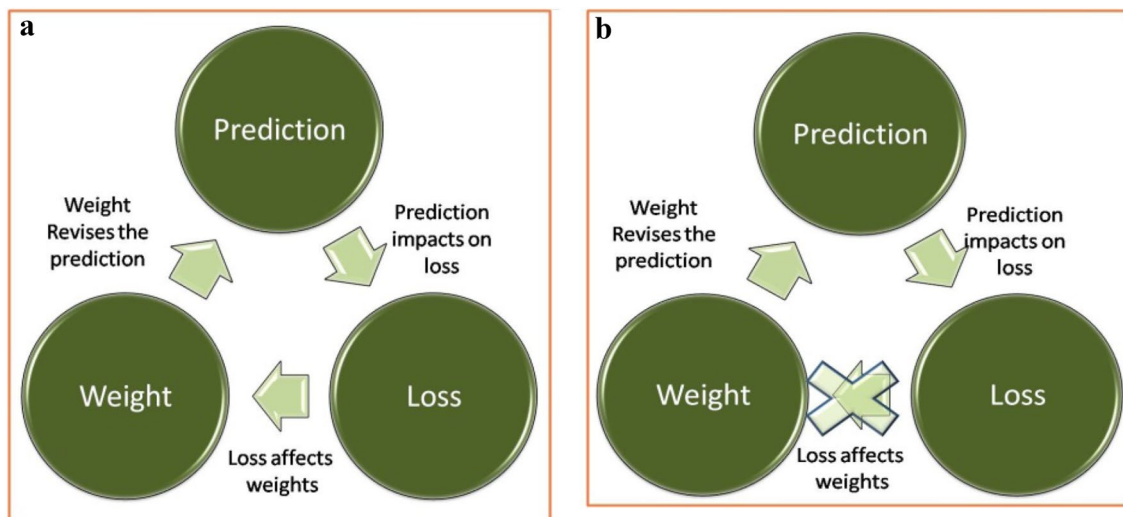


Fig. 3 a. Cyclic Learning Sequence, b Neurons and connection weights loses their connection when not adhered to the Loss

prediction. When the connection weights renew their magnitudes and directions in connection with the propagated Loss, the prediction moves closer to the expected result.

However, not all neurons and connection weights follow the loss direction; some may miss the track and get stuck in local minima, as shown in Fig. 3b. Nevertheless, those neurons and connection weights that follow the Loss play a significant role in contributing to the prediction [29]. It is impossible to determine those neurons and connection weights that renew according to the loss direction before training because the learning process is dynamic for the neural network. However, after training, as a post-hoc explanation, analyzing weight distribution along with loss distribution can provide insight into the study by effectively measuring the degree of association between Loss and weight distribution using the correlation matrix, as given in Eq. (1), for feature importance [30–34].

$$r = \frac{n(\sum XY) - (\sum X)(\sum Y)}{\sqrt{[n \sum X^2 - (\sum X)^2][n \sum Y^2 - (\sum Y)^2]}} \tag{1}$$

The correlation coefficient measures the correlation between two variables, and *r* ranges from -1 to $+1$. When the correlation coefficient reaches the two extreme ranges, i.e., -1 or 1 , it implies a strong correlation or contribution. Being away from the range indicates less contribution, while being close to 0 means very little contribution [35].

After analyzing the importance of loss parameters for feature importance from the cyclic learning sequence, another purpose of our study is to incorporate domain/business knowledge into the neural network. These aspects require a novel neural representation that goes beyond the tensor

representation and can satisfy the requirements. The neural representation must combine relevant parameters such as neuron value, Loss, and connection weights so that loss and connection weight are bound together to compute feature importance through a correlation matrix [36, 37].

We have chosen Object-Oriented Modeling (OOM) for the neural network representation as it best fits the requirements. OOM has implicit encapsulation and extensibility properties, which make it convenient to combine relevant parameters. Extensibility can play a crucial role in the effortless incorporation of new parameters, allowing domain/business-specific parameters to be incorporated into the neural network model. Therefore, we have provided a representation to combine relevant parameters and an easy plug-in to incorporate domain/business-specific parameters into a neural network using OOM and following the design principles of the interpreter, filter, and factory patterns [36, 37].

In backpropagation, the goal is to minimize the error or loss by adjusting the weights in the neural network. This is done by repeatedly computing the gradient of the loss function with respect to the weights and adjusting the weights in the direction of the negative gradient. This process is repeated iteratively until the loss function converges to a minimum value, indicating that the neural network has learned to predict the output.

Whereas, in the proposed feature importance method, the loss is not used to minimize a function. Instead, loss is considered in harmony with co-related parameters such as prediction and weight, as part of a cyclic learning sequence. This cyclic property is used to derive feature importance by evaluating whether a particular feature is adhering to the propagated loss or stuck in a local minimum without contributing significantly to the prediction.

To determine feature importance, the proposed method calculates a correlation coefficient value between the loss and connection weight using a correlation equation. By using this correlation equation, the proposed method is able to determine which features are important for making predictions or decisions and which features are not contributing significantly. This allows for a more transparent and understandable prediction process, providing assurance over the result of the predictions.

Furthermore, the proposed model has been expanded to utilize its object-oriented extensibility property, making it capable of easily integrating domain or business rules. Another important aspect of the proposed model is that it does not rely on conventional abstract methods like model.fit() for training the neural network. Instead, it employs a differential calculus-based process to derive equations that facilitate the network’s training. This approach adds transparency by identifying the key factors that contribute to the machine learning.

5 Proposal of an object-oriented neural representation model

In this section, we propose a model for the proposed methodology.

We inspect neural networks from an object-oriented perspective to provide an enhanced representation. The neural network is an arrangement of nodes and layers. Each layer consists of nodes, and the nodes from a layer are connected to the successive layer. From a conceptual background, the Node is the most primitive encapsulation that binds value, Loss, and connection weights to the successive layer. We have modeled Node with these highly cohesive attributes. Connections from one Node to other nodes are established through a Map: a key-value pair [Map <Node, WeightPair>]. Designing the Node class with highly relevant and cohesive parameters is essential to find feature importance through loss and connection weight correlation. The layer is a composition of a list of nodes, and a list of layers constitutes a baseline neural network, as depicted in the class diagram in Fig. 4 [38–40].

In addition to the baseline neural network model, we utilize the extensibility property of object-oriented programming to incorporate a business or domain-specific rules into the baseline representation. We have derived supplemental classes from the baseline ANN: “ExtendedNodeFunction” and “ExtendedLayerFunction” are inherited from the base Node and Layer classes. “ExtendedNodeFunction” inherits the node class and contains objects to assert the result with the business rules. Thus, the derived class provides composition to embed business or domain rules along with primitive

