International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems

Generating Deep Learning Model-Specific Explanations at the End User's Side

Rami Haffar
Najeeb Jebreel
David Sánchez
Josep Domingo-Ferrer

Universitat Rovira i Virgili, Department of Computer Engineering and Mathematics, CYBERCAT-Center for Cybersecurity Research of Catalonia, Av. Països Catalans 26, 43007 Tarragona, Catalonia.
{*rami.haffar, najeeb.jebreel, david.sanchez, josep.domingo*}*@urv.cat*

Received (received date)
Revised (revised date)

End users who cannot afford to collect and label big data to train accurate deep learning (DL) models resort to Machine Learning as a Service (MLaaS) providers, who provide paid access to accurate DL models. However, the lack of transparency in how the providers' models make predictions causes a problem of trust. A way to increase trust (and also to align with ethical regulations) is for predictions to be accompanied by explanations locally and independently generated by the end users (rather than by explanations offered by the model providers). Explanation methods using internal components of DL models (a.k.a. model-specific explanations) are more accurate and effective than those relying solely on the inputs and outputs (a.k.a. model-agnostic explanations). However, end users lack white-box access to the internal components of the providers' models. To tackle this issue, we propose a novel approach allowing an end user to locally generate model-specific explanations for a DL classification model accessed via a provider's API. First, we approximate the provider's model with a local surrogate model. We then use the surrogate model's components to locally generate model-specific explanations that approximate the explanations obtainable with white-box access to the provider's DL model. Specifically, we leverage the surrogate model's gradients to generate adversarial examples that counterfactually explain why an input example is classified into a specific class. Our approach only requires the end user to have unlabeled data of size 0.5% of the provider's training data and with a similar distribution; given the small size and unlabeled nature of these data, they can be assumed to be already available to the end user or even to be supplied by the provider to build trust in his model. We demonstrate the accuracy and effectiveness of our approach through extensive experiments on two ML tasks: image classification and tabular data classification. The locally generated explanations are consistent with those obtainable with white-box access to the provider's model, thus giving end users an independent and reliable way to determine if the provider's model is trustworthy.

Keywords: End-user explanations, Deep learning classification models, Counterfactual explanations, Adversarial examples.

1. Introduction

The classification performance of deep learning (DL) models has significantly improved over the past years in a variety of areas, including computer vision, natural language processing and healthcare.[1,2] However, building highly accurate DL classification models requires a large amount of training data, whose collection and labeling involve a significant effort.[3] Therefore, small businesses and ordinary users, who cannot afford this effort, resort to big technology companies that provide paid API access to highly accurate DL models via Machine Learning as a Service (MLaaS) platforms.[4] These users then query those models with their (small) data and obtain the final classification predictions. In this paper, we refer to the user of a black-box model offered through a provider's MLaaS as an end user.

Even though end users are interested in using MLaaS with highly accurate DL models, they may not entirely trust such models due to the lack of transparency of DL predictions. Obtaining explanations alongside predictions helps end users understand why a DL model produces a specific prediction, which increases the trust in the model and contributes to clearer decision-making.[5,6] Moreover, explaining predictions aligns with ethical regulations, such as the European General Data Protection Regulation (GDPR),[7] which gives end users the right to explanation.

Existing explanation methods for DL models can be divided into model-agnostic methods (*i.e.*, applicable to different types of models, including DL models) and DL model-specific methods (*i.e.*, for specific DL models). Although model-agnostic methods, such as LIME,[8] SHAP[9] and LRP,[10] can explain any model and in particular DL classification models, they only look at models "from the outside", without considering their internal components. In contrast, DL model-specific methods leverage the internal components of a DL model, such as the gradients, to generate more efficient and accurate explanations.[11,12]

End users may certainly obtain the explanations from the MLaaS provider, but this entails blindly trusting the provider and her explanations. To avoid the need for such blind trust, it would be preferable for end users to be able to locally generate explanations using any explanation method they prefer, either model-agnostic or DL model-specific. An end user can generate explanations using model-agnostic methods since they only require the input and the output of the model. However, it is challenging for the end user to generate (the more accurate) DL model-specific explanations because she does not have white-box access to the provider's model. To the best of our knowledge, there is no work that enables an end user with only API access to a remote DL classification model to locally generate explanations using DL model-specific explanation methods.

Contributions and plan of this paper

We propose a novel approach that allows an end user to locally generate DL model-specific explanations for a DL classification model accessed via a provider's API. The approach consists of two main phases: i) approximating the provider's model

by a local surrogate model, and ii) using the surrogate model to locally generate DL model-specific explanations that approximate the explanations obtainable with white-box access to the provider's model.

The originality of our proposed method lies in its being the first proposal that generates accurate explanations on the end-user's side using the mimicked gradients of the provider's model.Our approach requires the end user to have unlabeled data of size about 0.5% of the provider's training data, but it does not require prior knowledge of the provider's model architecture or training hyper-parameters.

First, the (small) unlabeled data available to the end user is augmented[13] for it be more representative of the input data distribution. Then, knowledge distillation (KD)[14] is leveraged to approximate the provider's model by a local surrogate model with nearly equivalent accuracy. To generate DL model-specific explanations, we leverage the surrogate model's gradients to generate adversarial examples[15] that counterfactually explain why an input example is classified into a specific class. In addition, we design a novel method for explaining predictions on tabular data, which makes minimal changes on the smallest number of features to generate understandable counterfactual examples (CE).

We demonstrate the accuracy and the effectiveness of our approach through extensive experiments on two types of ML classification tasks: image classification and tabular data classification. Our results show accurate local explanations consistent with the explanations generated by the provider's model, which give end users an independent and reliable way to determine if the provider's predictions and explanations are trustworthy.

The remainder of this paper is organized as follows. Section 2 discusses related works on generating deep learning model explanations. Section 3 introduces a number of techniques and methods we leverage to design our solution, and presents the assumptions on the end-user data and knowledge. Section 4 describes our method for generating DL model-specific explanations at the end-user's side. Section 5 details the experimental setup, and reports and discusses empirical results. Finally, in Section 6 we gather conclusions and sketch future research lines.

2. Related work

Several methods in the literature have been proposed to explain decisions made by deep learning classification models. As introduced above, they fall into two main classes:

- *Model-agnostic*. The methods in this class treat the model as a black box and can explain any ML model, including DL models. They approximate the relationship between the input and the output prediction of the black-box model. For example, LIME[8] creates an interpretable surrogate model to approximate the relationship between a prediction and the perturbed examples of the input example. Anchors[16] approximates the relationship between the models prediction and the input example using simple if-then

rules. SHAP[9] measures the contribution of each feature of an example to the output prediction by computing Shapley values for each feature. Unfortunately, model-agnostic methods are either unstable,[17] computationally expensive or prone to misinterpretation.[11] Moreover, they disregard the internal components of DL models that may be useful to generate more accurate explanations.

- *DL model-specific.* This class of methods assume white-box access to the DL model pipeline and use the internal components of the model, such as the gradients, to generate more accurate and robust explanations. The main idea of these methods is to identify the features of the input example that are important to the output classification. For example, saliency maps[18] highlight the pixels of an input image by visualizing the gradient of the model prediction w.r.t. those pixels. Grad-CAM++[19] highlights the regions of important input features computed by the weighted gradients of an output classification w.r.t. the final convolutional layer of a DL model. Another option[20] is to generate adversarial examples that counterfactually explain the predictions of a DL model. In fact, adversarial examples are a sub-class of CEs and therefore can be used for explanations.[11] Counterfactual explanations are understandable and human-friendly: if you make this change in the values of these features, the prediction will change from this to that class. Several black-box methods that do not require access to the internal model components to generate CEs have been proposed.[21–23] However, gradient-based methods perform better in terms of the generality and accuracy of the explanations they generate and have much lower computational costs than methods based on black-box access.[23,24]

End users can use any model-agnostic method to explain a DL model with black-box access, but they have to blindly trust the provider when they need the more precise explanations obtained by exploiting the DL model's internal components. Unlike previous work that confines end users in the model-agnostic class, our goal is to enable them to locally generate DL model-specific explanations.

