


---


# EXPLAINING NEURAL NETWORKS WITHOUT ACCESS TO TRAINING DATA

---

A PREPRINT

 **Sascha Marton**

Institute for Enterprise Systems  
University of Mannheim  
Mannheim, 68161  
marton@es.uni-mannheim.de

 **Stefan Lüdtkke**


Institute for Enterprise Systems  
University of Mannheim  
Mannheim, 68161  
luedtke@es.uni-mannheim.de

 **Christian Bartelt**

Institute for Enterprise Systems  
University of Mannheim  
Mannheim, 68161  
bartelt@es.uni-mannheim.de

 **Andrej Tschalzev**

Institute for Enterprise Systems  
University of Mannheim  
Mannheim, 68161  
tschalzev@es.uni-mannheim.de

 **Heiner Stuckenschmidt**

Data and Web Science Group  
University of Mannheim  
Mannheim, 68159  
heiner@informatik.uni-mannheim.de

## 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,  $\mathcal{I}$ -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

## 1 Introduction

Artificial neural networks achieve impressive results for various modeling tasks [LeCun et al., 2015, Wang et al., 2020]. However, a downside of their superior performance and sophisticated structure is the comprehensibility of the learned models. In many domains, it is crucial to understand the function learned by a neural network, especially when it comes to decisions that affect people [Samek et al., 2019, Molnar, 2020].

A common approach to tackle the problem of interpretability without sacrificing the superior performance is using a surrogate model as gateway to interpretability [Molnar, 2020]. Most existing global surrogate approaches use a distillation procedure to learn the surrogate model based on the predictions of the neural network [Molnar, 2020, Frosst and Hinton, 2017]. Therefore, they query the neural network 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, or at least follows its distribution [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 and 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 neural networks. To gain insight into the decision-making process of the neural network, we can learn a global surrogate model. Unfortunately, if the training data is not accessible, we can't ensure that the neural network is properly queried, and the explanation contains the relevant information when using a traditional, sample-based distillation.

Figure 1a shows such 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. However, in reality, this fidelity cannot be computed when the training data is not available, such that these misconceptions might go unnoticed. This can lead to wrong assumptions about what the network actually learned.

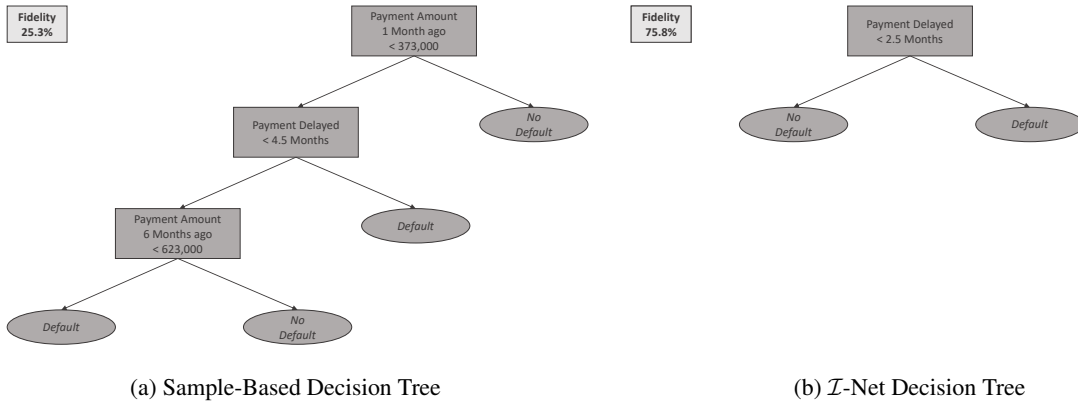


Figure 1: **Explaining Neural Networks for Credit Card Default Prediction.** The DT on the left is learned by a sample-based distillation without access to training data, and the DT on the right is predicted by the  $\mathcal{I}$ -Net. The  $\mathcal{I}$ -Net makes reasonable splits and achieves a significantly higher fidelity on the real data.

As shown in *Example 1*, knowing the training data is crucial for sample-based methods and without them, 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  $\mathcal{I}$ -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 neural network (the so-called  $\mathcal{I}$ -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 Figure 1b: The  $\mathcal{I}$ -Net achieves a high fidelity and the resulting surrogate model encodes the rule that *Default* is predicted if the payment for the last month was delayed for more than two months, which is a reasonable explanation for this scenario.

The  $\mathcal{I}$ -Net was originally devised for lower-order polynomials as a surrogate model. While polynomials can be reasonable explanations for regression tasks, they are not well-suited for representing decision boundaries of a classification task, which is the focus of our paper. In contrast, decision trees (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 and Hinton, 2017]. In the recent literature, soft DTs (SDTs) are successfully used as interpretable surrogate models [Frosst and 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 contributions:

- We propose an improved data generation method for the training of an  $\mathcal{I}$ -Net (Section 3.1.1) which allows more robust and distribution-independent explanations on real-world datasets.

- We present an  $\mathcal{I}$ -Net design that is able to represent standard DTs and SDTs as surrogate model (Section 3.2) with a high fidelity.
- We introduce univariate SDTs as complexity-fidelity trade-off (Section 3.2.2) and show that they can be implemented efficiently using the  $\mathcal{I}$ -Net.
- We empirically evaluate our approach against sample-based approaches for learning standard DTs and SDTs and show that it achieves superior results when training data is not accessible (Section 4.2).

## 2 $\mathcal{I}$ -Nets as a Sample-Free Approach to Global Model Interpretability

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

### 2.1 Global Explanations for Neural Networks

The general task of globally explaining neural networks can be formalized as finding a function  $g : X \rightarrow P(Y|X)$  (i.e., a surrogate model) that approximates the decision function of a neural network  $\lambda : X \rightarrow P(Y|X)$ , such that  $\forall \mathbf{x} \in X : \lambda(\mathbf{x}) \approx g(\mathbf{x})$ , where  $X$  is a set of feature vectors and  $Y$  is a set of classes.

Since the  $\mathcal{I}$ -Net approach implements a learning task, it is convenient to distinguish between the functions  $\lambda$  and  $g$  and their representations  $\theta_\lambda \in \Theta_\lambda$  and  $\theta_g \in \Theta_g$  [Marton et al., 2022]. The representation  $\theta_\lambda$  consists of the network parameters, i.e., the weights and biases of the neural network. Similarly,  $\theta_g$  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  $\lambda$  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, as shown in Figure 2 (I-II). They generate a new dataset, where the labels are obtained by querying  $\lambda$  based on a set of data points. In the next step, a surrogate model is trained using the generated dataset, maximizing the fidelity between  $\lambda$  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 also strongly depends on the data used for querying the model. Information that is not properly queried cannot be contained in the explanation, as already shown in *Example 1*. Therefore, in the literature it is suggested to use the original train data or data from the same distribution for querying the model [Molnar, 2020, Lopes et al., 2017], which usually yields to meaningful explanations as depicted in I in Figure 2. However, if the training data is not accessible or not existing anymore, the model has to be queried based on some sampled data (II in Figure 2). 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 neural network. 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 also take the data distribution into account: 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 Figure 3. In Section 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 neural network as proposed by Marton et al. [2022]. Therefore, we transform the task of explaining neural networks into a machine learning task, as shown in Figure 2 III. The concept of  $\mathcal{I}$ -Nets is depicted more detailed in Figure 4 and involves two major steps:

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

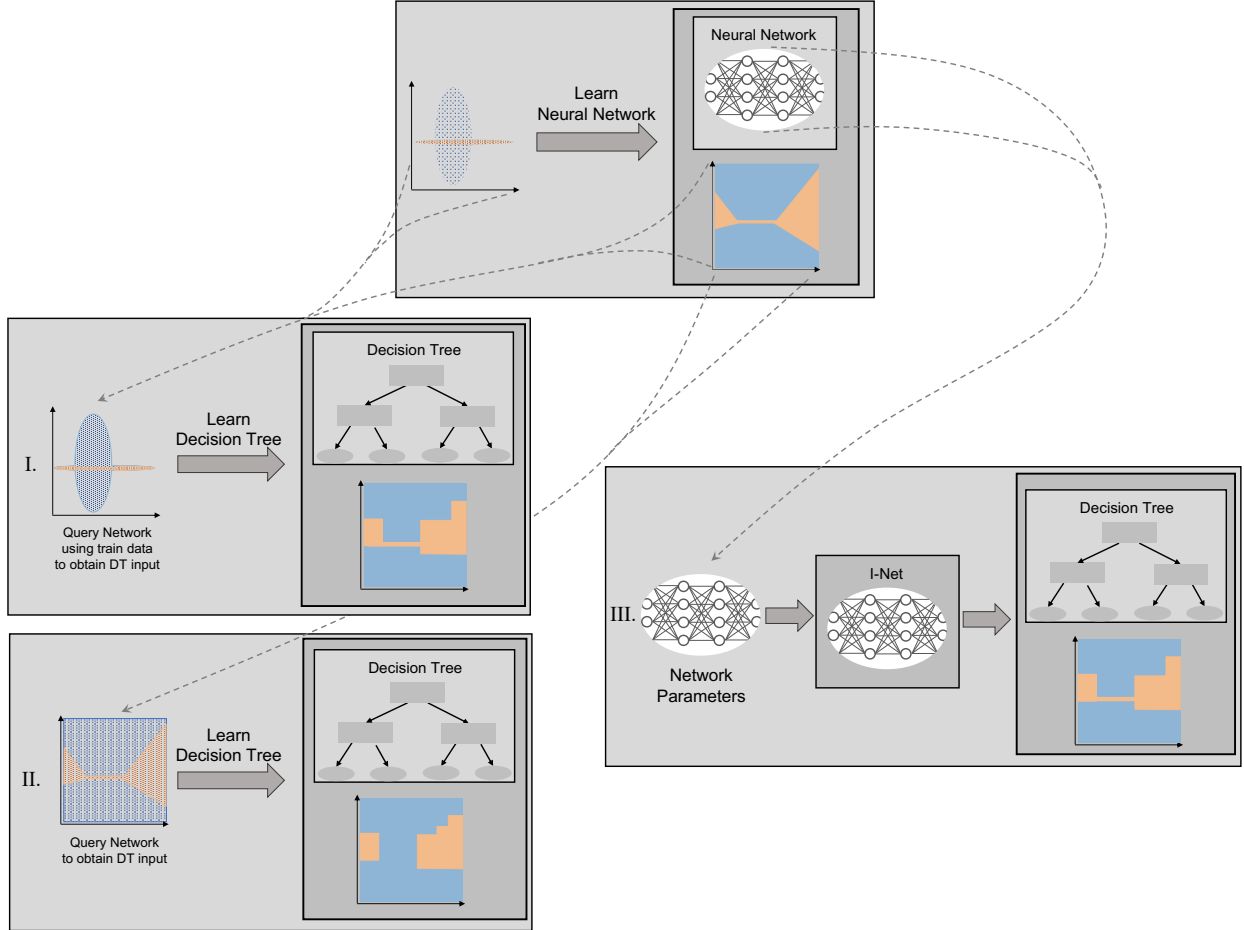


Figure 2: **Sample-Based and  $\mathcal{I}$ -Net Approach.** Sample-based approaches query the target network based on a set of data points. Using the train data to query the network (I) usually generates a meaningful explanation. If the training data is not available, we have to query the network based on randomly sampled data, e.g., from a uniform distribution (II), which often cannot generate a meaningful explanation since relevant parts are not queried properly. The  $\mathcal{I}$ -Net uses the network parameters as an input to generate a reasonable explanation (III) and does not rely on querying the neural network.

Thereby, no supervision in terms of actual labels is required during the training. Instead, the fidelity between  $\lambda$  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 crucial part of the  $\mathcal{I}$ -Net approach is an efficient training procedure. Thereby, as for most machine learning tasks, good training data (in our case, a set of network parameters  $\Theta_\lambda$ ) is important. Therefore, we present an improved data generation method making  $\mathcal{I}$ -Nets applicable for real world scenarios in Section 3.1.1.

### 3 Robust $\mathcal{I}$ -Nets for Decision Trees

In this section, we present the main contributions of this 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. Therefore, we will introduce an improved data generation method that considers multiple data distributions and creates reasonable learning tasks (Section 3.1.1).

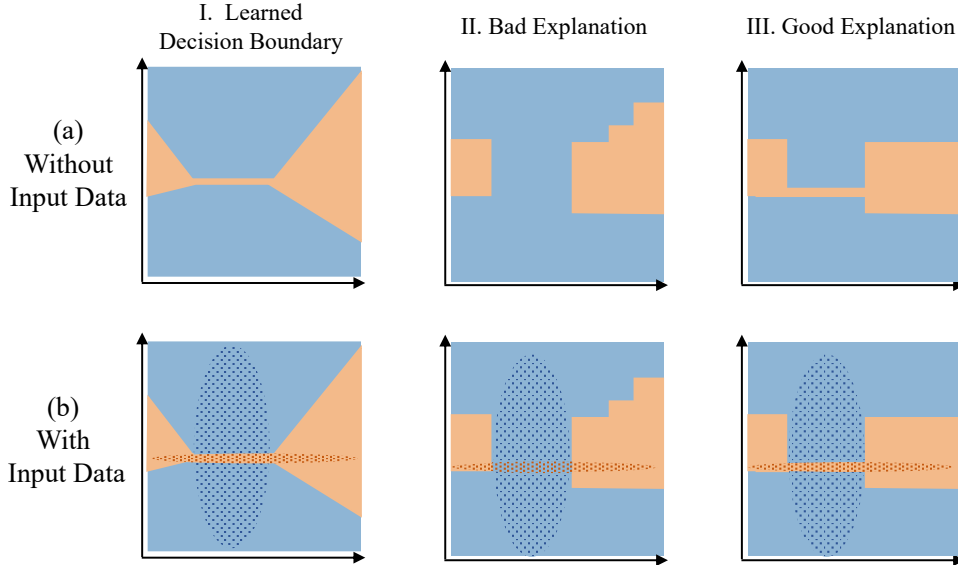


Figure 3: **Good and Bad Explanations.** This figure shows an exemplary decision boundary of a bad (II) and a good (III) explanation for the model we want to interpret (I). Without considering the data (a), the explanation shown in II appears very reasonable, since the areas created by the decision boundary cover most of the decision boundary of the original model. However, when taking the data into account (b), we can see that the small area in the center of the picture is very important, since there are many samples located. This is neglected by the explanation shown in II and only considered by the explanation shown in III.

Furthermore, Marton et al. [2022] focus on regression tasks. In contrast, we focus on classification tasks and therefore present an adjusted loss function for the  $\mathcal{I}$ -Net (Section 3.1.2).

In general, the  $\mathcal{I}$ -Net framework can be applied to arbitrary function families for  $g$ , as long as a suitable representation  $\theta_g$  is available. In Section 3.2, we introduce different DT variants and propose corresponding representations  $\theta_g$  that allow an efficient training.

### 3.1 Improved Data Generation and Training Procedure

#### 3.1.1 Data Generation Method

The data generation method proposed by Marton et al. [2022] focused on maximizing the performance of the  $\mathcal{I}$ -Net during training by learning functions  $\lambda$  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  $\lambda$ . This procedure ensures that the functions  $\lambda$  are representative of  $g$ , which enables efficient training. However, a high training performance does not necessarily mean that the model generalizes well to unseen data, i.e., neural networks trained on real-world datasets.

Additionally, Marton et al. [2022] use a uniform data distribution to query  $\lambda$  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 Section 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 [Marton et al., 2022]: For each  $\theta_\lambda \in \Theta_\lambda$ , we know the data that was used for learning  $\lambda$ . 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, it can generate meaningful explanations solely based on the network parameters  $\theta_\lambda$  at test time.

In general, generating the training data for the  $\mathcal{I}$ -Net  $\Theta_\lambda$  involves three major steps:

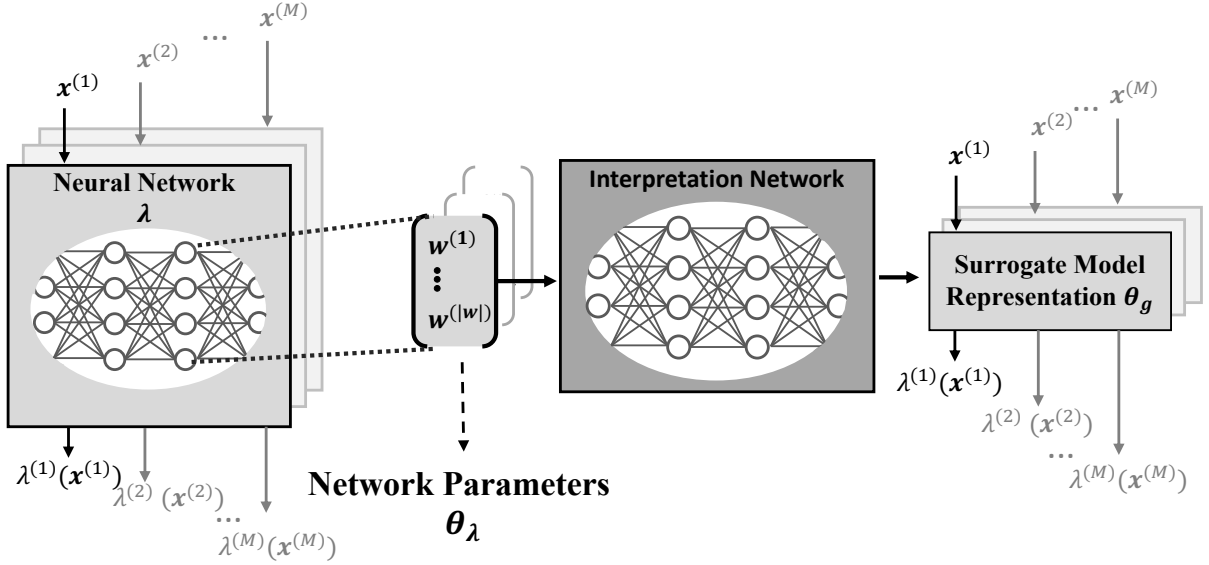


Figure 4: **Overview of the  $\mathcal{I}$ -Net Approach.** The neural network parameters as input are translated into a surrogate model Marton et al. [2022].

Distribution	Parameter $p_1$	Parameter $p_2$	Symbol
Uniform	minimum	maximum	$\mathcal{U}(p_1, p_2)$
Normal	location (mean)	scale (standard deviation)	$\mathcal{N}(p_1, p_2)$
Gamma	shape ( $k$ )	scale ( $\theta$ )	$\Gamma(p_1, p_2)$
Beta	$\alpha$	$\beta$	$\mathbf{B}(p_1, p_2)$
Poisson	lambda	-	$\text{Poi}(p_1)$

Table 1: **Data Generation Distributions.** This table summarizes the different distributions used for the data generation of the  $\mathcal{I}$ -Net with their parameters and used symbol.

1. Generate  $N$  datasets  $\mathcal{D}_\lambda = \{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^M$ , each comprising  $M$  samples.
2. For each dataset  $\mathcal{D}_\lambda$ , train a network  $\lambda$ , extract the network parameters  $\theta_\lambda$  and add them to the training set  $\Theta_\lambda$ .
3. Use  $\Theta_\lambda$  to train an  $\mathcal{I}$ -Net for the respective function family.

The data generation is visualized in Figure 5: 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  $\{\mathcal{U}, \mathcal{N}, \Gamma, \text{B}, \text{Poi}\}$  for each feature. The distributions considered within this paper are summarized in Table 1. The distributions were selected to cover a wide range of diverse distributions that are reasonable for many different real-world phenomena [Leemis and 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*. This procedure is formalized in Algorithm 1. We can see the proposed data generation method as a generalization of common, synthetic machine learning problems (as for instance `make_blobs`<sup>1</sup>), that is able to generate more realistic tasks.

