On the Effectiveness of Interpretable Feedforward Neural Network

Miles Q. Li
School of Computer Science
McGill University
Montreal, Canada
miles.qi.li@mail.mcgill.ca

Benjamin C. M. Fung*
School of Information Studies
McGill University
Montreal, Canada
ben.fung@mcgill.ca

Adel Abusitta
School of Information Studies
McGill University
Montreal, Canada
adel.abusitta@mcgill.ca

Abstract—**Deep learning models have achieved state-of-the-art performance in many classification tasks. However, most of them cannot provide an explanation for their classification results. Machine learning models that are interpretable are usually linear or piecewise linear and yield inferior performance. Non-linear models achieve much better classification performance, but it is usually hard to explain their classification results. As a counter-example, an interpretable feedforward neural network (IFFNN) is proposed to achieve both high classification performance and interpretability for malware detection. If the IFFNN can perform well in a more flexible and general form for other classification tasks while providing meaningful explanations, it may be of great interest to the applied machine learning community. In this paper, we propose a way to generalize the interpretable feedforward neural network to multi-class classification scenarios and any type of feedforward neural networks, and evaluate its classification performance and interpretability on interpretable datasets. We conclude by finding that the generalized IFFNNs achieve comparable classification performance to their normal feedforward neural network counterparts and provide meaningful explanations. Thus, this kind of neural network architecture has great practical use.**

I. Introduction

Deep learning models are achieving state-of-the-art performance in an increasing number of tasks [1]–[4]. They work as black-boxes, in which when a large number of training samples are fed to them, they learn patterns that correlate with different classes and then the patterns are used to classify unseen samples. However, most deep neural networks only implicitly learn and use the patterns, and do not explicitly explain the reasons for which a sample belongs to a class. This causes concerns about applying deep learning in some critical fields, such as healthcare and automatic pilot systems [5]–[9].

That being said, there are interpretable machine learning classification models, such as linear regression, softmax regression, and decision trees [10]. These models can explain their classification results in a clear and simple way. However, as linear or piecewise linear models, their expressive abilities are very limited, i.e., they cannot model complex interactions between different features. Linear regression and softmax regression can be seen as neural networks with no hidden layers. They can tell to what extent each feature contributes to a classification result. The interpretability comes from that fact that the relation between a feature and the class of a sample is computed independently without any interactions. Even though this simplicity allows the model to explain its classification results, it yields inferior results compared to multi-layer neural networks. In this era, classification performance typically has higher priority than interpretability. Hence, these simple models are usually less useful than the complex and non-interpretable models [7].

In an attempt to solve the dilemma of choosing either high classification performance or interpretability, some techniques have been proposed to explain the classification results of complex machine learning models. For example, integrated gradients [11] and permutation feature importance [12]–[14] can explain many kinds of machine learning models. However, the model needs to be run many times to explain one prediction. The computational cost for an explanation is too expensive. Some others, such as surrogate model methods [15], [16], use another interpretable model (e.g., a decision tree) as a surrogate to approximate the target model and use the surrogate model's explanation to explain the target model's prediction. However, the expressive ability of a surrogate model is usually not as good as a complex target model; thus, the former cannot very accurately approximate the latter, and the explanation also cannot be accurate.

To address the aforementioned limitations, some researchers have turned to creating deep neural networks that are intrinsically interpretable. Choi et al. [5] propose RETAIN for classifying sequential data with an explanation on how much each variable in a sequence contributes to the classification result. Li et al. [17] propose an interpretable feedforward neural network (IFFNN) for malware detection. It classifies vectorial data and provides an explanation on how much each feature in the vector contributes to the detection result. The proposed IFFNN architecture is promising for solving the dilemma between classification performance and interpretability. However, their exploration of IFFNN is very limited because it was only applied to binary classification, and the architecture contains fully connected layers and only accepts vectors as its input. In addition, the classification performance and interpretability were not comprehensively evaluated on general classification problems.

To explore whether the IFFNN [17] can be extended to

* Corresponding author.

general classification scenarios and achieve excellent classification performance and provide meaningful explanations, in this paper we generalize IFFNN to multi-class classification scenarios and any type of feedforward neural networks, and perform a comprehensive evaluation on the classification performance and interpretability of two interpretable datasets. The source code of this work is released at https://github.com/McGill-DMaS/IFFNN. The contributions of this paper are summarized as follows:

- We propose ways to generalize the IFFNN to multi-class classification and any type of feedforward neural network that takes any tensors of a fixed shape as its input.
- We conduct comprehensive experiments to evaluate the classification performance and interpretability of the IFFNNs. We compare the classification accuracy of the IFFNNs with their non-interpretable counterparts to show that they have similar classification performance.
- We propose a synthetic interpretability benchmark dataset to evaluate the interpretability of classification models. It can generate an unlimited number of samples with the reasons why they belong to a specific class.

II. Related Works

The terms explainability and interpretability can be confusing. There has been works that try to address the distinction [6], [18]. Explainability is the ability of a post-hoc method to explain the prediction of a model, while interpretability is the intrinsically self-explaining ability of a model without relying on a post-hoc explanation process.