3. Methods and assumptions

Data augmentation. The use of the original training data to generate virtual examples that are similar but different from the original ones is known as data augmentation.[13] The virtual examples are used to enlarge the support of the training distribution and help the trained models generalize better. For instance, virtual examples of an image can be defined as the set of its horizontal reflections, rotations, and scalings when performing image classification. A well-known data-agnostic and efficient data augmentation method is the Mixup augmentation method.[25] Mixup creates a new virtual example from two different original training examples by applying the same linear combination to the two examples' input features and their corresponding target labels. We use a modified version of Mixup to enlarge the

unlabeled training data of end users.

Knowledge distillation (KD). Distillation methods[14,26] transfer the knowledge from a complex "master" to a simpler "student" model. Our goal of approximating the provider's black-box DL model can be viewed as a special case of KD where the end user leverages some unlabeled data examples to transfer knowledge from the provider's model to a local surrogate model. Note that most existing methods for KD exploit soft labels (probability vector predictions),[14] feature maps of the provider's model intermediate layers[27] or the relationships between different layers or data samples.[28] However, in our case, we only have black-box access to the provider's model and thus access to the hard labels only. Therefore, we use hard labels to transfer knowledge, which is shown to be more effective than the previous approaches.[29]

Adversarial examples. An adversarial example is obtained from an original input example by making minimal feature changes in order to cause an ML model to make a wrong prediction. Most existing methods suggest minimizing the distance between the adversarial and the original example while changing the prediction to the desired (adversarial) label. A gradient-based optimization approach[30] can be used to find adversarial examples for DL models. An improvement is the fast gradient sign method (FGSM) for generating adversarial images.[31] FGSM uses the sign of the gradient of the underlying model to find adversarial examples.

Assumptions. We consider a scenario where an end user u uses a trained DL classification model f_p via a prediction API of a service provider p. Examples of such scenarios are cloud-based platforms (*e.g.*, AmazonML and AzureML), where the end user queries an API with his unlabeled data and obtains the final predictions. We assume the end user knows the input-output shape, but knows nothing about the model architecture and training hyper-parameters. Moreover, we assume p has used a big and representative labeled data set $D_p = \{(x_p^j, y_p^j)\}_{j=1}^{m_p}$ to train f_p, whereas u has only a small unlabeled data set $D_u = \{x_u^i\}_{i=1}^{m_u}$ (with $m_u \ll m_p$). Specifically, we assume the ratio between m_u and m_p to be around 0.5% and for D_u and the unlabeled version of D_p to be similarly distributed. Assuming possession of D_u by u is realistic; given the small size and unlabeled nature of D_u, it can be assumed to be already available to the end user or even to be supplied by the provider to build trust in his f_p model.

4. Explaining deep learning classification model predictions on the user's side

We want to rid end users of the need to blindly trust the providers' explanations and allow them to generate DL model-specific explanations locally. In this way, users can reliably understand how the providers' models make their predictions and determine whether these predictions are trustworthy.

However, in order to generate DL model-specific explanations for the predictions of the provider's model f_p, an end user u needs white-box access to f_p, which he does

not have under MLaaS. To remedy this, we propose a two-phase approach that first approximates f_p through another local surrogate model f_u that has performance near-equivalent to that of f_p by using knowledge distillation.[14] After that, we use f_u's internal components to generate local explanations that approximate the explanations that would be generated using the internal components of f_p. Specifically, we leverage f_u's gradients to generate adversarial examples that counterfactually explain why an input example is classified into a specific class.

The following subsections describe in detail the design of the proposed approach.

4.1. *Data augmentation and surrogate model training*

In the first phase, we need to approximate the provider's model f_p by a local surrogate model f_u having an accuracy as close to that of f_p as possible. This raises two challenges:

(1) The small unlabeled data set D_u available to the end user may not be representative of the distribution of the data D_p used to train f_p. Therefore, using only D_u to distill the knowledge from f_p into f_u may yield poor accuracy and poor explanations based on f_u.
(2) u does not know the suitable model architecture and training hyper-parameters of f_u that bring its accuracy close to that of f_p.

Data augmentation and labeling. To tackle the first challenge, we employ a modified version of the Mixup method[25] to augment D_u and obtain more representative training data. Mixup constructs a virtual input example

$$\hat{x} = \lambda x^i + (1-\lambda)x^j, \tag{1}$$

$$\hat{y} = \lambda y^i + (1-\lambda)y^j, \tag{2}$$

where (x^i, y^i) and (x^j, y^j) are two different examples drawn from the training data with x^i being the input example and y^i is its corresponding target label, and $\lambda \sim \text{Beta}(\alpha, \alpha) \in [0, 1]$, for $\alpha \in (0, \infty)$. Mixup is easy for the end user to implement, it incurs little computation overhead, and it produces valuable augmented data that help train more robust and accurate models.[25,32]

In our case, D_u contains only input examples that do not have corresponding target labels. Thus, we only compose new virtual input examples by mixing each input example $x^i \in D_u$ with other $m_u - 1$ examples with $m_u = |D_u|$. Fig. 1 shows an example of mixing two input images to construct a new virtual image.

Once we obtain the augmented data D_{aug} that contain both the original input examples and the virtual input examples, we query the provider's API with the examples in D_{aug} to obtain the augmented labeled examples D_{lbld}. Note that D_{lbld} will contain $m_u + \binom{m_u}{2}$ different training examples (the original ones plus the virtual ones obtained from pairs of original examples) that can be expected to be more representative of the training data distribution.

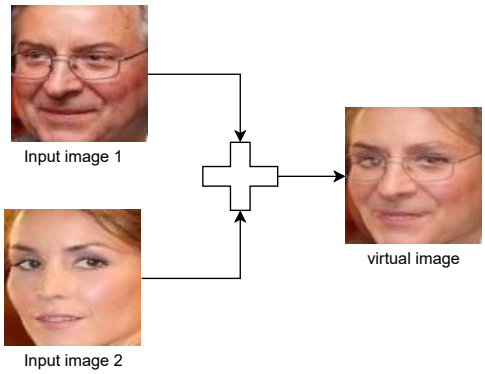


Fig. 1: Virtual input image generated by mixing two original input images with $\lambda = 0.5$

Algorithm 1 describes the methodology we use to augment the small unlabeled data set and then label it using the provider's model f_p.

Algorithm 1 Augmenting unlabeled data and labeling the augmented data

Input: API access to the provider's model f_p, the end user's unlabeled data D_u, parameter α of the beta distribution.
Output: Augmented labeled data D_{lbld}.
$m_u = |D_u|$
$D_{aug} = [\,]$
for $i \in [1, m_u]$ **do**
  $x_i \leftarrow D_u[i]$
  Add(x_i, D_{aug})
  for $j \in [i+1, m_u]$ **do**
    $x_j \leftarrow D_u[j]$
    $\lambda \leftarrow Beta(\alpha, \alpha)$
    $\hat{x} = \lambda x_i + (1-\lambda)x_j$
    Add$(\hat{x}, D_{aug})$
  end for
end for
$D_{lbld} = [\,]$
for each $x_k \in D_{aug}$ **do**
  Add$((x_k, f_p(x_k)), D_{lbld})$
end for
return D_{lbld}

Choosing the model architecture and training hyper-parameters. After obtaining D_{lbld}, the end user needs to define the appropriate model architecture and training hyper-parameters to distill the knowledge from f_p into the local surrogate model f_u using D_{lbld}.

