UNIVERSITAT
ROVIRA i VIRGILI

Contributions to Explainability and Attack Detection in Deep Learning

Author:
Rami HAFFAR



DOCTORAL THESIS
2023

Rami HAFFAR

Contributions to Explainability and Attack Detection in Deep Learning

DOCTORAL THESIS

Supervisors:
Josep DOMINGO-FERRER
David SÁNCHEZ

Department of Computer Engineering and Mathematics



UNIVERSITAT ROVIRA i VIRGILI

November, 2023

UNIVERSITAT ROVIRA i VIRGILI

FEM CONSTAR que aquest treball, titulat “Contributions to Explainability and Attack Detection in Deep Learning”, que presenta Rami Haffar per a l’obtenció del títol de Doctor, ha estat realitzat sota la meva direcció al Departament d’Enginyeria Informàtica i Matemàtiques d’aquesta universitat.

HACEMOS CONSTAR que el presente trabajo, titulado “Contributions to Explainability and Attack Detection in Deep Learning”, que presenta Rami Haffar para la obtención del título de Doctor, ha sido realizado bajo mi dirección en el Departamento de Ingeniería Informática y Matemáticas de esta universidad.

We STATE that the present study, entitled “Contributions to Explainability and Attack Detection in Deep Learning”, presented by Rami Haffar for the award of the degree of Doctor, has been carried out under my supervision at the Department of Computer Engineering and Mathematics of this university.

Tarragona 19/07/2023

Els directors de la tesi doctoral
Los directores de la tesis doctoral
Doctoral Thesis Supervisors

David Sánchez Ruenes - DNI 47763566H (TCAT)
Firmado digitalmente por David Sánchez Ruenes - DNI 47763566H (TCAT)
Fecha: 2023.07.19 10:24:41 +02'00'

David Sánchez Ruenes

DOMINGO FERRER JOSEP - 33890313C
Digitally signed by DOMINGO FERRER JOSEP - 33890313C
Date: 2023.07.19 10:50:43 +02'00'

Josep Domingo Ferrer

Abstract

Artificial intelligence (AI) is used for various purposes that are critical to human life. However, most state-of-the-art AI algorithms, and in particular deep-learning (DL) models, are black-box, meaning humans cannot understand how such models make decisions. To forestall an algorithm-based authoritarian society, decisions based on machine learning ought to inspire trust by being *explainable*. For AI explainability to be practical, it must be feasible to obtain explanations systematically and automatically. There are two main methodologies to generate explanations. 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, the users of the black-box model lack white-box access to the internal components of the providers' models. Nevertheless, the only way for users to trust predictions and for these to align with ethical regulations is for predictions to be accompanied by explanations locally and independently generated by the users (rather than by explanations offered by the model providers). Furthermore, those models can be vulnerable to various security and privacy attacks targeting their training. In this thesis, we leverage both model-specific and model-agnostic explainability techniques. First, we propose a model-agnostic explainability method using random decision

forests as surrogates. The surrogate model can explain the predictions of the black-box models in both centralized and decentralized settings. In addition, it uses those explanations to protect the models from attacks that might target them. We also propose a model-specific explainability method that uses the gradients of the model to generate adversarial examples that counterfactually explain why an input example is classified into a specific class. We also generalize this method so that external users can use it by training a local surrogate model that mimics the black-box model's behavior and using the surrogate gradients to generate the adversarial examples. Extensive experimental results show that our methods outperform the state-of-the-art techniques by providing more representative explanations and model protections while requiring a low computational cost.

Resum

La intel·ligència artificial (IA) es fa servir per a diversos propòsits que són crítics per a la vida humana. Tanmateix, la majoria dels algorismes d'IA d'última generació, i en particular els models d'aprenentatge profund (AP), són models de caixa negra, en el sentit que els humans no podem entendre com prenen les decisions. Per evitar una societat autoritària basada en algorismes, les decisions basades en l'aprenentatge automàtic haurien de ser *explicables* per tal d'inspirar confiança. Perquè l'explicabilitat de la IA sigui pràctica, ha de ser factible obtenir explicacions de manera sistemàtica i automàtica. Hi ha dues metodologies principals per generar explicacions. Els mètodes d'explicació que utilitzen components interns dels models d'AP (també coneguts com a explicacions específiques del model) són més precisos i efectius que els que es basen únicament en les entrades i en les sortides (també coneguts com a explicacions independents del model). Malauradament, els usuaris dels models de caixa negra no tenen accés de caixa blanca als components interns dels models dels proveïdors. Tot i així, l'única manera perquè els usuaris confiïn en les prediccions i perquè aquestes s'alineïn amb les normatives ètiques és que les prediccions s'acompanyin d'explicacions generades localment i independentment pels usuaris (en lloc d'explicacions fornides pels proveïdors de models). A més, aquests models poden ser vulnerables a diversos atacs a la seguretat i a la privadesa dirigits contra llur entrenament. En aquesta tesi, explotarem les tècniques d'explicabilitat tant específiques del model com agnòstiques del model. Primer, proposem un mètode d'explicació agnòstic del model basat en l'ús de boscos de decisió aleatoris com

a model substitut. El model substitut pot explicar les prediccions dels models de caixa negra en entorns centralitzats i descentralitzats. A més, utilitza aquestes explicacions per protegir els models dels atacs que se'ls puguin dirigir. D'altra banda, proposem un explicador específic del model que fa servir els gradients del model per generar exemples contradictoris que expliquen de manera contrafactual per què un exemple d'entrada es classifica en una classe específica. També generalitzem aquest mètode perquè qualsevol usuari del model pugui utilitzar-lo entrenant un model substitut local que imiti el comportament del model de caixa negra i fent servir els gradients substituts per generar els exemples adversaris. Els nostres resultats experimentals detallats mostren que els nostres mètodes superen les tècniques d'última generació i forneixen explicacions més representatives i més protecció del model alhora que requereixen un cost computacional més baix.

Resumen

La inteligencia artificial (IA) se utiliza para varios propósitos que son críticos para la vida humana. Sin embargo, la mayoría de los algoritmos de IA de última generación, y en particular los modelos de aprendizaje profundo (AP), son modelos de caja negra, en el sentido de que los humanos no podemos entender cómo esos modelos toman las decisiones. Para evitar una sociedad autoritaria basada en algoritmos, las decisiones basadas en el aprendizaje automático han de ser *explicables* para inspirar confianza. Para que la explicabilidad de la IA sea práctica, debe ser factible obtener explicaciones de forma sistemática y automática. Hay dos metodologías principales para generar explicaciones. Los métodos de explicación que utilizan componentes internos de los modelos de AP (también conocidos como explicaciones específicas del modelo) son más precisos y efectivos que aquellos que se basan únicamente en las entradas y salidas (también conocidas como explicaciones independientes del modelo). Desafortunadamente, los usuarios de los modelos de caja negra carecen de acceso de caja blanca a los componentes internos de los modelos de los proveedores. Sin embargo, la única forma de que esos usuarios confíen en las predicciones y estas se alineen con las normas éticas es que las predicciones vayan acompañadas de explicaciones generadas local e independientemente por los usuarios (en lugar de explicaciones ofrecidas por los proveedores del modelo). Además, estos modelos pueden ser vulnerables a varios ataques de seguridad y privacidad dirigidos contra su entrenamiento. En esta tesis, explotaremos las técnicas de explicabilidad específicas de modelo y

agnósticas de modelo. Primero, proponemos un método de explicación independiente del modelo basado en el uso de bosques aleatorios de decisión como modelo sustituto. El modelo sustituto puede explicar las predicciones de los modelos de caja negra en entornos centralizados y descentralizados. Además, utiliza esas explicaciones para proteger a los modelos de los ataques que podrían tenerlos como objetivo. Asimismo, proponemos un explicador específico del modelo que usa los gradientes del modelo para generar ejemplos contradictorios que explican de manera contrafactual por qué un ejemplo de entrada se clasifica en una clase específica. También generalizamos este método para que cualquier usuario del modelo pueda usarlo entrenando un modelo sustituto local que imita el comportamiento del modelo de caja negra y usando los gradientes sustitutos para generar los ejemplos contradictorios. Nuestros extensos resultados experimentales muestran que nuestros métodos superan las técnicas de vanguardia al proporcionar explicaciones más representativas y mayor protección del modelo, al mismo tiempo que requieren un menor costo computacional.

Acknowledgements

This dissertation would not have been possible without the support of many people.

Foremost, I would like to express my sincere gratitude to my supervisors Prof. Josep Domingo-Ferrer and Prof. David Sánchez for their guidance during the development of this thesis, their patience, their motivation, their enthusiasm, and their immense knowledge.

I am indebted to all CRISES group members as well, especially to Najeeb Jebreel for his helpful assistance, support, and encouragement. Thanks also to my wife, my mother, brother and sisters for their unconditional support and love. And to all my friends for always being there.

Finally, I am grateful to Prof. Anna Monreale and the rest of KDD research group, for their hospitality during my stay in Pisa (Italy) in 2022.

This work was partially funded by the European Commission (projects H2020-871042 "SoBigData++" and H2020-101006879 "MobiDataLab"), the Government of Catalonia (ICREA Acadèmia Prizes to J. Domingo-Ferrer and D. Sánchez), and the Spanish Government (project RTI2018-095094-B-C21 "Consent"), Grant PRE2019-089210 funded by MCIN/AEI/ 10.13039/501100011033 and "ESF Investing in your future".