<sup>1</sup>Accessible using sklearn under [https://scikit-learn.org/0.15/modules/generated/sklearn.datasets.make\\_blobs.html](https://scikit-learn.org/0.15/modules/generated/sklearn.datasets.make_blobs.html) (Accessed 15.05.2022)

	Feature 1	Feature 2	Feature 3	...	Feature n-2	Feature n-1	Feature n	Class
number of samples (M)	$D_{1,0}$	$D_{2,0}$	$D_{3,0}$	...	$D_{n-2,0}$	$D_{n-1,0}$	$D_{n,0}$	$c_0$
	$D_{1,1}$		$D_{3,1}$	...			$D_{n,1}$	
		$D_{2,1}$			$D_{n-2,1}$	$D_{n-1,1}$		$c_1$
	number of feature (n)							

Figure 5: **Data Generation Visualization.** This graphic visualizes the generation of a balanced, random dataset used for training a network  $\lambda$  where  $D \in \{\mathcal{U}, \mathcal{N}, \Gamma, \mathcal{B}, \text{Poi}\}$ . For each feature, a random distribution with two random parametrizations is chosen and a random number of data points is sampled from each distribution.

---

**Algorithm 1** Generate Multi-Distribution Data

---

```

1: function GENERATE( $n, D, M$ )
2:   for  $i = 1, \dots, n$  do
3:      $D_i \sim \mathcal{U}\{\mathcal{U}, \mathcal{N}, \Gamma, \mathcal{B}, \text{Poi}\}$  ▷ Randomly chose distribution
4:      $M_0 \sim \lceil \mathcal{U}(1, M - 1) \rceil$ 
5:      $\mathbf{p}_0 \sim \mathcal{U}(0, p)$  ▷ Randomly sample parameters for  $D_i, 0$ 
6:      $\mathbf{p}_1 \sim \mathcal{U}(0, p)$  ▷ Randomly sample parameters for  $D_i, 1$ 
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:     $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i - \min(\mathbf{x}_i)}{\max(\mathbf{x}_i) - \min(\mathbf{x}_i)}$  ▷ Scale feature to  $[0, 1]$ 
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

```

---

### 3.1.2 Adjusted Loss Function

While Marton et al. [2022] focused on regression tasks, we focus on binary classification tasks within this paper. Therefore, the loss function has to be adjusted by using binary cross-entropy as distance measure to quantify the fidelity between  $\lambda$  and  $g$ :

$$\begin{aligned}
\text{BC}(\theta_\lambda, \theta_g) = \frac{1}{M} \sum_{j=1}^M [\lambda(\mathbf{x}^{(j)})] \times \log(g(\mathbf{x}^{(j)})) \\
+ (1 - \lfloor \lambda(\mathbf{x}^{(j)}) \rfloor) \times \log(1 - g(\mathbf{x}^{(j)}))
\end{aligned} \tag{1}$$

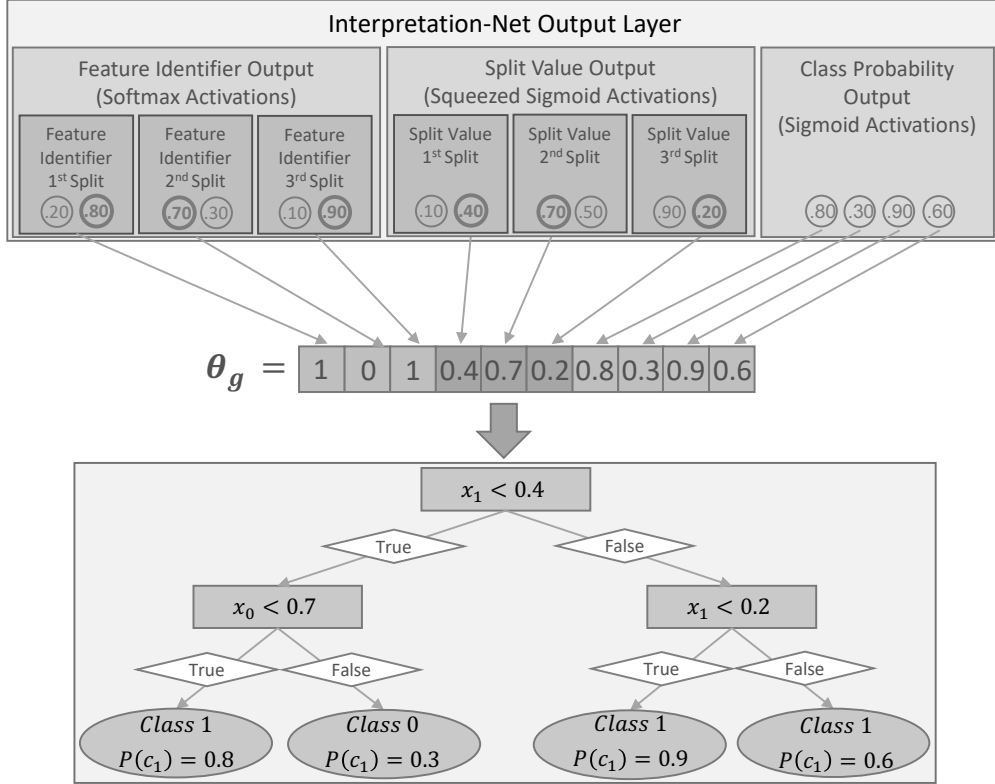


Figure 6: **Exemplary  $\mathcal{I}$ -Net Output for DTs.** The DT representation is predicted by the  $\mathcal{I}$ -Net using three separate output layers with different activation functions. The output shows an exemplary DT of depth two for a binary classification task on a dataset with two features.

Here,  $\lfloor \cdot \rfloor$  denotes rounding to the closest integer, which is required to calculate the binary cross-entropy correctly. The  $\mathcal{I}$ -Net loss for a set of network parameters  $\Theta_\lambda = \{\theta_\lambda^{(i)}\}_{i=1}^N$  is then computed as

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

### 3.2 Function Families and $\mathcal{I}$ -Net Output Representation

#### 3.2.1 $\mathcal{I}$ -Nets for Standard Decision Trees

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].

**Standard Decision Tree Representation in the  $\mathcal{I}$ -Net Framework** The  $\mathcal{I}$ -Net approach requires a suitable representation  $\theta_g$  for standard DTs to enable efficient learning. Specifically, we need a one-dimensional encoding of internal and leaf nodes, as shown in Figure 6.

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 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 Equation 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 (one softmax for each leaf node).

### 3.2.2 $\mathcal{I}$ -Nets for Soft Decision Trees

SDTs were proposed by Frosst and Hinton [2017] to overcome the interpretability problem that arises from distributed hierarchical representations when using neural networks by expressing the knowledge using hierarchical decisions of a tree-based structure. 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].

Figure 7 shows a SDT with a single internal node following the design of Frosst and Hinton [2017]. Each internal node  $j$  comprises a filter  $w^j$  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 by

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

where  $x$  is an input sample and  $S$  is a sigmoid function defined as  $S(x) = \frac{1}{1+e^{-x}}$ . Each leaf nodes  $l$  comprise a probability distribution  $Q^l$ , which for the binary case is defined as

$$Q_k^l = \frac{e^{\phi_k^l}}{e^{\phi_0^l} + e^{\phi_1^l}} \quad (4)$$

where  $k \in \{0, 1\}$  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.

While Frosst and Hinton [2017] focused on image datasets for a better visualization of the explanation, they also reported that their algorithm can efficiently be applied to tabular data. However, for tabular data, especially when the feature space is high-dimensional, using SDTs as a surrogate model can make the explanation hard to understand for humans due to their high complexity. To counteract the strong correlation between the model complexity and the number of features, we propose univariate SDTs. Univariate SDTs are similar to standard SDTs, but the filter at each internal node comprises only one value  $\neq 0$ . While the complexity of univariate SDTs is similar to standard DTs, they can represent decision boundaries that are not parallel to the feature axis. Simultaneously, they maintain the hierarchical structure, making them easier to comprehend for humans.

Unfortunately, univariate SDTs cannot be trained using the algorithm proposed by Frosst and Hinton [2017] without adjustments and there is to the best of our knowledge no existing learning algorithm for univariate SDTs. However,

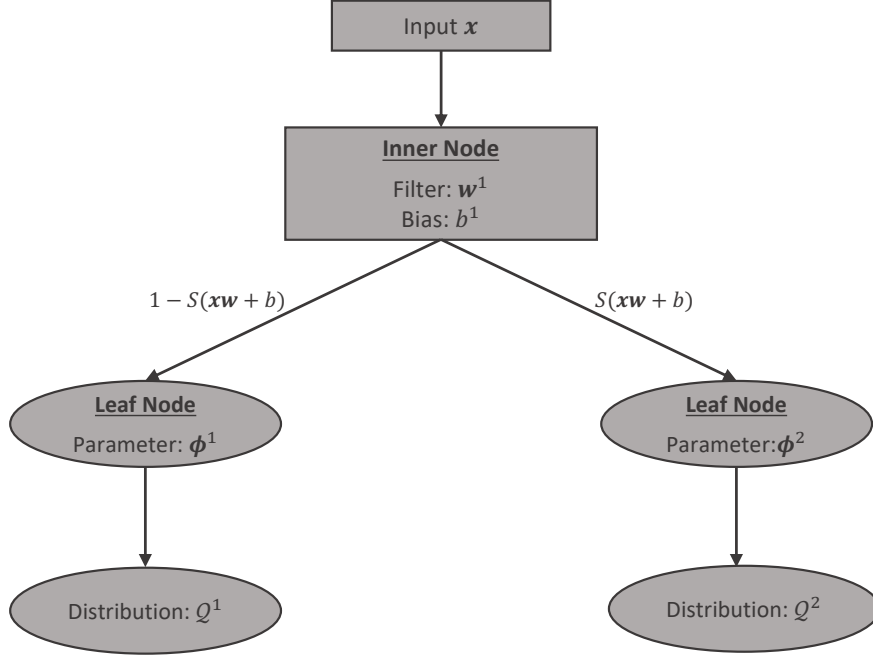


Figure 7: **Soft Decision Tree** This figure shows a minimal SDT with a single inner node and two leaf nodes.

we can constrain the learning algorithm to just consider the feature with the highest absolute value (which is also the value that according to the design has the highest influence on the decision) when calculating the path probabilities. We can implement this by multiplying the filter with a binary mask generated by an argmax or exaggerated softmax of the absolute filter values. Accordingly, during the forward pass, the filter values are calculated as:

$$\mathbf{w}_i = \mathbf{w}_i \times \sigma(\beta_2 \times |\mathbf{w}_i|) \quad (5)$$

By increasing the temperature  $\beta_2$ , we increase the extent to which the filter with the highest absolute value influences the calculation of the path probability. When  $\beta_2$  is sufficiently high, we approximate an argmax and the path probability is calculated only based on the filter with the highest absolute value.