Given the complexity of the classification task and its data distribution, it is possible to estimate a set of candidate models with different complexities that can solve the task.[33] However, we also need to find the appropriate training hyper-parameters, such as learning rate and batch size, that maximize the accuracy of each candidate model.

To this end, we use the *cross-validation search* (CV-SEARCH) technique to select the appropriate hyper-parameters for each candidate model. Specifically, we conduct CV-SEARCH by five-fold cross-validation, which works as follows. The labeled training data are divided into five non-overlapping chunks for each hyper-parameter combination. Then, the average accuracy is aggregated over the validation chunks, saved, and tested in the next hyper-parameter combination. The process is repeated five times, each with a different validation chunk. After that, the hyper-parameter combination that produces the best accuracy on validation chunks is selected for the candidate model.

Given the ample search space, testing out all parameter combinations will take a long time. We leverage Bayesian hyper-parameter optimization[34] to find the appropriate combination efficiently. Bayesian hyper-parameter optimization estimates the validation accuracy of certain hyper-parameter combinations. Then the next validation hyper-parameter combinations are chosen with high expected accuracy. The hyper-parameters we search for are the training epochs, the learning rate and the batch size. The candidate model architecture and the corresponding hyper-parameter combination that have the highest expected validation accuracy are selected to train the local surrogate model f_u.

Note that an alternative solution to train an accurate surrogate model f_u is to use an AutoML platform, such as Google AutoML[a] or Microsoft AutoML[b]. These platforms have advanced optimization algorithms that determine the appropriate model architecture and training hyper-parameters for a given task and training data. In this case, the end user does not need to incur any effort other than paying the AutoML provider for training the model. However, doing so also implies trusting the provider to operate the platform honestly.

4.2. *Generating counterfactual adversarial explanations*

Once the end user obtains a trained local surrogate model f_u that is nearly as accurate as the provider's model f_p, he can use f_u to generate accurate explanations for the predictions of f_p. Since f_u has almost learned the same decision boundaries

[a] `https://cloud.google.com/automl`
[b] `https://www.microsoft.com/en-us/research/project/automl/`

as f_p, explanations generated using f_u's internal components can be expected to accurately approximate the explanations generated using f_p's internal components.

In our work, we explain the provider's model by generating counterfactual explanations[35,36] of a specific example. Counterfactual explanations tell us how to change the example's features so that its predicted label also changes. In this way, we can understand how the model makes its predictions and explain individual predictions. Moreover, counterfactual explanations can be viewed as by-example explanations, which are the preferable kind of explanations for end users.[11,37] We use adversarial training[30,31] as a means to generate adversarial examples that counterfactually explain the model predictions. Adversarial examples are easy to compute when we have white-box access to the gradients of a DL model.[31] In fact, adversarial examples are aimed at fooling the model rather than explaining it, but, in the end, they serve the same purpose as CEs by slightly changing the features of input examples to modify their predicted labels.[11]

We use the following expression to generate adversarial examples:

$$x_* = x - \epsilon \cdot \frac{\partial}{\partial x}\mathcal{L}(f(x), y_*), \tag{3}$$

where x is an input example (represented as a vector of features), x_* is the created adversarial example, y_* is the desired class label, and parameter ϵ is used to minimize the changes made on the original input example to create the adversarial example. Note that we perform a gradient descent optimization by moving the features of the input example in the opposite direction of the gradient, thus causing the model to classify the generated adversarial example into the desired class. If we take y_* as the second most probable class label in the local model prediction vector, we will get a CE with the fewest possible feature changes on the input example that modify the predicted label.

Expression (3) works well for images because the pixel values of an image carry a lot of context, and we can identify the changed pixels and visualize them easily. However, it is more challenging to represent tabular data in a meaningful way because an example may consist of hundreds of (less regular) features. Listing all feature values to describe an example is usually not desirable for end users who look for the minimum changes needed to change the example's prediction. For these reasons, we build on Expression (3) to design two different algorithms that take into account the generation of appropriate explanations, either for image data or tabular data.

Generating explanations for images. To generate an explanation for a specific image x^i, we create an adversarial example x^i_* with small pixel perturbations that cause f_p to change its current prediction $\hat{y^i}_p = f_p(x^i)$ to another desired prediction y^i_*. Finally, we visualize the generated explanations by superimposing a heatmap on the regions of important pixels in x^i according to the difference between x^i and x^i_*.

Algorithm 2 describes the method we use to generate explanations for image

data. Given API access to the provider's model f_p, white-box access to the trained local surrogate model f_u and the maximum allowed value ϵ_{max} for the ϵ parameter, we generate a visualized explanation for an input example x as follows. First, we check whether the provider's model and the surrogate model predict the same class for x. If $f_p(x) \neq f_u(x)$, we cannot go further in the adversarial example generation. This should occur relatively seldom because f_u closely mimics the predictions of f_p. If $f_p(x) = f_u(x)$, let $probs_u$ be the probability vector f_u outputs for x. Then, we set the desired class label y_* to be the index of the second most probable class in $probs_u$. Choosing the second most probable class instead of the other classes guarantees that we make the smallest possible changes to modify the actual prediction of f_p on x. Note, however, that the end user can also set y_* to any class label she wants. After that, we keep repeating the gradient descent step in Expression (3) until one of two following conditions is satisfied: i) an adversarial example x_* is obtained that fools f_p into labeling it as y_* or ii) the maximum value ϵ_{max} is reached for ϵ. Note that we start with a small $\epsilon = 0.005$ and we increase it by 0.005 at each step.

Once we create the adversarial example x_*, we identify the pixel perturbations values that caused the change of the prediction of f_p for x to the desired class y_*. We compute the absolute values of the added perturbations $perturbs = abs(x_* - x)$. Note that the vector *perturbs* has the same size as the original input image x.

Finally, we superimpose the vector *perturbs* on the original image x to visually explain the important pixels that cause the prediction to change from $f_p(x)$ to y_*.

Algorithm 2 Explaining predictions for image data

Input: API access to the provider's model f_p, surrogate model f_u, image to be explained x, maximum allowed value ϵ_{max} for ϵ.
Output: Visualized explanation of x.
if $f_p(x) = f_u(x)$ **then**
  $probs_u \leftarrow$Get_Probabilities($f_u(x)$)
  $y_* \leftarrow argmax(probs_u, 2)$
  $\epsilon = 0.005$
  $x_* \leftarrow x$
  while $f_p(x_*) \neq y_*$ and $\epsilon \leq \epsilon_{max}$ **do**
    $x_* \leftarrow x - \epsilon \cdot \frac{\partial}{\partial x}\mathcal{L}(f_u(x), y_*)$
    $\epsilon \leftarrow \epsilon + 0.005$
  end while
  $perturbs \leftarrow abs(x_* - x)$
  Superimpose($perturbs, x$)
end if

Generating explanations for tabular data. Our method generates counterfactual adversarial examples of tabular data by introducing changes to a small number of attributes of an example to modify its predicted label. These slight

changes are favored by end users when explaining tabular data predictions:[11] instead of overwhelming the user by changing a lot of attributes, changing a small number of attributes facilitates understanding the explanation. The method is very similar to creating adversarial examples for images with the difference of limiting the number of changed attributes.