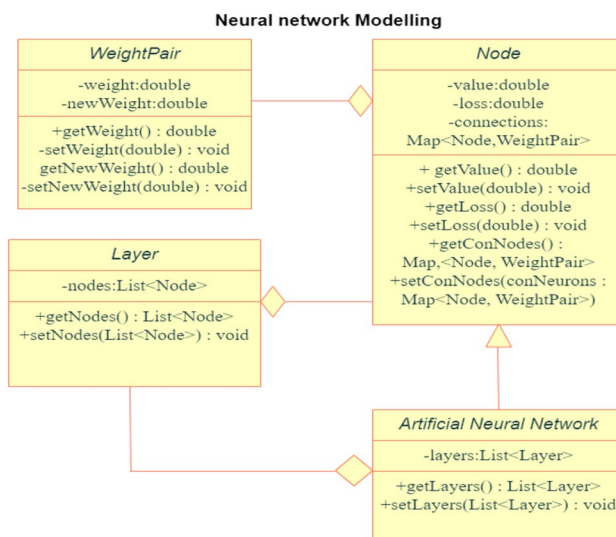


Fig. 4 Class diagram for baseline Artificial Neural Network

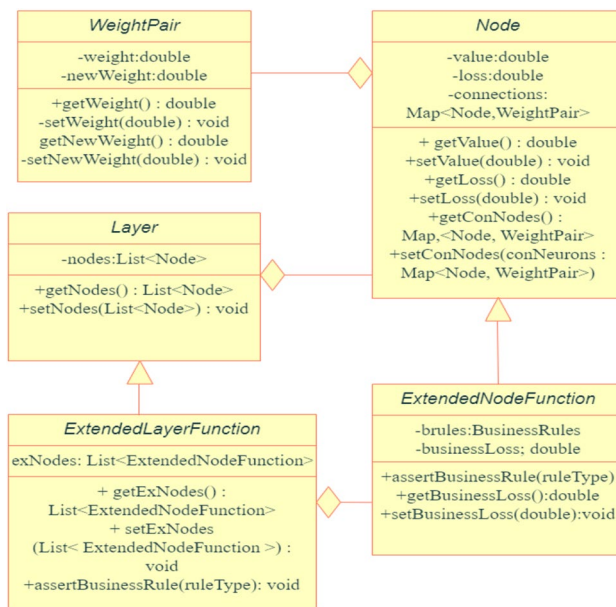


Fig. 5 Base classes are extended for domain/business rules

neural network properties, as depicted in the class diagram in Fig. 5. The incorporated business rules are specific to output layers and nodes [38–40].

Business or domain rules are unique and specific to a particular problem. However, we tried to provide a standard template flexible to any problem domain by adopting the standard design principles from interpreter, filter, and factory patterns [38–40].

From the Business Rule template, as shown in Fig. 6, the business rule is a public interface with an assert method. All the concrete classes implementing the base template or

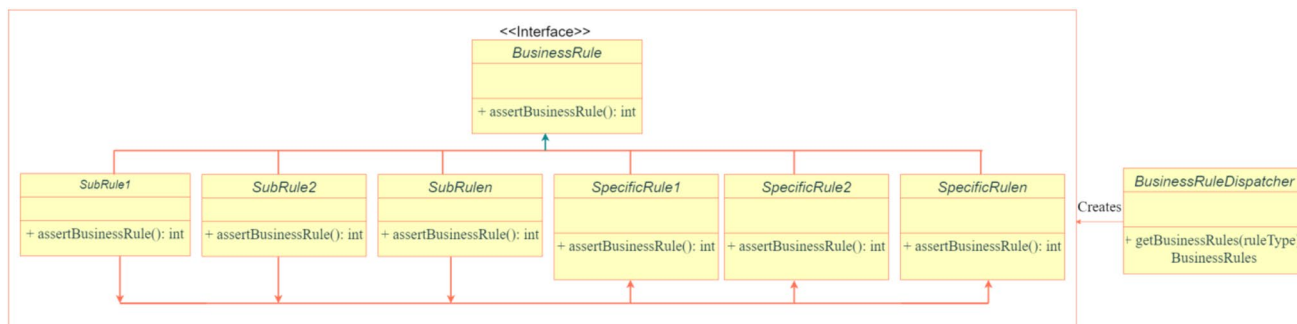


Fig. 6 General template class diagram for Business Rules following interpreter, filter, and factory patterns

“BusinessRules” interface have to provide a specific implementation in the assert method. “SpecificRule” classes use sub-rule classes to reach their objective. Each neural network output node is associated with the specific rule to assert with the business rule, which is “SpecificRule” from the business template [38–40].

For our experiment, we have taken XOR and XNOR functions; the business rule Module for XOR is represented in Fig. 7.

XOR and XNOR use the subrules(ComplementRule, AndRule, OrRule) to build their specific XOR and XNOR Rules, respectively, as depicted in Fig. 7.

“BusinessRuleDispatcher” acts as a dispatcher for business rules, providing the appropriate “SpecificRule” instance to the extended node function, as shown in Fig. 7. When there is a list of rules, the business dispatcher provides an appropriate rule for the extended Node by examining its type [38–40].

We designed a baseline neural network model for feature importance and then extended base functionality for incorporating domain/business rules. Therefore, the complete representation of the proposed model is an integration of extended neural representation and business rule model, as presented in Fig. 8.

During the training, the extended class obtains a business rule from the dispatcher, and the result is asserted with a business rule to obtain business loss. Obtained business loss is combined with L2 Loss.

We have designed a neural network in such a way as to embed appropriate business rules dynamically during the training process and combine the asserted business loss with the regular L2 Loss as shown in Fig. 9, [26–28]. Integrating the business loss assists the entire learning process by giving additional semanticity of the domain to the neural network, apart from the general input–output mapping representation. We will analyze the impact of the business loss in the experiment section.

6 Derivation for neural network training

To train the neural network, abstract methods such as model. fit() are not used. Instead, formulas have been derived for end-to-end training.

Consider a neural network learning the function $f(x) = Y$. The target function is learned through a series of hidden layers, where each layer acts as a sub-function that maps relevant features to the target function $f(x) = Y$. Successive layers learn more relevant representations by