**Soft Decision Tree Representations in the  $\mathcal{I}$ -Net Framework** To use SDTs as surrogate models within the  $\mathcal{I}$ -Net framework, we again need a suitable representation  $\theta_g$ . Fortunately, the encoding for standard SDT shown in Figure 8 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  $Q_k^l$  and no specific range for  $\phi^l$  is required.

Unfortunately, the representation of univariate SDTs as  $\mathcal{I}$ -Net output is not as straightforward. This is mainly caused by the fact that neural networks hardly predict a value of 0, which would be necessary when using the same representation we proposed for standard SDTs. This would result in many small filter values and therefore still multivariate trees. One solution is using a feature identifier and a separate filter value output, similar to the representation of the internal nodes for standard DTs (Figure 6). Accordingly, we can use  $n$  output neurons and a softmax activation for each internal node to indicate at which position to set the filter value (i.e., we can see this as a classification task for which filter value is  $\neq 0$ ). We represent the filter value again using  $n$  neurons with a linear activation for each internal node. The bias and leaf parameters are adopted from the standard SDT. An exemplary representation for univariate SDTs is shown in Figure 9.

## 4 Evaluation

The goal of our evaluation is to show that  $\mathcal{I}$ -Nets are able to interpret neural networks 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:

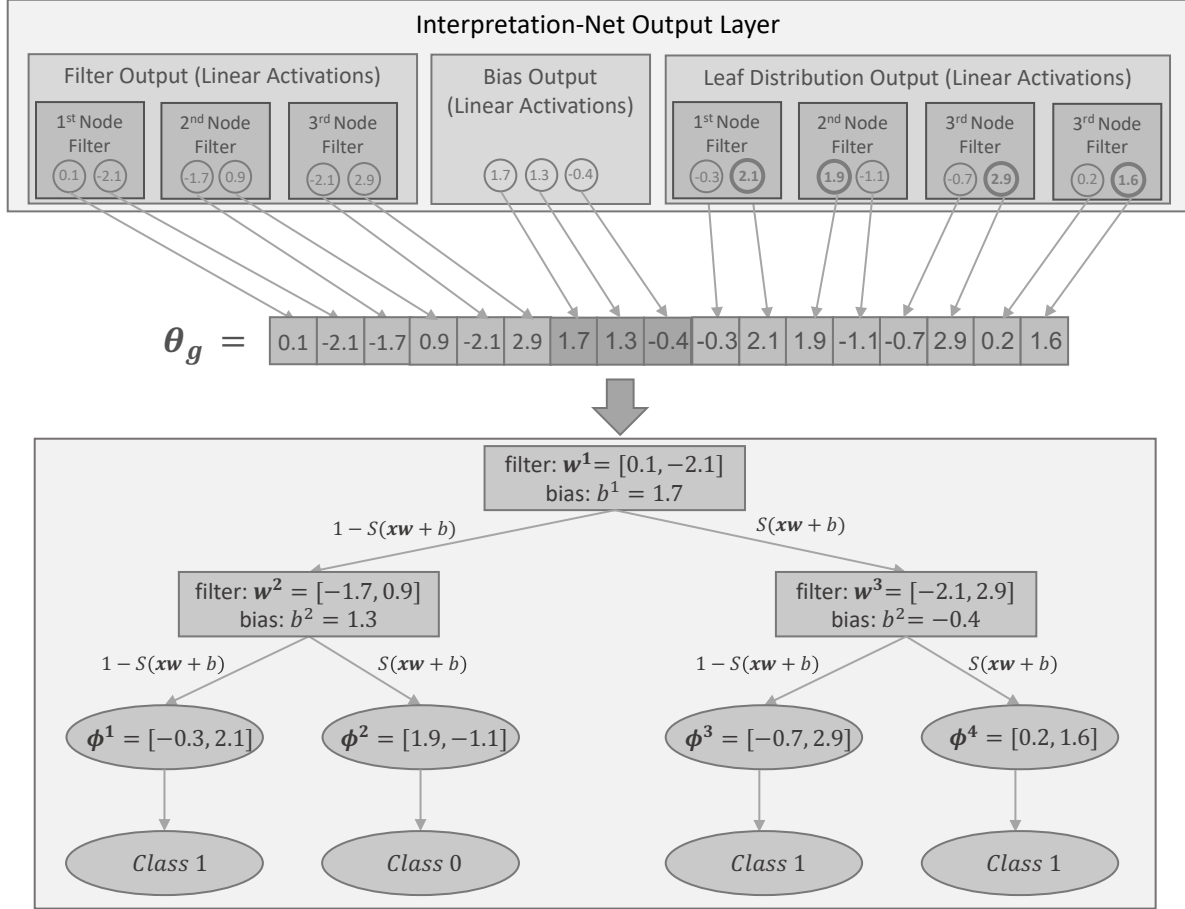


Figure 8: **Exemplary  $\mathcal{I}$ -Net Output for SDTs.** The SDT representation is predicted by the  $\mathcal{I}$ -Net using three separate output layers with linear activation functions. The output shows an exemplary SDT of depth two for a binary classification task on a dataset with two features.

- 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 (Section 4.2.1).
- We empirically investigate the fidelity of the  $\mathcal{I}$ -Net in comparison to sample-based approaches on real-world datasets if the training data is not accessible (Section 4.2.2).
- We perform an ablation study showing the effect of the improved data generation method introduced in Section 3.1.1 on the  $\mathcal{I}$ -Net performance for real-world datasets (Section 4.2.3).
- We present a case study of credit card default prediction, comparing the explanations for a neural network generated by the  $\mathcal{I}$ -Net and sample-based approaches without access to the training data (Section 4.2.4).

#### 4.1 Experimental Setup

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

- **Standard Decision Trees:** For standard DTs, we used the implementation from sklearn<sup>2</sup> (Accessed 15.05.2022) which uses the CART algorithm for DT induction [Breiman et al., 1984].
- **Univariate Soft Decision Trees:** For univariate SDTs, we used the learning algorithm introduced by Frosst and Hinton [2017] with our adjustments described in Section 3.2.2.

<sup>2</sup>Available under: <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>

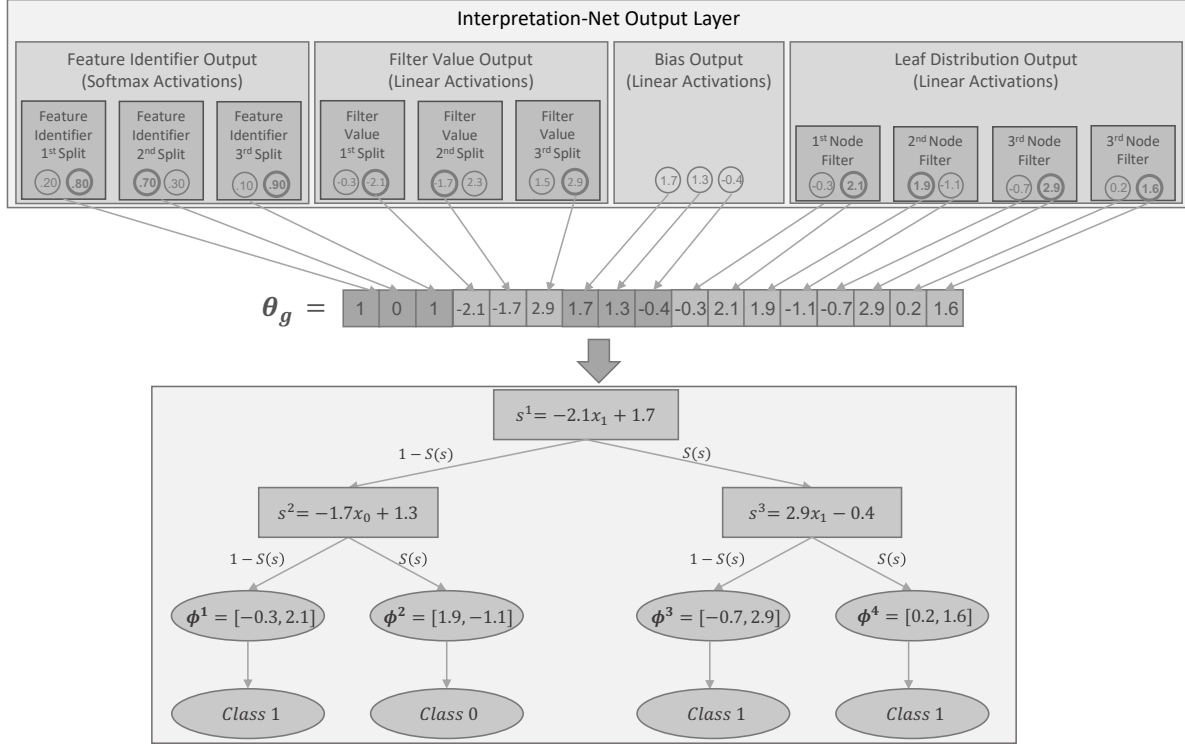


Figure 9: **Exemplary  $\mathcal{I}$ -Net Output for Univariate SDTs.** The univariate SDT representation is predicted by the  $\mathcal{I}$ -Net using four separate output layers with different activation functions. The output shows an exemplary univariate SDT of depth two for a binary classification task on a dataset with two features.