Contents

Abstract **v**

Resum **vii**

Resumen **ix**

Acknowledgements **xi**

1 Introduction **1**
1.1 Objectives 5
1.2 Thesis structure 6

2 Background and State of the Art **9**
2.1 Machine learning 9
2.1.1 Classical machine learning methods 10
Support-vector machines 10
Decision trees 11
Random decision forests 11
2.1.2 Neural networks 11
Deep learning neural networks 12
2.2 Explainability methods for black-box models 15
2.2.1 Requirements of the surrogate models used for explaining the black box models 17

2.3 Federated learning 19
2.3.1 Attacks in federated learning 20
2.3.2 Countermeasures against attacks 22
2.4 Conclusion 25

3 Explaining Centralized DL via Random Forests 27
3.1 Contributions and plan of this chapter 28
3.2 Random forest-based surrogate model 29
3.3 Experimental results 35
3.3.1 Experiments on synthetic numerical data . . 35
3.3.2 Experiments on real numerical data 38
3.3.3 Experiments on real categorical data 40
3.4 Conclusions 42

4 Explaining FL via Random Forests 45
4.1 Contributions and plan of this chapter 46
4.2 Constructing surrogate models 47
4.3 Experimental results 52
4.3.1 Data sets 52
4.3.2 Results on the explainability of the predictions of the black box model 53
4.3.3 Results on the attack detection performance 63
4.4 Conclusions 70

5 Explaining DL via Adversarial Examples 73
5.1 Contributions and plan of this chapter 74
5.2 Our proposals 75
5.2.1 Adversarial examples 75
5.2.2 Explaining model predictions on the developer's side 76

5.2.3 Explaining model predictions on the user's side . . . 78
5.3 Experimental results . . . 81
5.3.1 Explanations for the developer . . . 81
5.3.2 Explanations for the user . . . 82
5.4 Conclusions . . . 86

6 Generating DL Model-Specific Explanations 87
6.1 Contributions and plan of this chapter . . . 88
6.2 Methods and assumptions . . . 90
6.3 Explaining DL model predictions on the user's side 92
6.3.1 Data augmentation and surrogate model training . . . 93
6.3.2 Generating counterfactual adversarial explanations . . . 97
6.4 Empirical analysis . . . 103
6.4.1 Experimental setup . . . 103
6.4.2 Results and discussion . . . 108
Accuracy of surrogate models . . . 108
Surrogate model explanations . . . 110
Comparison with LIME . . . 115
6.5 Conclusions . . . 117

7 Conclusion and Future Work 119
7.1 Contributions and publications . . . 120
7.2 Future work . . . 122

Bibliography 125

List of Figures

2.1 An example of input, hidden, and output layers of a basic neural network 12
2.2 An example of a federated learning model architecture 20

4.1 Pearson correlations between the feature importances obtained by the surrogate for each pair of black-box models on the Activity and Adult data sets 61
4.2 Structure of a tree from a random forest with depth limit 6 built on a synthetic data set 62
4.3 Standalone decision tree covering the whole set of features of the same synthetic data set used in Figure 4.2 62
4.4 FNR and FPR of attack detection with the proposed method and with the method based on update statistics 69

5.1 Architecture of the CNN used as original model in the experiments 82
5.2 Four examples of explanations generated on the developer's side 83

5.3 Four examples of explanations generated on the user's side 85

6.1 Architecture of the MLaaS 89
6.2 Virtual input image generated by mixing two original input images with $\lambda = 0.5$ 94
6.3 Visual explanations generated by the surrogate models in comparison with those generated by the provider's models 112
6.4 Comparison of the explanations generated by the proposed method with those generated by LIME . . 117

List of Tables

3.1 Importance of the features of the synthetic data set on wrong predictions by the black box model 36
3.2 Attacking the three most important features in the synthetic data set 37
3.3 Attacking the three least important features in the synthetic data set 38
3.4 Attacking the three most important features in the physical activity data set 40
3.5 Attacking the three least important features in the physical activity data set 40
3.6 Importance of the features of the Adult data set on wrong predictions by the black box model 41
3.7 Attacking the three most important features in the Adult data set 42
3.8 Attacking the three least important features in the Adult data set 42

4.1 Accuracy of the black-box models and the surrogate random forest for the Activity and Adult data sets. The fidelity of the surrogate model to each black box model is also displayed (a smaller score indicates more fidelity) 56

4.2 Ten highest feature importances for wrong *black box1* predictions computed by a peer using Algorithm 3 on the two data sets, and corresponding importances computed by the LIME algorithm 57
4.3 Accuracy of the *black box1* model after removing the features with the highest importance on the wrong predictions, as identified by the proposed method and LIME, in comparison with the accuracy after removing random subsets of features. Subset1 and Subset2 are example random subsets. Subset1 consists of *(magnetometer_z_chest, magnetometer_z_ankle)* for the Activity data set, and *(native-country, workclass)* for the Adult data set. Subset2 consists of *(acceleration_16_z_ankle, acceleration_6_y_chest, acceleration_16_x_hand)* for the Activity data set, and *(relationship, occupation, marital-status)* for the Adult data set. 59
4.4 Differences between average feature importances and importances of attacked features for different attacks on the Adult data set. Values in boldface correspond to the epoch in which the attack was detected. 65
4.5 Attack detection rate of the proposed method and the update statistics method on the two data sets. The detection rate of the best-performing method for a certain configuration of data set, attack type and number of peers is depicted in boldface. 67

4.6 Differences between average feature importances and importances of attacked features for the model manager's mGAN-AI privacy attack on the Adult data set. Values in boldface correspond to the epoch in which the attack was detected. 71

6.1 Number of data samples owned by the service provider and the user before and after the Mixup augmentation 105
6.2 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 0, and a padding 1; $MP(2, 2)$ denotes a max-pooling layer with a kernel of size 2×2 and a stride 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. 105
6.3 Architectures of the provider's black-box and end-user's surrogate models used in the experiments for the MNIST data set 106
6.4 Architectures of the provider's black-box and end-user's surrogate models used in the experiments for the Adult data set 107
6.5 Accuracy of surrogate models, without and with data augmentation, compared to the accuracy of the provider's model 109

6.6 Similarity between the adversarial examples generated by the surrogate models and those generated by the provider's model on the Gender and MNIST data sets 110
6.7 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. 115
6.8 Runtimes for training the surrogate models and execute the LIME algorithm, and for generating the explanations on the Gender and MNIST data sets . 116

List of Algorithms

1 Determine the importance of features in wrong predictions 31
2 Discover the feature under attack 34
3 Computation by each peer of the importance of features in the black box model's wrong predictions . . 48
4 Detection by each peer of attacks on the federated black-box model 50
5 Explaining the model predictions on the developer's side 78
6 Drawing the saliency maps 79
7 Explaining the model predictions on the user's side 80
8 Augmenting unlabeled data and labeling the augmented data 95
9 Explaining predictions for image data 101
10 Explaining predictions for tabular data 102

List of Abbreviations

AI	**A**rtificial **I**ntelligence
DL	**D**eep **L**earning
FL	**F**ederated **L**earning
SVM	**S**upport-**V**ector **M**achines
GDPR	**G**eneral **D**ata **P**rotection **R**egulation
NN	**N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
ANN	**A**rtificial **N**eural **N**etwork
DNN	**D**eep **N**eural **N**etwork
lr	**l**earning **r**ate
LIME	**L**ocal **I**nterpretable **M**odel-Agnostic **E**xplanations
CV	**C**omputer **V**ision
NLP	**N**atural **L**anguage **P**rocessing
SHAP	**SH**apley **A**dditive ex**P**lanations
LRP	**L**ayer-wise **R**elevance ex**P**ropagation
MLaaS	**M**achine **L**earning as a **S**ervice
API	**A**pplication **P**rogramming **I**nterface
CE	**C**ounterfactual **E**xamples
iid	**i**dentically **i**ndependently **d**istributed
GAN	**G**enerative **A**dversarial **N**etwork
KD	**K**nowledge **D**istillation

I would like to dedicate this thesis to
my Family,
for their endless love, support and
encouragement.
And to my wife Rogina.
You made my life so much better in so many ways that it is hard to imagine doing this without you.

Chapter 1

Introduction

The past two decades have witnessed major advances in artificial intelligence (AI) systems. Deep learning (DL) models and convolutional neural networks (CNNs) are the cornerstones of many modern machine learning (ML) systems due to their ability to solve many complex tasks such as computer vision (CV), natural language processing (NLP), and speech recognition (Deng and Yu, 2014; LeCun, Bengio, and Hinton, 2015). However, those models are opaque decision systems because humans cannot understand the reasoning behind their predictions. This is why they are called *black-box models* (Ljung, 2001).

On the other hand, companies increasingly release market services and products by embedding data mining and machine learning components, often in safety-critical industries such as self-driving cars, robotic assistants, or personalized medicine (Guidotti et al., 2019). However, by blindly relying on black-box machine learning models, we risk leaving daily decisions that affect companies (and, ultimately, citizens' lives) to systems that we do not understand. This impacts not only on ethics, but also on accountability (Joshua et al., 2017), safety (Danks and London, 2017), and

industrial liability (Kingston, 2016).

On the other hand, ML models are often trained on data compiled by human analysts, who cluster these data to teach the models how to make decisions. As a result, they may containing human biases and prejudices (Guidotti et al., 2019), which may lead the model to wrong and unfair predictions. Another inherent risk of these ML models is the possibility of unintentionally making wrong predictions. Those may have been learned from spurious correlations in the training data –such as recognizing an object in a picture by the properties of the background or lighting–, or due to a systematic bias in training data collection –such as in the well-known example of Ribeiro, Singh, and Guestrin, 2016, where the animal will be classified as a wolf if the image has snow background and a husky dog if the image has grass background, because in the training pictures all the wolves were displayed in a snowy landscape and the huskies were not–.