Explanations for machine learning models can be acquired in different ways. For linear or piecewise linear models, such as linear regression, softmax regression, decision trees, and k-nearest neighbors, their simple classification mechanics make them intrinsically interpretable. Their expressive ability is quite limited so they achieve inferior classification performance when the features have complex interactions [7]–[9].

Most complex machine learning models are not easily interpretable in themselves. Some post-hoc explanation techniques have been proposed to explain their classification results. Some explanation methods do not require knowledge of the models. They just need the input and output pairs of the models to provide an explanation. The permutation feature importance method [12]–[14] is one example of a model-agnostic method. The values of the features are permuted and then their impact on the classification results give a clue on how important they are. The computational cost is high since a model needs to be run multiple times. Surrogate model methods [15], [16] train an interpretable model, such as a decision tree, to approximate the target model to explain, and use the explanations given by the surrogate models to explain the results of the target model. As the expressive abilities of the surrogate models are usually lower than the target models, neither the approximation nor the explanations are accurate. There are other explanation techniques that work in a model-agnostic manner [19], [20].

Other techniques are proposed to explain certain types of machine learning models. The integrated gradients method [11] is proposed to explain the classification results of neural networks (i.e., differentiable models) by cumulating the gradients along the path from a base sample to the target sample. As this also requires running the target model multiple times, its efficiency is still limited. The fuzzy rule extraction method is proposed especially for explanation of classification results for support vector machines [21]. Other explanation techniques are proposed for different types of neural networks, such as feedforward neural networks [17], [22], recurrent neural networks [5], [23], convolutional neural networks [24]–[26], and deep graphical models [22].

III. Problem Definition

The term *explanation* can be defined in different ways. To clarify the explanation we discuss in this paper, we give the following formal definition of explanation in a classification problem.

Definition 1 (Explanation): Let a sample be a p-th-order tensor $\boldsymbol{X} \in \mathbb{R}^{m_1 \times m_2 \ldots \times m_p}$. The sample belongs to one of c classes. An interpretable classification model should predict its class $y \in \{1, 2, .., c\}$ and give an explanation $\boldsymbol{I} \in \mathbb{R}^{c \times m_1 \times m_2 \ldots \times m_p}$. $\boldsymbol{I}_{j,i_1,i_2,\ldots,i_p}$ represents the importance/contribution of feature $\boldsymbol{X}_{i_1,i_2,\ldots,i_p}$ for classifying the sample $\boldsymbol{X}$ to class j.

As can be seen from the definition of explanation, it provides the importance value of a feature not only for the predicted class, but also for other classes. In practice, the explanation does not have to be organized as a tensor $\boldsymbol{I}$. As long as an importance score of each element in $\boldsymbol{X}$ for each class can be computed, it is equivalent to having $\boldsymbol{I}$. The interpretability mentioned in this paper, refers to the intrinsic ability of classification models to provide the kind of explanation we define.

IV. Interpretable Feedforward Neural Network

The interpretable feedforward neural network proposed by Li et al. [17] contains a series of fully connected layers, which is similar to a normal feedforward neural network. The difference is that the output of the top layer $\boldsymbol{w}(\boldsymbol{x})$ is a vector that has the same dimension as the input feature vector and is used as a dynamically computed weight for the features. The last step is the same as logistic regression, which uses the dot product of the $\boldsymbol{w}(\boldsymbol{x})$ and $\boldsymbol{x}$, followed by sigmoid as the probability that a sample is positive.

The full computation is as follows. Let $\boldsymbol{x} \in \mathbb{R}^m$ be the feature vector of a sample. It is fed to l fully connected hidden layers:

$$\boldsymbol{v}_l(\boldsymbol{x}) = FC^l(...FC^1(\boldsymbol{x})...) \tag{1}$$

$$where\ FC^i(\boldsymbol{v}_{i-1}(\boldsymbol{x})) = f(\boldsymbol{W}_1^i \boldsymbol{v}_{i-1}(\boldsymbol{x}) + \boldsymbol{b}_1^i) \tag{2}$$

where $\boldsymbol{W}_1^i \in \mathbb{R}^{d^i \times d^{i-1}}$, $\boldsymbol{b}_1^i \in \mathbb{R}^{d^i}$, f is the activation function (e.g., $Relu$, $tanh$), and $\boldsymbol{v}_l(\boldsymbol{x}) \in \mathbb{R}^{d^l}$. Another normal fully connected layer where the output vector has the same dimension as $\boldsymbol{x}$ is applied:

$$\boldsymbol{w}(\boldsymbol{x}) = \boldsymbol{W_2} \boldsymbol{v}_l(\boldsymbol{x}) + \boldsymbol{b_2} \tag{3}$$

where $\boldsymbol{W_2} \in \mathbb{R}^{m \times d^l}$, $\boldsymbol{b_2} \in \mathbb{R}^m$, and $\boldsymbol{w}(\boldsymbol{x}) \in \mathbb{R}^m$. $\boldsymbol{w}(\boldsymbol{x})$ serves as a weight vector for each feature in $\boldsymbol{x}$. The final confidence that the input sample belongs to the positive class (in malware detection, positive means malicious) is calculated as follows:

$$y = IFFNN(\boldsymbol{x}) = \sigma(\boldsymbol{w}(\boldsymbol{x})^T \boldsymbol{x} + b) \tag{4}$$

$$where\ \sigma(z) = \frac{1}{1 + e^{-z}}, b \in \mathbb{R} \tag{5}$$

This IFFNN has the expressive ability of a non-linear model since $\boldsymbol{w}(\boldsymbol{x})$ is computed through a multi-layer fully connected neural network. The interpretability of it is like logistic regression: the contribution of feature x_i to the positive class is calculated as $w(\boldsymbol{x})_i x_i$ and the contribution of feature x_i to the negative class is $-w(\boldsymbol{x})_i x_i$.

