Machine Learning
https://doi.org/10.1007/s10994-023-06428-4

Explaining neural networks without access to training data

Sascha Marton[1] · Stefan Lüdtke[2] · Christian Bartelt[1] · Andrej Tschalzev[1] · Heiner Stuckenschmidt[3]

Received: 5 June 2023 / Revised: 2 August 2023 / Accepted: 7 October 2023

Abstract
We consider generating explanations for neural networks in cases where the network's training data is not accessible, for instance due to privacy or safety issues. Recently, Interpretation Nets ($\mathcal{I}$-Nets) have been proposed as a sample-free approach to post-hoc, global model interpretability that does not require access to training data. They formulate interpretation as a machine learning task that maps network representations (parameters) to a representation of an interpretable function. In this paper, we extend the $\mathcal{I}$-Net framework to the cases of standard and soft decision trees as surrogate models. We propose a suitable decision tree representation and design of the corresponding $\mathcal{I}$-Net output layers. Furthermore, we make $\mathcal{I}$-Nets applicable to real-world tasks by considering more realistic distributions when generating the $\mathcal{I}$-Net's training data. We empirically evaluate our approach against traditional global, post-hoc interpretability approaches and show that it achieves superior results when the training data is not accessible.

Keywords Explainable AI (XAI) · Interpretability · Neural networks · Decision trees

Editors: Vu Nguyen, Dani Yogatama.

✉ Sascha Marton
sascha.marton@uni-mannheim.de

Stefan Lüdtke
stefan.luedtke@uni-rostock.de

Christian Bartelt
christian.bartelt@uni-mannheim.de

Andrej Tschalzev
andrej.tschalzev@uni-mannheim.de

Heiner Stuckenschmidt
heiner.stuckenschmidt@uni-mannheim.de

[1] Institute for Enterprise Systems, University of Mannheim, L 15, 1, 68161 Mannheim, Germany

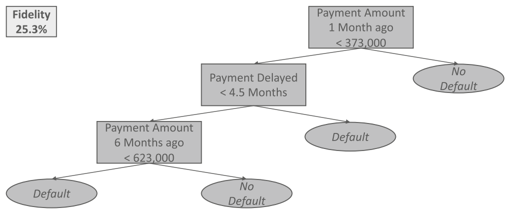
[2] Institute for Visual & Analytic Computing, University of Rostock, Albert-Einstein-Str. 21, 18059 Rostock, Germany

[3] Data and Web Science Group, University of Mannheim, B 6, 26, 68159 Mannheim, Germany

1 Introduction

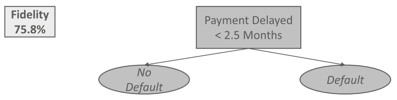
Artificial neural networks (NNs) achieve impressive results for various modeling tasks (LeCun et al., 2015; Wang et al., 2020). A downside of their superior performance is the interpretability of the learned models. However, in many domains, it is crucial to understand the function learned by a NN (Samek et al., 2019; Molnar, 2020). A common approach to tackle the problem of interpretability without sacrificing the superior performance is using a global surrogate model as gateway to interpretability (Molnar, 2020). Most global surrogate approaches use a distillation procedure to learn the surrogate model based on the predictions of the NN (Molnar, 2020; Frosst & Hinton, 2017). Therefore, they query the NN based on a representative set of samples and the resulting input–output pairs are then used to train the surrogate model. This representative sample usually comprises the training data of the original model (Molnar, 2020; Lopes et al., 2017). However, there are many cases where the training data cannot easily be exposed due to privacy or safety concerns (Lopes et al., 2017; Bhardwaj et al., 2019; Nayak et al., 2019). Without having access to the training data, traditional approaches can fail to provide meaningful explanations since the querying strategy can easily miss dense regions of the training data such that the resulting samples are a poor approximation of the true function, as we will show in the following example:

Example 1 The Credit Card Default dataset (Yeh & Lien, 2009) comprises personal, confidential data which usually cannot be exposed to external authorities. The task is to predict whether a client will default the payment in the current month, which can be solved efficiently using NNs. To gain insight into the decision-making process of the NN, we can learn a global surrogate model. Unfortunately, if the training data is not accessible, we need to query the model using random samples and therefore can't ensure that the NN is properly queried. Figure 1a shows a scenario, where the explanation generated by a sample-based distillation without training data contains a misconception: It encodes the rule that we should always predict *No Default* if the payment amount of the last month is larger than 373, 000 without taking the payment history of the client into account. This mismatch between network and surrogate model is also reflected in the low fidelity between the network and surrogate model on the training data. Since we can't compute this fidelity in a real scenario, such misconceptions might go unnoticed, which can lead to wrong assumptions about what the network actually learned.



Fidelity 25.3%
Payment Amount 1 Month ago < 373,000
Payment Delayed < 4.5 Months
No Default
Payment Amount 6 Months ago < 623,000
Default
Default
No Default

(a) Sample-Based DT



(b) $\mathcal{I}$-Net DT

Fig. 1 Explaining NNs for credit card default prediction. The DT on the left is learned by a sample-based distillation without access to training data, and the DT on the right is predicted by the $\mathcal{I}$-Net. The $\mathcal{I}$-Net DT makes reasonable splits and achieves a significantly higher fidelity on the real data

As shown in Example 1, knowing the training data is crucial for sample-based methods and without access, it is often not possible to generate reasonable explanations. Recent approaches tackle this issue by using only a subset of the training data and/or layer activations to generate a representative set of samples (Lopes et al., 2017; Bhardwaj et al., 2019; Nayak et al., 2019). However, they still rely on a proper querying of the model and use a sample-based distillation. In contrast, the Interpretation Net ($\mathcal{I}$-Net) approach introduced by Marton et al. (2022) is a sample-free approach that only accesses the network parameters and therefore does not rely on a proper querying. This is achieved by using a second NN (the $\mathcal{I}$-Net) which learns a mapping from the network parameters to a human-understandable representation of the network function. Following this approach, we can generate reasonable explanations, even when the training data is not accessible, as shown in Example 1.

The $\mathcal{I}$-Net was originally devised for regression tasks and lower-order polynomials as a surrogate model. In this paper, we extend $\mathcal{I}$-Nets to classification tasks and use decision trees (DTs) as intrinsically interpretable surrogate models. DTs are frequently used as an explainable model for classification tasks since they make hierarchical decisions and therefore are easy to comprehend for humans (Frosst & Hinton, 2017). Furthermore, in recent literature, soft DTs (SDTs) are successfully used as interpretable surrogate model (Frosst & Hinton, 2017). While SDTs make multivariate splits, they usually achieve a higher fidelity than standard DTs, but also have a higher level of complexity.

In this paper, we make the following key contributions:

- We work out the importance of the data distribution to assess reasonable explanations (Sect. 2.2 and Sect. 4.2.1).
- We extend $\mathcal{I}$-Nets to classification tasks and propose an improved data generation method, making $\mathcal{I}$-Nets applicable in real-world scenarios (Sect. 3.1).
- We present an $\mathcal{I}$-Net design that is able to represent standard DTs and SDTs as surrogate model (Sect. 3.2) with a high fidelity.

We empirically evaluate our approach against a sample-based distillation and show that it achieves superior results when training data is not accessible (Sects. 4.2.1–4.2.3). Additionally, we present a case study (Sect. 4.2.4) illustrating that sample-based approaches are ineffective in producing satisfactory explanations when the training data is unavailable. In contrast, $\mathcal{I}$-Nets as sample-free approach are capable of generating reasonable explanations even when the training data is not available.

2 $\mathcal{I}$-Nets as sample-free surrogate models

In this section, we summarize the task of explaining NNs, focusing on the case where the networks' training data is not available, followed by a brief introduction to $\mathcal{I}$-Nets. For a more in-depth explanation of the $\mathcal{I}$-Nets, we refer to Marton et al. (2022).