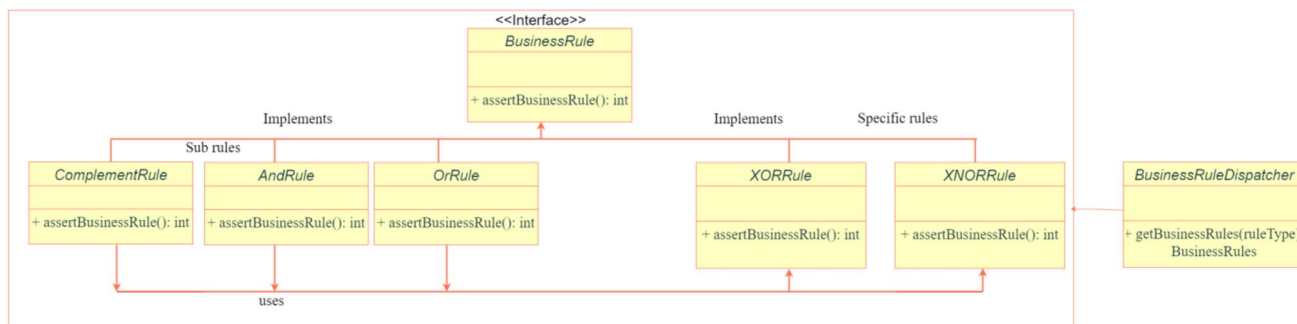


Fig. 7 Business rules template for XOR and XNOR functions

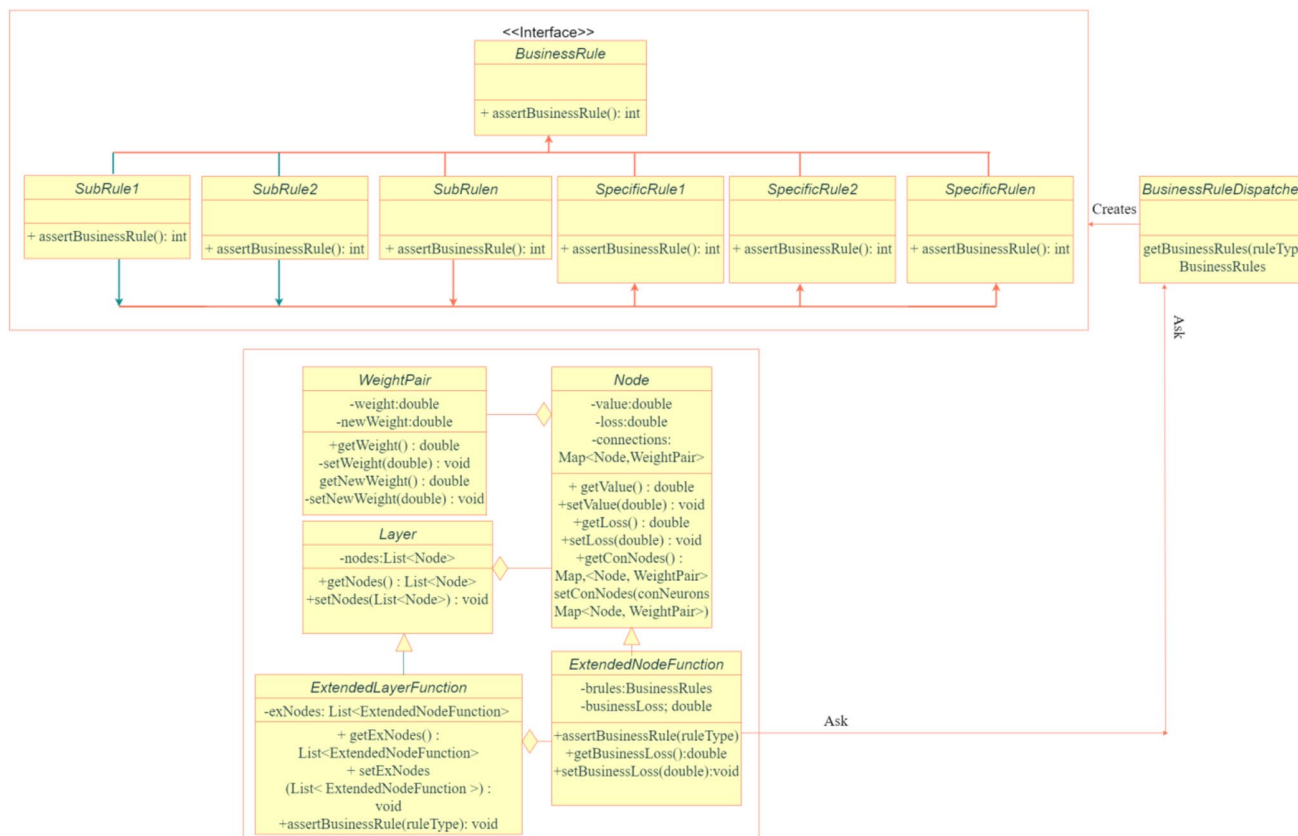


Fig. 8 Object-oriented class diagram representing the proposed model after combining neural network and business model



Fig. 9 Total Loss = L2 + Domain/Business Loss

obtaining features from the previous layer. As a result, the neural network approximates the target function $f(x)$ by composing multiple functions, as shown in Fig. 10 and Eq. (2), [34, 41–44].

$$g(x) = Y^l = f_4(f_3(f_2(f_1(x, w^1), w^2), w^3)) \tag{2}$$

The output $\sum_{i=1}^N Y_i^l$ from a neural network can be approximated as a sum of the products of the node values and their connected weights, each passed through an activation function across all the nodes in the network. This approximation is obtained by a process called forward propagation, which is represented in Eq. (3).

Formally defined as,

$$\sum_{i=1}^n Y_i^l = \sum_{i=2}^n \left(\sum_{j=1}^n \left(\sigma \sum_{i=2}^n X_i^l \times W_{ij}^l \right) \right) \tag{3}$$

The difference between the desired and obtained output is the error that needs to be minimized. The Mean Square function, $[E = \frac{1}{2M} \sum_{i=1}^N (Y^l - Y)^2]$ is considered for this purpose. Based on the loss value, the weights $[w^1 - w^3]$ are updated to reach the target function (x). The sensitivity or derivative of the loss function provides the magnitude and direction in which the weights have to be changed, as shown in Eq. (4)

$$E^l = \sum_{i=1}^n (Y_i - Y_i^l) \left(Y_i^l (1 - Y_i^l) \right) \tag{4}$$

E^l is the derivative of the loss function along with the derivative of the sigmoid function.

E^l is back propagated through intermediate layers. Partial derivative $\frac{\partial E^l}{\partial w^4}$ gives the sensitivity at which the weight w^4 has to be adjusted.

Similarly, for the other layers, partial derivatives are given as

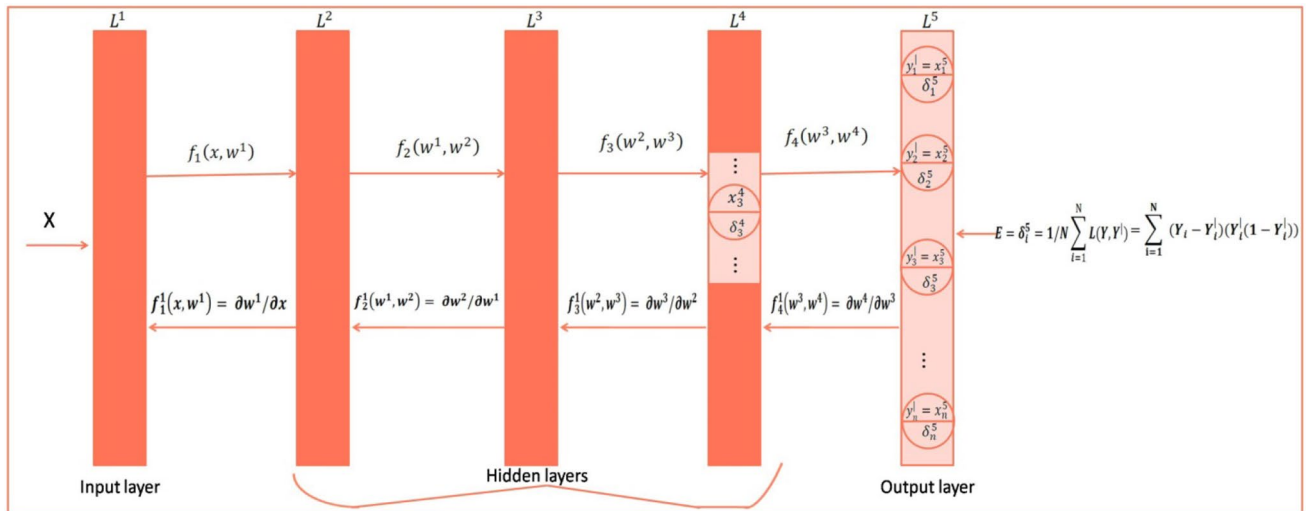


Fig. 10 Layers of the neural network representing the composition of functions

$$\frac{\partial E^1}{\partial w^4}, \frac{\partial w^4}{\partial w^3}, \frac{\partial w^3}{\partial w^2}, \frac{\partial w^2}{\partial w^1}, \frac{\partial w^1}{\partial x}$$

From the Jacobean composition of function, the derivate of composition (layer) is the product of the partial derivative.

$$\frac{\partial E}{\partial w^1} = \frac{\partial E}{\partial w^4} \times \frac{\partial w^4}{\partial w^3} \times \frac{\partial w^3}{\partial w^2} \times \frac{\partial w^2}{\partial w^1} \times \frac{\partial w^1}{\partial x} \tag{5}$$