The lack of transparency of ML is problematic both for the individuals affected by the predictions of the models and for the developers who train the black-box models:

- Individuals are affected by a growing number of automated decisions: credit rating, loan granting, insurance premiums, medical diagnoses, job selection, etc. While transparency measures are being implemented by public administrations worldwide, there is a danger that automated decisions will become a ubiquitous black box. To protect citizens, explainability requirements are beginning to show up in legal regulations and ethics guidelines, such as article 22 of the European Union's General Data Protection Regulation (GDPR)

(Regulation, General Data Protection, 2016), which states the right of citizens to an explanation of automated decisions on them, and the European Commission's Ethics Guidelines for Trustworthy AI (European Commission's High-Level Expert Group on Artificial Intelligence, 2019) –which insists that the organizations making automated decisions be prepared to explain them at the request of the affected citizens–, and the IEEE report on ethically aligned design for intelligent systems (Shahriari and Shahriari, 2017). Furthermore, the recent EU proposal for a regulation on artificial intelligence (nicknamed Artificial Intelligence Act (*Proposal for a Regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts* 2021)) also emphasizes the explainability requirement. On the other hand, obtaining explanations alongside predictions helps the users understand why an ML model produces a specific prediction, which increases the trust in the model and contributes to a more transparent decision-making (Chazette, Karras, and Schneider, 2019; Blanco-Justicia et al., 2020).

- Developers would like to know how the black-box models make predictions to ensure the algorithm considers the relevant features during the training phase and that the training data lack bias. This allows them to develop more accurate, fair and robust models.

For explanations on black-box models to be practical and scalable, their generation must be automated. 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 (Ribeiro, Singh, and Guestrin, 2016), SHAP (Lundberg and Lee, 2017) and LRP (Bach et al., 2015) can explain any model, they only look at models "from the outside", that is, 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 (Molnar, 2020; Vermeire et al., 2022).

On the other hand, training an ML model requires a large amount of data, which may be difficult to compile due to several reasons, among which privacy concerns stand out. To mitigate this issue, two solutions have been proposed. First, the federated learning (FL) framework (Konečný et al., 2016) –which is described in more detail in Chapter 2–, is a decentralized machine learning technique that aggregates local models trained by a set of participants on their private data to obtain a global model without sharing the data with other participants. Second, big companies that own sufficient data to train an accurate ML model may provide paid API access to those models to other companies or end users via Machine Learning as a Service (MLaaS) platforms (Ribeiro, Grolinger, and Capretz, 2015). Users can then query those models with their own data to obtain accurate predictions.

However, like in centralized deep learning, FL and MLaaS produce unexplainable (black-box) models. In addition, due to their distributed architectures, they are vulnerable to security and

privacy attacks (described in Chapter 2). To protect models against those attacks, various methods have been proposed in the literature, such as Krum aggregation (Blanchard et al., 2017) and the coordinate-wise median (Yin et al., 2018). Unfortunately, most attack detection methods focus on improving the model's performance, and do not provide information in the event of an attack, such as the identity of the attacker or the target of the attack.

1.1 Objectives

In this thesis, we aim to develop methods to automatically generate explanations for black box models' predictions, and to detect attacks that may target those models. As such, we introduce the following set of goals:

1. Developing a model-agnostic method based on random decision forests as a surrogates to explain the predictions of the black box models. This method can be applied both to centralized and decentralized settings without hampering the model's performance. The surrogate model provides the user with two types of explanations: i) small decision trees that are interpretable by humans, and ii) the features' importances according to their effect on the prediction made by the black-box model.

2. Leveraging the explanations provided by the surrogate model to detect the attack targeting the training data, and to detect security and privacy attacks that may target the model.

3. Developing DL model-specific explainability methods based on counterfactual examples (CE), which can be used with different input data types such as tabular and images.

4. Expanding the use of the method based on counterfactual examples so that any party with API access to the black-box model can execute it on their end, using only their local unlabeled data.

1.2 Thesis structure

- Chapter 2 gives background on centralized and decentralized machine learning, and reviews works on explainability for black-box models. It also lists the most outstanding security and privacy attacks that affect decentralized ML architectures and the main countermeasures proposed in the literature to tackle them.

- Chapter 3 describes our proposed model-agnostic method to explain black-box model predictions and to detect attacks on the training data in centralized settings.

- Chapter 4 describes the application of the former method to a decentralized setting.

- Chapter 5 introduces the model-specific approach based on adversarial examples, and applies it to black-box image classification.

- Chapter 6 discusses the application of the former method to a scenario in which users access ML models via API access (i.e., MLaaS). This method counterfactually explains the black-box model predictions regardless of the input data type.
- Chapter 7 summarizes the main contributions of this thesis and presents some lines of future research.

Chapter 2

Background and state of the art

This chapter provides background on classical machine learning and deep learning in Section 2.1. In Section 2.2, we discuss previous works attempting to interpret black-box models. In Section 2.3, we describe the federated learning architecture, list the possible attacks that can be orchestrated against it, and the current countermeasures that aim to protect the model.

2.1 Machine learning

Machine learning (ML) is a field of AI that mimics the experiential "learning" associated with human intelligence while also having the capacity to learn and improve its analyses through the use of computational algorithms (Bini, 2018; Naylor, 2018). These algorithms use large sets of data inputs and outputs to recognize

patterns in order to train the machine to make autonomous recommendations or decisions. After sufficient repetitions and modifications of the algorithm, the machine can take an input and predict an output (Bini, 2018; Naylor, 2018). Outputs are then compared with a set of known outcomes in order to judge the accuracy of the algorithm, which is then iteratively adjusted to perfect the ability to predict further outcomes (Haeberle et al., 2019; Helm et al., 2020).

2.1.1 Classical machine learning methods

In classical machine learning algorithms, models are trained on handcrafted features, which are extracted by the developer from the data. Also, the models are either linear or graph-based. This makes them self-explainable, since the features are known, and humans can analyze linear and graph-based models. This section summarizes the best-known methods of this type.

Support-vector machines

Support vector machines (SVMs) (Gunn et al., 1998) are linear classifiers based on the margin maximization principle. They perform structural risk minimization, which improves the complexity of the classifier with the aim of achieving excellent generalization performance. SVMs accomplish the classification task by constructing the hyperplane in a higher dimensional space that optimally divides the data into two categories (Adankon and Cheriet, 2009).

Decision trees

Decision trees (Quinlan, 1986) are decision support models that classify patterns using a sequence of well-defined rules. They are tree-like graphs in which each branch node represents an option between a number of alternatives, and each leaf node represents an outcome of the cumulative choices (Tong and Ranganathan, 2013).

Random decision forests

A random forest (Ho, 1995) is a robust machine learning algorithm suitable for various tasks, including regression and classification. It is an ensemble method, in that a random forest comprises many small decision trees, called estimators or trees, each one producing its own predictions. Then, the random forest model combines the predictions of the estimators to produce a more accurate global prediction.

2.1.2 Neural networks

Neural networks (NNs), also known as artificial neural networks (ANNs), are at the heart of deep learning algorithms. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another.

Artificial neural networks (ANNs) contain nodes (artificial neurons) structured in layers: an input layer, one or more hidden layers, and an output layer. Each node connects to another and has an associated weight and threshold, as shown in Figure 2.1. If

the output of any individual node is above the specified threshold value, that node is activated, sending data to the next layer of the network. Otherwise, no data is passed along to the next layer of the network (Chen et al., 2017b).

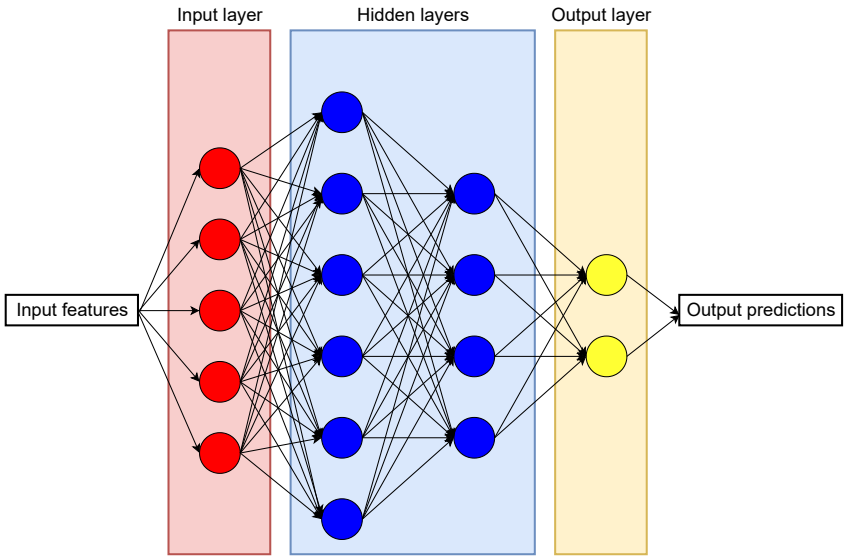


FIGURE 2.1: An example of input, hidden, and output layers of a basic neural network

Deep learning neural networks

Deep learning neural networks (DNNs) are neural networks with multiple hidden layers and multiple nodes in each hidden layer, which allow them to determine significantly higher complex patterns than traditional machine-learning methods. DNNs have been a crucial asset to foster Artificial Intelligence (AI) in the recent few

years, which have achieved remarkable or yet superior to human-level performance on image classification He et al., 2016, speech identification Xiong et al., 2018, and reading knowledge Devlin et al., 2018. Deep learning models consist of different input and hidden layers:

- *Convolutional layers:* These layers are the core of convolutional neural networks (CNN), which are most commonly applied to analyze visual imagery and NLP (Valueva et al., 2020). A convolutional layer contains a set of filters whose parameters need to be learned. The height and weight of the filters are smaller than those of the input volume. Each filter is convolved with the input volume to compute an activation map made of neurons. In other words, the filter is slid across the width and height of the input, and the dot products between the input and filter are computed at every spatial position. There are different activation functions such as ReLU (Agarap, 2018), Sigmoid (Pratiwi et al., 2020), and Tanh (Karlik and Olgac, 2011). The output volume of the convolutional layer is obtained by stacking the activation maps of all filters along the depth dimension. Since the width and height of each filter are designed to be smaller than the input, each neuron in the activation map is only connected to a small local region of the input volume. That is, the receptive field size of each neuron is small and equal to the filter size. The local connectivity of the convolutional layer allows the network to learn filters that maximally respond to a local region of the input, thus exploiting the local spatial correlation of the input (*e.g.*, for an input image, a

pixel is more correlated to the nearby pixels than to the distant pixels). In addition, as the activation map is obtained by convolving the filter and the input, the filter parameters are shared for all local positions. The weight sharing reduces the number of parameters, which results in more efficient learning and better generalization (Ke et al., 2018).