2.1 Global explanations for NNs

We can formalize the task of explaining NNs globally as finding a function $g : X \rightarrow P(Y|X)$ (i.e., a surrogate model) that approximates the decision function of a NN $\lambda : X \rightarrow P(Y|X)$, such that $\forall \mathbf{x} \in X : \lambda(\mathbf{x}) \approx g(\mathbf{x})$ for some data X and corresponding labels Y. Since the $\mathcal{I}$

-Net approach implements a learning task, it is convenient to distinguish between the functions λ and g and their representations $\theta_\lambda \in \Theta_\lambda$ and $\theta_g \in \Theta_g$ (Marton et al., 2022). The representation θ_λ consists of the network parameters, i.e., the weights and biases of the NN. Similarly, θ_g is the parameter vector of the surrogate model and depends on the selected function family.

The process of generating explanations can be formalized as a function $\mathcal{I} : \Theta_\lambda \rightarrow \Theta_g$ that maps representations of λ to representations of g (Marton et al., 2022). Traditional approaches for generating global surrogate models post-hoc implement $\mathcal{I}$ via a sample-based procedure. They generate a new dataset, where the labels are obtained by querying λ based on a set of data points. In the next step, a surrogate model is trained using the generated dataset, maximizing the fidelity between λ and g. As shown by Marton et al. (2022), this process is time-consuming, which can be a huge drawback if timely explanations are required, as for instance in an online learning scenario. Additionally, it strongly depends on the data used for querying the model. Information that is not properly queried cannot be contained in the explanation, as shown in Example 1. Therefore, in the literature it is suggested to use the original training data or data from the same distribution for querying the model (Molnar, 2020; Lopes et al., 2017), which usually yields to meaningful explanations. However, if the training data is not accessible or not existing anymore, the model has to be queried based on some sampled data. In this case, it is often not possible to generate meaningful explanations with sample-based approaches, since we cannot ensure a proper querying and therefore the explanation does not necessarily focus on the relevant aspects.

2.2 Reasonable explanations

In the following, we discuss what constitutes a meaningful explanation for a NN. In general, the decision boundary of the surrogate model should closely match the decision boundary of the network we want to interpret to achieve a high fidelity. However, we argue that it is necessary to take the data distribution into account as well: A decision boundary should assign as many samples as possible to the correct class. Therefore, it is crucial that the decision boundary is composed correctly in the areas where many samples are located. Accordingly, for a reasonable explanation, the decision boundary should match the model we want to interpret especially in regions where many samples are located, while it is less important that the decision boundaries match in regions with low data density. In other words, we are less interested in an explanation that shows us how the model behaves when making predictions on data points that do not occur in reality. This concept is visualized in Fig. 2. In Sect. 4.2.1, we show that traditional, sample-based approaches cannot generate such reasonable explanations when the training data is not available.

2.3 Explanations for neural networks by neural networks

To renounce the dependency on a proper querying of the model, we can implement $\mathcal{I}$ as a NN (Marton et al., 2022) and learn how to distill a model. Accordingly, we transform the task of explaining NNs into a machine learning task. The concept of $\mathcal{I}$-Nets involves two key steps and is visualized in Fig. 3:

1. We train a set of NNs on synthetic data and extract their parameters.
2. We train a second NN, the $\mathcal{I}$-Net, using the parameters extracted in the first step as input data.

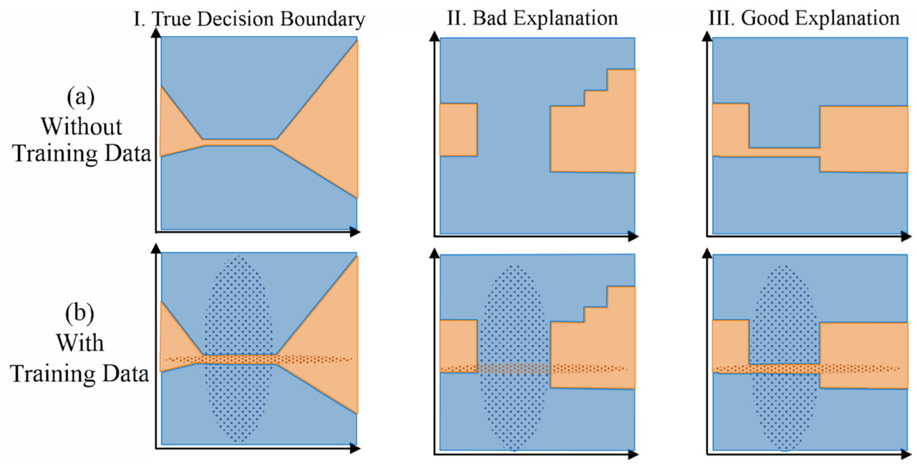


Fig. 2 Good and bad explanations. This figure shows an example of a bad (II) and a good (III) explanation for a given model we want to interpret (I). Without considering the data that the model was trained on (a), the explanation shown in (II) appears to be reasonable, since the decision boundary of the explanation has a large overlap with the decision boundary of the original model. However, when taking the training data distribution into account (b), we can see that the small area in the center of the picture is very important, since most samples are located in that area. This fact is neglected by the explanation shown in (II) and only considered by the explanation shown in (III)

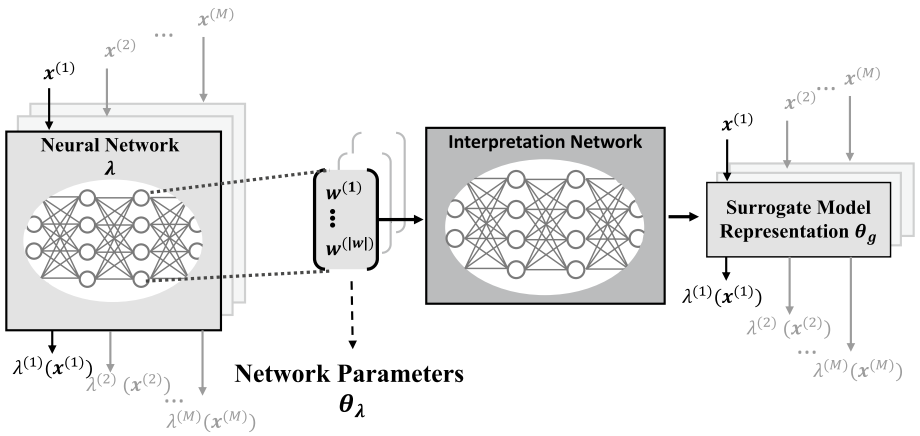


Fig. 3 Overview of the $\mathcal{I}$-Net approach. The $\mathcal{I}$-Net translates network parameters $\theta_\lambda = w^{(1)}, \dots, w^{(|w|)}$ of a NN into a surrogate model representation θ_g (e.g. a DT) (Marton et al., 2022)

Thereby, no supervision in terms of actual labels is required during the training. Instead, the fidelity between λ and g is computed using a distance measure over a set of data points in the loss function. Since the loss is only computed during the training, no data except the network parameters is required when applying the $\mathcal{I}$-Net. This is a major advantage to sample-based approaches, where the training data is required for each network we want to interpret. Accordingly, to generate an explanation, $\mathcal{I}$-Nets only need access to the network parameters and therefore, the approach can be applied in scenarios where the training data is not accessible without suffering a performance deficit. The most important part of the $\mathcal{I}$ -Net approach is an efficient training procedure. As for most machine learning tasks, good training data (in our case, a set of network parameters Θ_λ) is crucial. Therefore, we present an improved data generation method making $\mathcal{I}$-Nets applicable for real world scenarios in Sect. 3.1.