Given an input record x containing n attributes, we only change $c \leq n$ attributes, where c is a hyper-parameter defining the number of attributes to be changed in x to generate the CE x_*. To choose the c attributes, we start by computing the gradient of the loss between the surrogate model output $f_u(x)$ and the desired output y_* w.r.t. the attributes of the input record. Then, we take the L_1-norm of the computed gradient ∇_x. After that, we identify the attributes with the highest c L_1-norms as the attributes to be changed when generating x_*. We do this by using a weighting vector w that contains 0s for the unchanged attributes and 1s for the changed attributes. In this way, when we compute the gradients w.r.t. the attributes of the input record, we only change the values corresponding to the c attributes. Finally, we return the generated CE x_*. In tabular data, a similar record with slightly changed attribute values counterfactually explains the attributes responsible for changing the predicted label.[11,36]

Note that our method also allows the end user to choose the attributes she prefers to change instead of making an automatic choice. Although there is a negative view on the fairness of tabular CEs,[38] some authors[39] propose that plausible tabular CEs can be used instead of closest CEs to improve the robustness and consequently the individual fairness of counterfactual explanations.

5. Empirical analysis

In this section, we evaluate the performance of the proposed approach on two ML tasks: image classification and tabular data classification. First, we show how accurate the local surrogate models were at approximating the provider's models. Then, we evaluate the DL model-specific explanations generated by the surrogate models, and show their consistency with those generated by the provider's models.

Finally, we compare the visual explanations generated by our method for image data with those generated by a model-agnostic method (LIME[40]), which shows that our explanations are more accurate and understandable.

Our code is available for reproducibility purposes[c].

5.1. *Experimental setup*

We ran our experiments on a machine with AMD Ryzen 5 CPU at a base speed of 3.6 GHz, 32 GB of RAM, and a GPU NVIDIA GeForce GTX 1660 with 6 GB of RAM.

[c]`https://github.com/anonymous16534/User-End-Explanations-of-the-Predictions-of-Deep-Learning-Black-Box-Models`

Algorithm 3 Explaining predictions for tabular data

Input: API access to the provider's model f_p, local surrogate model f_u, record to be explained x, maximum allowed value ϵ_{max} for ϵ, number of attributes to be changed c.
Output: Counterfactual example x_*.
if $f_p(x) = f_u(x)$ **then**
  $probs_u \leftarrow$Get_Probabilities($f_u(x)$)
  $y_* \leftarrow argmax(probs_u, 2)$
  $n \leftarrow$ Number of attributes in x
  $|\nabla_x| \leftarrow abs(\frac{\partial}{\partial x}\mathcal{L}(f_u(x), y_*))$
  $w \leftarrow$ Zero vector of length n
  $idxs \leftarrow$ Indices of the highest c values in $|\nabla_x|$
  for $idx \in idxs$ **do**
    $w[idx] = 1$
  end for
  $x_* \leftarrow x$
  $\epsilon \leftarrow 0.005$
  while $f_p(x_*) \neq y_*$ and $\epsilon \leq \epsilon_{max}$ **do**
    $x_* \leftarrow x - w \cdot \epsilon \cdot \frac{\partial}{\partial x}\mathcal{L}(f_u(x), y_*)$
    $\epsilon \leftarrow \epsilon + 0.005$
  end while
  Return x_*
end if

Data sets and provider models. We evaluated the proposed approach on three data sets:

- **Gender**[d] is a binary classification data set from Kaggle. It consists of RGB images of male and female faces with a balanced training set of 23,200 female images and 23,800 male images. The validation set contains 11,600 images evenly divided into the two classes. Since the images have large and different sizes, we first resized them to 100×100 pixels in order to train our models faster. The provider's model was the deep CNN used in a previous paper.[41] The model was trained for 10 epochs with a batch size 64, the binary cross-entropy loss function and the Adam optimizer[42] with a learning rate 0.001.
- **MNIST** contains 70K handwritten images corresponding to digits 0 to 9.[43] The images are divided into a training set (60K examples) and a validation set (10K examples). We used the LeNet model[44] as the provider's model. It was trained for 100 epochs with a batch size 128, the cross-entropy loss, and the same optimizer and learning rate as the Gender benchmark.

[d]`https://www.kaggle.com/cashutosh/gender-classification-dataset`

- **Adult**[e] is a tabular data set that contains 48,842 records of census income information with 14 numerical and categorical attributes. From these, we dropped the final weights (*fnlwgt*), which reveal too much information to the model, and the *education* attribute, which is redundant with *education-num*. We also encoded categorical attributes as numbers. The class label is the attribute *income*, that classifies records into either > 50K or ≤ 50K. We used 80% of the data as training data, and the remaining 20% as validation data. The provider's DL model for this data set consisted of three hidden layers of 100 neurons each. The sigmoid activation function followed the final layer to produce probabilities. The model was trained for 10 epochs with a batch size 128, and the same loss and optimizer as the two former benchmarks.

Data augmentation and local surrogate models. The end user was assigned 0.5% of each training data set's examples without their classification labels, which corresponds to 235 out of 47,000 images for the Gender data set, 300 out of 60,000 images for the MNIST data set, and 195 out of 39,073 records for the Adult data set. This data was augmented following Algorithm 1. We set the parameter α of the beta distribution to 0.25 for the Adult data set, while we used a constant value $\lambda = 0.5$ for the image data sets. We found that augmenting the image data by taking the average of two images led to distilling more knowledge from the provider's models. Table 1 reports the number of samples the user owns before and after augmenting the data.

Table 1: Number of data samples owned by the service provider and the user before and after the Mixup augmentation

Data set	Provider data	User data before augmentation	User data after augmentation	Time needed to create each example
Gender	47,000	235	27,495	0.045 s
MNIST	60,000	300	22,893	0.0015 s
Adult	39,073	195	56,745	0.0002 s

We used four surrogate models with different architectures for the Gender and the MNIST data sets. Surrogate 1 was shallower than the provider's model, surrogate 2 had the same architecture, and surrogates 3 and 4 were larger than the provider's model. Tables 2 and 3 summarize the architectures of these surrogate models.

On the other hand, in the case of the Adult data set we used a simpler surrogate architecture than the provider's model: same depth as the provider's model but fewer neurons in each layer. The rationale is that this classification task is simple and does not require deep feature extraction, unlike the Gender and the MNIST

[e] `https://archive.ics.uci.edu/ml/datasets/adult`

Table 2: Architectures of the provider's black-box and end user's surrogate models used in the experiments for the Gender classification data set. $C(3, 32, 3, 0, 1)$ denotes a convolutional layer with 3 input channels, 32 output channels, a kernel of size 3×3, a stride of 0, and a padding of 1; $MP(2, 2)$ denotes a max-pooling layer with a kernel of size 2×2 and a stride of 2; and $FC(18432, 2048)$ indicates a fully connected layer with 18,432 inputs and 2,048 output neurons. We used $ReLU$ as an activation function in the hidden layers; lr stands for learning rate.