V. Generalization of Interpretable Feedforward Neural Networks

The IFFNN can be generalized in different ways to be a more versatile neural network architecture for additional classification scenarios. We describe our methods of generalization in this section.

A. Generalization to Multi-class Classification

The original IFFNN is proposed for binary classification. It works as a logistic regression function with "dynamically" computed weights. Thus, a generalization of the original IFFNN to multi-class classification is to make it a software regression with "dynamically" computed weights.

Let c be the number of classes and $\boldsymbol{W} \in \mathbb{R}^{c \times m}$ be a parametric matrix. Softmax regression can be expressed as follows:

$$\boldsymbol{y} = softmax(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) \tag{6}$$

$$where\ softmax(\boldsymbol{z}) = \frac{1}{\sum_{j=1}^{c} e^{z_j}}(e^{z_1}, ..., e^{z_c}), \boldsymbol{b} \in \mathbb{R}^c \tag{7}$$

The output is a vector of dimension c, and each element is the probability that the sample belongs to a class. Therefore, $W_{i,j} x_j$ is the contribution of feature x_j to class i.

For a multi-class classification scenario, rather than mapping the output of the last fully connected layer to a vector of dimension m, in the generalized IFFNN, the last fully connected layer requires a tensor to map the feature vector to a matrix that has the shape $c \times m$.

The complete computation of the generalized IFFNN for multi-class classification can be expressed as follows:

$$\boldsymbol{v}_l(\boldsymbol{x}) = FC^l(...FC^1(\boldsymbol{x})...) \tag{8}$$

$$\boldsymbol{W}(\boldsymbol{x}) = \boldsymbol{T}\boldsymbol{v}_l(\boldsymbol{x}) + \boldsymbol{B_2} \tag{9}$$

$$\boldsymbol{y} = softmax(\boldsymbol{W}(\boldsymbol{x})\boldsymbol{x} + \boldsymbol{b}) \tag{10}$$

where $\boldsymbol{T} \in \mathbb{R}^{c \times m \times d^l}$, $\boldsymbol{B_2} \in \mathbb{R}^{c \times m}$, $\boldsymbol{W}(\boldsymbol{x}) \in \mathbb{R}^{c \times m}$, and $\boldsymbol{b} \in \mathbb{R}^c$. The contribution of feature x_i to class j is $W(\boldsymbol{x})_{j,i} x_i$.

In practice, it is equivalent to replace the tensor T with a matrix $\boldsymbol{W_2} \in \mathbb{R}^{(cm) \times d^l}$. This matrix maps $\boldsymbol{v}_l(\boldsymbol{x})$ to a vector of dimension cm, which can be reshaped to a matrix with the expected shape $c \times m$. The complete equivalent computation of the generalized IFFNN for multi-class classification can be expressed as follows:

$$\boldsymbol{v}_l(\boldsymbol{x}) = FC^l(...FC^1(\boldsymbol{x})...) \tag{11}$$

$$\boldsymbol{W}(\boldsymbol{x}) = Reshape(\boldsymbol{W_2}\boldsymbol{v}_l(\boldsymbol{x}), (c \times m)) + \boldsymbol{B_2} \tag{12}$$

$$\boldsymbol{y} = softmax(\boldsymbol{W}(\boldsymbol{x})\boldsymbol{x} + \boldsymbol{b}) \tag{13}$$

where the $Reshape(\boldsymbol{z}, target\ shape)$ operation re-organizes the elements of $\boldsymbol{z}$ to the target shape.

B. Generalization to Any Feedforward Neural Networks with Any Tensor of Fixed Shape as Input

The original IFFNN can only be applied on vectors of fixed dimensions and only includes fully connected layers. These two constraints can be removed to build more expressive feedforward neural networks for wider applications. Rather than being a vector of a fixed dimension, the input can be any tensor of a fixed shape. Vectors as first-order tensors are the most commonly seen feature form. Matrices as second-order tensors are also common as the input to feedforward neural networks. Greyscale images serve as a good example of this type. Furthermore, RBG images can be represented as third-order tensors. The feedforward neural networks that classify these high order tensors usually contain other kinds of layers besides fully connected layers, such as convolutional layers and pooling layers. We describe ways to handle the generalized situations as follows.