3 $\mathcal{I}$-Nets for decision trees

In this section, we present the main contributions of our paper. Marton et al. (2022) argue that $\mathcal{I}$-Nets can be trained solely based on synthetic data. However, it is crucial that this synthetic data comprises reasonable learning problems to assure that an application of the $\mathcal{I}$-Net is possible in a real-world setting. To achieve this, we will introduce an improved data generation method that considers multiple data distributions to create reasonable learning tasks (Sect. 3.1). Furthermore, Marton et al. (2022) focus solely on regression tasks. We extend $\mathcal{I}$-Nets to classification tasks and therefore present an adjusted loss function. In general, the $\mathcal{I}$-Net framework can be applied to arbitrary function families for g, as long as we can define a suitable representation θ_g. In Sect. 3.2, we introduce different DT variants and propose corresponding representations θ_g that allow an efficient training.

3.1 Improved data generation and training procedure

Data generation method The data generation method proposed by Marton et al. (2022) focuses on maximizing the performance of the $\mathcal{I}$-Net during training by learning functions λ that are similar to the function family of g. This is achieved by randomly sampling a set of functions from the family of g. These functions are queried to generate labels for a uniformly sampled dataset, which is used to learn λ. This procedure ensures that the functions λ are representative of g, enabling an efficient training. However, a high training performance does not necessarily mean that the model generalizes well to unseen data, i.e., NNs trained on real-world datasets. Additionally, Marton et al. (2022) use a uniform data distribution to query λ for the fidelity calculation in the $\mathcal{I}$-Net loss. However, if we only consider a uniform distribution during the training of the $\mathcal{I}$-Net, we might not be able to make reasonable predictions if the network we want to interpret was trained using data from a substantially different distribution, as we will show in Sect. 4.2.3. This problem is related to the general problem that occurs for a machine learning task, if the data we are actually interested in (i.e., the test data) is from a different distribution than the data used for training the model.

To tackle this issue, we propose using multiple, different distributions during the training of the $\mathcal{I}$-Net to make it more robust and therefore applicable on real-world datasets. In this process, we can also utilize the fact that an $\mathcal{I}$-Net can be trained in a controlled, synthetic environment: For each $\theta_\lambda \in \Theta_\lambda$, we know the data that was used for learning λ. Therefore, we can use these data points to compute the $\mathcal{I}$-Net loss on a meaningful set of samples during the training. The $\mathcal{I}$-Net utilizes this additional knowledge to generalize. Since the loss is only calculated during training, we can generate meaningful explanations solely based on the network parameters θ_λ. In summary, the process of generating training data for $\mathcal{I}$-Net Θ_λ involves three steps:

1. Generate N datasets $\mathcal{D}_\lambda = \left\{ \left(\mathbf{x}^{(j)}, y^{(j)} \right) \right\}_{j=1}^{M}$, each comprising M samples.

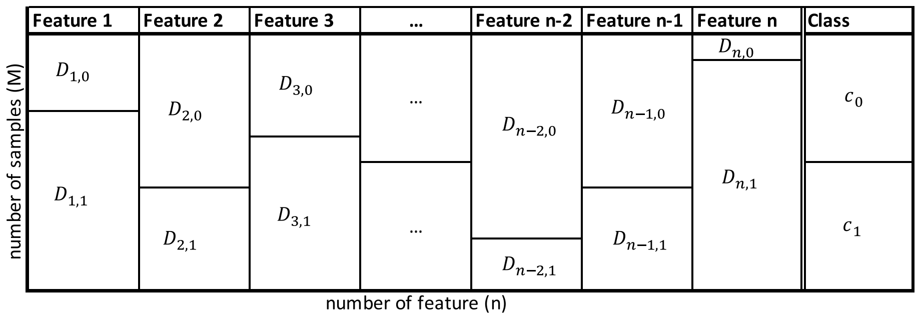


Fig. 4 Data generation visualization. This graphic visualizes the generation of a balanced dataset used for training a network λ where $D \in \{\mathcal{U}, \mathcal{N}, \Gamma, \mathrm{B}, \mathrm{Poi}\}$. For each feature, a random distribution with two random parametrizations is chosen and a random number of data points is sampled from each distribution

2. For each dataset $\mathcal{D}_\lambda$, train a network λ, extract the network parameters θ_λ and add them to the training set Θ_λ.
3. Use Θ_λ to train an $\mathcal{I}$-Net for the respective function family.

Our improved data generation is visualized in Fig. 4 and formalized in Algorithm 1.

Algorithm 1 Generate Data

```
1: function GENERATE($n, D, M$)
2:     for $i = 1, \dots, n$ do
3:         $D_i \sim \mathcal{U}\{\mathcal{U}, \mathcal{N}, \Gamma, \mathrm{B}, \mathrm{Poi}\}$
4:         $M_0 \sim \lceil \mathcal{U}(1, M-1) \rceil$
5:         $\mathbf{p_0} \sim \mathcal{U}(0, p)$
6:         $\mathbf{p_1} \sim \mathcal{U}(0, p)$
7:         for $j = 1, \dots, M_0$ do
8:             $x_i^{(k)} \sim D_i(\mathbf{p_0})$
9:         end for
10:        for $j = 1, \dots, M - M_0$ do
11:            $x_i^{(M_0+j)} \sim D_i(\mathbf{p_1})$
12:        end for
13:        $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i - min(\mathbf{x}_i)}{max(\mathbf{x}_i) - min(\mathbf{x}_i})$
14:    end for
15:    for $j = 1, \dots, \lceil \frac{M}{2} \rceil$ do
16:        $y^{(j)} \leftarrow 0$
17:    end for
18:    for $j = 1, \dots, \lfloor \frac{M}{2} \rfloor$ do
19:        $y^{(j)} \leftarrow 1$
20:    end for
21:    $\mathcal{D} \leftarrow \{\mathbf{x}^{(j)}, y^{(j)}\}_{j=1}^{M}$
22:    return $\mathcal{D}$
23: end function
```

For each feature i, we sample data points from one distribution with k different parametrizations, where k is the number of classes. For this paper, we focus on binary classification tasks and therefore set $k = 2$. The distribution $D_{i,k}$ is sampled uniformly from $\{\mathcal{U}, \mathcal{N}, \Gamma, \mathrm{B}, \mathrm{Poi}\}$ for each feature. The distributions were selected to cover a wide range of diverse distributions that are reasonable for many real-world phenomena (Leemis & McQueston, 2008; Mun, 2015). The parametrization for the distributions $D_{i,0}$ and $D_{i,1}$ are again randomly drawn from $\mathcal{U}(0, p)$, where p is a hyperparameter for the data generation procedure. The number of samples is selected randomly, where $M_0 = \lceil \mathcal{U}(1, M - 1) \rceil$ data points are sampled from $D_{i,0}$ and $M_1 = M - M_0$ data points are sampled from $D_{i,1}$. The generated datasets are balanced and for each feature and the first $\frac{M}{2}$ data points are associated with *Class 0* and the subsequent $\frac{M}{2}$ data points are associated with *Class 1*.

We can see the proposed data generation method as a generalization of common, synthetic machine learning problems that is able to generate more realistic tasks. We also want to note that even though the data generation focuses on balanced datasets, we can still use $\mathcal{I}$-Nets to interpret models for imbalanced real-world datasets, as we will show in our evaluation (Sect. 4).

Adjusted loss function While Marton et al. (2022) focused on regression tasks, we extend their approach to binary classification tasks within this paper. Therefore, we adjust the loss function by using binary cross-entropy to quantify the fidelity between λ and g as

$$\mathrm{BC}(\theta_\lambda, \theta_g) = \frac{1}{M} \sum_{j=1}^{M} \lfloor \lambda(\mathbf{x}^{(j)}) \rceil \times log(g(\mathbf{x}^{(j)})) + (1 - \lfloor \lambda(\mathbf{x}^{(j)}) \rceil) \times log(1 - g(\mathbf{x}^{(j)})), \tag{1}$$

where $\lfloor \cdot \rceil$ denotes rounding to the closest integer. The $\mathcal{I}$-Net loss for a set of network parameters $\Theta_\lambda = \{\theta_\lambda^{(i)}\}_{i=1}^{N}$ is then computed as

$$\mathcal{L}_{\mathcal{I}} = \frac{1}{|\Theta_\lambda|} \sum_{\theta_\lambda \in \Theta_\lambda} \mathrm{BC}(\theta_\lambda, \mathcal{I}(\theta_\lambda)). \tag{2}$$

3.2 Function families and $\mathcal{I}$-Net output representation

$\mathcal{I}$-Nets for standard DTs The first function family we will consider as surrogate models are standard DTs. DTs and decision rules are frequently used as explanations, since they are comparatively easy to understand for most humans (Molnar, 2020).