Model	Model architecture	Hyper-parameters
Provider	C(3,32,3,0,1), C(32,64,3,1,1), MP(2,2),C(64,128,3,0,1), C(128,128,3,1,1), MP(2,2), C(128,256,3,0,1), C(256,256,3,1,1), MP(2,2), C(256,512,3,0,1), C(512,512,3,1,1), MP(2,2), FC(18432,2048), FC(2048,1024), FC(1024,512), FC(512,128), FC(128,32), FC(32,2)	lr = 0.001 epochs = 10 batch = 64
Surrogate 1	C(3,32,3,0,1), C(32,64,3,1,1), MP(2,2), C(64,128,3,0,1), C(128,256,3,1,1),MP(2,2), C(256,512,3,0,1), C(512,256,3,1,1), MP(2,2), FC(36864,1024), FC(1024, 256), FC(256,32), FC(32,2)	lr = 0.0001 epochs = 10 batch = 128
Surrogate 2	Same architecture as the provider's model.	lr = 0.0001 epochs = 10 batch = 128
Surrogate 3	C(3,32,3,0,1), C(32,64,3,1,1), MP(2,2), C(64,128,3,0,1), C(128,256,3,1,1), MP(2,2), C(256,512,3,0,1), C(512,256,3,1,1), MP(2,2), C(256,512,3,0,1), C(512,512,3,1,1), MP(2,2), C(512,512,3,0,1), C(512,512,3,1,1), MP(2,2), FC(4608,1024), FC(1024, 256), FC(256,32), FC(32,2)	lr = 0.0001 epochs = 10 batch = 128
Surrogate 4	C(3,32,3,0,1), C(32,64,3,1,1), MP(2,2), C(64,128,3,0,1), C(128,128,3,1,1), MP(2,2), C(128,256,3,0,1), C(256,256,3,1,1), MP(2,2),C(256,512,3,0,1), C(512,512,3,1,1), MP(2,2), C(512,512,3,0,1), C(512,512,3,1,1), MP(2,2), FC(4608,1024), FC(1024,256), FC(256,128), FC(128,32), FC(32,2)	lr = 0.0001 epochs = 10 batch = 128

tasks. Table 4 depicts the details of both the provider's model and the surrogate model for the Adult data set.

We limited the search space of the Bayesian hyper-parameter optimization[34] to find the best hyper-parameter combination for each surrogate model quickly. The learning rate was searched between 0.0001 and 0.01, the training epochs between 5 and 150, and the batch size between 32 and 128.

Evaluation metrics. We used the following evaluation metrics to measure the performance of the trained surrogate models and the generated explanations:

- *Accuracy*: number of correct predictions divided by the total number of predictions. We used this metric to measure and compare the performance of the provider's and the surrogate models.
- *Structural Similarity Index Measure* (SSIM): similarity between two images x and y measured as

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)},$$

where μ_x is the average of x, μ_y is the average of y, σ_x^2 the variance of x, σ_y^2 the variance of y, σ_{xy} the covariance of x and y, and c_1 and c_2 are two variables to stabilize the division. The values of c_1 and c_2 are calculated as

$$c_1 = (K_1L)^2, c_2 = (K_2L)^2,$$

Table 3: Architectures of the provider's black-box and end user's surrogate models used in the experiments for the MNIST data set

Model name	Model architecture	Hyper-parameters
Provider model	LeNet 5[44]	lr = 0.001, epochs = 100, batch = 128
Surrogate model 1	C(1, 16,0, 2), MP(2, 0) C(16, 32, 0, 2), MP(2, 0), C(32, 64, 0, 2), FC(16384, 200), FC(200, 10)	lr = 0.001, epochs = 100, batch = 128
Surrogate model 2	LeNet 5[44]	lr = 0.001, epochs = 100, batch = 128
Surrogate model 3	C(1, 64, 0, 2), MP(2,0), C(64, 128, 0, 2), MP(2, 0), C(128, 256, 0, 2), FC(1638400, 1024), FC(1024, 200), FC(200, 10)	lr = 0.001, epochs = 100, batch = 128
Surrogate model 4	C(1, 32, 0, 2), MP(2, 0), C(32, 64, 0, 128), MP(2, 0), C(64, 128, 0, 2), C(128, 128, 0, 2), C(128, 64, 0, 2), C(64, 64, 0, 2), FC(16384, 256), FC(256, 10)	lr = 0.001, epochs = 100, batch = 128

Table 4: Architectures of the provider's black-box and end user's surrogate models used in the experiments for the Adult data set

Model name	Model architecture	Hyper-parameters
Provider model	FC(12,100), FC(100,100), FC(100,2)	lr = 0.001, epochs = 10, batch = 128
Surrogate model	FC(12, 64), FC(64, 64), FC(64, 2)	lr = 0.0001, epochs = 100, batch = 64

where L is the dynamic range of the pixel-values, $K_1 = 0.01$, and $K_2 = 0.03$. We used SSIM to measure the similarity between the adversarial images generated by the provider's model and those generated by the surrogate models.

- *Overlap similarity*: similarity between two records x and y with categorical attributes computed as

$$Overlap(x, y) = \Sigma_{k=1}^{d} w_k S(x_k, y_k),$$

where d is the number of the categorical attributes, x_k denotes the k-th attribute in x, y_k denotes the k-th attribute in y, w_k denotes the weight assigned to the k-th attribute, and

$$S(x_k, y_k) = \begin{cases} 1, & \text{if } x_k = y_k, \\ 0, & \text{otherwise.} \end{cases}$$

In our experiments, we set $w_k = 1/d$ for all the attributes. We used this metric to measure the similarity between the counterfactual records generated by the provider's model and those generated by the surrogate model for tabular data.

5.2. *Results and discussion*

5.2.1. *Accuracy of surrogate models*

Table 5 reports the accuracy of the provider's models and the trained surrogate models. We can notice that all the surrogates trained on the small non-augmented data performed very poorly on the unseen data: their predictions were almost random guesses. The best surrogate with the Gender benchmark achieved 53.5% accuracy compared to 96.3% for the provider's model; the best surrogate with the MNIST benchmark achieved 10.35% compared to 99.22% for the provider's model; and the best surrogate with the Adult benchmark achieved 47.2% compared to 84.85% for the provider's model. The reason of that poor performance is that the surrogates overfitted the small non-representative training data sets, which were insufficient to distill the knowledge from the provider's model.

On the other hand, when the surrogates were trained on the more representative augmented data, their performance was nearly equivalent to that of the provider's model. The best surrogate with the Gender benchmark achieved 94.47% accuracy compared to 96.3% for the provider's model; the best surrogate with the MNIST benchmark achieved 96.1% compared to 99.22% for the provider's model, and the best surrogate with the Adult benchmark achieved 84.58% compared to 84.85% for the provider's model. Moreover, *Surrogate 1*, which is simpler than the provider's model, achieved the highest accuracy among all the surrogates. This is not surprising because transferring knowledge from a large "master" DL model to a simpler "student" DL model is more effective and produces highly regularized models.[14]

Table 5: Accuracy of surrogate models, without and with data augmentation, compared to the accuracy of the provider's model

Data set	Provider model	Surrogate 1		Surrogate 2		Surrogate 3		Surrogate 4	
		Non-aug.	Aug.	Non-aug.	Aug.	Non-aug.	Aug.	Non aug.	Aug.
Gender	96.3%	53.8%	94.47%	43.7%	92.89%	48.26%	93.86%	49.68	93.35%
MNIST	99.22%	7.69%	96.1%	8.9%	95.8%	10.35%	94.8%	6.3%	94.5%
Adult	84.85%	47.2%	84.58%	-	-	-	-	-	-

5.2.2. *Surrogate model explanations*

In this section we compare the explanations generated by the surrogate models with those generated by the provider's models.

Image data. We generated an adversarial image for each image in the Gender and MNIST data sets by using the gradients of the provider's model and the four surrogate models. Then, we computed SSIM between the adversarial examples generated by the provider's model and those generated by the surrogates to numer-

ically measure their similarity. Table 6 reports the average SSIM for the Gender and MNIST validation images. We can see that the simplest model (Surrogate 1) generated adversarial examples with the highest similarity to those generated by the provider's model.