Let $\boldsymbol{X} \in \mathbb{R}^{m_1 \times m_2 ... \times m_p}$ be an order p tensor representing the features of a sample. Let $m = m_1 \times m_2 ... \times m_p$. For binary classification, we have:

$$\boldsymbol{v}(\boldsymbol{X}) = f(\boldsymbol{X}) \tag{14}$$

$$\boldsymbol{w}(\boldsymbol{X}) = \boldsymbol{W_2}\boldsymbol{v}(\boldsymbol{X}) + \boldsymbol{b_2} \tag{15}$$

$$\boldsymbol{x'} = flatten(\boldsymbol{X}) \tag{16}$$

$$\boldsymbol{y} = \sigma(\boldsymbol{w}(\boldsymbol{X})^T \boldsymbol{x'} + b) \tag{17}$$

where f represents an arbitrary feedforward neural network with any kind of layers, $\boldsymbol{v}(\boldsymbol{X}) \in \mathbb{R}^d$, $\boldsymbol{W_2} \in \mathbb{R}^{m \times d}$, $\boldsymbol{b_2}, \boldsymbol{x'} \in \mathbb{R}^m$, the $flatten$ operation re-organizes the elements of a tensor to a 1d array to form a vector, and $b \in \mathbb{R}$. The contribution of feature $X_{i_1,...,i_p}$ to the positive class is $w(\boldsymbol{X})_i x'_i$ where $i = (i_1 - 1) \times (m_2 m_3 ... m_p) + (i_2 - 1) \times (m_3 m_4 ... m_p) + ... + i_p$.

For multi-class classification, we have:

$$\boldsymbol{v}(\boldsymbol{X}) = f(\boldsymbol{X}) \tag{18}$$

$$\boldsymbol{W}(\boldsymbol{X}) = Reshape(\boldsymbol{W_2}\boldsymbol{v}(\boldsymbol{X}), (c \times m)) + \boldsymbol{B_2} \tag{19}$$

$$\boldsymbol{x'} = flatten(\boldsymbol{X}) \tag{20}$$

$$\boldsymbol{y} = softmax(\boldsymbol{W}(\boldsymbol{X})\boldsymbol{x'} + \boldsymbol{b}) \tag{21}$$

where $\boldsymbol{v}(\boldsymbol{X}) \in \mathbb{R}^d$, $\boldsymbol{W_2} \in \mathbb{R}^{(cm) \times d}$, $\boldsymbol{B_2} \in \mathbb{R}^{c \times m}$, $\boldsymbol{x'} \in \mathbb{R}^m$, and $\boldsymbol{b} \in \mathbb{R}^c$. The contribution of feature $X_{i_1,...,i_p}$ to class j is $W(\boldsymbol{X})_{j,i} x'_i$ where $i = (i_1 - 1) \times (m_2 m_3 ... m_p) + (i_2 - 1) \times (m_3 m_4 ... m_p) + ... + i_p$.

It should be noted that assuming $\boldsymbol{v}(\boldsymbol{X})$, the output of $f(\boldsymbol{X})$ to be a vector of a fixed dimension does not cause the loss of generality. When $f(\boldsymbol{X})$ is a higher order tensor rather than a vector, its shape is still fixed, so it can always be converted to a vector by applying a $flatten$ operation.

C. Discussion

In some cases, in the input tensor, multiple elements correspond to the same object. When the contribution of each object is expected, the contributions of these elements should be added up. For instance, an RGB image can be represented as a third-order tensor $\boldsymbol{X} \in \mathbb{R}^{3\times h\times w}$. $X_{1,i,j}$, $X_{2,i,j}$, and $X_{3,i,j}$ are the red, green, and blue values of the same pixel. The contribution of pixel (i,j) is the summation of the contributions of $X_{1,i,j}$, $X_{2,i,j}$, and $X_{3,i,j}$.

VI. Experiments

In this section, we evaluate various versions of IFFNNs on different datasets. The objectives are to answer the following questions:

- Is classification performance harmed when the feedforward neural networks are organized in our interpretable way compared to normal feedforward neural networks?
- Do the explanations given by the IFFNNs make sense?
- Do the generalized versions of IFFNNs work well in terms of classification performance and interpretability?

A. Datasets

We evaluate the models on two datasets: MNIST and INBEN. They complement each other in the evaluation procedure. MNIST is an image classification dataset that allows us to evaluate IFFNNs with convolutional layers and to qualitatively evaluate the interpretability of IFFNNs. However, it cannot be used to quantitatively evaluate their interpretability, since there is no exact answer on how important each pixel is for the classification results. With our created dataset INBEN, the gold standard explanations of the samples are known, and thus allows us to achieve this purpose.

TABLE I
Statistics of the datasets used for evaluation.

Dataset	Training	Valid	Test	X Shape
MNIST 10 cls	50,000	10,000	10,000	(28,28)
MNIST 2 cls	10,554	2,111	2,115	(28,28)
INBEN 10 cls	100,000	10,000	10,000	(1000,)
INBEN 2 cls	20,000	2,000	2,000	(1000,)

1) MNIST: MNIST is a handwritten digit dataset. It is a common benchmark for image classification models. This dataset works well for our purposes because of its easily interpretable characteristic. The IFFNNs applied on this dataset can point out which pixels are important to classify a sample to a certain digit. It is easy for humans to determine whether these pixels are good indicators for the predictions.

We create two scenarios with MNIST. **Scenario 1** uses samples of all 10 classes. In this scenario, we can evaluate the versions of IFFNNs for multi-class classification. **Scenario 2** uses samples of only two classes (digits of ”0” and ”1”). In this scenario, we can evaluate the versions of IFFNNs for both binary classification and multi-class classification.