$\mathcal{I}$-Nets require a suitable representation θ_g for standard DTs to enable efficient learning. Specifically, we need a one-dimensional encoding of internal and leaf nodes, as shown in Fig. 5. The inner node of a DT comprises two major parts: The first part is the feature that is considered within the split, and the second part is the split value. The operator is fixed to less (<) as it is common practice for representing DTs. Furthermore, we can fix the left path to be the true path and the right path as the false path. The feature x_i considered in the split can be defined by enumerating the features, where $i \in \{0, 1, \dots, n\}$. We can represent this using n neurons and a softmax activation for each inner node (i.e., we can see it as a classification task for which feature to consider at a certain split).

For the split value, we can assume that all features are scaled to be within [0, 1], as it is common practice. To represent this in the $\mathcal{I}$-Net output, we can use sigmoid activations, to constraint the output interval. However, due to the functional form of the sigmoid activation, the $\mathcal{I}$-Net prefers split values close to 0.5. To counteract this tendency, we used a squeezed sigmoid activation, which we define as $\frac{1}{1+e^{-3x}}$. This supports the $\mathcal{I}$-Net in

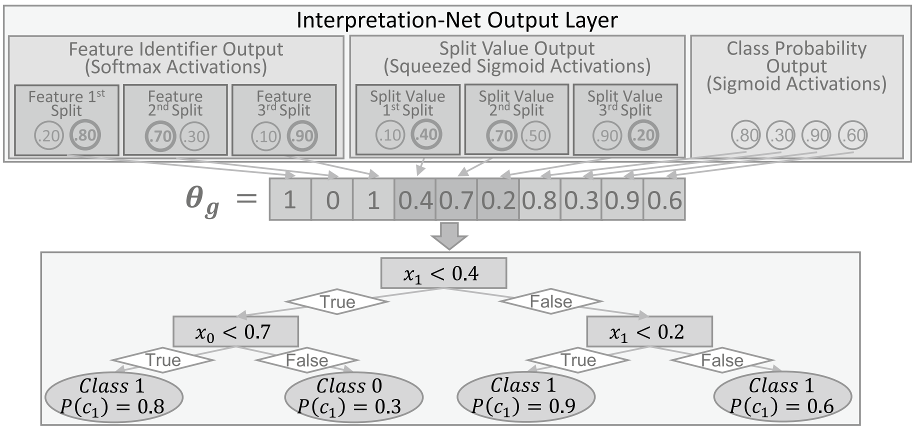


Fig. 5 Exemplary $\mathcal{I}$-Net output for DTs. The DT representation is predicted by the $\mathcal{I}$-Net using three separate output layers with different activation functions. The output shows a DT of depth two for a binary classification task with two features

choosing more distinct split values. Furthermore, the output layer does not comprise one split value for each split, but n split values for each split (one for each feature). To construct the DT, we always use the split value at the index indicated by the feature identifier. This design choice is influenced by the fact that we always need to consider the meaning of a split value in context with the corresponding feature. In other words, while the split value 0.7 might be a reasonable threshold for the feature x_0, it might not be reasonable at all for the feature x_1. Designing the $\mathcal{I}$-Net output with one split value for each feature and each inner node, we can make this interaction easier to learn.

In a standard DT, the leaf nodes comprise the decision for a certain path (i.e., the class to be predicted). However, to compute the $\mathcal{I}$-Net loss in Eq. 2, it is necessary that g has probabilities as an output. Therefore, we adjust the DTs to not just have a class in the leaf node, but a probability. This is similar to the purity in the leaf node of a DT, which is also often used as a gateway to predicting probabilities using a standard DT. In a binary classification case, we can use a single value to represent the probabilities of predicting *Class 1* and thereby, the probability of *Class 0* is the complementary probability. In the output layer of the $\mathcal{I}$-Net, we can represent this using a total of 2^d neurons with sigmoid activations (one neuron for each leaf node). This can easily be extended for a multi-class classification problem with k classes by using $k \times 2^d$ neurons and one softmax activation over k neurons.

$\mathcal{I}$-Nets for SDTs SDTs were proposed to overcome the interpretability problem that arises from distributed hierarchical representations when using NNs by expressing the knowledge using hierarchical decisions of a tree-based structure (Frosst & Hinton, 2017). Unlike standard DTs, SDTs do not make hard true/false splits at each internal node, but use soft decisions associated with probabilities for each path. In the following, we will shortly introduce the functioning of SDTs. For a more in-depth description, especially concerning the learning algorithm, we refer to Frosst and Hinton (2017).

For SDTs, each internal node j comprises a filter $\mathbf{w}^j$ and a bias b^j. While the bias is a single, learned value, the filter consists of one value for each feature. Accordingly, in contrast to a standard DT with univariate decisions, a SDT has a multivariate decision at each internal node. This comes with a significantly higher model complexity, especially with an increasing number of features. At each internal node, the probability of taking the right branch is calculated as

$$P^j(\mathbf{x}) = S\big(\mathbf{x}w^j + b^j\big), \tag{3}$$

where x is an input sample and S is a sigmoid function defined as $S(x) = \frac{1}{1+e^{-x}}$. Each leaf node l comprises a probability distribution $Q^l_.$, which is defined as

$$Q^l_k = \frac{e^{\phi^l_k}}{e^{\phi^l_0} + e^{\phi^l_1}} \tag{4}$$

for the binary case. Thereby, $k \in \{0, 1\}$ is the output class and $\phi^l_.$ is a learned parameter for each leaf l. Usually when using SDTs, there is not only a single leaf node considered when making a prediction, but all leaf nodes are multiplied with their path probabilities to calculate the final probability distribution. However, Frosst and Hinton (2017) suggest increasing the interpretability of SDTs by just considering the path with the maximum path probability when calculating the final probability distribution. Since this does not significantly affect the performance, we will only consider SDTs using the maximum path probability in this paper.

To use SDTs as surrogate models within the $\mathcal{I}$-Net framework, we again need a suitable representation θ_g. Fortunately, the encoding for SDT shown in Fig. 6 is straightforward: We can represent the internal nodes with n output neurons for the filter (one for each feature) and one output neuron for the bias. Since there are no specific ranges for the filter and bias value in the SDT, we use linear activations. The same accounts for $\phi^l_.$, where we need k output neurons for each leaf node. Again, we can use linear activations here, since the final probabilities are calculated by Q^l_k and no specific range for $\phi^l_.$ is required.