Derivative of the output layer is the direct derivative of the loss function (MSE) Therefore, $\sum_{i=1}^N \delta_i^5$ is obtained from Eq. (4).

$$\text{Has } \sum_{i=1}^N \delta_i^5 = \sum_{i=1}^n (Y_i - Y_i^l)(Y_i^l(1 - Y_i^l))$$

For the f_4 layer is composed by $f_5: f_5(f_4, w^4)$

From the Jacobean property derivative of f_4 is:

$$\sum_{i=1}^n \delta_i^5 \times \frac{\partial f}{\partial w} \tag{6}$$

Partial derivative of f_4 w.r.t. connection weight $\frac{\partial f_4}{\partial w} = x$
In General, all the nodes in L^4 can be represented as

$$\sum_{j=1}^n \sum_{i=1}^n \frac{\partial f_4}{\partial W_{ji}^4} = \sum_{j=1}^n \sum_{i=1}^n \delta_i^5 \times x_j^4 \tag{7}$$

Gradient descent to change the connection weight w^4 is

$$\sum_{j=1}^n \sum_{i=1}^n W_{ji}^4 = \sum_{j=1}^n \sum_{i=1}^n \left(W_{ji}^4 - \eta \times \frac{\partial}{\partial W_{ji}^4} \right) \tag{8}$$

where η is the learning rate.

Similarly, for the remaining non-output layers L^4 to L^1 , partial derivatives with respect to $[w^3-w^1]$ are required to be transferred from the successive layer, with a magnitude of connection weight [34, 41–44].

For the nodes in L^4 is obtained as

$$\sum_{j=1}^n \delta_j^4 = \sum_{j=1}^n \sum_{i=1}^n (\delta_i^5 \times w_{j,i}^4) \times x_j^4 (1 - x_j^4) \tag{9}$$

In General, for the non-output layer $[\delta^4-\delta^2]$ is computed as

$$\sum_{l=1}^{n-2} \sum_{j=1}^n \delta_j^{n-l} = \sum_{l=1}^{n-2} \sum_{j=1}^n \sum_{i=1}^n (\delta_i^{n-l} \times w_{j,i}^{n-l-1}) \times (x_j^{n-l-1} (1 - x_j^{n-l-1})) \tag{10}$$

Equation (10) gives δ for non-output layers $[\delta^4-\delta^2]$ from this, we expand the weight update for all the layers of the neural network $[w^1 - w^4]$ [34, 41–44].

Therefore, Eqs. (7 , 8) are also extended for the non-output layer using Eq. (10) as

$$\sum_{l=0}^{n-2} \sum_{j=1}^n \sum_{i=1}^n \frac{\partial}{\partial W_{ji}^{n-l-1}} = \sum_{l=0}^{n-2} \sum_{j=1}^n \sum_{i=1}^n \delta_i^{n-l} \times x_j^{n-l-1} \tag{11}$$

$$\sum_{l=1}^{n-1} \sum_{j=1}^n \sum_{i=1}^n W_{ji}^{n-l} = \sum_{l=1}^{n-1} \sum_{j=1}^n \sum_{i=1}^n (W_{ji}^{n-l} - \eta \times \frac{\partial}{\partial W_{ji}^{n-l}}) \tag{12}$$

7 Algorithms

7.1 Part a—Algorithm for training the proposed model

This section provides an algorithm for training neural networks and recording the Loss and weight instances for correlation analysis between Loss and weight.

PART A- ALGORITHM 1: Algorithm for training the proposed model*INPUT* : *input_file, epochs, batch_size, model, use_business_loss, business_rule_type**OUTPUT* : *Loss and weight instances of neural networks.***A. Train for every epoch****FOR EACH** *epochIndex* **FROM** *epochs* **DO**:

```

a. Initialize_input(model, batchIndex)
b. ForwardFlow:
LayerList←model.getLayers()
FOR EACH Layer IN LayerList DO:
    PrevList←getPrevNodes(Layer)
    NodeList←getNodes(Layer)
    FOR EACH Node IN NodeList DO:
        ConnNodeValFromPrevNode(Node,PrevList,batchIndex)
        //above line computes equation 3 from section 6
        IF Layer is LastLayer THEN
            FOR EACH Node IN NodeList DO:
                ExtendedNode.businessLoss←
                assetNodeWithBusinessRule(ExclusiveNode,business_rule_type)
            END FOR
        END IF
        NNParameters←getInstanceOfNodeAndConnection(Node)
        // above line of code stores loss and weight for feature importance
    END FOR
END FOR

```

c. BackwardFlow :

```

ExtendedNodeList←getLastLayer(LayerList)
FOR EACH enode IN ExtendedNodeList DO:
    actualOutput← extractOutputFromFile(Input_file, batchIndex)
    L2_Loss←computeRegressionLoss(enode,actualOutput)
    //Computes derivative at the output layer which is equation 4 from section 6
    IF use_businessLoss IS true THEN
        enode.delta← L2_Loss + enode.BusinessLoss
        // above line combines business loss L2 computed in ForwardPass with
    ELSE
        enode.delta← L2_Loss // line computes only L2 loss
    END IF
END FOR
//Transfers the computed derivative to other layers in reverse order
FOR EACH Layer IN LayerList TRAVERSE -IN REVERSE-ORDER :
    NodeList←getNodes(Layer)
    FOR EACH Node IN NodeList DO:
        ComputePropagatedLoss(Node,Conn← Node.Map)
        // above line computes equations 11 from section 6
    END FOR
END FOR
B. Updating with new weights at end of each epoch
FOR EACH Layer IN LayerList DO:
    NodeList←getNodes (Layer)
    FOR EACH Node IN NodeList DO:
        updateWeights(Conn ← Node.Map)
        //above line computes equation 12 from section 6.
    END FOR
END FOR

```

END FOR

PART B - Algorithm 2 : ALGORITHM COMPUTES CO-RELATION CO-EFFICIENT BETWEEN LOSS AND WEIGHT DISTRIBUTION FOR FEATURE IMPORTANCE

// The algorithm computes the correlation coefficient between Loss and weight distribution

INPUT :NNparameters - a Map containing neural network instances computed in Algo 1.

OUTPUT : A correlation coefficient matrix between Loss and weight distribution

Row \leftarrow 0 // Initialize to 0

FOR EACH delta IN deltaList FROM NNparameters EXCEPT inputLayer DO:

// in the above line delta is a propagated loss

conectionList \leftarrow getConnection(delta) //return list of connected weight

Column_num \leftarrow 0 // Initialize to 0

FOR EACH connection IN conectionList DO:

Corr \leftarrow ComputeCorrelationCoefficient(delta,connection)

// above line computes eqn(1) from section 4

CorrMatrix[Row][Column_num] \leftarrow Corr

Column_num \leftarrow Column_num+ 1

END FOR

Row \leftarrow Row + 1

END FOR

Return CorrMatrix

8 Implementation

We have implemented the proposed object-oriented neural network representation in Java using the Eclipse IDE. We designed the proposed model to be scalable and flexible, allowing us to add or remove layers and nodes as required. For graphical plots, we utilized the XChart open-source tool.