- *Pooling layer:* A pooling layer is usually incorporated between two successive convolutional layers. The pooling layer reduces the number of parameters and computation by downsampling the representation. The pooling function can be max or average (Ke et al., 2018).

- *Fully connected layers:* In those layers, all the inputs from one layer are connected to every activation unit of the next layer. In most popular machine learning models, the last few layers are fully connected layers that compile the data extracted by previous layers to form the final output.

DL models are trained until they achieve the desired accuracy, or for a fixed number of *epochs* specified by the developer. Each time the entire training data set is passed forward and backward through the neural network is called an *epoch*. Usually, the data set is too big to be fed to the machine at once, so it gets divided into several smaller batches. The total number of training examples present in a single batch is called *batch size*, and it is defined based on the machine's capacity. The number of batches needed to complete one epoch is called an *iteration*. After finishing each iteration, the model executes a back-propagation algorithm to adjust the weights of each neuron in the network according to the loss

calculated during the iteration and the *learning rate* (*lr*) selected by the developer.

In general, DL models consist of thousands or even millions of parameters, which means they are very complex. At the same time, while they can approximate any function, studying their structure does not give any insights into the structure of the function being approximated. Therefore, *they are considered black boxes because the model's behavior cannot be understood, even when one is able to see its internal structure and weights.*

2.2 Explainability methods for black-box models

Several methods in the literature have been proposed to explain decisions made by deep learning models. 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. Furthermore, they approximate the relationship between the input and the output prediction of the black-box model. A standard methodology for generating model-agnostic explanations is building a surrogate model based on simpler and more understandable machine learning algorithms. This surrogate is trained to mimic the black-box model behavior. Afterwards, the causes of the surrogate model predictions are used as explanations of the black-box model predictions. For example, LIME (Ribeiro, Singh, and Guestrin, 2016) creates an interpretable linear surrogate model

to approximate the relationship between a prediction and the perturbed examples of the input example. Anchors (Ribeiro, Singh, and Guestrin, 2018) approximates the relationship between the model's prediction and the input example using simple if-then rules. SHAP (Lundberg and Lee, 2017) 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 (Alvarez-Melis and Jaakkola, 2018), computationally expensive or prone to misinterpretation (Molnar, 2020). Moreover, they disregard the internal components of DL models that may be useful for generating more accurate explanations.

- *DL model-specific*. This class of methods assumes white-box access to the DL model pipeline. They 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 (Simonyan, Vedaldi, and Zisserman, 2014) highlight the pixels of an input image by visualizing the gradient of the model prediction w.r.t. those pixels. Grad-CAM++ (Chattopadhyay et al., 2018) 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 is to generate counterfactual examples (CEs) to explain the predictions of a DL model (Molnar, 2020). Counterfactual explanations

are understandable and human-friendly: if you make targeted changes in the values of specific features, the prediction will change from one class to another. Several black-box methods that do not require access to the internal model components to generate CEs have been proposed (Lash et al., 2017; Laugel et al., 2018; Verma, Dickerson, and Hines, 2020). However, gradient-based methods perform better in terms of the generality and accuracy of the explanations they generate, and have much lower computational cost than methods based on black-box access (Mahajan, Tan, and Sharma, 2019; Verma, Dickerson, and Hines, 2020).

2.2.1 Requirements of the surrogate models used for explaining the black box models

As discussed in Molnar, 2020, explanations for black-box models through surrogates should satisfy the following properties:

- *Accuracy:* This property refers to how well an explanatory surrogate model predicts unseen data.
- *Fidelity:* The decisions of the explanatory surrogate model should be close to the decisions of the black-box model on unseen data. If the black-box model has high accuracy and the explanation has high fidelity, then the explanatory surrogate model also has high accuracy.
- *Consistency:* The explanations should apply equally well to any machine learning algorithm trained on the same data set.

- *Stability:* Decisions by the black-box model on similar instances should yield similar explanations.
- *Representativeness:* If the surrogate model can be applied to several decisions on several instances, we can call it a highly representative explanation.
- *Certainty:* If the black-box model at study provides a measure of assurance in its predictions, an explanation of these predictions should reflect this measure.
- *Novelty:* It indicates the ability of the explanations of the surrogate model to cover cases far from the training data.
- *Degree of importance:* The explanation should highlight the important features.
- *Comprehensibility:* The explanations provided by the surrogate model should be understandable to humans. Depending on the target users, more or less complex explanations can be acceptable, but short explanations are generally more comprehensible.

No single explanation model in the current literature is able to satisfy all the above properties (Blanco-Justicia et al., 2020). In particular, LIME and Anchors focus on satisfying fidelity, stability, degree of importance, and comprehensibility, whereas SHAP focuses on fidelity, stability, representativeness, and degree of importance (Molnar, 2020).

2.3 Federated learning

Federated learning (Konečný et al., 2016) is a decentralized machine learning technique that aggregates local models trained by a set of peers on their private data to obtain a global model. Since private data do not leave the peers' devices, FL provides more intrinsic privacy to the participating peers than other approaches that require uploading the peers' data to a central server. Another advantage of FL systems is that the learning effort is distributed among peers instead of being centralized in a single entity, as shown in Figure 2.2. However, FL models are also black boxes. In more detail, in FL a model manager (a.k.a. server) uses a learning algorithm such as *FedAvg* (McMahan et al., 2017) to learn a global shared model θ by cooperating with m participating peers. In the beginning, the server initializes an ML model, for example, a neural network, with parameters θ^0, and asks the m participating peers to update the model by training it on their local data with a set of predefined hyper-parameters. The main hyper-parameters are the number of local epochs e, the local batch size bs and the learning rate α. After that, following the prescribed learning algorithm, at epoch t each peer p among the m participating peers uses her private data set D_p to train θ^t locally, and sends the resulting local model θ_p^{t+1} or the computed gradient δ_p^{t+1} back to the server. Once the server receives the local updates, it computes the global model θ^{t+1} for epoch $t+1$ by averaging the received updates. This process is iteratively repeated for a fixed number of epochs or until the global model converges.

FL is particularly vulnerable to security and privacy attacks because the server has no control over the participating peers.

For example, a malicious peer may spoil the learning process by sending bad updates to the model manager. Ideally, honest peers should be able to *detect* and *understand* such attacks, because the dishonest behavior of one of the participants may significantly alter the expected behavior of the black-box model.

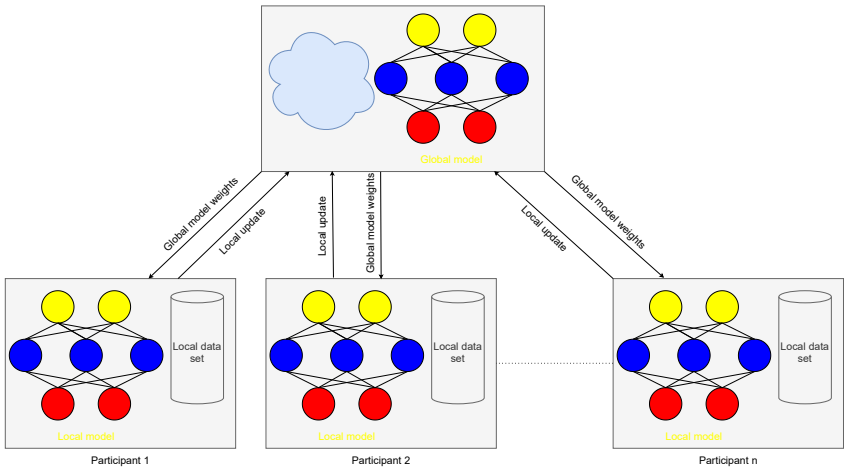


FIGURE 2.2: An example of a federated learning model architecture

2.3.1 Attacks in federated learning

Federated learning, due to its distributed nature, is vulnerable to a number of security attacks orchestrated by malicious peers:

- *Byzantine attacks*: These attacks try to prevent the model from converging. Lack of convergence may be caused by transmission errors, attackers that alter updates from other peers, or malicious peers that submit random updates (Lamport, Shostak, and Pease, 2019).

- *Poisoning attacks*: These attacks aim at misleading the global model into misclassifying a specific set of inputs. For example, in a recommendation system trained via federated learning, a possible aim would be to cause the recommendation system to suggest a specific item or to recommend it more often than due (Fang et al., 2020). In (Zhong and Deng, 2021), a powerful poisoning approach based on transferable adversarial examples has been proposed: a surrogate model is leveraged to create adversarial examples that can poison a black-box deep learning model with only API access (*i.e.* without knowing its structure or having access to the data it has been trained on).

- *Label-flipping attacks*: In these attacks, the attacker is assumed to have access to a part or all of a peer's training data. The attacker uses this access to alter (some of) the labels in the training data (Taheri et al., 2020).

External entities may also attack the model manager and may even mount attacks to break the privacy of the contributing peers:

- *Security attacks against the model manager*: Since the model manager receives all local updates, an external attacker who takes control of the model manager gains access to all updates and can manipulate the aggregation process (Kairouz et al., 2021). Thus, the attacker can easily carry out Byzantine or poisoning attacks.