- **Standard Soft Decision Trees:** For SDTs, we used the algorithm proposed by Frosst and Hinton [2017].

The hyperparameters used during our experiments are summarized in Table 8-9.

Since we assume that the training data is not available, we needed to generate data for querying the neural network to distill a surrogate model. 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 10000 data points. However, this is only necessary for the sample-based approaches and the  $\mathcal{I}$ -Net as sample-free approach does not rely on querying to generate explanations. Increasing the number of sampling points further did not enhance the fidelity of sample-based approaches (see Figure 14), but only increases their runtime. We further assumed, that the data used for training the neural network was scaled to be in  $[0, 1]$  and therefore, we also scaled the sampled data to the same interval.

The network parameters  $\Theta_\lambda$  for training the  $\mathcal{I}$ -Net were generated according to Algorithm 1 using the hyperparameters in Table 7. The hyperparameter  $p$  which defines the maximum value for the distribution parameters was fixed to 5 during the data generation for all experiments. Furthermore, we excluded all datasets that were linearly separable during the data generation to focus on more complex and reasonable datasets. The  $\mathcal{I}$ -Net hyperparameters are summarized in Table 6 and were tuned using a greedy neural architecture search according to Jin et al. [2019] followed by a manual fine-tuning of the selected values. We selected one  $\mathcal{I}$ -Net architecture for each of the three function families.

The neural networks considered during the evaluation in Section 4.2 were trained with the hyperparameters summarized in Table 7. The fidelity between the surrogate model and the neural network is always calculated based on the test split of the original data.

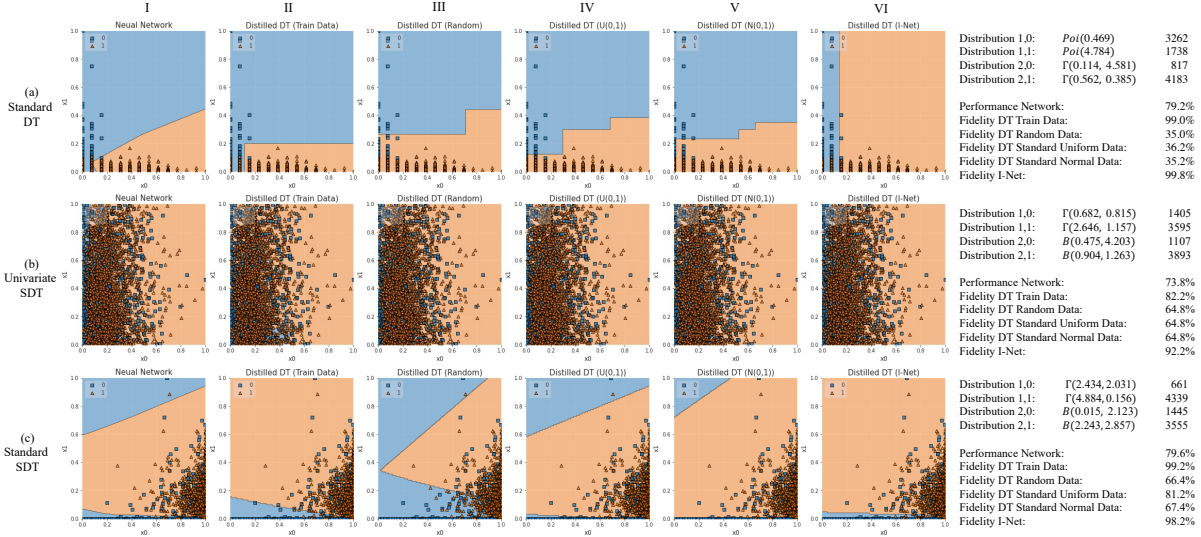


Figure 10: **Visual Decision Boundary Evaluation.** This figure shows the decision boundaries of the neural network we want to interpret (I), followed by the decision boundary of explanations generated by different approaches, along with their performance for three different datasets and function families. 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.

## 4.2 Experimental Results

### 4.2.1 Visual Evaluation for Different Distributions

In this experiment, we show the importance of knowing the distribution of the training data in a controlled, synthetic setting. We use a two-dimensional dataset which allows a visual comparison of the decision boundaries (Figure 10). The data used for training the neural networks 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 10(a) shows a decision boundary learned by a neural network that ranges from the bottom left corner to the middle right. Thereby, many data points that were assigned to *Class 0* by the neural network 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 neural network, 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 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 Section 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 univariate and standard SDTs 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 non-relevant 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

In this experiment, we compare the performance of the  $\mathcal{I}$ -Net and a sample-based distillation without access to train data on real-world datasets. We selected 8 commonly used datasets, mostly focusing on the banking and medical domain, comprising personal, confidential data where it is realistic to assume that the training data cannot be exposed. A description of the datasets, the preprocessing that was performed, and the performance of the neural networks we want to interpret is summarized in Appendix B.

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

Table 2: Real-World Evaluation Results for Standard Decision Trees.

Dataset	$\mathcal{I}$ -Net	Multi-Distribution	Standard Uniform	Standard Normal
Titanic (n=9)	<b>95.51</b> $\pm$ <b>0.00</b>	47.75 $\pm$ 14.70	67.19 $\pm$ 10.50	61.80 $\pm$ 8.35
Medical Insurance (n=9)	<b>84.96</b> $\pm$ <b>0.00</b>	56.47 $\pm$ 13.40	48.05 $\pm$ 10.97	49.62 $\pm$ 10.79
Breast Cancer Wisconsin Original (n=9)	<b>73.91</b> $\pm$ <b>0.00</b>	30.87 $\pm$ 2.25	31.88 $\pm$ 0.00	33.33 $\pm$ 7.45
Wisconsin Diagnostic Breast Cancer (n=10)	<b>82.14</b> $\pm$ <b>0.00</b>	48.57 $\pm$ 22.64	28.57 $\pm$ 0.00	28.57 $\pm$ 0.00
Heart Disease (n=13)	<b>63.33</b> $\pm$ <b>0.00</b>	60.00 $\pm$ 0.00	60.00 $\pm$ 0.00	60.00 $\pm$ 0.00
Cervical Cancer (n=15)	83.53 $\pm$ 0.00	77.06 $\pm$ 20.70	84.59 $\pm$ 0.35	<b>84.71</b> $\pm$ <b>0.00</b>
Loan House (n=16)	<b>100.00</b> $\pm$ <b>0.00</b>	13.11 $\pm$ 0.00	13.11 $\pm$ 0.00	13.11 $\pm$ 0.00
Credit Card Default (n=23)	<b>75.03</b> $\pm$ <b>0.00</b>	65.25 $\pm$ 16.47	74.73 $\pm$ 0.00	65.57 $\pm$ 16.60
Mean Fidelity	82.30	49.89	51.02	49.59

Table 3: Real-World Evaluation Results for Univariate Soft Decision Trees.

Table 2-4 show the results on the selected datasets for the  $\mathcal{I}$ -Net and a sample-based distillation using the data generation methods introduced in Section 4.1 for the different function families. For the sample-based distillation, the results show 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. For a better visual comparison, we highlighted a value bold if it shows a significantly higher fidelity for a certain dataset with 95% confidence according to an unpaired t-test<sup>3</sup>.

**Standard Decision Trees** Comparing the results for standard DTs as surrogate model in Table 2, the  $\mathcal{I}$ -Net had a significantly higher fidelity on 5/8 datasets. A sample-based distillation achieved the highest fidelity on 2/8 datasets. Even though the multi-distribution sampling strategy did not achieve the best performance on any dataset, it had the highest average performance among the sample-based distillation methods, with a mean fidelity of 73.2%. Nevertheless, the performance of the  $\mathcal{I}$ -Net was significantly better and achieved a mean fidelity of 86.19%. Especially for the *Breast Cancer Wisconsin Original* and *Wisconsin Diagnostic Breast Cancer*, the sample-based distillation was not able to generate accurate explanations if the training data was not accessible. For those datasets, the fidelity of sample-based distillation was often even worse than a random guess, which, as already shown in Section 4.2.1, highlights the importance of querying the model on reasonable data points.

**Soft Decision Trees** For univariate SDTs, the performance difference between  $\mathcal{I}$ -Nets and sample-based methods is even more pronounced:  $\mathcal{I}$ -Nets achieved a significantly higher fidelity in 7/8 cases with a mean fidelity of 82.3%, while the best sampling strategy only achieved a mean fidelity of 51.02%.

Finally, for standard SDTs, the  $\mathcal{I}$ -Net achieved a significantly higher fidelity on 3/8 datasets. The best sampling strategy for SDTs was a standard uniform distribution, which achieved a significantly higher accuracy than the  $\mathcal{I}$ -Net on 2/8 datasets. Comparing the mean fidelities, the performance of the  $\mathcal{I}$ -Net with a fidelity of 92.1% was considerably higher than sampling from a standard uniform distribution (82.2%). This was mainly caused by the superior performance of the  $\mathcal{I}$ -Net on the *Cervical Cancer* and *Credit Card Default* dataset, where the fidelity was more than 40 percentage points higher than sampling from standard uniform distribution.

<sup>3</sup>For the t-test calculation, we only compared the  $\mathcal{I}$ -Net with the distillation approach for the sampling strategy with the highest fidelity. Therefore, a bold value only means that a sampling strategy was significantly better than the  $\mathcal{I}$ -Net and not that it was significantly better than the other sampling strategies. A bold value for the  $\mathcal{I}$ -Net means that it achieved a significantly higher accuracy than the best sampling strategy for the respective dataset.

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

Table 4: Real-World Evaluation Results for Standard Soft Decision Trees.