On the other side, as the surrogate models got larger and deeper (Surrogates 2, 3 and 4), the similarity between their generated adversarial examples and those generated by the provider's model decreased. Therefore, we can conclude that choosing the simplest model architecture, which gave the highest accuracy in the classification task, leads to a more accurate approximation of the explanations generated by the provider's model.

Table 6: Similarity between the adversarial examples generated by the surrogate models and the ones generated by the provider's model on the Gender and MNIST data sets.

Data set	Surrogate model 1	Surrogate model 2	Surrogate model 3	Surrogate model 4
Gender	**96.79%**	93.42%	91.81%	91.36%
MNIST	**98.27%**	95.63%	93.22%	92.68%

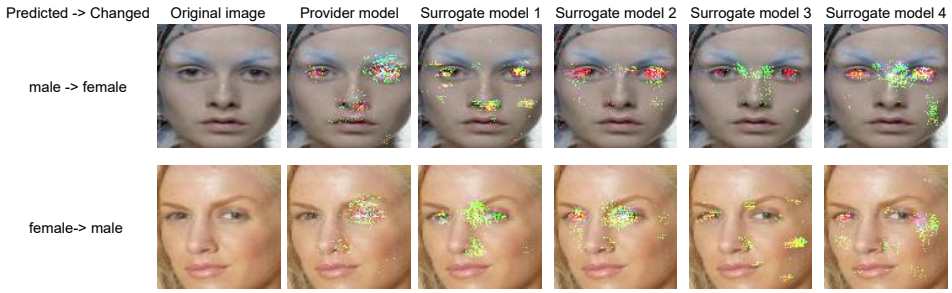
To visually compare the explanations generated by the surrogates with those generated by the provider's models, we computed the absolute perturbations added to the original validation images (*perturbs*) by using Algorithm 2. We then overlaid them as heat maps on the original images. Figure 2a shows two examples of these visual explanations generated for the Gender data set. In the first example (row 1), the provider's model wrongly predicted the original image as male, and the added pixel perturbations changed its prediction to the true prediction female. In the example (row 2), the provider's model correctly predicted the original image as female, and the added pixel perturbations changed its prediction to the wrong prediction male.

By looking at the pixels that caused the prediction to change, we can see that, in general, the explanations generated by the surrogates were consistent with those generated by the provider's model: all the models, including the provider's, identified the pixels corresponding to the regions of the eyes, the nose, the cheek and, sometimes, the lips, as the pixels responsible for changing their prediction. This is in line with the literature on gender recognition,[45,46] which found that the eyes, the nose, the cheek and the lips are the most relevant features for differentiating between male and female faces.

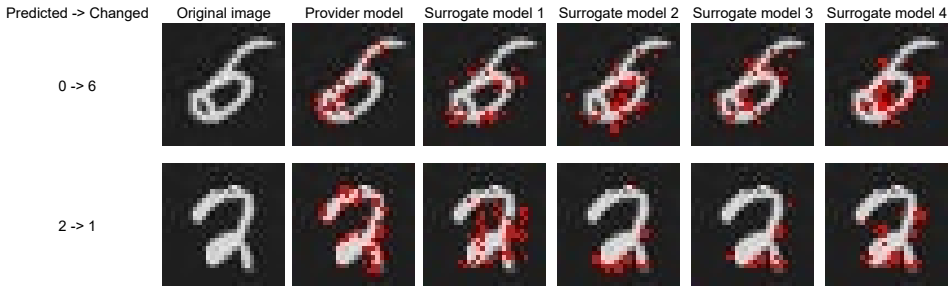
Figure 2b shows two examples of the explanations generated for MNIST. In the first example (row 1), the provider's model wrongly predicted the original image as 0, and the added pixel perturbations changed its prediction to the true prediction 6. In the example (row 2), the provider's model correctly predicted the original

image as 2, and the added pixel perturbations changed its prediction to the wrong prediction 1. Again, the explanations generated by the surrogates were consistent with the explanations generated by the provider's model: all the models identified pixels on the edges of the digits as the important ones. This is also in line with previous works on digit recognition.[47]

To summarize, the visual explanations generated by the surrogates were understandable and consistent with those generated by the provider's models.



(a) Two examples of the explanations generated for the images of the Gender data set



(b) Two examples of the explanations generated for the images of the MNIST data set

Fig. 2: Visual explanations generated by the surrogate models in comparison with those generated by the provider's models.

Tabular data. We generated a CE for each record in the Adult data set by using the gradients of the provider's model and those of the surrogate model. First, we used Algorithm 3 with $c = n$ (that is, $c = 12$) to allow changing all the attributes when generating the examples. Then, we computed the overlap between the records generated by the provider's model and those generated by the surrogate model. We found that the generated records were almost identical, with an average overlap of around 1. This is not surprising because the accuracy of the surrogate model on the classification task was extremely close to the accuracy of the provider's model (84.58% vs. 84.85%).

To generate CEs for a specific validation record while limiting attribute changes,

we used Algorithm 3 with $c \in \{1, \ldots, 12\}$. Table 7 shows two original records, each with four generated CE. CE 1 was generated by changing the *educational number* attribute only. CE 2 was generated by allowing the attributes *educational number* and *age* to be changed. CE 3 was generated by allowing the attributes *educational number*, *age* and *working hours per week* to be changed. Finally, CE 4 was generated by allowing all the attributes to be changed.

Record 1's income was classified as $\leq 50K$ by the provider's model. CE 1 for that record shows that we can change the original record's prediction to $> 50K$ by increasing *educational level* from 13 to 16. This seems logical: in most cases, the higher the education level, the higher the income. CE 2 shows that we can change the prediction by increasing *age* from 31 to 41 and *educational level* from 13 to just 15. This also makes sense: in addition to the impact of the educational level (which is lower in this case), an older age means more working experience and, therefore, higher income.

CE 3 shows that we can change the prediction by increasing *age* from 31 to just 39, *educational level* from 13 to 15 and *working hours per week* from 60 to 61. Certainly, higher education, higher age, and more working hours usually correlate with higher income. Finally, CE 4 shows that we can change the prediction by increasing *age* from 31 to 36, *educational level* from 13 to 14 and *occupation* from *craft repairer* to *executive managerial*. This also makes sense, since besides the age and educational level, the type of occupation also impacts on the income.

On the other hand, record 2's income was classified as $> 50K$ by the provider's model. In this case, the CEs changed the attribute values the other way around. CE 1 shows that we can change the original record's prediction to $\leq 50K$ by decreasing *educational level* from 14 to 7. CE 2 shows that we can change the prediction by decreasing *age* from 46 to 45 and *educational level* from 14 to 7. CE 3 shows that we can change the prediction by decreasing *age* from 46 to 45, *educational level* from 14 to 9 and *working hours per week* from 60 to 51. Finally, CE 4 shows that we can change the prediction by decreasing *educational level* from 14 to 12, *occupation* from *executive managerial* to *handlers cleaners*, *marital relationship* from *husband* to *not-in-family* and *working hours per week* from 60 to 56.

These results show that our method generated logical and understandable explanations for tabular data, which can help end users understand which attributes influenced the predictions of DL models. Furthermore, it gives the end users the flexibility to specify the attributes that can or cannot be altered to change predictions.

5.2.3. *Comparison with LIME*

To illustrate the advantages of our DL model-specific method over model-agnostic methods, we compared it with LIME[8] in terms of runtime and quality of the explanations.

Table 8 reports the time required to train the surrogates, to generate the ex-

Table 7: Explanations provided by the proposed method on two records of the Adult data set. Symbol '-' indicates that the value of the feature did not change during the creation of the CE.