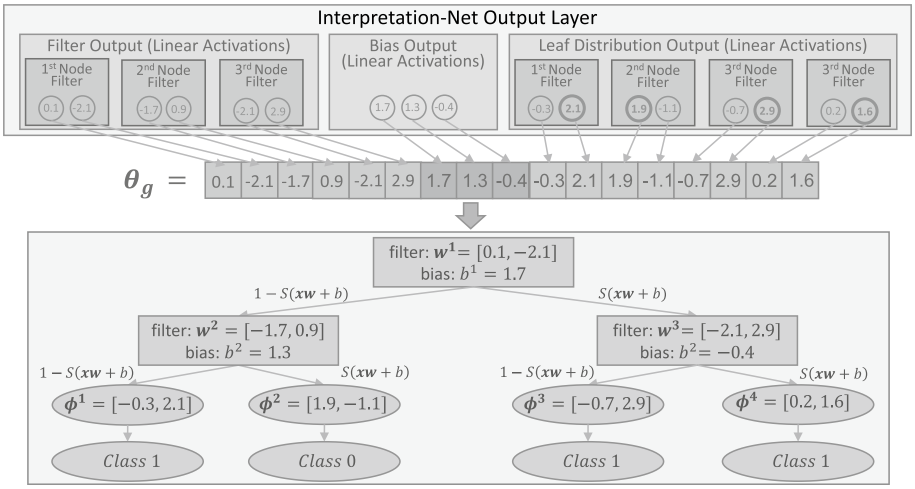


Fig. 6 Exemplary $\mathcal{I}$-Net Output for SDTs. The SDT representation is predicted by the $\mathcal{I}$-Net using three separate output layers with linear activation functions. Here, we show a SDT of depth two for a binary classification task with two features

4 Evaluation

The goal of our evaluation is to show that $\mathcal{I}$-Nets are able to interpret NNs trained on real-world datasets without requiring access to the training data, and achieve a higher fidelity than sample-based approaches in most of the cases. Therefore, we will address the following in our evaluation:

- We illustrate which effects occur once the training data is not accessible for querying the model and thereby show that it is crucial for sample-based approaches to access the training data (Sect. 4.2.1).
- We evaluate the fidelity of the $\mathcal{I}$-Net in comparison to sample-based approaches on real-world datasets if the training data is not accessible (Sect. 4.2.2).
- We perform an ablation study showing the impact of our improved data generation method for real-world datasets (Sect. 4.2.3).
- We present a case study of credit card default prediction, comparing the explanations for a NN generated by the $\mathcal{I}$-Net and sample-based approaches (Sect. 4.2.4).

4.1 Experimental setup

In our experiments, we compare $\mathcal{I}$-Nets with standard distillation approaches for a scenario where the training data is not available. For $\mathcal{I}$-Nets, we used the representations Θ_g described in Sect. 3.2. The sample-based distillation was conducted as follows:

- **Standard DTs:** We used the implementation from sklearn, which uses the CART algorithm for DT induction (Breiman et al., 1984).
- **SDTs:** We used the algorithm proposed by Frosst and Hinton (2017).

Since we assume that the training data is not available, we needed to generate data to query the NN for the distillation using sample-based approaches. Therefore, we selected three sampling strategies for generating the query data as benchmarks:

1. **Multi-Distribution:** According to Algorithm 1, i.e., considering different data distributions to allow for a fair comparison with the $\mathcal{I}$-Net.
2. **Standard Uniform:** A standard uniform distribution $U(0, 1)$.
3. **Standard Normal:** A standard normal distribution $\mathcal{N}(0, 1)$.

For each sampling strategy, we sampled 10, 000 data points. Increasing the number of sampling points further did not enhance the fidelity of sample-based approaches, but only increased their runtime.

The network parameters Θ_λ for training the $\mathcal{I}$-Net were generated using data according to Algorithm 1. We excluded all datasets that were linearly separable during the data generation to focus on more complex and reasonable datasets. The hyperparameter p which defines the maximum value for the distribution parameters was fixed to 5 for all experiments. The $\mathcal{I}$-Net hyperparameters were optimized using a greedy neural architecture search according to Jin et al. (2019). We selected one $\mathcal{I}$-Net architecture for each of the three function families. The code of our implementation along with all datasets and used hyperparameters is available under: https://github.com/s-marton/explaining-neural-networks-without-training-data.

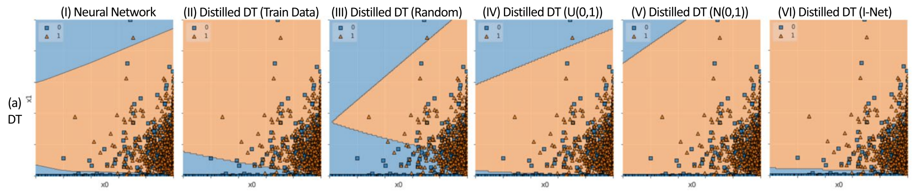
4.2 Experimental results

In the following, we will summarize our results and findings. The fidelity between the surrogate model and the NN was calculated based on the test split of the original data.

4.2.1 Visual evaluation for different distributions

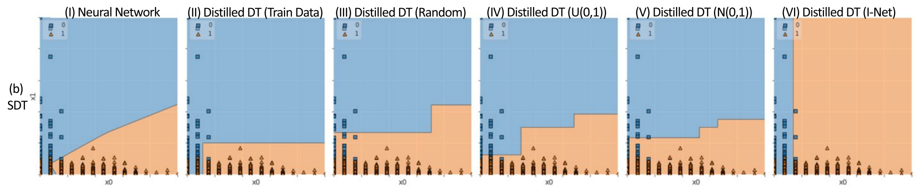
In this experiment, we assess the importance of knowing the distribution of the training data in a controlled, synthetic setting. We use a two-dimensional dataset to allow a visual comparison of the decision boundaries (Fig. 7). The data used for training the NNs for this experiment was generated randomly according to Algorithm 1 but is distinct to the data used for training the $\mathcal{I}$-Net to ensure a fair comparison.

Figure 7a shows a decision boundary learned by a NN that ranges from the bottom left corner to the middle right. Thereby, many data points that were assigned to *Class 0* by the NN are located in the bottom left corner. In contrast, the top right part contains few to no data points. When the training data is available (II), the standard DT learned a decision boundary that closely matches the decision boundary of the NN, including the area in the bottom left. However, if the training data is not available, the sample-based approach (III–V) only comprises the large area towards the top and neglects the small area



Distribution 1,0:	$Poi(0.469)$	n = 3262
Distribution 1,1:	$Poi(4.784)$	n = 1738
Distribution 2,0:	$\Gamma(0.114,\ 4.581)$	n = 817
Distribution 2,1:	$\Gamma(0.562,\ 0.385)$	n = 4183
Performance NN (I):		79.2%

Fidelity DT Train Data (II):	99.0%
Fidelity DT Random Data (III):	35.0%
Fidelity DT Standard Uniform Data (IV):	36.2%
Fidelity DT Standard Normal Data (V):	35.2%
Fidelity I-Net (VI):	99.8%



Distribution 1,0:	$\Gamma(2.434,\ 2.031)$	n = 661
Distribution 1,1:	$\Gamma(4.884,\ 0.156)$	n = 4339
Distribution 2,0:	$B(0.015,\ 2.123)$	n = 1445
Distribution 2,1:	$B(2.243,\ 2.857)$	n = 3555
Performance NN (I):		79.6%

Fidelity DT Train Data (II):	99.2%
Fidelity DT Random Data (III):	66.4%
Fidelity DT Standard Uniform Data (IV):	81.2%
Fidelity DT Standard Normal Data (V):	67.4%
Fidelity I-Net (VI):	98.2%

Fig. 7 Visual decision boundary evaluation. This figure shows the decision boundaries of the NN we want to interpret (I), followed by the decision boundary of explanations generated by different approaches, along with their performance for three different datasets and function families. Only when the training data is accessed (II) and when using the $\mathcal{I}$-Net (VI), the explanation comprises the relevant aspects of the model. When the training data is not accessible (III)-(V), sample-based approaches are not able to generate reasonable explanations

at the bottom left. Considering just the shapes and size of the areas created by the decision boundary, this seems to be a reasonable explanation. However, as explained in Sect. 2.2, if we take the data into account, it becomes apparent that the neglected part of the decision boundary at the bottom left is much more important, since many data points are located in this area. In contrast, the explanation generated by the $\mathcal{I}$-Net as sample-free approach (VI) correctly separates the samples at the bottom left with its decision boundary and neglects the part at the top right, which is not relevant when taking the data into account. We can confirm this by taking the fidelity scores into account: The $\mathcal{I}$-Net achieved a fidelity of 99.8%, while the sample-based distillation without training data only achieved a maximum fidelity of 36.2%. For SDTs in Fig. 7(b), we can observe similar results: The fidelity for the sample-based approaches (III)–(V) significantly decreased if the training data is not available and the explanation focused on irrelevant areas. In contrast, the $\mathcal{I}$-Net (VI) was able to generate high-fidelity explanations without accessing the training data.