- *Privacy attacks by the model manager*: A powerful and hard-to-detect privacy attack was proposed in Hitaj, Ateniese, and Pérez-Cruz, 2017. The authors use a multi-task Generative

Adversarial Network (GAN) for Auxiliary Identification, called mGAN-AI. The typical GAN (Goodfellow et al., 2020) consists of the generator G and the discriminator D. G is trained to generate data from the same target data distribution, while D is trained to classify the output of G as a real or a fake data sample. In mGAN-AI, the updates (or the model) sent by a peer are used as a discriminator so that the model manager can take advantage of it to generate data from the same distribution as the data owned by a participating peer. This attack may break the privacy of the peer's data.

2.3.2 Countermeasures against attacks

Security attacks by malicious peers can be detected (and filtered) by the model manager, provided the manager has access to the individual updates. Several methods to detect malicious updates have been proposed in the literature:

- *Detection of malicious peers via model metrics*: This class of methods use a custom validation set to assess the updates sent by the participating peers. The performance metrics (*e.g.*, accuracy) of the model as updated by each peer are computed on the validation set. The model manager filters out the peer updates that yield poor performance before aggregating the remaining updates to obtain the new global model.

- *Detection of malicious peers via update statistics*: As the model converges, the magnitudes of the gradients δ_p^{t+1} tend toward zero. This attack detection method checks the magnitudes of

the updates and identifies those updates as anomalous outside a given range. Usually, this range is related to the interquartile range of the set of updates sent by the participating peers during the training. However, since the gradients become close to zero only when models approach convergence, it is not realistic to apply this methodology throughout the training phase (Tukey, 1977; Domingo-Ferrer et al., 2020; Jebreel et al., 2020). In Cao et al., 2019, a different way of filtering out label-flipping attacks by analyzing updates is presented. The authors propose to construct a graph of all the updates sent by the participants. This graph is built according to the parameters of each update, and Euclidean distances are calculated to identify updates that are not similar to the majority, which are filtered out. This method, however, requires a large number of costly operations per epoch.

- *Outlier-excluding aggregation*: Other countermeasures, such as Krum aggregation (Blanchard et al., 2017) or the coordinate-wise median (Yin et al., 2018), consist in aggregation mechanisms that exclude outlying values (which are regarded as probably malicious updates). However, these countermeasures may reject updates from honest peers whose private data distribution legitimately differs from that of the majority.

Other measures to prevent or filter out attacks exist, but they rely on cryptography, secure channels, and trusted hardware (Chen et al., 2020), thereby imposing significant deployment requirements.

Some recent research works (Kim et al., 2020; Salah et al., 2019) combine federated learning with blockchain technologies. The

idea is to save global model parameters and weights in an unchangeable blockchain ledger in order to ensure the security of the global ML model. However, although the blockchain can save the current model, it cannot protect against bad updates uploaded by malicious participants. Furthermore, very substantial computational power is needed to create each block in the blockchain.

Even though the security countermeasures discussed above can filter out malicious updates and thereby increase the model accuracy, they can only be implemented on the server side, and they do not explain the operation of the attack, that is, the data features modified by it. In this thesis, we use surrogate models built on the side of any participant in the training, using only their data not only to detect but also to *explain* potential security attacks.

Regarding privacy attacks by the model manager, these can be proactively prevented by the participating peers themselves by distorting their updates via differential privacy (Wei et al., 2020) or by securely aggregating updates of several peers before sending them to the server (Bonawitz et al., 2017). However, differential privacy severely deteriorates the accuracy of the learned model (Domingo-Ferrer, Sánchez, and Blanco-Justicia, 2021; Blanco-Justicia et al., 2022), whereas secure aggregation is incompatible with countermeasures against security attacks because it hides from the model manager the individual updates provided by the clients (Blanco-Justicia et al., 2021).

2.4 Conclusion

Although NNs and DL models are opaque and require interpretation, they perform better than classical machine learning methods on classification, segmentation, and detection tasks (Janiesch, Zschech, and Heinrich, 2021), which calls for their use to accomplish these tasks efficiently. For this reason, there is an urgent need to explain the predictions of these models. Previous attempts to generate these interpretations have used either simpler models (*i.e.*, model-agnostic approaches), which cannot properly approximate the black boxes' behavior due to their low performance, or the internal parameters of the black boxes (*i.e.*, model-specific approaches), which prevents the model users from generating their local explanations. This calls for the research of an explainer that either uses a more robust surrogate model such as random decision forests (see Section 2.1.1) or allows the user to create local model-specific explanations without having to share the internals of the models with all the users. Furthermore, since DL and, particularly, FL models are vulnerable to various security and privacy attacks that might target the model, the information provided by some explainers can be used to detect and understand these attacks.

Chapter 3

Explaining Misclassification and Attacks in Centralized Deep Learning via Random Forests

In centralized deep learning, the training data are gathered by a single entity, and the training of the black-box model is done locally on that entity's side, which makes it the only one responsible for preserving the privacy and the security of the data. Nevertheless, this forces all interested parties to share their data with that central entity. Besides, some portions of the data could be wrongly clustered or attacked to alter some of its values. Those corrupted portions could affect the performance of the model. Furthermore, this effect will lead the model to wrong classifications. In this

chapter, we will focus on explaining the wrong model's predictions, and we will evaluate if the data were corrupted or not during the training by using a model-agnostic approach based on random forests as a surrogates.

3.1 Contributions and plan of this chapter

We present an approach that assumes that the party who generates the explanations has unrestricted access to the black-box model and the training data set. The training data set of the surrogate model can be smaller than the entire data set used to train the black-box model: a sufficiently representative subset may be enough.

We use random decision forests in this chapter to build our surrogate model. Random forests (Ho, 1995) consist of a fixed number of decision trees, each of which has a controlled depth and a measure reflecting the feature's importance. The surrogate random forest will be trained on the same data used to train the black-box model.

Single decision trees have already been employed in the literature as surrogate models (Blanco-Justicia et al., 2020; Singh, Ribeiro, and Guestrin, 2016). The originality of our proposal lies in using a random decision forest rather than a single decision tree. In this forest, all trees have limited depth, but the structure of each tree may differ. The diversity in the random decision forest makes it possible to have trees whose predictions match the prediction of the black-box model, even if the majority of the trees do not match it. This allows scanning the forest for trees that agree with

the black-box model and using these trees to explain the black-box decision.

In this chapter, we concentrate on the cases where the black box model made wrong predictions and aim to identify the features involved in those predictions. For example, if some features of the training data set were altered due to an attack, we can identify which of the altered features were more influential in the wrong predictions.

The rest of this chapter is structured as follows. Section 3.2 describes our surrogate model based on a random decision forest. Then, experimental results are provided in Section 3.3. Finally, in Section 3.4 we gather conclusions.

3.2 Random forest-based surrogate model

To fulfill the most outstanding properties listed in Section 2.2.1, such as fidelity and accuracy, degree of importance, consistency, and comprehensibility, we need to build a surrogate model that can provide information about the black-box model predictions with high accuracy while keeping complexity at bay. Random decision forests described in Section 2.1.1 are a promising solution because they are able to provide accurate predictions, and the decision trees they build are intrinsically understandable (Blanco-Justicia et al., 2020).

Moreover, one of the advantages of random forests over decision trees is that the latter become too deep to be understandable if they have to make accurate decisions based on a large number of features. In contrast, random forests can offer accuracy in the

case of many features by using several trees with limited depth and stay thus understandable.

Random decision forests also satisfy consistency because the surrogate model does not depend on the internal structure of the black-box model. Moreover, since forests are built from the actual data, they can explain the predictions of any black-box model trained on the same data.

Another distinguishing trait of random forests over other classical ML models is that each tree in the forest focuses on a different subset of features, and at least one of those trees agrees with the black-box model prediction. Our method allows deterministically choosing the trees/feature sets that have the greatest influence on the predictions that match the predictions of the black-box model, thereby yielding more accurate explanations.

When the surrogate model is trained on a fraction of the whole training data set, it provides the feature importances vector, which assesses the importance of each feature in the data on the prediction made by the black box.

Algorithm 1 attempts to determine the causes of wrong predictions by the black-box model. First, the algorithm uses the training data set to train the black-box and random forest models. Second, it evaluates both models to make sure they can be compared with each other. The random forest's accuracy should not be inferior to that of the black-box model; otherwise, the explanations obtained from the random forest would be useless. Third, for each wrong prediction of the black-box model on the test data set, the algorithm scans the forest and stores the vector of feature importances of each tree in the forest whose prediction coincides

with the wrong black-box prediction. Finally, the algorithm averages all stored feature importance vectors in order to obtain a vector containing the average importance of every feature in causing wrong decisions.

Algorithm 1 Determine the importance of features in wrong predictions

input: Data set X, CNN model *model*
$Train_X, Test_X \leftarrow Split_Train_Test(X)$
$Black_Box \leftarrow Train_Black_Box(Train_X)$
$Forest \leftarrow Build_Random_Forest(Train_X)$ {The accuracy of the black-box model and the random forest are evaluated}
$Score_Black_Box \leftarrow Evaluate_Black_Box(Test_X)$
$Score_Forest \leftarrow Evaluate_Forest(Test_X)$
$Feature_Importances_List \leftarrow \{\}$ {*}This list will contain the vectors of feature importances for selected trees in the forest
for *Sample in Test_X* **do**
  if $Predict(Black_Box, Sample)$ not correct **then**
    for *Tree in Forest* **do**
      if $Predict(Black_Box, Sample) == Predict(Tree, Sample)$ **then** {The vector with the feature importances of each tree agreeing with the black-box model is appended to the list}
        $Append(Feature_Importances_List, Tree.Feature_Importances)$
      end if
    end for
  end if
end for{The vectors in the list are averaged to obtain the average feature importance vector}
$Feature_Importances \leftarrow Average(Feature_Importances_List)$
return $Black_Box, Score_Black_Box, Forest, Feature_Importances$

In Algorithm 1, feature importances are computed according to Pedregosa et al., 2012, as follows. First, let us define the notion

of impurity of a data set, which is a measure of the homogeneity of its values. Impurity can be measured in several values, including Gini impurity and Shannon entropy. In particular, the Gini impurity of a data set is

$$C = \sum_{i=1}^{L} f_i(1 - f_i),$$

where L is the number of possible different labels, and f_i is the relative frequency of values with the i-th label. Clearly, if all the values in the data set correspond to the same label, then $C = 0$; the more diverse the values, the higher C.