9 Experiment

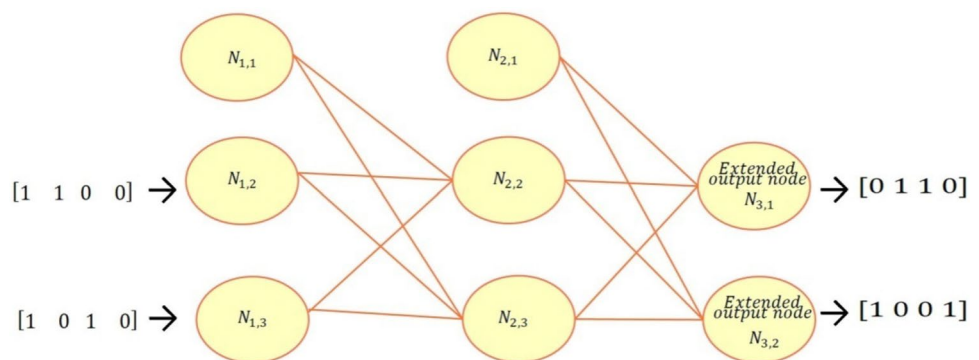
We have utilized an ANN feed-forward neural network that learns XOR and XNOR functions with three layers to experiment with the proposed feature importance technique. We followed the naming convention of prefixing with “N” and including the column and row numbers. For instance, “ $N_{1,1}$ ” represents the first Node in the first layer. Generally, “ $N_{x,y}$ ” represents the node in layer “x” and position “y,” as depicted in the feed-forward neural network in Fig. 11.

The experiment was carried out for 3000 epochs, using a learning rate of 0.1 and a batch size of 04 for the input

combinations (0,0), (0,1), (1,0), and (1,1). The Loss and weight distributions were analyzed to determine the feature importance through Loss and weight correlation based on the cyclic learning principle.

We took the neurons and their corresponding connection weight distribution from the training and analyzed their correlation on a graph for two different circumstances: when the model fit the training set with the least RMSE error, and when the model did not fit the training set with considerable RMSE error. In the properly fit model, the neuron and connection weights adhered to the propagated Loss, as we discussed in Sect. 4. Therefore, we observed a clear spike or dip in the loss distribution, and the distribution of the corresponding weights learned by moving towards a converging point for most of the neurons and connection weights, as shown in Fig. 12a, b. Similarly, for the underfit model, most of the neuron and connection weights did not adhere to the propagated Loss. Therefore, a clear spike or dip in the loss distribution was unseen, the Loss never decreased throughout the learning, and the weight distribution was barely a straight line (unlearned), as shown in Fig. 12c, d.

Fig. 11 A neural network trained to perform XOR and XNOR functions



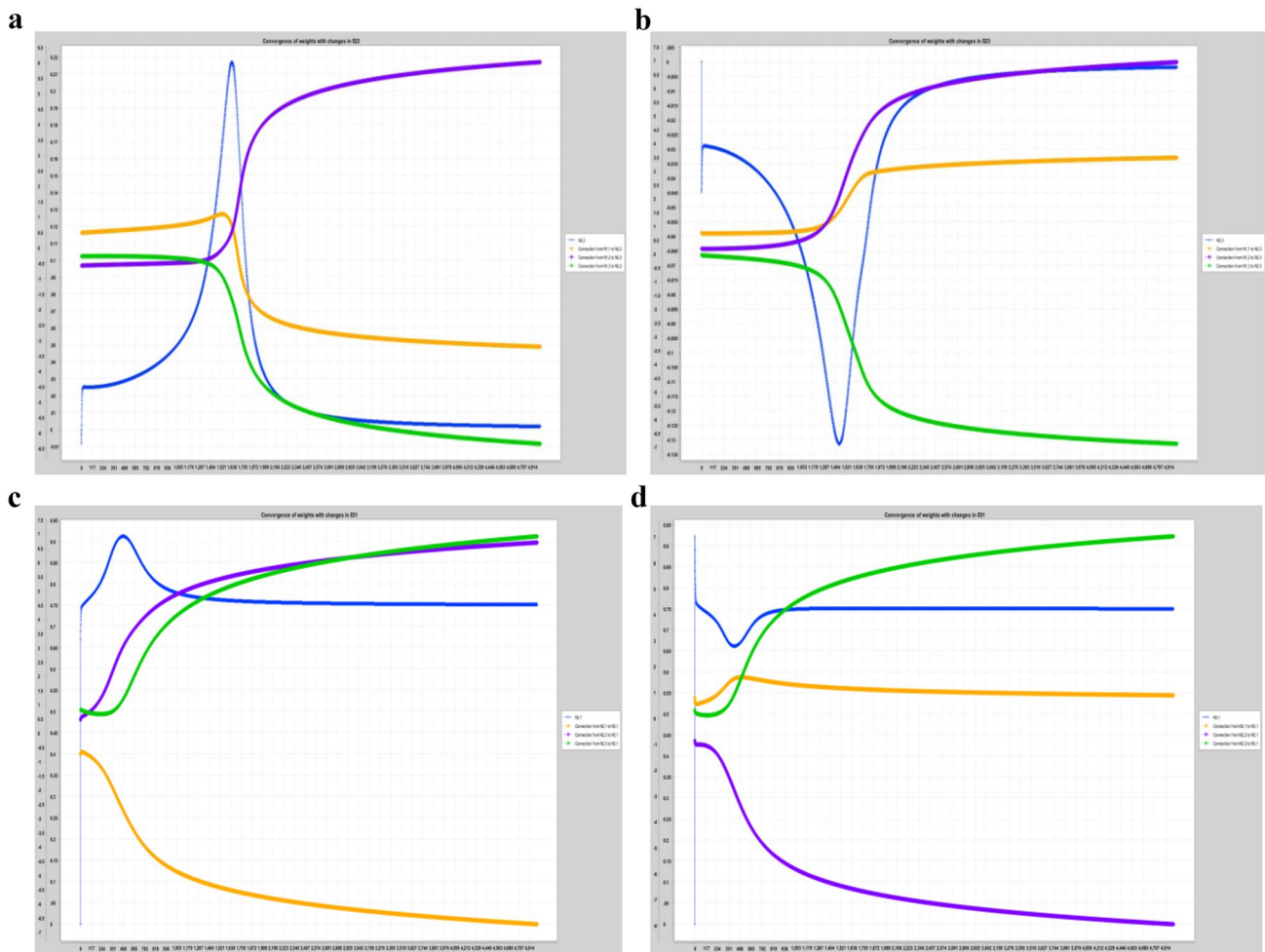


Fig. 12 a Neurons and connection weight that follow Loss (apparent spike), b Neurons and connection weight that follow Loss (apparent Dip), c Neurons and connection weight that does not adhere to Loss

(misses clear spike), Fig. 12 d. Neurons and connection weight that does not adhere to Loss (misses clear dip)

Note: In the Fig. 12, blue curve represents loss distribution, while the other curves, such as green, yellow, and purple, represent the weight distributions. Furthermore, to measure the correlation between all the connection weights and the propagated Loss, we used the correlation coefficient as discussed in the methodology section, which is calculated using Eq. (1).

To determine feature importance, we measured the correlation coefficient of all the connection weights in the neural network across multiple trials.

Table 1 describes the feature importance derived from the correlation between Loss and weight distribution. Since the loss curve can either be a dip or a spike, the correlation values can be either positive or negative. We found that the correlation coefficient for all the connection weights was maximum across multiple trials, indicating that this is the optimal neural network configuration for learning the XOR and XNOR functions. To obtain varying levels of feature

importance, we added an additional layer to the neural network, as shown in Fig. 11.