	Attribute	age	workclass	edu	M.S.	occupation	relationship	race	gender	C. G.	C. L.	H.P.W.	country	Prediction original model
Record 1	Original	31	Self-emp-not-inc	13	unmarried	Craft-repair	Unmarried	White	Male	0	0	60	U.S.	$\leq 50K$
	CE 1	**-**	-	**16**	-	-	-	-	-	**-**	**-**	**-**	-	$> 50K$
	CE 2	**41**	-	**15**	-	-	-	-	-	-	-	-	-	$> 50K$
	CE 3	**39**	-	**15**	-	-	-	-	-	-	-	**61**	-	$> 50K$
	CE 4	**36**	-	**14**	-	**Exec-managerial**	-	-	-	-	-	-	-	$> 50K$
Record 2	Original	46	Private	14	married	Exec-managerial	husband	White	Male	0	0	60	U.S.	$> 50K$
	CE 1	-	-	**7**	-	**-**	-	-	-	-	-	**-**	-	$\leq 50K$
	CE 2	**45**	-	**7**	-	-	-	-	-	-	-	-	-	$\leq 50K$
	CE 3	**45**	-	**9**	-	-	-	-	-	-	-	**51**	-	$\leq 50K$
	CE 4	**-**	-	**12**	-	**Handlers-cleaners**	**Not-in-family**	-	-	-	-	**56**	-	$\leq 50K$

planation of one prediction, to generate the prediction of the validation set, and to train the models and generate the explanations for both the proposed method and LIME. We can notice that LIME, on average, was ten times slower than our method to generate explanations.

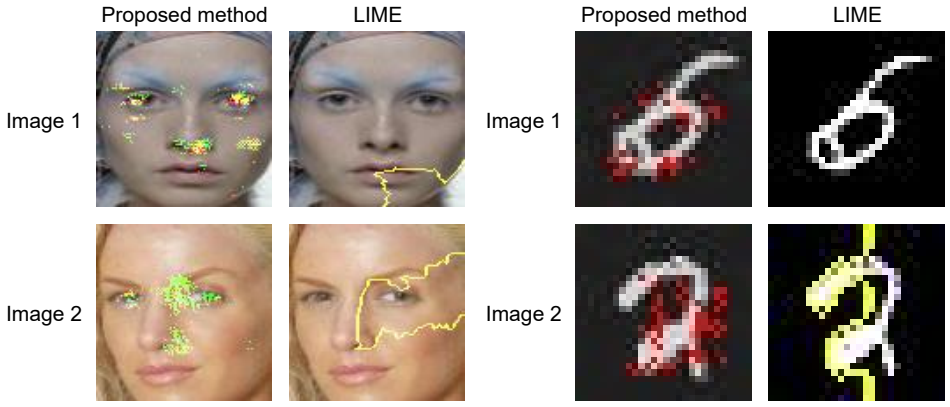
The high computational runtime of LIME was due to the way LIME generates its explanations. In order to explain the prediction of a black-box model f on an image x, LIME first decomposes x into d superpixels (a.k.a. image patches). Then, it creates n neighbor perturbed images $x_1, \ldots, x_n$ by randomly turning on and off those superpixels. After that, it queries the model to get predictions $y_i = f(x_i)$. Finally, it builds a local linear interpretable model fitting the y_is to the presence or absence of superpixels. Since each coefficient of the built model is associated with a superpixel of the image, the more positive the coefficient is, the more important the superpixel is for the prediction. Usually, the end user explains the prediction by highlighting the superpixels associated with the top positive coefficients. In our experiments, LIME created, on average, 104 and 84 perturbed images for each original image in the Gender data set and the MNIST data set, respectively. After that, it trained a unique local model to explain the black-box model prediction on each validation image. In contrast, our method used the same trained surrogate model to explain any example in 2 steps on average, which caused little computational overhead compared to LIME.

To compare the quality of the explanations generated by our method vs LIME, we generated explanations using both methods for two images of the Gender and MNIST data sets. For our method, we used Surrogate 1. Figure 3a shows the explanations generated for the Gender examples. We can see that our method highlighted specific image pixels that caused the model's prediction. Those pixels were clear, limited, and associated with facial features important for classifying faces.[45,46] On the other hand, LIME generated less clear explanations, which involved many non-relevant pixels; sometimes it could not find explanations at all, as shown in

Table 8: Runtimes for training the surrogate models and execute the LIME algorithm, and for generating the explanations on the Gender and MNIST data sets.

Data set	Task	Provider model	Surrogate model 1	Surrogate model 2	Surrogate model 3	Surrogate model 4	LIME
Gender	Training the model	1224 s	1231 s	**1154 s**	1468 s	1471 s	
	Explaining one image		0.24 s	**0.13 s**	0.2 s	0.24 s	2.77 s
	Explaining the val. set		2818 s	**1554 s**	2331 s	2784 s	32132 s
	Total		4049 s	**2708 s**	3799 s	4255 s	32132 s
MNIST	Training the model	328 s	124 s	**124 s**	192 s	177 s	
	Explaining one image		0.03 s	**0.03 s**	0.04 s	0.05 s	0.53 s
	Explaining the val. set		350 s	**330 s**	400 s	500 s	5300 s
	Total		474 s	**454 s**	592 s	677 s	5300 s

Figure 3b.



(a) Two examples from the Gender data set (b) Two examples from the MNIST data set

Fig. 3: Comparison of the explanations generated by the proposed method with those generated by LIME

6. Conclusions and future work

In the context of machine learning as a service (MLaaS), end users do not normally have white-box access to the MLaaS provider's DL models. We have presented a novel approach that enables end users to locally generate DL model-specific explanations that accurately approximate the explanations the user would obtain if she had white-box access to the provider's model.

First, we use a modified version of the Mixup augmentation method to enlarge the small unlabeled data set available to the end user and make it more representative of the input data distribution. Then, we leverage knowledge distillation to

build a local surrogate model at the end user's side that approximates the provider's model. Finally, we use the gradients of the surrogate model to generate adversarial examples that counterfactually explain the prediction of the provider's model on a specific input example. For image data, we visualize the explanation by superimposing on the input example the difference between its pixels and those of the counterfactual. For tabular data, we designed a method that makes small changes on few attributes to generate counterfactuals that are understandable by end users.

Our approach only requires the end user to have access to unlabeled data of size about 0.5% of the provider's training data and it does not require any knowledge about the provider's model architecture or the training hyper-parameters.

Our experiments on image classification and tabular classification data sets showed that our approach can locally generate DL model-specific explanations consistent with those generated by the provider's model, thereby giving end users an independent and reliable way to determine if the provider's predictions and explanations are trustworthy.

As future work, we plan to test the performance of our approach on other computer vision tasks, such as detection and segmentation, as well as natural language processing. We intend to test the use of the plausible counterfactual explanations[39] on the end user's side. We also plan to use adversarial examples as a means to assess the counterfactual fairness (individual fairness) of deep learning models.

Acknowledgments

We acknowledge support from the European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab") and from the Government of Catalonia (ICREA Acadèmia Prizes to J. Domingo-Ferrer and D. Sánchez, and FI grant to N. Jebreel).