2) INBEN: By visualizing the importance of each pixel of an image in MNIST, we can only qualitatively evaluate the interpretability of the IFFNNs. To quantitatively evaluate the interpretability, we propose a synthetic INterpretablility BENchmark (INBEN) dataset. It can be described as follows:

1) Each sample belongs to 1 of c classes.
2) Each sample is a vector of dimension m. Each entry corresponds to a fixed feature, and the value of it could be 0 or 1. For example, if $m = 5$, a sample could be (1 0 1 1 0).
3) For each class, there is a set of randomly generated patterns, where if a sample contains one of these patterns, it belongs to that class. For example, (1,3) is a pattern for class 2. It means that a sample x belongs to class 2 if $x_1 = 1$ and $x_3 = 1$. (1 0 1 1 0) is an example that contains this pattern.
4) There is a class priority sequence (e.g., [3,2,4,1,5]). If a sample contains patterns of multiple classes, it belongs to the class with the highest priority among them. For example, if a sample contains the patterns of both class 2 and class 5, it belongs to class 2.
5) There is a default class. If a sample contains no patterns, it belongs to the default class.

We also create two scenarios with INBEN datasets. **Scenario 1** contains samples of 10 classes, and **Scenario 2** contains samples of 2 classes.

The statistics of the datasets are given in Table I.

B. Models

We include four kinds of feedforward neural networks in our experiments to illustrate the classification performance and interpretability of the IFFNN architecture. They are fully connected neural networks (FC), convolutional neural networks (CNN) [27], fully connected neural networks with highways (HW) [28], and residual neural networks (ResNET) [29]. For each of the four kinds of neural networks, we have eight different variants. We use FC as the example to describe the variants:

- **FC-BC1** A feedforward neural network with fully connected layers for binary classification. The top fully connected layer maps the feature vector to a real number followed by a sigmoid layer. This is only applicable to **Scenario 2**.
- **FC-MC1** A feedforward neural network with fully connected layers for multi-class classification. The top fully connected layer maps the feature vector to a vector of dimension c followed by a softmax layer.
- **FC-IFFNN-BC** The interpretable version of FC-BC1 achieved by replacing the top layer with Eq.15$\sim$17. This is only applicable to **Scenario 2**.
- **FC-IFFNN-MC** The interpretable version of FC-MC1 achieved by replacing the top layer with Eq.19$\sim$ 21.
- **FC-BC2** Similar to FC-BC1, with the total number of trainable parameters about the same as FC-IFFNN-BC by increasing the dimensions of the layers but not increasing

TABLE II

CLASSIFICATION PERFORMANCE EVALUATION ON MNIST AND INBEN.

Model	10-class MNIST		2-class MNIST		10-class INBEN		2-class INBEN	
	Params	Acc	Params	Acc	Params	Acc	Params	Acc
FC-MC1	898.5K	98.46	894.5K	99.93	1.0M	97.80	1.0M	98.23
FC-MC2	4.8M	98.54	1.7M	99.94	6.0M	98.83	2.0M	98.45
FC-MC3	4.8M	98.49	1.7M	99.92	6.0M	98.69	2.0M	98.37
FC-IFFNN-MC	4.8M	98.06	1.7M	99.91	6.0M	98.19	2.0M	99.06
HW-MC1	2.4M	98.13	2.4M	99.93	2.5M	97.99	2.5M	98.57
HW-MC2	6.3M	98.10	3.2M	99.92	7.5M	97.81	3.5M	98.69
HW-MC3	6.3M	97.67	3.2M	99.93	7.5M	97.41	3.5M	98.68
HW-IFFNN-MC	6.3M	97.96	3.2M	99.90	7.5M	97.58	3.5M	99.28
ResNET-MC1	226.2K	99.50	201.1K	99.92	NA	NA	NA	NA
ResNET-MC2	24.7M	99.41	5.1M	99.99	NA	NA	NA	NA
ResNET-MC3	24.7M	99.39	5.1M	99.93	NA	NA	NA	NA
ResNET-IFFNN-MC	24.8M	98.92	5.1M	99.95	NA	NA	NA	NA
CNN-MC1	1.2M	98.88	1.2M	99.89	NA	NA	NA	NA
CNN-MC2	72.3M	98.95	14.5M	99.92	NA	NA	NA	NA
CNN-MC3	72.3M	98.99	14.5M	99.93	NA	NA	NA	NA
CNN-IFFNN-MC	72.3M	98.69	14.5M	99.96	NA	NA	NA	NA
SR	7.8K	92.82	1.6K	99.95	10.0K	87.53	2.0K	97.67
DT	NA	88.19	NA	99.66	NA	76.75	NA	98.93
FC-BC1	NA	NA	894.0K	99.95	NA	NA	1.0M	98.04
FC-BC2	NA	NA	1.3M	99.92	NA	NA	1.5M	98.47
FC-BC3	NA	NA	1.3M	99.91	NA	NA	1.5M	98.58
FC-IFFNN-BC	NA	NA	1.3M	99.94	NA	NA	1.5M	98.67
HW-BC1	NA	NA	2.4M	99.92	NA	NA	2.5M	98.71
HW-BC2	NA	NA	2.8M	99.91	NA	NA	3.0M	98.55
HW-BC3	NA	NA	2.8M	99.92	NA	NA	3.0M	98.57
HW-IFFNN-BC	NA	NA	2.8M	99.94	NA	NA	3.0M	99.34
ResNET-BC1	NA	NA	197.9K	99.98	NA	NA	NA	NA
ResNET-BC2	NA	NA	2.7M	99.95	NA	NA	NA	NA
ResNET-BC3	NA	NA	2.7M	99.96	NA	NA	NA	NA
ResNET-IFFNN-BC	NA	NA	2.7M	99.91	NA	NA	NA	NA
CNN-BC1	NA	NA	1.2M	99.93	NA	NA	NA	NA
CNN-BC2	NA	NA	7.2M	99.93	NA	NA	NA	NA
CNN-BC3	NA	NA	7.2M	99.91	NA	NA	NA	NA
CNN-IFFNN-BC	NA	NA	7.2M	99.94	NA	NA	NA	NA
LR	NA	NA	0.8K	99.95	NA	NA	1.0K	97.66