The additional layer neural network configuration (N4,2 and N4,1), as depicted in Table 2, shows a drop in the correlation coefficient for the connection weights N3,1—N4,2, N3,2—N4,1, and N3,1—N4,1, as well as negligible connection weights for the connections N3,2—N4,2 and N3,3—N2,1. Furthermore, other connections contribute the most to the prediction. We have also tabulated the correlation coefficient for the underfit or non-convergence case in Table 3.

For the underfit or non-convergence case, we can see that most connection weights provide either less or least contribution, or it is strange to find a strong contribution, as depicted in Table 3.

For the next part of our experiment, we have designed an output layer and nodes with extended functionality to incorporate business rules in neural network training, as discussed in Section. 5. The extended output node obtains its relevant

Table 1 The Correlation Coefficient (CC) with multiple trials

Connection weights	Trials1 CC	Trials2 CC	Trials3 CC	Trials4 CC	Trials5 CC	Trials6 CC	Trials7 CC	Trials8 CC	Trials9 CC	Trials10 CC
N2,3 to N3,2	1	1	1	1	1	1	1	1	1	1
N2,2 to N3,2	1	0.99	-1	0.99	0.99	0.99	-1	0.99	0.99	1
N2,2 to N3,1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
N2,3 to N3,1	1	1	1	1	1	1	1	1	1	1
N2,2 to N3,1	-0.99	0.99	-1	0.99	0.99	0.99	-1	1	0.99	1
N2,1 to N3,1	-0.99	-1	-1	-1	-1	-1	-1	-1	-1	-1
N1,3 to N2,3	-0.94	-0.95	0.99	0.96	0.96	0.96	0.99	0.99	-0.95	0.99
N1,2 to N2,3	0.95	0.95	0.99	0.96	0.96	0.96	-0.99	0.99	-0.95	-0.99
N1,1 to N2,3	-0.88	0.93	-0.99	-0.95	-0.95	-0.95	-0.99	-0.99	0.97	0.99
N1,3 to N2,2	0.99	-0.99	-0.99	1	1	1	-0.98	0.99	-0.99	0.99
N1,2 to N2,2	-0.99	0.99	-0.99	1	1	1	0.98	0.98	-1	-0.99
N1,1 to N2,2	-1	-0.99	0.99	-1	-1	-1	-0.98	-1	1	-0.98
Absolute average correlation coefficient	0.97	0.98	0.99	0.98	0.98	0.98	0.99	0.99	0.98	0.99

Refer Fig. 11 for neural network connection reference
 Bold values representing the Significance of Strong participation from all the connection weight

business rule from the business model for each epoch of the training. The result is asserted with the corresponding rule, and the asserted Loss is then combined with regular L2 Loss. We have conducted an ablation study to determine the impact of business loss.

The combined business loss performs exceptionally well compared to L2 Loss alone, as shown in Fig. 13. There is a faster decline in Loss with a few epochs, and improved predictions from the RMSE and R^2 metrics in Table 5. As seen from the result comparison in Fig. 14, the solid lines for all four combinations closely overlap and move towards the targeted results, i.e., either 0 or 1 for the combined business loss, followed by the dotted line for L2 Loss alone.

10 Results and discussion

In this section, we compare the proposed feature importance technique, Feature Importance through the Cyclic Learning Sequence (FICLS), with various XAI evaluation parameters, and then analyze the impact of business loss from an ablation study.

FICLS computes feature importance for the entire neural network, starting from the input to output layers, and therefore provides a global explanation as represented in Table 4. For implementation level and model dependency, FICLS falls under post-hoc and model-agnostic explanation. Since FICLS infers the feature importance from the post-trained Loss and weight distribution and measures feature importance for generalized ANN, it is relevant for other types of neural networks. From an interpretation perspective, proposed method provides improved interpretation as the cyclic learning sequence is straightforward and can be easily understood with minimal understanding of neural network training, compared to surrogating, gradient importance, and other techniques.

Moreover, the correlation coefficient matrix gives the end-user additional assurance for the decision. As part of the interpretation from developers’ and researchers’ perspectives, FICLS explains the feature importance of each connection with discrete measures. The developer and researcher can gain insights by determining which connection weight plays a significant role. Also, feature importance values are discrete, with positive and negative values, therefore adaptable to Parallel Coordinate Plots (PCP) for visual analytics.

We have done a time complexity analysis for the proposed technique:

- Let us consider ANN with,
- L Number of layers,
- N Number of nodes in each layer,

The number of connections in a layer is $C = (N \times N) + N$ (13)

Let E be the number of epochs for training.

Table 2 Correlation coefficient with an additional layer

	N3,3	N3,2	N3,1
N4,2	1	– 0.17	0.40
N4,1	1	– 0.21	0.36
	N2,3	N2,2	N2,1
N3,3	– 0.74	0.84	– 0.05
N3,2	0.98	– 0.95	– 0.99
	N1,3	N1,2	N1,1
N2,3	– 0.88	– 0.89	0.89
N2,2	0.98	0.99	– 0.95