Now, given a decision tree, the importance of each node j in it is

$$ni_j = w_j C_j - \sum_{k \in \text{Children}_j} w_k C_k,$$

where w_j is the weighted number of samples reaching node j, C_j is the Gini impurity of the samples reaching node j, and Children_j are the children nodes of node j. Thus, the higher the homogeneity gain of a node, the more important it is, where the homogeneity gain is the reduction of impurity between the input set of the node and its output subsets (those that go to its children nodes). In other words, an important node is one that "neatly" classifies the samples that reach it. Then, the raw importance of each feature i is

$$fi_i = \frac{\sum_{j:\text{node } j \text{ splits on feature } i} ni_j}{\sum_{k \in \text{all nodes}} ni_k}.$$

Finally, the normalized feature importance of each feature i is a number between 0 and 1 computed as

$$normfi_i = \frac{fi_i}{\sum_{j \in \text{all features}} fi_j}.$$

From now on, in this thesis, when we mention feature importances, we will refer to normalized feature importances.

Algorithm 2 attempts to discover whether the training of a black-box model is done on corrupted/attacked data or on clean data, and which training features are most likely to have been attacked. The algorithm takes as inputs the outputs of Algorithm 1, that is, a trained black-box model, its reported accuracy score, a random forest explaining the black box model, and the vector of reported feature importances associated with the random forest. The algorithm also takes as input a *reliable* test data set Test_X that will be used to test whether the training of the black-box model was on attacked or clean data. First, the algorithm evaluates the accuracy of the black-box model on the reliable test data Test_X. If the accuracy Score_Black_Box2 on Test_X is much lower than the reported accuracy Score_Black_Box, this suggests that the black-box model was trained on attacked data. In this case:

1. For each wrong prediction of the black-box model on the reliable test data Test_X, the algorithm scans the forest and stores the vector of feature importances of each tree in the forest whose prediction coincides with the wrong black-box prediction.

2. The algorithm averages all stored feature importance vectors in order to obtain a vector Feature_Importances2 containing the average importance of every feature in causing wrong decisions.

3. The likelihood of each feature being attacked in the training data is proportional to the difference in the importance of that feature in Feature_Importances2 and the reported Feature_Importances. In particular, the feature with the largest difference is the most likely to have been attacked.

Algorithm 2 Discover the feature under attack

input: $Black_Box, Score_Black_Box, Forest, Feature_Importances, Test_X$
$Score_Black_Box2 \leftarrow Evaluate_Black_Box(Test_X)$
if Score_Black_Box − Score_Black_Box2 > threshold **then**
  $Feature_Importance_List2 \leftarrow \{\}$
  for *Sample in Test_X* **do**
    if $Predict(Black_Box, Sample) notcorrect$ **then**
      for *EachTreeinForest* **do**
        if $Predict(Black_Box, Sample) == Predict(Tree, Sample)$ **then**
          $Append(Feature_Importance_List2, Tree.Feature_Importances)$
        end if
      end for
    end if
  end for
  $Feature_Importance2 \leftarrow Average(Feature_Importance_List2)$
  $Feature_Attack_Likelihoods = |$Feature_Importance − Feature_Importance2$|$
  return *Feature_Attack_Likelihoods*
end if

3.3 Experimental results

We applied the above-described methodology to three data sets: a synthetic numerical data set, a real numerical data set, and a real data set with a mix of categorical and numerical attributes. To keep computation simple, we made a small change when testing Algorithm 2. Instead of attacking the training data (which would require training both the black box model and the random forest first with the original training data to get the real importance of features and then again with the attacked training data), we attacked the test data Test_X used by Algorithm 2. This avoids re-training but has the same effect: the test data Test_X used in Algorithm 2 depart from the data used to train the model. We can take Test_X as the good data and the training data as having been attacked.

3.3.1 Experiments on synthetic numerical data

We generated a data set consisting of $1,000,000$ records, each with 10 numeric continuous attributes and a single binary class labeled using the `make_classification` method from *Scikit learn*[1]. We reserved $2/3$ of the records to train the models and the remaining $1/3$ to test them. As a black-box model, we took a neural network denoted by an artificial neural network (ANN) with three hidden layers of 100 neurons each, which achieved 96.55 % classification accuracy. We also trained a random forest with 1000 trees with a maximum depth of 5 and an average size of 62 nodes. The forest classification accuracy was 90.8%; this is less than the accuracy

[1] `https://scikit-learn.org/stable/index.html`

of the black-box model but still high enough for the explanations obtained from the random forest to be useful.

Table 3.1 shows the importances of features on the wrong decisions of the black-box model, as computed by Algorithm 1. Features are sorted in descending order of importance. Feature number 5 turns out to be the one with the most influence on wrong decisions.

TABLE 3.1: Importance of the features of the synthetic data set on wrong predictions by the black box model

Feature name	*Feature importance*
X[5]	0.2916
X[1]	0.2564
X[4]	0.0924
X[6]	0.0769
X[3]	0.0646
X[9]	0.0532
X[7]	0.0462
X[0]	0.0395
X[2]	0.0394
X[8]	0.0392

Also, we ran Algorithm 2 after attacking each feature individually with a threshold for the drop in the accuracy set to 4%. Our attacks were sufficient to exceed the threshold for all features. Algorithm 2 correctly detected the attacked feature in 60% of these single-feature attacks. Table 3.2 shows the result of attacking the three features with the highest importance (according to

Table 3.1). It can be seen that the algorithm correctly detected attacks on the two most important features $X[5]$ and $X[1]$ (the only two whose importance is above 10%). However, the algorithm failed when the third most important feature $X[4]$ was attacked (it mistook it for the second most important feature $X[1]$). Table 3.3 shows the performance of Algorithm 2 when the three features with the least importance were attacked. The algorithm detected well the attacks on the least important feature $X[8]$ and the third least important feature $X[0]$. Still, it failed for the second least important feature $X[2]$ (which was mistaken for the second most important feature $X[1]$).

A comparison between Table 3.2 and Table 3.3 also suffices to see that the drop in the accuracy of the black box model was greater when the high-importance features were under attack. Indeed, attacks on low-importance features entailed a milder degradation of accuracy. This also confirms that the feature importances identified by Algorithm 1 are coherent with the impact of those features on accuracy.

TABLE 3.2: Attacking the three most important features in the synthetic data set

Attacked feature	*black box accuracy*	*Feature detected by Algorithm 2*
X[5]	64.81%	X[5]
X[1]	64.9%	X[1]
X[4]	61.97%	X[1]

TABLE 3.3: Attacking the three least important features in the synthetic data set

Attacked feature	*black box accuracy*	*Feature detected by Algorithm 2*
X[8]	76.45%	X[8]
X[2]	73.38%	X[1]
X[0]	75.99%	X[0]

3.3.2 Experiments on real numerical data

To experiment with large real numerical data, we used the "PAMAP2 Physical Activity Monitoring" data set from the UCI Machine Learning Repository[2]. This data set contains continuous measurements of three inertial body sensors (placed on the arm, chest, and ankle) and a heart-rate monitor worn by nine subjects who performed 18 different activities such as walking, cycling, watching TV, etc. First, as recommended by the releasers of the activity data set (Reiss and Stricker, 2012), we created our data set by discarding the transient activity (*e.g.*, going to the next activity location). Second, for simplicity, we mapped the various types of activity into two categories indicating whether the activity involved displacement or not (*e.g.*, walking and cycling were mapped to "displacement" and watching TV to "not displacement"). As a result, we obtained a data set containing $1,942,872$ records of which $1,136,540$ records were labeled as "displacement" and $806,332$ as "not displacement".

Each record contained 54 numerical attributes corresponding to timestamp, label, heart rate, and 17 sensor data feeds for each

[2] `https://archive.ics.uci.edu/ml/datasets/PAMAP2+Physical+Activity+Monitoring`

of the three inertial sensors. Given an unlabeled record, the experiment's classification task consisted of deciding whether the subject was performing an activity involving physical displacement at that instant. We used the same black-box and random forest models as in the synthetic data set. In classifying this data set, the black box achieved 99.9% accuracy and the forest 94.6% accuracy. Algorithm 1 computed the importances of the various features on wrong decisions of the black-box model: these importances ranged from 0.1616 down to 0.00017802.

We then applied Algorithm 2 with a threshold of 5% drop in the accuracy. Only 58.97% of the attacks on single features exceeded this threshold. Algorithm 2 correctly detected the attacked feature in 78.26% of these single-feature attacks. Table 3.4 shows that the attacks on the three most important features were detected correctly. Table 3.5 shows three examples of attacked features with low importance: two of those attacked features were correctly detected, but the algorithm was not able to detect the attack on acceleration_6_x_chest.

A comparison between Table 3.4 and Table 3.5 shows that the accuracy drop was greater when attacking a feature with high importance than a feature with low importance. The exception was acceleration_6_x_chest which, in spite of being of low importance, caused a substantial accuracy drop and was, in fact, wrongly detected by Algorithm 2 as the high-importance feature acceleration_6_x_ankle. However, other than that, our proposed method showed very promising results in detecting the importance of the features causing the black box wrong predictions on real numerical data and in detecting attacks on specific features.