4.2.2 Real world datasets performance comparison

For the evaluation, we selected 8 commonly used real-world datasets, mostly from the banking and medical domain, comprising confidential data where it is realistic to assume that the training data cannot be exposed. In this experiment, we compare the performance of the $\mathcal{I}$-Net and a sample-based distillation without access to training data for standard DTs (Table 1) and SDTs (Table 2). We report the mean and standard deviation over 10 trials. While for the standard uniform and standard normal sampling only the sampled data points differ, we sampled a new set of distributions and parameters for each trial in the multi-distribution case. Results for the best method, as well as results that are not significantly different in a paired t-test ($\alpha = 0.05$) are highlighted in bold (Tables 1, 2, 3).

Standard DTs Comparing the results for standard DTs as surrogate model in Table 1, the $\mathcal{I}$-Net was the best method on 6/8 datasets and achieved the highest average performance (86.19%). For a sample-based distillation, multi-distribution sampling achieved the best average performance 73.20%, even though it did not achieve the best results on any dataset. Sampling from a standard uniform distribution achieved a similar average performance (72.75%) and was the best method on 2/8 datasets. A standard normal sampling strategy achieved the worst average performance but still achieved the best results

Table 1 Real-world evaluation results for standard DTs

Dataset	$\mathcal{I}$-Net	Multi-distribution	Standard uniform	Standard normal
Titanic (n=9)	**95.51** ± 0.00	71.12 ± 17.16	86.07 ± 3.30	86.29 ± 7.75
Medical Insurance (n=9)	82.71 ± 0.00	88.12 ± 6.71	89.47 ± 4.19	**90.75 ± 8.83**
Breast Cancer Wisconsin Original (n=9)	**97.10 ± 0.00**	83.62 ± 13.09	39.42 ± 13.90	31.88 ± 0.00
Wisconsin Diagnostic Breast Cancer (n=10)	**80.36 ± 0.00**	56.43 ± 17.65	37.86 ± 15.56	33.39 ± 5.42
Heart Disease (n=13)	73.33 ± 0.00	74.67 ± 9.45	**85.67 ± 5.97**	80.33 ± 7.67
Cervical Cancer (n=15)	**84.71 ± 0.00**	65.41 ± 27.77	71.88 ± 9.64	60.82 ± 30.29
Loan House (n=16)	**100.00 ± 0.00**	77.05 ± 24.41	**96.89 ± 7.42**	59.84 ± 33.84
Credit Card Default (n=23)	**75.80 ± 0.00**	69.16 ± 17.58	74.76 ± 0.05	34.33 ± 20.31
Mean Fidelity	**86.19**	73.20	72.75	59.70

Table 2 Real-world evaluation results for SDTs

Dataset	$\mathcal{I}$-Net	Multi-distribution	Standard uniform	Standard normal
Titanic (n=9)	**95.51 ± 0.00**	88.31 ± 3.80	92.47 ± 1.24	92.81 ± 0.75
Medical Insurance (n=9)	77.44 ± 0.00	79.25 ± 20.79	**91.20 ± 7.59**	78.50 ± 0.60
Breast Cancer Wisconsin Original (n=9)	**100.00 ± 0.00**	96.67 ± 4.49	**100.00 ± 0.00**	**31.88 ± 0.00**
Wisconsin Diagnostic Breast Cancer (n=10)	94.64 ± 0.00	84.82 ± 16.98	**97.50 ± 2.55**	28.57 ± 0.00
Heart Disease (n=13)	**100.00 ± 0.00**	**90.67 ± 12.45**	**99.33 ± 1.33**	**60.00 ± 0.00**
Cervical Cancer (n=15)	**85.88 ± 0.00**	58.35 ± 21.36	43.29 ± 11.54	25.76 ± 2.38
Loan House (n=16)	**100.00 ± 0.00**	50.82 ± 39.36	**100.00 ± 0.00**	17.21 ± 12.30
Credit Card Default (n=23)	**83.30 ± 0.00**	59.86 ± 21.99	38.77 ± 6.06	75.40 ± 1.32
Mean Fidelity	**92.10**	76.09	82.82	51.27

on one dataset. Especially for the *Wisconsin Diagnostic Breast Cancer* and *Cervical Cancer*, the sample-based distillation was not able to generate accurate explanations if the training data was not accessible. For *Wisconsin Diagnostic Breast Cancer*, the fidelity of sample-based distillation was mostly even worse than a random guess, which highlights the importance of querying the model on reasonable data points, as already shown in Sect. 4.2.1.

SDTs For SDTs, the $\mathcal{I}$-Net achieved the highest average performance (92.10%) and was the best method on 6/8 datasets. Sampling from a standard uniform distribution resulted in the best performance for sample-based approaches with 5/8 wins, which is slightly worse than the $\mathcal{I}$-Net. However, the mean performance was significantly worse, with an average fidelity of (82.82%). This was mainly caused by the superior performance of the $\mathcal{I}$-Net on the *Cervical Cancer* and *Credit Card Default* datasets, where the fidelity was more than 40 percentage points higher. The second-best results were achieved using a multi-distribution sampling strategy (76.09%) and sampling from a standard normal distribution was again significantly worse, with an average of 51.27%. Furthermore, we observed that the average fidelity of SDTs is considerably higher than the fidelity standard DTs. We can trace this back to the fact that SDTs have a significantly higher complexity, especially with an increasing number of variables, as discussed in Sect. 3.2. This can also explain why the performance difference between a sample-based distillation and the $\mathcal{I}$-Net is smaller for SDTs compared to standard DTs: While using meaningful samples for querying the NN is very crucial when the surrogate model has low complexity, it is less crucial if the surrogate model is more complex, making it less reliant on focusing on the most important information. Accordingly, it is less likely that relevant areas are neglected with an increasing complexity of the surrogate model. However, for interpretability, we are usually interested in surrogate models with a comparatively low complexity that are understandable for humans. In this scenario, $\mathcal{I}$-Nets substantially outperformed sample-based methods.

Summed up, the $\mathcal{I}$-Net outperformed a sample-based distillation on the majority of datasets when training data was not accessible and was the best model in 12/16 evaluated scenarios. Especially for surrogate models with low complexity, sample-based approaches are dependent on proper querying. Therefore, using the $\mathcal{I}$-Net in such scenarios can achieve a higher fidelity. This can be crucial since wrong explanations can lead to wrong decisions, as we will evaluate more in-depth in Sect. 4.2.4.