Bolded values represent a lesser contribution from the connection weight

Table 3 The correlation coefficient for underfit or non-convergence training

	N2,1	N2,2	N2,3
N3,2	– 0.64	– 0.32	1
N3,1	– 0.38	– 0.38	1
	N1,1	N1,2	N1,3
N2,3	– 0.52	0.29	0.30
N2,2	0.05	0.45	0.46

Bolded values represent a lesser contribution from the connection weight

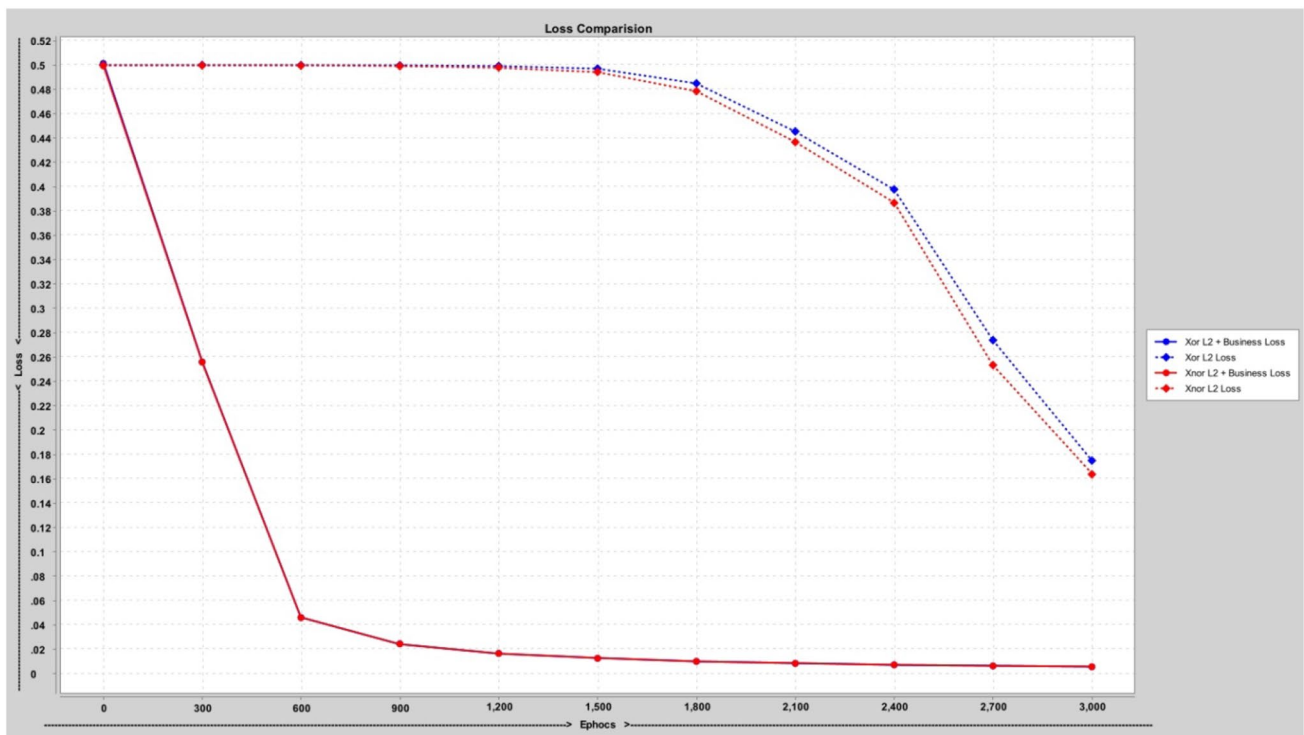


Fig. 13 Loss Comparison: L2 + business Loss with L2 alone

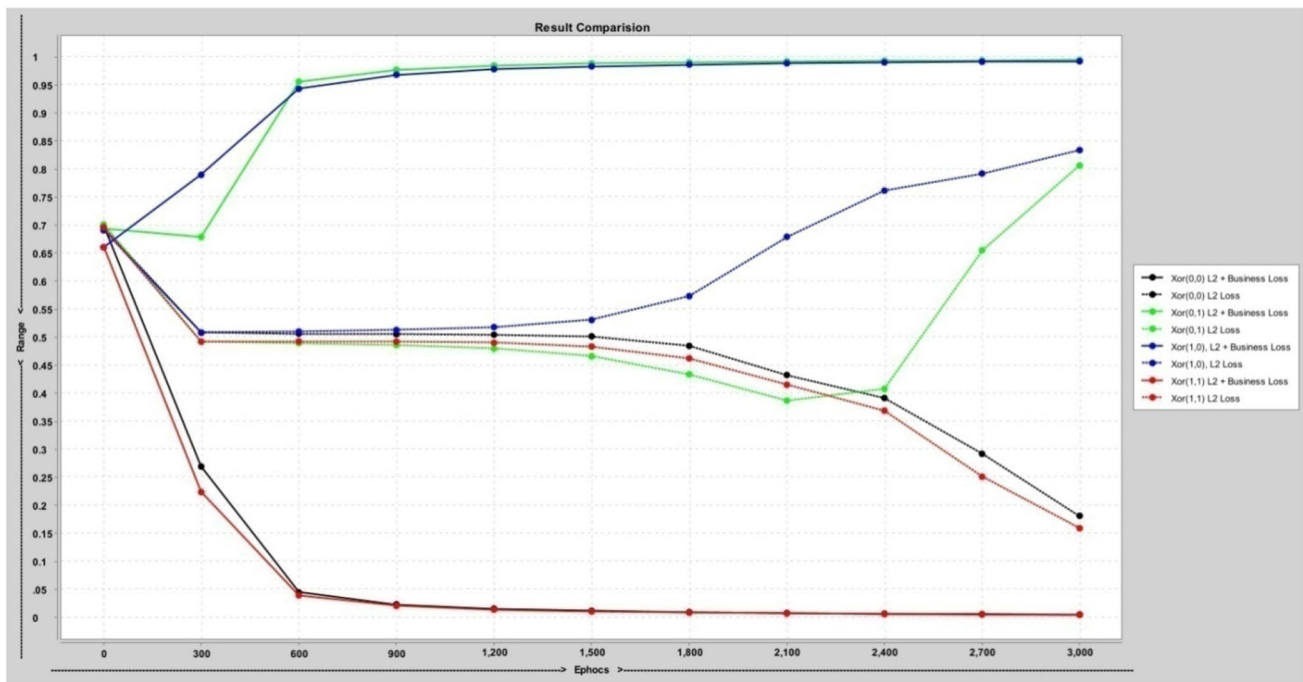


Fig. 14 XOR function approximation with L2 + business loss and L2 Loss alone

Table 4 Comparison of Feature Importance methods with XAI evolution parameter

XAI Methods	Explanation level		Implementation Level		Model Dependency		Interpretation Level			Computation Level		
	Global	Local	Intrinsic Explanation	Post-Hoc Explanation	Model- specific	Model- Agnostic	Low	Mid	high	Low	Mid	high
Anchors	✓	✓		✓		✓		✓				✓
Bayesian Rule Lists	✓		✓		✓				✓	✓		
Distillation	✓		✓			✓		✓			✓	
Grad-CAM		✓		✓		✓		✓		✓		
Integrated Gradients(IG)		✓		✓		✓		✓		✓		
LIME	✓			✓		✓	✓				✓	
SHAP		✓		✓		✓		✓			✓	
FICLS(Proposed method)	✓			✓		✓			✓	✓		

The time complexity (TC) to compute the correlation coefficient for a layer is.

$$TC \text{ For single layer} = E \times C \tag{14}$$

For all the layers

Table 5 Ablation study for the combined domain/business loss

Experiments	L2 Loss	business Loss + L2	RMSE	R ²
L2 Loss	✓		0.1693 ↓	0.73 ↑
L2 + Business Loss	✓	✓	0.0070 ↓	0.99 ↑

$$TC = (L - 1) \times (E \times C) \tag{15}$$

The time complexity (TC) for FICLS is the same as training the neural network, which is optimistic.

Business loss plays a crucial part by giving progress in the training, which proved to decline the mean Loss faster and close to 0 as shown in Fig. 13. The performance of the model improved when the business loss was combined with L2 Loss by a significant decrease in the RMSE Loss and an improved prediction by a substantial increase in R² metrics, as represented in Table 5. Therefore, apart from the mapping learned from the dataset, the known rules and predicates

can substantially ease the effort required to extract features for prediction by giving early convergence and improving performance for the appropriately designed and combined conceptual part into the general neural network structure.

11 Conclusions

We have introduced a feature importance technique through Cyclic Learning Sequence (CLS) by analyzing the learning integrity of prediction, Loss, and Weight. We designed a neural network representation for CLS using Object Oriented Programming (OOP) and extended the representation to embed domain/business rules from the researcher's and expert's perspectives. The designed proposed model is effective enough to draw feature importance from the association between Loss and Weight for different learning conditions with optimal time complexity. CLS gives the degree of participation of each connection weight for the decision with discrete measures. The discrete measure assists end users and AI developers/researchers with insight from the training. Moreover, the correlation coefficient matrix adds assurance to the decision. AI developers/researchers can mitigate the black box nature by inspecting the feature importance of connection weight for different inputs and their corresponding decision. Further to the baseline representation, we have extended the representation to embed domain/business rules using standard design principles, which is effective in the dynamic embedding of business rules and showed a positive impact on performance in terms of early convergence, RMSE, and R^2 metrics. Therefore, the integration of business loss adds additional semantics to the neural network.

Modern technological evolution has uncovered many exceptional aspects of natural phenomena that were previously inscrutable. One such aspect is the ability of creatures to learn and reason, which has been better understood through progress in machine learning (ML) and neural networks (NN), leading to a better realization of biological neural networks from artificial neural networks [45]. In order to avoid abstract "magic" functions, we must delve into the depth and length of neural network derivation, from which we can answer the most transformative question of this decade: "from how machines work to how they learn?" The optimization techniques from differential calculus, data organization, and restless computation play crucial roles in discerning patterns from data. This transformation has indeed provided a glimpse of natural phenomena from biological neural networks. However, understanding and ensuring the decisions made by a "mindless" machine is more challenging. Explainable AI (XAI) is a new and evolving field, but XAI methods and their advanced derivatives have

tremendous potential to mitigate this challenge and enhance the reliability of the decision.