TABLE 3.4: Attacking the three most important features in the physical activity data set

Attacked feature	*black box accuracy*	*Feature detected by Algorithm 2*
magnetometer_z_chest	82.66%	magnetometer_z_chest
gyroscope_z_ankle	76.23%	gyroscope_z_ankle
acceleration_6_x_ankle	86.25%	acceleration_6_x_ankle

TABLE 3.5: Attacking the three least important features in the physical activity data set

Attacked feature	*black box accuracy*	*Feature detected by Algorithm 2*
acceleration_6_x_hand	92.47%	acceleration_6_x_hand
acceleration_6_x_chest	86.73%	acceleration_16_x_ankle
gyroscope_x_ankle	90.68%	gyroscope_x_ankle

3.3.3 Experiments on real categorical data

To experiment on a real data set containing categorical data, we used the Adult data set, which is a standard data set hosted in the UCI Machine Learning Repository[3]. Adult contains 48,842 records of census income information and has 14 attributes reporting both numerical and categorical values.

We recoded categories as numbers for each categorical attribute to obtain a numerical version of the attribute. We reserved two thirds of the records to train the models and the remaining one third to validate them. We used the same black box and forest as in the synthetic and physical activity data sets. The black box achieved 84.53% classification accuracy, and the forest achieved 84.42%. Table 3.6 lists the importance of features on the wrong decisions of the black-box model, as computed by Algorithm 1. Features are sorted in descending order of importance.

[3]https://archive.ics.uci.edu/ml/data sets/adult

TABLE 3.6: Importance of the features of the Adult data set on wrong predictions by the black box model

Feature name	*Feature importance*
marital-status	0.2459
capital-gain	0.1967
relationsip	0.1902
educational-num	0.1645
age	0.012
hours-per-week	0.0451
capital-loss	0.0384
occupation	0.021
gender	0.0177
workclass	0.0035
native-country	0.00172
fnlwgt	0.00171
race	0.00094

We applied Algorithm 2 with a threshold 3% in the accuracy drop. Only 53.84% of the attacks on single attributes exceeded this threshold. Among these 53.84%, Algorithm 2 correctly detected the attacked attribute in 85.71% of the cases. Table 3.7 shows the results for the three most important features: attacks on them were all correctly detected. Table 3.8 reports on three features with low importance: two were well detected, but capital-loss was wrongly detected. Yet capital-loss was the only attribute in the data set whose attack exceeded the accuracy drop threshold but was not correctly detected by Algorithm 2.

A comparison between Table 3.7 and Table 3.8 also shows that the drop in the accuracy was greater when attacking a feature with high importance (according to Algorithm 1) than a feature with

lower importance.

Our approach holds promise because (i) in the case of the real categorical data set, only one single feature was wrongly detected as attacked by Algorithm 2; (ii) the feature importances computed by Algorithm 1 anticipate the black-box model accuracy drop when each respective feature is attacked.

TABLE 3.7: Attacking the three most important features in the Adult data set

Attacked feature	*black box accuracy*	*Feature detected by Algorithm 2*
martial-status	80.96%	martial-status
capital-gain	63.62%	capital-gain
relationship	77.51%	relationship

TABLE 3.8: Attacking the three least important features in the Adult data set

Attacked feature	*black box accuracy*	*Feature detected by Algorithm 2*
hours-per-week	81.30%	hours-per-week
capital-loss	79.18%	capital-gain
educational-num	76.48%	educational-num

3.4 Conclusions

We have presented an approach based on random decision forests with small tree depth that provides explanations of the decisions made by black-box machine learning models. Specifically, we have focused on investigating and explaining wrong decisions. Algorithm 1 computes the importance of the various features on the wrong black-box model decisions. Additionally, the visualization

of the random forest trees affords a further understanding of the decision-making process. Finally, Algorithm 2 introduces a new way to protect against attacks that alter the training data because it detects which features have been attacked.

Chapter 4

Explaining Predictions and Attacks in Federated Learning via Random Forests

Federated learning (introduced in Section 2.3) allows participants to train robust and accurate DL black-box models collaboratively. Moreover, FL improves privacy since the participants' sensitive data do not leave their premises. Still, the distributed nature of FL makes it vulnerable to security and privacy attacks, as discussed in Section 2.3.1. Participants or even the model manager can orchestrate those attacks. In this chapter, we present an explainability method that interprets the predictions of the federated black-box models. In addition, the proposed method can detect various attacks that target federated training with a high detection success rate.

4.1 Contributions and plan of this chapter

As in the previous chapter, we also use random decision forests (Ho, 1995) as black-box FL models surrogates. Those surrogates accomplish two tasks. The first task is to create explanations for the predictions of the black-box FL model. Even though decision trees are commonly used as explainability tools for DL models in the literature, *ours is the first attempt to use random forests of limited depth to explain the (wrong) predictions of black-box models trained in a decentralized setting.* The second –and most relevant– task consists in leveraging the random forest surrogates to detect security and privacy attacks against FL model training. Again, even though a variety of FL attack detection mechanisms have been proposed in the literature (see Section 2.3.1), *our approach is novel in that we do not only* detect *the attack, but we also* explain *its operation at the peer's side*.

The purpose of most attacks targeting FL is to affect the model predictions, and this usually results in lower model accuracy. Therefore, we focus on wrong predictions by the FL model because they may signal possible attacks. The empirical results we report demonstrate the effectiveness of our proposal at detecting and explaining attacks against FL.

The remainder of this chapter is organized as follows. First, Section 4.2 describes our surrogate model based on random decision forests. Next, experimental results on the explainability performance of the surrogate and the attack detection accuracy are reported in Section 4.3. Finally, in Section 4.4 we gather conclusions.

4.2 Constructing explainable surrogate models via random forests

In FL systems, the training data are held by the participating peers. Therefore, the peers themselves can build surrogates of the global FL model by leveraging the updated global model they receive at each epoch and their data, provided the performance of this surrogate satisfies the accuracy property (see Section 2.2.1), thereby allowing peers to obtain explanations of the global model's predictions. Again, random decision forests (see Section 2.1.1) are suitable surrogates since they can be trained on a portion of the data and still achieve acceptable accuracy.

Moreover, since peers can get explanations at each epoch, they can also observe significant changes in the model's behavior, indicating attacks orchestrated by malicious entities. Thus, *random forests can equip peers with explanations and attack detection capabilities*.

The proposed method for extracting explanations of the predictions of the black-box model is formalized in Algorithm 3. Peers interested in creating interpretations of black-box predictions follow the learning protocol, and, at the same time, they test the black-box model with their own data. To this end, they divide the data they own into two parts. They use the first part to build the random decision forest in the first epoch and to train the black-box model at each epoch. They use the second part of the data to test the updated black-box model they receive at each epoch and to obtain explanations (by means of the random forest they built) on the predictions the black-box model may give.

We again focus on the black-box model's wrong predictions,

which may indicate malicious manipulations and attacks. For each wrong prediction of the black-box model on the peer's test data, the algorithm scans the forest and stores a vector with the feature importances (see Section 3.2) of each tree in the forest whose prediction matches the wrong black-box prediction. As mentioned above, the forest diversity makes it very likely to find trees that match the (wrong) black box predictions. Finally, the algorithm averages all the stored feature importance vectors to obtain a vector containing the average importance of every feature causing wrong predictions.

Algorithm 3 Computation by each peer of the importance of features in the black box model's wrong predictions

1: **Input:** Data set X owned by peer P_X, epoch number $Epoch_No$, black-box model $Black_Box$ obtained via FL at the current epoch;
2: **if** $Epoch_No == 1$ **then**
3: $Train_X, Test_X \leftarrow Split_Train_Test(X)$;
4: $Forest \leftarrow Build_Random_Forest(Number_trees, Max_depth, Train_X)$;
5: **else**
6: Retrieve $Test_X$ and $Forest$ computed in the first epoch;
7: **end if**
8: $Score_Black_Box \leftarrow Evaluate_Black_Box(Black_Box, Test_X)$;
9: $Feature_Importances_List \leftarrow \{\}$;
10: **for** each $Sample$ in $Test_X$ **do**
11: **if** $Predict(Black_Box, Sample)$ not correct **then**
12: **for** each $Tree$ in $Forest$ **do**

```
            if Predict(Black_Box, Sample) == Predict(Tree, Sample) then
                Feature_Importances_List.Append(Tree.Feature_Importances);
            end if
        end for
    end if
end for
Feature_Importances ← Average(Feature_Importances_List);
Updated_Black_Box ← Train(Black_Box, Train_X);
Return Score_Black_Box, Forest, Feature_Importances, Updated_Black_Box.
```

In Algorithm 3, feature importances are computed as described in Section 2.1.1. By relying on feature importances, Algorithm 4 attempts to detect whether a malicious manipulation or an attack happened during the training of the federated black-box model. The idea is that the algorithm compares the changes in the feature importances across the training process. Particularly, starting from the second epoch, the peer that built the surrogate model calculates the changes in the feature importances between every two consecutive epochs. In this way, she can monitor whether there are big changes in the values of the feature importances, which may indicate that an attack affecting the performance of the black-box model is in progress. To automate the attack detection task, she needs to compute the average of those changes from the start of the training to the current epoch and use the successive averages as baselines for comparison at each epoch. Then, starting from the third epoch, Algorithm 4 checks the changes in feature importances and compares them with the average changes over

the training epochs up to the current epoch. Since all peers update the global model at every training epoch, we expect small changes in feature importances. However, if the current changes are greater than the threshold α, where $\alpha > 1$, this may indicate that an attack has happened. Parameter α controls how strictly Algorithm 4 reacts to changes in the feature importances; the smaller α, the more sensitive the algorithm is to small changes. Due to how the model training naturally evolves at every single iteration (where small changes are expected), it is not recommended to set $\alpha < 1.2$. Also, if $\alpha > 2$, the algorithm will only detect the attacks after the model has been significantly affected. Thus, we recommend values within the range $1.2 \leq \alpha \leq 2$.

On the other hand, by looking at changes at the feature level, the algorithm is able to find out which features were attacked: it selects those features having an impact on changes in the feature importance greater than β, where $\beta < 1$ is the threshold to select the affected feature according to the weight of the single feature in the total changes in the feature importance. The set of selected features explains how the attack operated. Since the total weight of changes in the feature importances is normalized to 1, we recommend setting $0.15 \leq \beta \leq 0.4$, meaning that a feature should influence the total changes by at least 15% to be selected. This weight can be tuned according to the number of features.