the number of layers. This is only applicable to **Scenario 2**.

- **FC-MC2** Similar to FC-MC1, with the total number of trainable parameters about the same as FC-IFFNN-MC by increasing the dimensions of the layers but not increasing the number of layers.
- **FC-BC3** Similar to FC-BC1, with the total number of trainable parameters about the same as FC-IFFNN-BC by increasing the number of layers, and adjusting the dimension of each layer. This is only applicable to **Scenario 2**.
- **FC-MC3** Similar to FC-MC1, with the total number of trainable parameters about the same as FC-IFFNN-MC by increasing the number of layers, and adjusting the dimension of each layer.

For the other three kinds of neural networks, there are the same eight variants. When we apply FC and HW networks on the MNIST dataset, we flatten the input to a vector. We don't apply CNN and ResNET on INBEN because those two networks are mainly for input of matrices or third-order tensors.

We also compare with other interpretable models, including logistic regression (LR), softmax regression (SR), and decision trees (DT). We use grid search to tune the hyper-parameters of decision trees, including its split criterion and maximum depth. The candidate values are given in Table III.

TABLE III

CANDIDATE VALUES FOR HYPER-PARAMETERS OF DECISION TREE.

Hyperparameter	Candidate Values
Split Criterion	gini,entropy
Maximum Depth	10,25,50,100,200,300,400,500,1000

C. Evaluation Metrics

We describe the evaluation metrics for classification performance and interpretability in this section.

For the classification performance, following the tradition, we use **accuracy** as the metric, which is the number of correctly classified samples over the total number of samples.

We cannot use MNIST to quantitatively evaluate the interpretability of the models, but we can use INBEN. With INBEN, we know the reason why a sample belongs to a class. It is the pattern(s) that decides its class. The ideal explanations should give the features included in the patterns the greatest contribution values. Therefore, we use the average of **accuracy@N** as our evaluation metric for interpretability. We formally define it as follows:

Definition 2 (Accuracy@N): Let S_1 be the set of features in the pattern(s) that determines a sample $\boldsymbol{x}$ belong to class c. Let $N = |S_1|$. Let S_2 be the set of top N important features for classifying $\boldsymbol{x}$ to class c by an interpretable classification system. Let $S_3 = S_1 \cap S_2$ and $n = |S_3|$. Then, $Accuracy@N = n/N$.

As can be seen, N is variant to different samples. Below is an example.

A sample $\boldsymbol{x}$ belongs to class 2 because it contains the two patterns of class 2: (113,251) and (35,72,99,217,251). We thus have $S_1 = \{35, 72, 99, 113, 217, 251\}$ and $N = 6$. Let the top six most important features for classifying it to class 2 determined by an interpretable classification model be: 113,251,7,35,12,308. Then, we have $S_2 = \{7, 12, 35, 113, 251, 308\}$, $S_3 = \{35, 113, 251\}$ and thus $n = 3$. $Accuracy@N = \frac{3}{6} = 0.5$.

We use the average of accuracy@N over all correctly classified test samples as the evaluation metric for interpretability. We do not include wrongly classified samples because the $Accuracy@N$ of explanations for wrong predictions do not mean anything.

D. Experiment Setting

We train and evaluate the models on a server with two Xeon E5-2697 CPUs, 384 GB of memory, and four Nvidia Titan XP graphics cards. Only one graphics card is used for each run. The operating system is Windows Server 2016. We use Python 3.7.9 and PyTorch 1.6.0 [30] to implement the models. We use the implementation of DT in scikit-learn 0.23.2 [31].

We use Adam [32] with the initial learning rate $1e-3$ to train all the neural networks including LR and SR. The batch size is 256 and maximum epoch is 200. The accuracy on the test set at the epoch in which the accuracy on the validation set is the best is reported.

We repeat each group of experiments five times and report the average. We use random seeds from 0 to 4 for model initialization.