Declarations

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

References

1. Patil S, Patil KR, Patil CR, Patil SS (2020) Performance overview of an artificial intelligence in biomedics: a systematic approach. *Int J Inf Technol* 12(3):963–973
2. Singh N, Panda SP (2022) Artificial Neural Network on Graphical Processing Unit and its emphasis on ground water level prediction. *Int J Inf Technol* 14(7):3659–3666
3. Gupta S, Saini AK (2021) An artificial intelligence based approach for managing risk of IT systems in adopting cloud. *Int J Inf Technol* 13:2515–2523
4. Tavakoli A, Honjani Z, Sajedi H (2023) Convolutional neural network-based image watermarking using discrete wavelet transform. *Int J Inf Technol* 15(4):2021–2029
5. Mahajan A, Singh HP, Sukavanam N (2017) An unsupervised learning based neural network approach for a robotic manipulator. *Int J Inf Technol* 9:1–6
6. Hospedales T, Antoniou A, Micaelli P, Storkey A (2021) Meta-learning in neural networks: a survey. *IEEE Trans Pattern Anal Mach Intell* 44(9):5149–5169
7. Kratsios A (2021) The universal approximation property. *Annals Mathematics Artificial Intelligence* 89(5):435–469
8. Speith, T. (2022) A review of taxonomies of explainable artificial intelligence (XAI) methods. In *2022 ACM Conference on Fairness, Accountability, and Transparency* (pp. 2239–2250).
9. Creel KA (2020) Transparency in complex computational systems. *Philosophy of Science* 87(4):568–589
10. Arulprakash E, Aruldoss M (2022) A study on generic object detection with emphasis on future research directions. *J King Saud University-Computer Inform Sci* 34(9):7347–7365
11. Arulprakash E, Martin A, Lakshmi TM (2022) A study on indirect performance parameters of object detection. *SN Computer Science* 3(5):1–11
12. Jiang P, Suzuki H, Obi T (2023) XAI-based cross-ensemble feature ranking methodology for machine learning models. *Int J Inf Technol* 15(4):1759–1768
13. Alicioglu G, Sun B (2022) A survey of visual analytics for explainable artificial intelligence methods. *Comput Graph* 102:502–520
14. Šefčík, F., Benesova, W., (2023) Improving a neural network model by explanation-guided training for glioma classification based on MRI data. *International Journal of Information Technology*, 1–9.
15. Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. (2016) "Why should I trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144.
16. Lundberg, Scott M., and Su-In Lee. (2017) "A unified approach to interpreting model predictions." *Advances in neural information processing systems* 30 .
17. Alwadi M, Chetty G, Yamin M (2023) A framework for vehicle quality evaluation based on interpretable machine learning. *Int J Inform Tech* 15(1):129–136

18. Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. (2018) “Anchors: High-precision model-agnostic explanations.” In *Proceedings of the AAAI conference on artificial intelligence*, 32 (1).
19. Zhou, Bolei, AdityaKhosla, AgataLapedriza, AudeOliva, and Antonio Torralba. 2016 “Learning deep features for discriminative localization.” In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929.
20. Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and DhruvBatra. 2017 “Grad-cam: Visual explanations from deep networks via gradient-based localization.” In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626.
21. Shrikumar, A., Greenside, P. and Kundaje, A., 2017. Learning important features through propagating activation differences. In *International conference on machine learning* (pp. 3145–3153). PMLR.
22. Bach S, Binder A, GrégoireMontavon FK, Müller K-R, Wojciech-Samek. (2015) On pixel-wise explanations for nonlinear classifier decisions by layer-wise relevance propagation. *PLoS ONE* 10(7):e0130140
23. Caruana, Rich, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and NoemieElhadad. 2015 “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission.” In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1721–1730.
24. Breiman L. Manual on setting up, using, and understanding random forests 125 v3. Tech Rep 2002;4 (1):29, https://www.stat.berkeley.edu/~breiman/Using_126_random_forests_V3.1.pdf. [Accessed 26 Dec 2022].
25. Pasricha, Sahil. 2020 Visually Explaining the Weight Distribution of Neural Networks over Time.
26. Confalonieri R, LudovikCoba BW, Besold TR (2021) A historical perspective of explainable artificial intelligence. *Wiley Interdisciplinary Rev: Data Mining Knowledge Discovery* 11(1):e1391
27. Dikmen M, Burns C (2022) The effects of domain knowledge on trust in explainable AI and task performance: A case of peer-to-peer lending. *Int J Hum Comput Stud* 162:102792
28. Islam, Sheikh Rabiul, William Eberle, Sheikh K. Ghafoor, AmbareenSiraj, and Mike Rogers. (2019) “Domain knowledge aided explainable artificial intelligence for intrusion detection and response.” *arXiv preprint arXiv:1911.09853*.
29. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. *J Machine Learn Res* 15(1):1929–1958
30. Slack D, Hilgard A, Singh S, Lakkaraju H (2021) Reliable post hoc explanations: Modeling uncertainty in explainability. *Adv Neural Inf Process Syst* 34:9391–9404
31. zmailov, P., Podoprikin, D., Garipov, T., Vetrov, D., and Wilson, AG., (2018). Averaging weights leads to wider optima and better generalization. *arXiv preprint arXiv:1803.05407*.
32. Smith, L. N. (2017) Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)* (pp. 464–472). IEEE.
33. Bengio, Y. (2012) Practical recommendations for gradient-based training of deep architectures. In *Neural Networks: Tricks of the Trade: Second Edition* (pp. 437–478). Berlin, Heidelberg: Springer Berlin Heidelberg.
34. Goodfellow, I., Bengio, Y., Courville, A. 2016. *Deep learning*. MIT press.
35. Schober P, Boer C, Schwarte LA (2018) Correlation coefficients: appropriate use and interpretation. *Anesth Analg* 126(5):1763–1768
36. Alam M (2020) Object oriented software security: goal questions metrics approach. *Int J Inf Technol* 12(1):175–179
37. Black AP (2013) Object-oriented programming: Some history, and challenges for the next fifty years. *Inf Comput* 231:3–20
38. Blaha, Michael R., and James R. Rumbaugh. 2020 “Object Oriented Modeling and Design with UML.”
39. Edwin, NjeruMwendi. 2014 “Software frameworks, architectural and design patterns.” *Journal of Software Engineering and Applications*.
40. Mu, Huaxin, and ShuaiJiang.2011 “Design patterns in software development.” In *2011 IEEE 2nd International Conference on Software Engineering and Service Science*, pp. 322–325. IEEE.,
41. P.K. Biswas 2019 . “Deep Learning IIT KGP: Back Propagation Learning”, NPTEL course.
42. Galatolo FA, Cimino MGCA, Vaglini G (2021) Formal derivation of mesh neural networks with their forward-only gradient propagation. *Neural Process Lett* 53(3):1963–1978
43. Kim D, June-HaakEe CY, Lee J (2021) Derivation of Jacobian formula with Dirac delta function. *Eur J Phys* 42(3):035006
44. Magnus, Jan R., and Heinz Neudecker. (2019) *Matrix differential calculus with applications in statistics and econometrics*.John Wiley and Sons.
45. Lillicrap TP, Santoro A, Marris L, Akerman CJ, Hinton G (2020) Backpropagation and the brain. *Nat Rev Neurosci* 21(6):335–346

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.