Dataset	Standard DT		Univariate SDT		Standard SDT	
	new	old	new	old	new	old
Titanic (n=9)	<b>95.51</b>	39.33	<b>95.51</b>	60.67	<b>95.51</b>	86.52
Medical Insurance (n=9)	<b>82.71</b>	72.93	<b>84.96</b>	77.44	77.44	<b>92.48</b>
Breast Cancer Wisconsin Original (n=9)	<b>97.10</b>	31.88	<b>73.91</b>	31.88	<b>100.00</b>	98.55
Wisconsin Diagnostic Breast Cancer (n=10)	<b>80.36</b>	28.57	<b>82.14</b>	28.57	<b>94.64</b>	83.93
Heart Disease (n=13)	<b>73.33</b>	60.00	<b>63.33</b>	60.00	<b>100.00</b>	80.00
Cervical Cancer (n=15)	84.71	84.71	<b>83.53</b>	15.29	<b>85.88</b>	84.71
Loan House (n=16)	<b>100.00</b>	13.11	<b>100.00</b>	13.11	100.00	100.00
Credit Card Default (n=23)	<b>75.80</b>	74.73	<b>75.03</b>	25.27	<b>83.30</b>	64.97
Mean Fidelity	<b>86.19</b>	50.66	<b>82.30</b>	39.03	<b>92.10</b>	86.40

Table 5: Data Generation Performance Comparison.

Furthermore, we observed that the average fidelity of standard SDTs is considerably higher than the fidelity of univariate SDTs and 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 shown in Section 3.2.2. This can also explain why the performance difference between a sample-based distillation and the  $\mathcal{I}$ -Net is comparatively small for SDTs: While using meaningful samples for querying the neural network 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 for each type of surrogate model considered in this paper. In total, the  $\mathcal{I}$ -Net achieved a significantly higher fidelity in 15/24 evaluated cases and only in 5/24 cases a significantly lower fidelity. 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 of the surrogate model. This can be crucial since wrong explanations can lead to wrong decisions, as we will evaluate more in-depth in Section 4.2.4.

### 4.2.3 Ablation Study

In Section 3.1.1 we introduced an improved data generation method which should be 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 5, we were able to outperform the standard data generation method in 7/8 datasets for standard DTs, 8/8 datasets for univariate SDT and 6/8 datasets for the standard SDT. Summed up, the improved data generation method achieved a higher fidelity in 21/24 cases, while the old data generation only achieved a higher fidelity in 1/24 cases. 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 and  $\approx 43$  percentage points for univariate SDTs. For standard 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 standard STDs, as already discussed in Section 4.2.2.

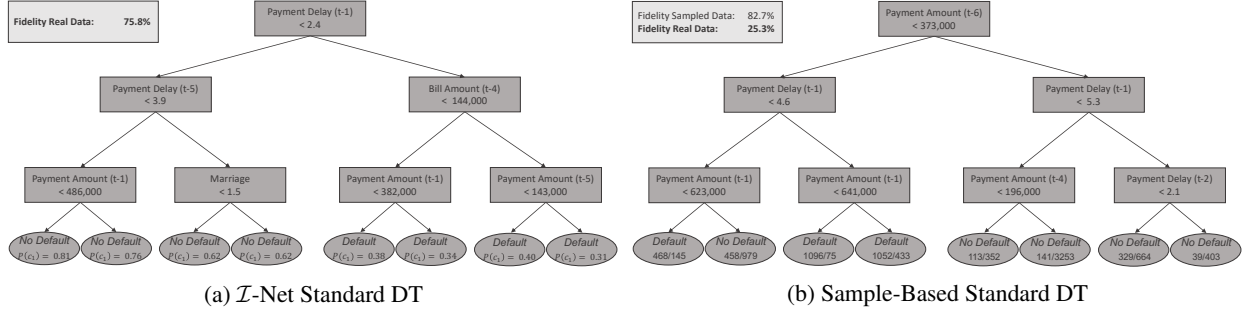


Figure 11: **Explanation Comparison for Standard Decision Trees.** The surrogate model on the right is learned by a sample-based distillation with a multi-distribution sampling strategy. The DT on the left is predicted by the  $\mathcal{I}$ -Net. The  $\mathcal{I}$ -Net makes reasonable splits and achieves a significantly higher fidelity on the real data.

#### 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 Section 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 neural network. We want to note that without access to the training data, it is not possible to identify for a specific surrogate model whether it contains a misconception or not, since we are not able to calculate a representative fidelity.

The *Credit Card Default* dataset is concerned with credit card default prediction based on 23 features including demographic data and the credit history of clients in Taiwan [Yeh and Lien, 2009]. The demographic features include the sex, the education level, the marital status and the age. Furthermore, the repayment status, the amount of the bill and the amount of previous payments, each for the past 6 months and the credit limit are listed for each client. The dataset is available in the UCI Machine Learning repository [Dua and Graff, 2017], and a more detailed summary of the features with a short explanation is given in Table 11.

Figure 11 shows the standard DT surrogate models generated by the  $\mathcal{I}$ -Net and a sample-based distillation using the multi-distribution sampling strategy. As shown in Figure 11a, the  $\mathcal{I}$ -Net achieved 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, if we assume that if a client was in default previously, there is a higher possibility that there will be a default again. If we take the probabilities at the leafs into account, we can get some more information on the decision process. If the payment for 5 months ago was also delayed less 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. Furthermore, as we can see in the split at the bottom-right, if the past payment amount was higher than 143,000, the chance of a default is approximately 10% higher in this branch.

In contrast, when taking a closer look at the DT generated by a sample-based distillation (Figure 11b), 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 whether there were defaults previously into account. This is confirmed by the poor fidelity of the surrogate on the real data, which was only 25.3% and worse than a random guess. However, the surrogate model had a very high fidelity of 82.7% on the sampled data used for querying the model which leads to this misconception, 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 this misconception and taking the high fidelity on the sampled data into account, we might assume that the surrogate actually represents what the model has learned and therefore would make wrong assumptions on its behavior.

We can observe similar misconceptions for the function family of univariate and standard SDTs. However, we will not discuss them here in detail, but refer to the Appendix C for an in-depth evaluation of the explanations generated for those function families.



## 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 neural networks into DTs to interpret the underlying decision function.

Model distillation is a common technique to transfer knowledge from a complex model into a surrogate model [Buciluă et al., 2006, Hinton et al., 2015]. It can be used to obtain more compact model representations for efficiency reasons [Buciluă et al., 2006, Hinton et al., 2015, Furlanello et al., 2018] or to interpret the model as a human-understandable function [Frosst and 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 and Shavlik, 1996, Boz and Hillman, 2000, Zhang et al., 2019] or to improve the performance of an interpretable algorithm to use it instead of the neural network at test time [Krishnan et al., 1999, Frosst and 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 and Shavlik, 1996, Krishnan et al., 1999, Boz and Hillman, 2000, Johansson and Niklasson, 2009, Frosst and Hinton, 2017, Liu et al., 2018, Zhang et al., 2019, Nguyen et al., 2020]. These approaches have in common that they transform a trained neural network into a surrogate function with a tree-like structure. This is usually achieved by maximizing the fidelity to the neural network 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 et al., 1999, Boz and Hillman, 2000, Liu et al., 2018],  $m$ -of- $n$  [Craven and Shavlik, 1996] or multivariate [Nguyen et al., 2020, Frosst and 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, the interpretation becomes harder.

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 et al. [1999] use ID3 [Quinlan, 1986] and C4.5 [Quinlan, 2014], Craven and Shavlik [1996] use ID2-of-3 [Murphy and Pazzani, 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.

In the literature, the data to maximize the fidelity is either the training data used for the neural network [Frosst and Hinton, 2017, Liu et al., 2018] or data from a distribution that was modeled based on the train data [Craven and Shavlik, 1996, Krishnan et al., 1999, Boz and Hillman, 2000, Johansson and Niklasson, 2009]. The latter has been claimed to be an effective way to improve the results [Boz and Hillman, 2000, Johansson and Niklasson, 2009].

In all cases, the predictions on data are the only source for understanding the black-box model behavior, and thus meaningful examples are crucial for the performance. Without information about the distribution of the training data, e.g., in the form of data points, the performance of sample-based methods decreases significantly. Recent model distillation 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 their 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 requires 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 neural network. 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 neural network 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 if no training data is available, 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 *Interpretation Networks* ( $\mathcal{I}$ -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 therefore 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, we allow the  $\mathcal{I}$ -Net to learn how to generalize to neural networks trained on different data distributions. Thereby, the  $\mathcal{I}$ -Net identifies which aspects learned by the neural network 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 once they have no 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 neural network 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.

## A Hyperparameter Summary

The hyperparameters for the  $\mathcal{I}$ -Net (Table 6) were tuned using a greedy neural architecture search according to Jin et al. [2019], followed by a manual fine-tuning of the selected values. To measure the performance during the optimization, we used the validation loss on a distinct validation set  $\Theta_\lambda$  comprising 1000 network parameters.

	Parameter	Value
DT	Hidden Layer Neurons	[1792, 512, 512]
	Hidden Layer Activation	Sigmoid
	Dropout	[0, 0, 0.5]
Univariate SDT	Hidden Layer Neurons	[4096, 2048]
	Hidden Layer Activation	Swish <sup>a</sup>
	Dropout	[0, 0.5]
Standard SDT	Hidden Layer Neurons	[1792, 512, 512]
	Hidden Layer Activation	Swish <sup>a</sup>
	Dropout	[0.3, 0.3, 0.3]
	Batch Size	256
	Optimizer	Adam
	Learning Rate	0.001
	Loss Function	$\mathcal{L}_{\mathcal{I}\text{-Net}}$
	Training Epochs	500
	Early Stopping	Yes
Number of Training Samples		9,000

<sup>a</sup> The Swish activation function proposed by Ramachandran et al. [2017] is defined by  $swish(x) = x \times sigmoid(x)$  and is claimed to consistently match or outperform a ReLU activation.

Table 6:  $\mathcal{I}$ -Net Training Parameters.