Table 3 Ablation study: data generation comparison

Dataset	Standard DT		SDT	
	new	old	new	old
Titanic (n=9)	**95.51**	39.33	**95.51**	86.52
Medical Insurance (n=9)	**82.71**	72.93	77.44	**92.48**
Breast Cancer Wisconsin Original (n=9)	**97.10**	31.88	**100.00**	98.55
Wisconsin Diagnostic Breast Cancer (n=10)	**80.36**	28.57	**94.64**	83.93
Heart Disease (n=13)	**73.33**	60.00	**100.00**	80.00
Cervical Cancer (n=15)	**84.71**	**84.71**	**85.88**	84.71
Loan House (n=16)	**100.00**	13.11	**100.00**	**100.00**
Credit Card Default (n=23)	**75.80**	74.73	**83.30**	64.97
Mean Fidelity	**86.19**	50.66	**92.10**	86.40

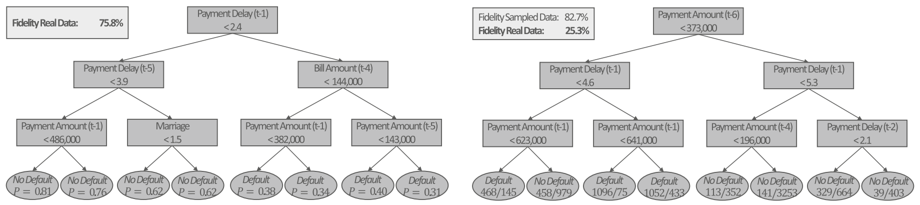
4.2.3 Ablation study

In Sect. 3.1 we introduced an improved data generation method which is more robust in a real-world scenario, since it considers multiple different distributions. In the following, we will compare our new data generation method with the data generation method introduced by Marton et al. (2022), which generates data based on the function family of the surrogate model and considers only a single distribution. As shown in Table 3, our improved data generation was the best method on 7/8 datasets for standard DTs and 7/8 datasets for SDTs. Comparing the average performance over all datasets, we also observed a significant increase in the accuracy using the new data generation method of $\approx$ 36 percentage points for standard DTs. For SDTs, the difference in the mean fidelity was significantly smaller, with only $\approx$ 6 percentage points. One explanation could be the higher complexity of the surrogate model for STDs, as already discussed in Sect. 4.2.2.

4.2.4 Case study: explaining neural networks for credit card default prediction

In this section, we will take a closer look at the explanations generated by sample-based approaches and the $\mathcal{I}$-Net by returning to Example 1 which we introduced in Sect. 1. The purpose of this experiment is to show in a real-world setting that without access to the training data, the surrogate model generated by sample-based approaches can lead to incorrect assumptions on the function learned by the NN. We want to emphasize that without access to the training data, it is not possible to identify whether a specific surrogate model contains a misconception or not, since we are not able to calculate a representative fidelity.

The *Credit Card Default* (Dua & Graff, 2017) dataset is concerned with credit card default prediction based on 23 features including demographic data along with the credit and payment history of clients in Taiwan (Yeh & Lien, 2009). Figure 8 shows the DT surrogate models generated by the $\mathcal{I}$-Net and a sample-based distillation. As shown in Fig. 8a, the $\mathcal{I}$-Net archived a fidelity of 75.8% and only considers a single split to decide whether there will be a payment or not. The split is based on whether the payment for the previous month was delayed for less than three months (left path) or not (right path). We can consider this as a very reasonable split, under the assumption that a client with a past default history has a higher probability of recurring defaults in the future. We can substantiate this



(a) $\mathcal{I}$-Net Standard DT **(b)** Sample-Based Standard DT

Fig. 8 Explanation comparison for standard DTs. The surrogate model on the right is learned by a sample-based distillation with a multi-distribution sampling strategy. The DT on the left is predicted by the $\mathcal{I}$-Net. The $\mathcal{I}$-Net makes reasonable splits and achieves a significantly higher fidelity on the real data

by taking the actual class distributions into account: The overall probability for *No Default* is $\approx 78\%$, while it is only $\approx 29\%$ for the right path of the tree. This aligns with our hypothesis, as the *Default* risk increases when taking the right path. By considering leaf node probabilities, we can enhance our understanding of the decision process. If the payment 5 months ago was also not delayed more than 4 months, the probability that there will be *No Default* is even higher, as shown in the left branch of the tree. If there was a delay of more than 4 months, the probability that there will be a *Default* is approximately 20% higher. We can again verify this based on the real data: The proportion of *No Default* in the data is 79% when taking the left branch, while it is 54% when taking the right branch. These numbers are also very close to the corresponding leaf probabilities (81% and 76% versus 62% respectively).

In contrast, when taking a closer look at the DT generated by a sample-based distillation (Fig. 8b), we can observe that the entire right branch of the tree has *No Default* as prediction. This prediction is made solely based on the first split, where the right branch is taken if the payment amount 6 months ago was larger than 373, 000. This translates to the rule that we should always predict that there will be *No Default* in the payment if there was a large payment amount in the past. However, it seems counter-intuitive to make this decision without taking for instance the credit history of the client and previous defaults into account. Upon investigating the real data, we can verify that this split is not reasonable: Only $\frac{6}{30{,}000}$ samples fall into the right branch of the tree where all 6 samples are from the majority class (non-default). Furthermore, all samples falling into the right subtree reach a single leaf, making the remaining splits dispensable. It becomes evident that this is in extreme contrast to the class distributions in the leaves that represent the random data used for querying. This discrepancy underlines once more the crucial role of data from the same distribution as the training data to query the model appropriately. We can further observe that almost all samples of the real data $\left(\frac{29{,}928}{30{,}000}\right)$ land in the leaf at the very left of the tree and therefore are assigned to the minority class (*Default*). Summed up, the tree predicts *No Default* for a total of only 7 samples of the training data. This is confirmed by the surrogate model's poor fidelity of 25.3% on the real data, which is worse than a random guess. In contrast, the fidelity on the sampled data used for querying the model was very high (82.7%) which can lead to misconceptions, since the model appears to have a high fidelity that does not hold on the real data. Without access to the training data, it is not possible to identify these misconceptions. Taking only the high fidelity on the sampled data into account, we might assume that the surrogate actually captures the model well, and therefore we could make wrong assumptions about its behavior.

5 Related work

Various methods to interpret black-box models have been proposed in the past decades. Overviews from different perspectives are given by Doshi-Velez and Kim (2017); Lipton (2018) and Molnar (2020). In this paper, we focus on methods that translate NNs into DTs.

Model distillation is a common technique to transfer knowledge from a complex model into a surrogate model (Buciluǎ et al., 2006; Hinton et al., 2015). It can be used to obtain more compact model representations for efficiency reasons (Buciluǎ et al., 2006; Hinton et al., 2015; Furlanello et al., 2018) or to interpret the model as a human-understandable function (Frosst & Hinton, 2017; Tan et al., 2018). With the focus on interpretability, model distillation is performed to either understand the function encoded by trained networks and how predictions are made (Craven & Shavlik, 1995; Boz & Hillman, 2000; Zhang et al., 2019) or to improve the performance of an interpretable algorithm to use it instead of the NN at test time (Krishnan & Sivakumar, 1999; Frosst & Hinton, 2017; Liu et al., 2018). Although those purposes differ, the methods can be interchangeably used for both.

Various sample-based methods using DTs as surrogate models were presented in the past quarter-century (Craven & Shavlik, 1995; Krishnan & Sivakumar, 1999; Boz & Hillman, 2000; Johansson & Niklasson, 2009; Frosst & Hinton, 2017; Liu et al., 2018; Zhang et al., 2019; Nguyen et al., 2020). They transform a trained NN into a surrogate function with a tree-like structure, which is usually achieved by maximizing the fidelity to the NN on a sample basis. The main differences among existing approaches are the type of the resulting DTs, the method to train the surrogate model, and the purpose of the surrogate model. The proposed trees make either univariate (Krishnan & Sivakumar, 1999; Boz & Hillman, 2000; Liu et al., 2018) or multivariate (Craven & Shavlik, 1995; Nguyen et al., 2020; Frosst & Hinton, 2017) decisions at each split. Trees that consider multiple variables can achieve higher fidelity and accuracy than univariate DTs. However, especially for tabular data, they are less interpretable. For training the surrogate model, differences exist regarding the data used, the decision how a split is determined, and the optimization technique used. Regarding the training of trees, most approaches rely on standard DT induction methods. Krishnan and Sivakumar (1999) use ID3 (Quinlan, 1986) and C4.5 (Quinlan, 2014), Craven and Shavlik (1995) use ID2-of-3 (Murphy, 1991) and Nguyen et al. (2020) use CART (Breiman et al., 1984). While these approaches greedily optimize the fidelity, Frosst and Hinton (2017) use gradient descent to distill the trees. The data to maximize the fidelity is either the original training data (Frosst & Hinton, 2017; Liu et al., 2018) or data from a distribution that was modeled based on the training data (Craven & Shavlik, 1995; Krishnan & Sivakumar, 1999; Boz & Hillman, 2000; Johansson & Niklasson, 2009).