E. Classification Results

The classification performance of all models is shown in Table II. The IFFNN version of different types of feedforward neural networks achieves slightly higher or lower accuracy compared with the non-interpretable ones in most cases (i.e., the difference is at most 1%). Between the same kind of neural networks with different amounts of trainable parameters, the difference in accuracy is minor as well. On datasets with 10 classes of samples, we can see a significant gap $(> 5\%)$ between SR, DT, and the neural networks. This means that forming feedforward neural networks in the proposed interpretable way does not harm the classification performance and is as effective as a normal multi-layer feedforward neural network. The generalized versions of IFFNNs on different feedforward neural networks for multi-class classification also perform well in terms of accuracy.

F. Interpretability Results

TABLE IV
EVALUATION OF INTERPRETABILITY WITH ACCURACY@N ON INBEN.

Model	10-class INBEN	2-class INBEN
SR	86.81	83.96
FC-IFFNN-MC	98.55	91.39
HW-IFFNN-MC	98.43	95.46
LR	NA	83.90
FC-IFFNN-BC	NA	90.58
HW-IFFNN-BC	NA	95.50

1) Quantitative Analysis: The Accuracy@N of LR, SR, and the IFFNNs on INBEN are reported in Table IV. As shown, the Accuracy@N of IFFNNs is always larger than 90%, which means when a sample is correctly classified, the IFFNNs can correctly point out the features in the patterns that determine its class. This indicates that the explanations provided by them are accurate.

We can also see that the explanations given by IFFNNs are even more accurate than those given by LR and SR. The reason is that the INBEN dataset we created is non-linear, thus these linear models cannot always capture the patterns that determine the class of a sample. To be more specific, LR and SR can only model the relation between a feature and a class independently, however, the patterns require the models to be able to model the co-occurrences of different features. Multi-layer neural networks model interactions of different features through the computations in the hidden layers. This also reflects the fact that as multi-layer networks, the IFFNNs have the pattern recognition ability of non-linear models.

2) Qualitative Analysis: In addition to the quantitative evaluation, we also qualitatively evaluate the models on MNIST to manually check whether the explanations make sense. We show the importance of a pixel to a class in a greyscale image that has the same shape as the original image, and the greyscale of a pixel is the importance of the pixel in the same position. The greyscale of the background in the original images is always 0, so their importance is also 0. Therefore, the pixels that are lighter than the background provide a positive contribution to the class and the darker pixels provide a negative contribution. We use the scenario with only ”0” and ”1” for the evaluation because there are areas of the images that only contain white pixels for only one of them and these pixels are good indicators of the digits.

Figure 1 show some images from the test set and the importance images of them for all classes. We can see that for the images of ”0”, the important pixels for the right class (i.e., ”0”) determined by all IFFNNs focus on the pixels of the left and right parts of the circle. This makes sense because ”1” is usually close to a vertical bar, so its white pixels rarely appear in those areas of the images of ”1”. Therefore, it makes sense that white pixels appearing in these areas contribute more to

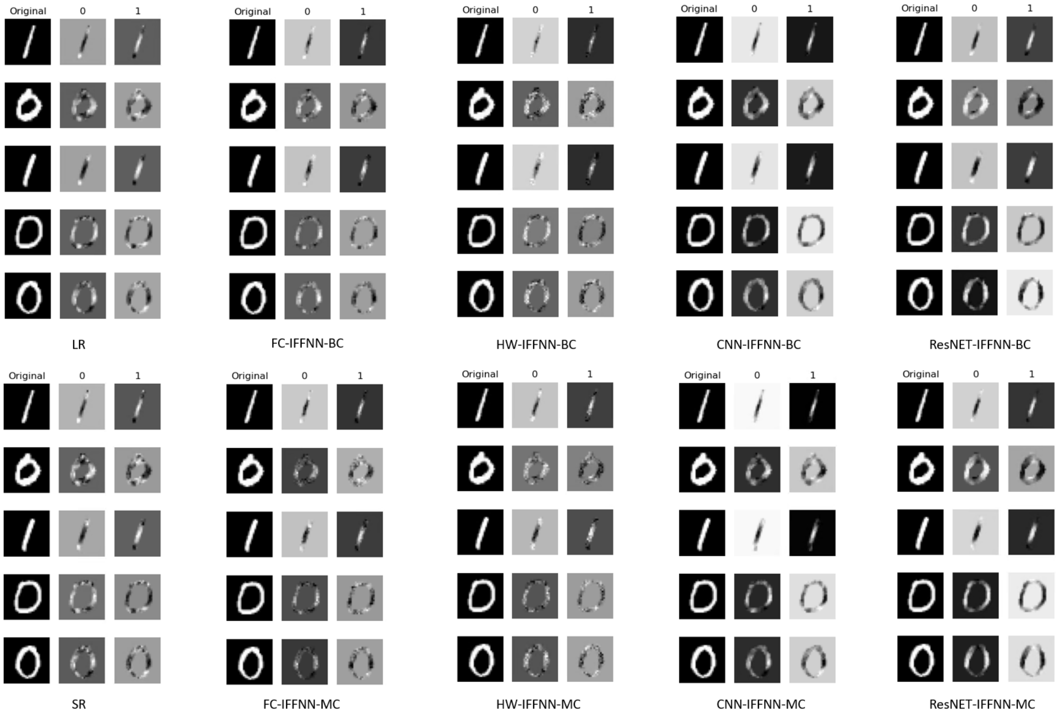

Fig. 1. Examples of images and the explanations for the classifications on MNIST with only 0 and 1.

the class of ”0”. The important pixels for images of ”1” are more concentrated in the center part of the stroke. This is also valid because there are rarely white pixels in the center areas of images of ”0”.