References

1. Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", *Nature* **521** (2015) 436–444.
2. S. Dargan, M. Kumar, M. R. Ayyagari and G. Kumar, "A survey of deep learning and its applications: A new paradigm to machine learning", *Archives of Computational Methods in Engineering* (2019) 1–22.
3. N. M. Jebreel, J. Domingo-Ferrer, D. Sánchez and A. Blanco-Justicia, "Keynet: An asymmetric key-style framework for watermarking deep learning models", *Applied Sciences* **11** (2021) 999.
4. M. Ribeiro, K. Grolinger and M. A. Capretz, "MLaaS: Machine learning as a service", in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)* (IEEE, 2015), pp. 896–902.
5. L. Chazette, O. Karras and K. Schneider, "Do end-users want explanations? Analyzing the role of explainability as an emerging aspect of non-functional requirements", in *2019 IEEE 27th International Requirements Engineering Conference (RE)* (IEEE, 2019), pp. 223–233.
6. A. Blanco-Justicia, J. Domingo-Ferrer, S. Martínez and D. Sánchez, "Machine learning explainability via microaggregation and shallow decision trees", *Knowledge-Based Systems* **194** (2020) 105532.

7. GDPR, "General Data Protection Regulation, Regulation EU 2016/679 of the European Parliament and of the Council of 27 April 2016", *Official Journal of the European Union* .
8. M. T. Ribeiro, S. Singh and C. Guestrin, ""Why should I trust you?" explaining the predictions of any classifier", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 1135–1144.
9. S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions", in *Proceedings of the 31st International Conference on Neural Information Processing Systems* (2017), pp. 4768–4777.
10. S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation", *PloS One* **10** (2015) e0130140.
11. C. Molnar, *Interpretable Machine Learning* (Lulu.com, 2020).
12. T. Vermeire and D. Martens, "Explainable image classification with evidence counterfactual", arXiv preprint arXiv:2004.07511, 2020.
13. P. Y. Simard, Y. A. LeCun, J. S. Denker and B. Victorri, "Transformation invariance in pattern recognitiontangent distance and tangent propagation", in *Neural Networks: Tricks of the Trade* (Springer, 1998), pp. 239–274.
14. G. Hinton, O. Vinyals and J. Dean, "Distilling the knowledge in a neural network", arXiv preprint arXiv:1503.02531, 2015.
15. A. Nguyen, J. Yosinski and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 427–436.
16. M. T. Ribeiro, S. Singh and C. Guestrin, "Anchors: High-precision model-agnostic explanations", in *Proceedings of the 32th AAAI Conference on Artificial intelligence* (2018), volume 1.
17. D. Alvarez-Melis and T. S. Jaakkola, "On the robustness of interpretability methods", arXiv preprint arXiv:1806.08049, 2018.
18. K. Simonyan, A. Vedaldi and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps", arXiv preprint arXiv:1312.6034, 2013.
19. A. Chattopadhay, A. Sarkar, P. Howlader and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks", in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (IEEE, 2018), pp. 839–847.
20. R. Haffar, N. M. Jebreel, J. Domingo-Ferrer and D. Sánchez, "Explaining image misclassification in deep learning via adversarial examples", in *International Conference on Modeling Decisions for Artificial Intelligence* (Springer, 2021), pp. 323–334.
21. M. T. Lash, Q. Lin, N. Street, J. G. Robinson and J. Ohlmann, "Generalized inverse classification", in *Proceedings of the 2017 SIAM International Conference on Data Mining* (SIAM, 2017), pp. 162–170.
22. T. Laugel, M.-J. Lesot, C. Marsala, X. Renard and M. Detyniecki, "Comparison-based inverse classification for interpretability in machine learning", in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (Springer, 2018), pp. 100–111.
23. S. Verma, J. Dickerson and K. Hines, "Counterfactual explanations for machine learning: A review", *arXiv preprint arXiv:2010.10596* .
24. D. Mahajan, C. Tan and A. Sharma, "Preserving causal constraints in counterfactual explanations for machine learning classifiers", *arXiv preprint arXiv:1912.03277* .
25. H. Zhang, M. Cisse, Y. N. Dauphin and D. Lopez-Paz, "mixup: Beyond empirical risk

minimization", in *International Conference on Learning Representations* (2018).
26. G. Chen, W. Choi, X. Yu, T. Han and M. Chandraker, "Learning efficient object detection models with knowledge distillation", in *31st Conf. on Neural Information Processing Systems–NIPS 2017* (2017).
27. X. Wang, T. Fu, S. Liao, S. Wang, Z. Lei and T. Mei, "Exclusivity-consistency regularized knowledge distillation for face recognition", in *European Conference on Computer Vision* (Springer, 2020), pp. 325–342.
28. J. Gou, B. Yu, S. J. Maybank and D. Tao, "Knowledge distillation: A survey", *International Journal of Computer Vision* **129** (2021) 1789–1819.
29. Z. Shen, Z. Liu, D. Xu, Z. Chen, K.-T. Cheng and M. Savvides, "Is label smoothing truly incompatible with knowledge distillation: An empirical study", *arXiv preprint arXiv:2104.00676* .
30. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow and R. Fergus, "Intriguing properties of neural networks", arXiv preprint arXiv:1312.6199, 2013.
31. I. J. Goodfellow, J. Shlens and C. Szegedy, "Explaining and harnessing adversarial examples", arXiv preprint arXiv:1412.6572, 2014.
32. E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan and Q. V. Le, "Autoaugment: Learning augmentation policies from data", arXiv preprint arXiv:1805.09501, 2018.
33. A. Srivastava, T. Shinde, R. Joshi, S. A. Ansari and N. Giri, "Auto-DL: A platform to generate deep learning models", in *International Conference on Soft Computing in Data Science* (Springer, 2021), pp. 89–103.
34. J. Snoek, H. Larochelle and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms", in *25th Conference on Neural Information Processing Systems–NIPS 2012* (2012).
35. S. Wachter, B. Mittelstadt and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the GDPR", *Harvard Journal of Law & Technology* **31** (2017) 841.
36. R. K. Mothilal, A. Sharma and C. Tan, "Explaining machine learning classifiers through diverse counterfactual explanations", in *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (2020), pp. 607–617.
37. J. V. Jeyakumar, J. Noor, Y.-H. Cheng, L. Garcia and M. Srivastava, "How can I explain this to you? An empirical study of deep neural network explanation methods", in *34th Conference in Neural Information Processing Systems–NeurIPS 2020* (2020).
38. D. Slack, A. Hilgard, H. Lakkaraju and S. Singh, "Counterfactual explanations can be manipulated", *Advances in Neural Information Processing Systems* **34**.
39. A. Artelt, V. Vaquet, R. Velioglu, F. Hinder, J. Brinkrolf, M. Schilling and B. Hammer, "Evaluating robustness of counterfactual explanations", in *2021 IEEE Symposium Series on Computational Intelligence (SSCI)* (IEEE, 2021), pp. 01–09.
40. M. T. Ribeiro, S. Singh and C. Guestrin, ""Why should I trust you?" Explaining the predictions of any classifier", in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 1135–1144.
41. R. Haffar, J. Domingo-Ferrer and D. Sánchez, "Explaining misclassification and attacks in deep learning via random forests", in *International Conference on Modeling Decisions for Artificial Intelligence* (Springer, 2020), pp. 273–285.
42. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", arXiv preprint arXiv:1412.6980, 2014.
43. Y. LeCun, P. Haffner, L. Bottou and Y. Bengio, "Object recognition with gradient-based learning", in *Shape, contour and grouping in computer vision* (Springer, 1999), pp. 319–345.
44. Y. LeCun et al., "LeNet-5, convolutional neural networks", `http://yann.lecun.com/`

`exdb/lenet`, accessed May 13, 2022.

45. E. Brown and D. I. Perrett, "What gives a face its gender?", *Perception* **22** (1993) 829–840.
46. J.-M. Fellous, "Gender discrimination and prediction on the basis of facial metric information", *Vision Research* **37** (1997) 1961–1973.
47. C.-L. Liu, K. Nakashima, H. Sako and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques", *Pattern Recognition* **36** (2003) 2271–2285.