In all cases, the predictions on data are the only source for understanding the model behavior, and thus meaningful samples are crucial for the performance. Without information on the distribution of the training data, e.g., in the form of data points, the performance of sample-based methods decreases significantly. Recent approaches deal with this problem using metadata, such as layer activations, to create good samples based on network information (Lopes et al., 2017; Bhardwaj et al., 2019; Nayak et al., 2019). However, they often still need access to the training data in some part of the distillation process. Lopes et al. (2017) use a fraction of the original training data to compute activations summaries to later compress the network without accessing the data. Similarly, Bhardwaj et al. (2019) require samples of the original training data to generate activation vectors, which are necessary for the distillation. However, they report requiring significantly fewer data points than Lopes et al. (2017). In contrast, Nayak et al. (2019) does not require access to training data, but only accesses the model internals. The model internals are used to generate a class similarity matrix based on the parameters of the softmax output layer of the NN. Based on the class similarity matrix, Nayak et al. (2019) generate meaningful samples called *Data Impressions* via Dirichlet sampling based on the output classes. However, the approach requires a softmax output for the NN and is tailored towards multi-class classification problems, since it utilizes the knowledge contained in the class similarity matrix for sampling. Accordingly, an application on a binary classification task is not adequate, since a class similarity matrix for two classes can contain only little information, which makes sampling difficult. Summed up, the main issue is that the majority of state-of-the-art sample-free approaches still need access to at least a subset of the training data. Only Nayak et al. (2019) is applicable without training data is, but the application is restricted, e.g., to multi-class tasks.

6 Conclusion and future work

In this paper, we proposed a new instance of $\mathcal{I}$-Nets for tree-based surrogate models and an improved data generation method, making $\mathcal{I}$-Net applicable in a real-world scenario. While traditional approaches generate explanations sample-based and rely on proper querying, $\mathcal{I}$-Nets utilize the model internals, which implicitly contain all relevant information about the network function. Therefore, $\mathcal{I}$-Nets can generate reasonable explanations in scenarios where the training data is not accessible. Using our new data generation method, the $\mathcal{I}$-Net generalizes to NNs trained on different data distributions. Thereby, the $\mathcal{I}$-Net identifies which aspects learned by the NN are most important based on the distribution of the training data and therefore should be contained in the explanation. The $\mathcal{I}$-Net can use this knowledge to generate meaningful explanations for new, unseen networks, even without access to the training data.

In our experiments, we showed that sample-based approaches strongly rely on proper querying and are often not able to generate reasonable explanations without access to the training data. In this scenario, the explanations of sample-based approaches frequently comprise misconceptions, since they focus on the global behavior and do not focus on the regions that are important for a reasonable explanation, as we demonstrated within our case study. Furthermore, we empirically showed that $\mathcal{I}$-Nets consistently outperform sample-based methods on real-world datasets when the training data is not available. Thus, using $\mathcal{I}$ -Nets, high-fidelity explanations can be generated when confidential training data can't be exposed.

Currently, the $\mathcal{I}$-Net comprises a feed-forward NN and the model internals used as an input are flattened to a one-dimensional array. In further work, we aim for a more sophisticated $\mathcal{I}$-Net architecture and a better-suited representation for the model input to improve the performance even further. Furthermore, the $\mathcal{I}$-Net is tailored towards generating fully grown DTs based on its structure. In further work this could be addressed by adjusting the output layer which allows using greater depths for the explanation without a significant increase in complexity.

Acknowledgements This research was supported by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) and the German Federal Ministry of Education and Research (BMBF).

Author contributions Conceptualization: SM, CB; Methodology: SM; Writing: SM, SL; Discussion and review: SM, SL, CB, AT; Supervision: CB, SL, HS.

Funding Open Access funding enabled and organized by Projekt DEAL. This research received no external funding.

Data availability All datasets are publically available.

Code availability Our code is publically available at https://github.com/s-marton/explaining-neural-networks-without-training-data.

Declarations

Conflict of interest The authors declare no competing interest.

Ethical approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Bhardwaj, K., Suda, N., & Marculescu, R. (2019). Dream distillation: A data-independent model compression framework. arXiv preprint arXiv:1905.07072

Boz, O., & Hillman, D. (2000). *Converting a trained neural network to a decision tree dectext-decision tree extractor*. Citeseer.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC Press.

Buciluă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 535–541).

Craven, M., & Shavlik, J. (1995). Extracting tree-structured representations of trained networks. *Advances in neural information processing systems***8**.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608

Dua, D., & Graff, C. (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml

Frosst, N., & Hinton, G. (2017). Distilling a neural network into a soft decision tree. arXiv preprint arXiv:1711.09784

Furlanello, T., Lipton, Z.C., Tschannen, M., Itti, L., & Anandkumar, A. (2018). Born again neural networks. arXiv preprint arXiv:1805.04770

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531

Jin, H., Song, Q., & Hu, X. (2019). Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 1946–1956). ACM.

Johansson, U., & Niklasson, L. (2009). Evolving decision trees using oracle guides. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*. IEEE.

Krishnan, R., Sivakumar, G., & Bhattacharya, P. (1999). Extracting decision trees from trained neural networks. *Pattern Recognition, 32*(12), 1999–2009.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*, 436–444.

Leemis, L. M., & McQueston, J. T. (2008). Univariate distribution relationships. *The American Statistician, 62*(1), 45–53.

Lipton, Z. C. (2018). The mythos of model interpretability. *Queue, 16*(3), 31–57.

Liu, X., Wang, X., & Matwin, S. (2018). Improving the interpretability of deep neural networks with knowledge distillation. arXiv preprint arXiv:1812.10924

Lopes, R.G., Fenu, S., & Starner, T. (2017). Data-free knowledge distillation for deep neural networks. arXiv preprint arXiv:1710.07535

Marton, S., Lüdtke, S., & Bartelt, C. (2022). Explanations for neural networks by neural networks. *Applied Sciences, 12*(3), 980.

Molnar, C. (2020). *Interpretable Machine Learning*. Lulu. com.

Mun, J. (2015). Understanding and choosing the right probability distributions. In *Advanced analytical models: Over 800 models and 300 applications from the basel II accord to Wall Street and beyond* (pp. 899–917).

Murphy, P. (1991). Constructive induction of m-of-n terms. In *Proceedings of the Eighth International Workshop on Machine Learning* (pp. 183–187).

Nayak, G.K., Mopuri, K.R., Shaj, V., Babu, R.V., & Chakraborty, A. (2019). Zero-shot knowledge distillation in deep networks. arXiv preprint arXiv:1905.08114

Nguyen, D.T., Kasmarik, K.E., & Abbass, H.A. (2020). Towards interpretable ANNs: An exact transformation to multi-class multivariate decision trees.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 81–106.

Quinlan, J. R. (2014). *C4. 5: Programs for Machine Learning*. Elsevier.

Samek, W., Montavon, G., Vedaldi, A., Hansen, L. K., & Muller, K.-R. (2019). *Explainable AI: Interpreting explaining and visualizing deep learning*. Springer.

Tan, S., Caruana, R., Hooker, G., Koch, P., & Gordo, A. (2018). Learning global additive explanations for neural nets using model distillation.

Wang, X., Zhao, Y., & Pourpanah, F. (2020). Recent advances in deep learning. *International Journal of Machine Learning and Cybernetics, 11*, 747–750.

Yeh, I.-C., & Lien, C.-H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Systems with Applications, 36*(2), 2473–2480.

Zhang, Q., Yang, Y., Ma, H., & Wu, Y.N. (2019). Interpreting cnns via decision trees. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 6261–6270).