Parameter	Value
Hidden Layer Neurons	[128]
Hidden Layer Activation	ReLU
Dropout	No
Batch Size	64
Optimizer	Adam
Learning Rate	0.001
Loss Function	binary_crossentropy
Training Epochs	1,000
Early Stopping	Yes
Number of Training Samples	5,000

Table 7:  $\lambda$ -Net Training Parameters.

Parameter	Value
max_depth	3
criterion	gini
min_samples_split	2
min_samples_leaf	1

Table 8: Standard DT Training Parameters.

Parameter	Value
depth	3
learning_rate	0.01
criterion	binary_crossentropy
lambda	0.001
beta	1
weight_decay	0.0005
maximum_path_probability	True

Table 9: SDT Training Parameters.

## B Real-World Dataset Specification

A specification of the datasets along with the source of the dataset and the performance of the neural network that was learned can be found in Table 10. A specification of the hyperparameters for learning the neural networks can be found in Table 7.

For the *Medical Insurance* dataset, the original objective is to predict the individual medical cost charged by the insurance. We transformed this to a classification task with the objective of predicting whether the medical cost is greater than 10000\$ or not. The *Heart Disease* dataset originally contains 75 attributes. However, only 13 are commonly used in published experiments. Therefore, we similarly only used these 13 features. Furthermore, We distinguish only between *Presence* (values 1, 2, 3, 4) and *Absence* (value 0), as it is common practice. For the *Cervical Cancer* dataset, we selected relevant features similar to the experiments conducted by Molnar [2020] and Fernandes et al. [2017].

For the remainder of the datasets, no feature selection or specific transformation was performed. We used standard preprocessing for all datasets which includes the following steps:

1. Remove all features comprising identifier features (e.g., IDs, Names).
2. Impute missing values: For numeric values we used the mean for imputation, for ordinal, categorical and nominal features, we used the mode.
3. Transform ordinal features to numeric values.
4. One-hot-encode categorical and nominal features.

5. Scale features in  $[0, 1]$  using min-max normalization.
6. Split data into distinct train (85%), valid (5%) and test set (10%).
7. Rebalance train data if number of samples of minority class is less than 25%.

Dataset	Number of Features Preprocessed (Raw)	Number of Samples (True/False)	Citation	Source	Network Performance
Titanic	9 (12)	891 (342/549)	-	<a href="https://www.kaggle.com/c/titanic/data">https://www.kaggle.com/c/titanic/data</a>	83.15
Medical Insurance	9 (7)	1338 (626/712)	Lantz [2019]	<a href="https://www.kaggle.com/datasets/mirichoi0218/insurance">https://www.kaggle.com/datasets/mirichoi0218/insurance</a>	95.49
Brest Cancer Wisconsin	9 (10)	699 (241/458)	Dua and Graff [2017] Mangasarian and Wolberg [1990]	<a href="https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29">https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29</a>	97.10
Wisconsin Diagnostic Breast Cancer	10 (10)	569 (212/357)	Dua and Graff [2017]	<a href="https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29">https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29</a>	98.21
Heart Disease	13 (65)	303 (164/139)	Dua and Graff [2017] Detrano et al. [1989]	<a href="https://archive.ics.uci.edu/ml/datasets/heart+disease">https://archive.ics.uci.edu/ml/datasets/heart+disease</a>	93.33
Cervical Cancer	15 (36)	858 (55/803)	Dua and Graff [2017] Fernandes et al. [2017]	<a href="https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29">https://archive.ics.uci.edu/ml/datasets/Cervical+cancer+%28Risk+Factors%29</a>	84.71
Loan House	16 (12)	614 (422/192)	-	<a href="https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/">https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/</a>	77.05
Credit Card Default	23 (23)	30000 (23364/6636)	Dua and Graff [2017] Yeh and Lien [2009]	<a href="https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients">https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients</a>	78.30

Table 10: **Dataset Specifications.** This table shows the dataset specifications, including the number of features and the number of samples for each class, along with the accuracy of the neural network that was learned. Furthermore, we list the source of the corresponding datasets. All datasets were accessed last on 15.05.2022.

## C Case Study: Credit Card Default Prediction

### C.1 Credit Card Default Dataset Description

In Table 11 we shortly describe the features of the *Credit Card Default* dataset along with the feature index and name as given by Yeh and Lien [2009].

### C.2 Explaining Neural Networks using Soft Decision Trees

**Univariate Soft Decision Trees** In Figure 12, we can see explanations for a neural network trained on the *Credit Card Default* with univariate SDTs as surrogate model. There are no hard splits in a SDT, but we can still interpret the surrogate model by inspecting the internal nodes  $s^i$ . Thereby, a positive filter value translates into a higher probability of taking the right branch when increasing the corresponding input value.

The univariate SDT predicted by the  $\mathcal{I}$ -Net achieved a fidelity of 75% which is similar to the fidelity of the standard DT. It considers the feature  $x_8$  in the first internal node  $s^1$  which corresponds to the repayment status 5 months earlier. Since we can assume a high correlation between the repayment status for the different months and the target variable, we can consider this as a similar decision in the standard DT predicted by the  $\mathcal{I}$ -Net. A high value for this feature leads to an increased probability of taking the right path where a default of the payment is the most probable outcome in each leaf, which we can again consider as a reasonable explanation for this decision.

For the univariate SDT generated by a sample-based distillation, there are many leaves where a parameter of 0 was learned for both classes, which translates in a prediction of *No Default*, but only with 50% certainty. Additionally, the repayment status is not considered in neither of the first three internal nodes. If the right branch is taken with a higher probability, the univariate SDT always predicts that there will be no default. For this decision, only the bill amount 3 months ago is considered. Similar to the standard DT generated by the sample-based distillation, this decision is made without considering the repayment status at all. Again, the fidelity on the sampled data with 67.1% was significantly higher than the performance on the real data with only 25.3%. Taking this model to get insights to the functioning of the model would again lead to strong misconceptions.

**Standard Soft Decision Trees** Figure 13 shows standard SDTs as surrogate model. For the  $\mathcal{I}$ -Net, the fidelity of the standard SDTs was approximately 8 percentage points higher than the fidelity of univariate DTs. However, this

Feature Index	Feature Name	Explanation
0	LIMIT_BAL	Amount of the given credit (NT dollar): it includes both the individual consumer credit and his/her family (supplementary) credit.
1	SEX	Gender (1 = male; 2 = female)
2	EDUCATION	Education (1 = graduate school; 2 = university; 3 = high school; 4 = others)
3	MARRIAGE	Marital status (1 = married; 2 = single; 3 = others)
4	AGE	Age (year)
5	PAY_0	Repayment status in September, 2005. The measurement scale for the repayment status is: 1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; . . . ; 8 = payment delay for eight months; 9 = payment delay for nine months and above.
6	PAY_2	Repayment status in August, 2005 (scale same as above)
7	PAY_3	Repayment status in July, 2005 (scale same as above)
8	PAY_4	Repayment status in June, 2005 (scale same as above)
9	PAY_5	Repayment status in May, 2005 (scale same as above)
10	PAY_6	Repayment status in April, 2005 (scale same as above)
11	BILL_AMT1	Amount of bill statement in September, 2005 (NT dollar)
12	BILL_AMT2	Amount of bill statement in August, 2005 (NT dollar)
13	BILL_AMT3	Amount of bill statement in July, 2005 (NT dollar)
14	BILL_AMT4	Amount of bill statement in June, 2005 (NT dollar)
15	BILL_AMT5	Amount of bill statement in May, 2005 (NT dollar)
16	BILL_AMT6	Amount of bill statement in April, 2005 (NT dollar)
17	PAY_AMT1	Amount of previous payment in September, 2005 (NT dollar)
18	PAY_AMT2	Amount of previous payment in August, 2005 (NT dollar)
19	PAY_AMT3	Amount of previous payment in July, 2005 (NT dollar)
20	PAY_AMT4	Amount of previous payment in June, 2005 (NT dollar)
21	PAY_AMT5	Amount of previous payment in May, 2005 (NT dollar)
22	PAY_AMT6	Amount of previous payment in April, 2005 (NT dollar)
target	target	Default payment (0=yes, 1=no)

Table 11: **Credit Card Dataset Feature Description.** The description of the feature is based on Yeh and Lien [2009].

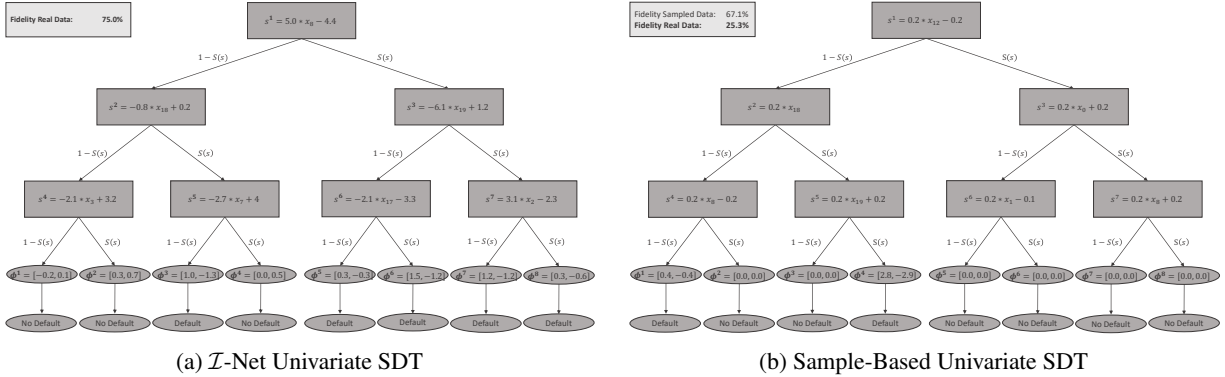


Figure 12: **Explanation Comparison for Univariate Soft Decision Trees.** The surrogate model on the right is learned by a sample-based distillation with a multi-distribution sampling strategy. The tree on the left is predicted by the  $\mathcal{I}$ -Net. The  $\mathcal{I}$ -Net makes reasonable splits and achieves a significantly higher fidelity on the real data.