VII. Conclusion

In this paper, we propose ways to generalize the IFFNN proposed by Li et al. [17] to multi-class classification and any type of feedforward neural networks. We also conduct comprehensive experiments to evaluate the classification performance and interpretability of the IFFNNs. We reached the conclusion that the IFFNNs achieve similar classification accuracy as their non-interpretable feedforward neural network counterparts and provide meaningful explanations. Therefore, the generalized IFFNN architecture is an excellent choice for real-world applications when explanations for classification results are expected for various reasons.

Acknowledgment

This research was funded by NSERC Discovery Grants (RGPIN-2018-03872), Canada Research Chairs Program (950-230623), and the Canadian National Defence Innovation for Defence Excellence and Security (IDEaS W7714-217794/001/SV1). The IDEaS program assists in solving some of Canada’s toughest defence and security challenges. The Titan Xp used for this research was donated by the NVIDIA Corporation.

References

[1] H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and T. Zhao, “Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization,” *arXiv preprint arXiv:1911.03437*, 2019.

[2] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, “Sharpness-aware minimization for efficiently improving generalization,” *arXiv preprint arXiv:2010.01412*, 2020.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, “Language models are few-shot learners,” *arXiv preprint arXiv:2005.14165*, 2020.

[4] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, “Informer: Beyond efficient transformer for long sequence time-series forecasting,” in *Proceedings of AAAI*, 2021.

[5] E. Choi, M. T. Bahadori, J. A. Kulas, A. Schuetz, W. F. Stewart, and J. Sun, “Retain: An interpretable predictive model for healthcare using reverse time attention mechanism,” *arXiv preprint arXiv:1608.05745*, 2016.

[6] C. Rudin, “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead,” *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, 2019.

[7] C. Molnar, *Interpretable machine learning*. Lulu.com, 2020.

[8] A. Das and P. Rad, “Opportunities and challenges in explainable artificial intelligence (xai): A survey,” *arXiv preprint arXiv:2006.11371*, 2020.

[9] P. Linardatos, V. Papastefanopoulos, and S. Kotsiantis, “Explainable ai: A review of machine learning interpretability methods,” *Entropy*, vol. 23, no. 1, p. 18, 2021.

[10] H. Laurent and R. L. Rivest, “Constructing optimal binary decision trees is np-complete,” *Information processing letters*, vol. 5, no. 1, pp. 15–17, 1976.

[11] M. Sundararajan, A. Taly, and Q. Yan, “Axiomatic attribution for deep networks,” in *International Conference on Machine Learning*. PMLR, 2017, pp. 3319–3328.

[12] A. Datta, S. Sen, and Y. Zick, “Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems,” in *2016 IEEE symposium on security and privacy (SP)*. IEEE, 2016, pp. 598–617.

[13] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1885–1894.

[14] P. Adler, C. Falk, S. A. Friedler, T. Nix, G. Rybeck, C. Scheidegger, B. Smith, and S. Venkatasubramanian, "Auditing black-box models for indirect influence," *Knowledge and Information Systems*, vol. 54, no. 1, pp. 95–122, 2018.

[15] G. Su, D. Wei, K. R. Varshney, and D. M. Malioutov, "Interpretable two-level boolean rule learning for classification," *arXiv preprint arXiv:1511.07361*, 2015.

[16] M. T. Ribeiro, S. Singh, and C. Guestrin, "Nothing else matters: model-agnostic explanations by identifying prediction invariance," *arXiv preprint arXiv:1611.05817*, 2016.

[17] M. Q. Li, B. C. M. Fung, P. Charland, and S. H. H. Ding, "I-MAD: Interpretable malware detector using Galaxy Transformers," *Computers & Security (COSE)*, vol. 108, no. 102371, pp. 1–15, September 2021.

[18] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[19] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st international conference on neural information processing systems*, 2017, pp. 4768–4777.

[20] S. I. Amoukou, N. J. Brunel, and T. Salaün, "The shapley value of coalition of variables provides better explanations," *arXiv preprint arXiv:2103.13342*, 2021.

[21] A. C. Chaves, M. M. Vellasco, and R. Tanscheit, "Fuzzy rule extraction from support vector machines," in *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*. IEEE, 2005, pp. 6–pp.

[22] D. Alvarez Melis and T. Jaakkola, "Towards robust interpretability with self-explaining neural networks," *Advances in neural information processing systems*, vol. 31, 2018.

[23] S. Wisdom, T. Powers, J. Pitton, and L. Atlas, "Interpretable recurrent neural networks using sequential sparse recovery," *arXiv preprint arXiv:1611.07252*, 2016.

[24] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *2010 IEEE Computer Society Conference on computer vision and pattern recognition*. IEEE, 2010, pp. 2528–2535.

[25] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2018–2025.

[26] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[28] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[30] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," *Neural Information Processing Systems NIPS 2017 Autodiff Workshop*, 2017.

[31] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

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