increase in the fidelity comes with a significant increase in the complexity, since each filter comprises 23 values that are considered for calculating the probabilities of taking the left or right path. This makes standard SDTs much harder to comprehend for humans. However, we can still get some insights if we inspect the filters thoroughly. In  $w_1$  of the SDT predicted by the  $\mathcal{I}$ -Net (Figure 13a), the value at index 6 and 18 have a considerably higher absolute value than the remainder of the values, which makes them especially important for calculating the path probabilities. Accordingly, a high value for the repayment status of the last month (negative filter value) and a low value for the amount of the payment two months ago (positive filter value) strongly increases the probability of taking the left path, which results in a payment default as prediction. Therefore, the surrogate model assumes that most clients that recently had a default in their payment and recently had a low payment amount are likely to default again. However, this does not account for all clients, since the filter comprises values  $\neq 0$  for all 23 features. Accordingly, there exist data points where the probability of taking the right path is higher, even if there is a high value for the repayment status of the last month and a low value for the amount of the payment two months ago. This makes it increasingly hard to understand all aspects of

the explanation, which is usually the goal when global surrogate models are selected as an explanation method. The SDT generated by the sample-based distillation (Figure 13b) is even more difficult to interpret. The filter  $w_1$  comprises many values with a similarly high absolute value and therefore many features have a similar importance, which makes it hard to formulate general explanations. It would be much easier to explain why there is a higher probability for taking a specific path for a certain sample. However, this is not the purpose of a global explanation, but in this instance it would be more reasonable to use a local explanation method. Furthermore, the surrogate model generated by a sample-based distillation again only achieved a high fidelity (90%) on the sampled data, but a very low fidelity on the real data (25%). Therefore, we can assume that the explanation is not able to give insights on the functioning of the neural network that hold in a real world scenario.

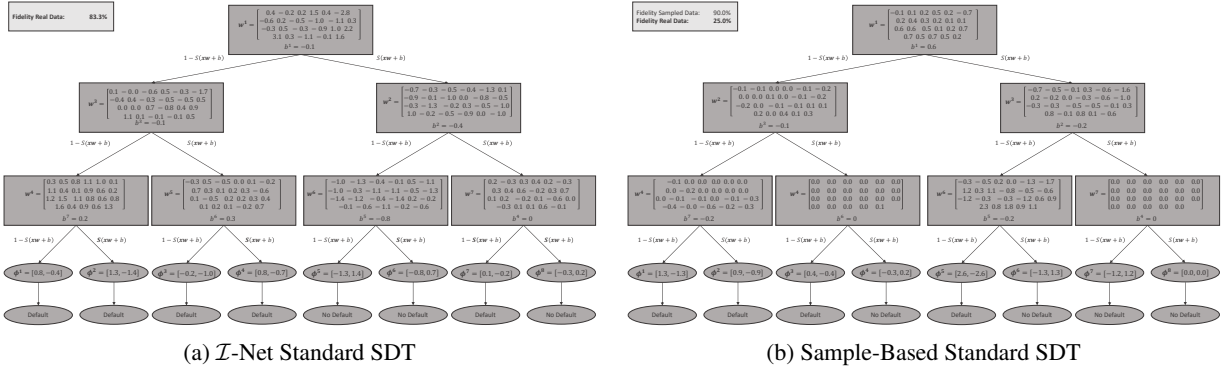


Figure 13: **Explanation Comparison for Standard Soft Decision Trees.** The surrogate model on the right is learned by a sample-based distillation with a multi-distribution sampling strategy. The tree on the left is predicted by the  $\mathcal{I}$ -Net. The  $\mathcal{I}$ -Net makes reasonable splits and achieves a significantly higher fidelity on the real data.

### D Dataset Size for Sample-Based Distillation

Selecting an appropriate number of data points to sample when using a sampling strategy to generate the query data is very crucial. However, with an increasing number of samples, the runtime also increases significantly. Figure 8 shows the mean performance of a sample-based distillation using standard DTs on the real-world datasets in Table 10 for an increasing number of sample points. For each number of samples, we ran 10 independent trials, similar to the experiments conducted in Section 4.2.2.

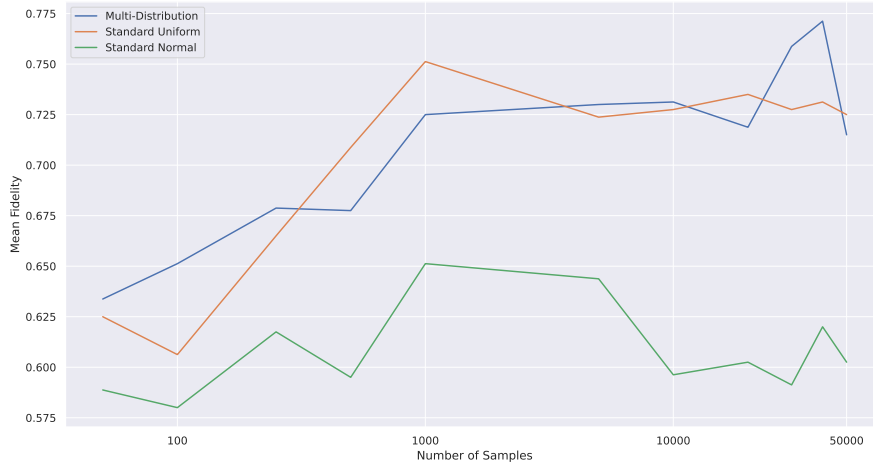


Figure 14: **Dataset Size for Sample-Based Distillation.** This figure shows the mean performance of sample-based approaches on the real-world datasets in Table 10 when increasing the number of samples generated using the different sampling strategies. We can see that there is no considerable performance increase when increasing the number of samples above 10000.

## References

- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.
- Xizhao Wang, Yanxia Zhao, and Farhad Pourpanah. Recent advances in deep learning, 2020.
- Wojciech Samek, Grégoire Montavon, Andrea Vedaldi, Lars Kai Hansen, and Klaus-Robert Müller. *Explainable AI: interpreting, explaining and visualizing deep learning*, volume 11700. Springer Nature, 2019.
- Christoph Molnar. *Interpretable Machine Learning*. Lulu. com, 2020.
- Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.
- Raphael Gontijo Lopes, Stefano Fenu, and Thad Starner. Data-free knowledge distillation for deep neural networks. *arXiv preprint arXiv:1710.07535*, 2017.
- Kartikeya Bhardwaj, Naveen Suda, and Radu Marculescu. Dream distillation: A data-independent model compression framework. *arXiv preprint arXiv:1905.07072*, 2019.
- Gaurav Kumar Nayak, Konda Reddy Mopuri, Vaisakh Shaj, R Venkatesh Babu, and Anirban Chakraborty. Zero-shot knowledge distillation in deep networks. *arXiv preprint arXiv:1905.08114*, 2019.
- I-Cheng Yeh and Che-hui Lien. 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, 2009.
- Sascha Marton, Stefan Lüdtke, and Christian Bartelt. Explanations for neural networks by neural networks. *Applied Sciences*, 12(3):980, 2022.
- Lawrence M Leemis and Jacquelyn T McQueston. Univariate distribution relationships. *The American Statistician*, 62(1):45–53, 2008.
- Johnathan Mun. Understanding and choosing the right probability distributions. *Advanced analytical models: Over 800 models and 300 applications from the basel II accord to Wall Street and beyond*, pages 899–917, 2015.
- Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.
- Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956. ACM, 2019.
- Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL <http://archive.ics.uci.edu/ml>.
- Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.
- Zachary C Lipton. The mythos of model interpretability. *Queue*, 16(3):31–57, 2018.
- Cristian Bucilua, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- Tommaso Furlanello, Zachary C Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. *arXiv preprint arXiv:1805.04770*, 2018.
- Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640*, 2018.
- Mark Craven and Jude W Shavlik. Extracting tree-structured representations of trained networks. In *Advances in neural information processing systems*, pages 24–30, 1996.
- Olcay Boz and Donald Hillman. *Converting a trained neural network to a decision tree dectext-decision tree extractor*. Citeseer, 2000.
- Quanshi Zhang, Yu Yang, Haotian Ma, and Ying Nian Wu. Interpreting cnns via decision trees. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6261–6270, 2019.
- R Krishnan, G Sivakumar, and P Bhattacharya. Extracting decision trees from trained neural networks. *Pattern recognition*, 32(12), 1999.
- Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. In *2018 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 905–912. IEEE, 2018.

- Ulf Johansson and Lars Niklasson. Evolving decision trees using oracle guides. In *2009 IEEE Symposium on Computational Intelligence and Data Mining*, pages 238–244. IEEE, 2009.
- Tung D Nguyen, Kathryn E Kasmarik, and Hussein A Abbass. Towards interpretable deep neural networks: An exact transformation to multi-class multivariate decision trees. *arXiv*, pages arXiv–2003, 2020.
- J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.
- J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.
- Patrick M Murphy and Michael J Pazzani. Id2-of-3: Constructive induction of m-of-n concepts for discriminators in decision trees. In *Machine Learning Proceedings 1991*, pages 183–187. Elsevier, 1991.
- Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.
- Kelwin Fernandes, Jaime S Cardoso, and Jessica Fernandes. Transfer learning with partial observability applied to cervical cancer screening. In *Iberian conference on pattern recognition and image analysis*, pages 243–250. Springer, 2017.
- Brett Lantz. *Machine learning with R: expert techniques for predictive modeling*. Packt publishing ltd, 2019.
- Olvi L Mangasarian and William H Wolberg. Cancer diagnosis via linear programming. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 1990.
- Robert Detrano, Andras Janosi, Walter Steinbrunn, Matthias Pfisterer, Johann-Jakob Schmid, Sarbjit Sandhu, Kern H Guppy, Stella Lee, and Victor Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American journal of cardiology*, 64(5):304–310, 1989.