Algorithm 4 Detection by each peer of attacks on the federated black-box model

1: **Input:** $Feature_Importances[1 : Max_Epochs][1 : Max_Features], Epoch_No$;
2: $i = Epoch_No$;

3: $Targeted_Features \leftarrow \{\}$;
4: **if** $i > 2$ **then**
5: **if** $i \geq 2$ **then**
6: $Changes_Feature_Importances[i] = |Feature_Importances[i] - Feature_Importances[i-1]|$; {The operands in this subtraction are vectors with $Max_Features$ components}
7: $Average_Overall_Changes_per_Epoch = AVG(Changes_Feature_Importances[1 : i])$; {$Average_Overall_Changes_per_Epoch$ is a vector with $Max_Features$ components}
8: $Total_Feature_Change_per_Epoch = SUM(Average_Overall_Changes_per_Epoch)$; {$Total_Feature_Change_per_Epoch$ is a scalar obtained as the sum of the components of $Average_Overall_Changes_per_Epoch$ over all features and it represents the mean of the total feature importance changes over the epochs until the current one}
9: $Total_Current_Changes = SUM(Changes_Feature_Importances[i])$; {Scalar containing the sum of all feature importance changes in the current epoch}
10: **if** $Total_Current_Changes \geq \alpha \times Total_Feature_Change_per_Epoch$ **then**
11: **for** j *in* $Changes_Feature_Importances[i]$ **do**
12: **if** $Changes_Feature_Importances[i][j] \geq \beta \times Total_Current_Changes$ **then**
13: $Targeted_Features.append(Feature[j])$;

end if
end for
end if
end if
end if
Return *Targeted_Features*.

4.3 Experimental results

In this section, we report experimental results on the explanatory usefulness of our surrogate model and on the ability of our algorithms to detect attacks against FL based on two different real data sets, one of them numerical and the other containing a mix of categorical and numerical attributes. The source code of the reported experiments is available for reproducibility purposes[1].

4.3.1 Data sets

We considered the following data sets:

- *Numerical data set*: We used the "PAMAP2 Physical Activity Monitoring" (a.k.a. Activity) data set described in Section 3.3.2.
- *Data set with a mix of categorical and numerical attributes*: We used the Adult data set described in Section 3.3.3.

In the experiments below, we used 100 peers, except in attack detection, where we tried various numbers of peers (30, 50, and

[1] `https://github.com/RamiHaf/Explainable-Federated-Learning-via-Random-Forests`

100). For the two data sets, 70% of the data were randomly split into as many disjoint shards as the number of peers, and each peer was assigned a different shard. Specifically, for 100 peers, each shard from Activity consisted of 13,600 records, and each shard of Adult consisted of 342 records.

The model manager kept the remaining 30% of the data set for validation purposes. The model manager used this validation data set at the end of each epoch to test the model's performance. At the same time, since the model was not trained on these data, they could be used for the final evaluation of the model.

Each peer P_X then further split her shard X into two parts: 70% of the data were used as *Train_X* for locally training the surrogate random forest, whereas the remaining 30% were used as *Test_X* to validate the black-box model and compute the feature importances (as per Algorithm 3).

4.3.2 Results on the explainability of the predictions of the black box model

We employed six federated black-box models with different internal structures to test the desirable properties of surrogate models introduced in Section 2.2.1. We built them by using the Keras[2] library (Gulli and Pal, 2017). The first model (*black box1*), which was employed in Blanco-Justicia et al., 2020 on the same data sets, had two hidden dense layers with 64 and 50 neurons, respectively. The second model(*black box2*) had three hidden layers, each with 100 neurons. The third (*black box3*) contained five hidden layers, with 100, 50, 30, 20 and 10 neurons, respectively. The fourth (*black*

[2] `https://keras.io/`

box4) also had five hidden layers, with 10, 20, 50, 20, and 6 neurons, respectively. The fifth model (*black box5*) had three hidden layers with 10, 6, and 4 neurons, respectively. Finally, the sixth (*black box6*) had seven hidden layers with 10, 40, 80, 30, 10, 6 and 4 neurons, respectively. We ran these models for ten epochs. We used the Adam optimization algorithm (Kingma and Ba, 2015), with batch size 32 and learning rate 0.001, with five local epochs for each peer. Using six different black-box models allowed us to test the proposed method's fidelity and accuracy, and compare the explanations it provided on different model architectures performing the same task. We built random forests at the peers' side using the command RandomForestClassifier[3] from the sklearn library with 1000 trees of maximum depth five and average size 62 nodes.

In what follows in this section, we report the experiments we carried out and the results we obtained on each of the properties listed in Section 2.2.1.

Fidelity and accuracy

Table 4.1 compares the accuracy of the six federated black-box models after converging and the random forest surrogate. One peer built the surrogate model on a shard from each of the two data sets introduced above. Table 4.1 also reports the fidelity score of the surrogate model vs each black-box model.

To train the black-box models, we used 70% of records of the corresponding full data set, whereas the random forests were trained

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`

on 70% of the shard of that data set held by the peer. Since the shards were created by randomly splitting the full data set, they can be assumed to consist of independent and identically distributed data and to be representative of the full data set.

The models with simpler architecture attained higher accuracy on the Activity data set: *black box1*, with only 2 hidden layers, achieved the highest accuracy (96.55%); *black box2* and *black box5*, with 3 hidden layers each, reached accuracy 93.32% and 93.11%, respectively; *black box3*, *black box4* and *black box6*, with 5, 5 and 7 hidden layers, respectively, achieved lower accuracy than the simpler models. On the other hand, for the Adult data set, *black box4*, with its five hidden layers having each a small number of neurons, attained the highest accuracy (87.56%); *black box3*, with the same architecture as *black box4* but with more neurons per layer, achieved the second-highest accuracy (84.14%); the rest of the models offered similar accuracy values, between 82.15% and 83.73%.

The fidelity reported in Table 4.1 is the absolute difference between the accuracy of each black-box model and the accuracy of the surrogate model. The smaller the fidelity score, the closer the performance of the surrogate model to the performance of the respective black-box model. For both data sets, the fidelity score of the surrogate model was at most 6.71%, which indicates that the surrogate model offered good fidelity to the black-box models.

Degree of importance

Our approach specifically focuses on computing the importance of features in the cases where the black-box model made a wrong

TABLE 4.1: Accuracy of the black-box models and the surrogate random forest for the Activity and Adult data sets. The fidelity of the surrogate model to each black box model is also displayed (a smaller score indicates more fidelity)

	Activity data set		Adult data set	
model name	Accuracy	Fidelity	Accuracy	Fidelity
black box1	96.55%	5.85%	82.28%	1.43%
black box2	91.32%	0.62%	82.15%	1.3%
black box3	89.33%	1.37%	84.14%	3.29%
black box4	88.58%	2.12%	87.56%	6.71%
black box5	93.11%	2.41%	83.73%	2.88%
black box6	90.74%	0.04%	83.51%	2.66%
Surrogate model	90.7%	N/A	80.85%	N/A

prediction.

We compared the explanations provided by our method with those obtained with the LIME method (Ribeiro, Singh, and Guestrin, 2016) mentioned in Section 2.2. First, LIME creates new samples of the data (as we do with adversarial examples) by adding small perturbations to the features of the data samples. Then, for each sample LIME intends to explain, it labels the newly created data samples using the black-box model and trains a linear SVM surrogate model to learn the relation between the features' value and the corresponding black-box prediction model. Finally, it classifies the features into two labels: one contains the features that support the prediction of the black box, and the other includes the features that contribute against it. Feature importances are computed by calculating the mean and standard deviation for the values of the feature in the original sample and the new perturbed samples and discretizing them into quartiles.

Table 4.2 reports the ten highest feature importances identified by our method (Algorithm 3) on the Activity and Adult data sets, and the corresponding importances computed by LIME.

TABLE 4.2: Ten highest feature importances for wrong *black box1* predictions computed by a peer using Algorithm 3 on the two data sets, and corresponding importances computed by the LIME algorithm

Activity data set			Adult data set		
Feature	Proposed method	LIME	Feature	Proposed method	LIME
gyroscope_z_ankle	12.38%	3%	*age*	18.17%	8%
magnetometer_z_chest	11.43%	17%	*educational-num*	15.63%	10%
acceleration_6_x_ankle	8.13%	2%	*capital-gain*	15.24%	68%
acceleration_16_x_ankle	7.63%	0%	*hours-per-week*	14.16%	7%
magnetometer_x_hand	6.15%	3%	*marital-status*	9.33%	2%
gyroscope_x_ankle	5.26%	2%	*relationship*	8.19%	3%
acceleration_6_y_chest	4.19%	1%	*occupation*	6.47%	1%
acceleration_16_y_chest	3.90%	0%	*race*	4.86%	0%
acceleration_6_z_chest	3.60%	2%	*workclass*	4.41%	3%
magnetometer_z_ankle	3.53%	4%	*native-country*	1.61%	4%

We can see that both LIME and our method agree on the features having the greatest importance. The difference is that, whereas LIME gives a dominant weight to just a single feature (*magnetometer_z_chest* for the Activity data set and *capital_gain* for the Adult data set), our method prioritizes a set of features (*gyroscope_z_ankle*, *magnetometer_z_chest* and *acceleration_6_x_ankle* for the Activity data set, and *age*, *educational-num*, *capital-gain* and *hours-per-week* for the Adult data set).

Since no approach in the literature allows comparing two explanation methods and since our proposed method is only concerned with investigating the reasons for the black-box wrong predictions, we devised an indirect comparison. Our idea was

to evaluate the accuracy benefits obtained after removing the features identified by the two explanation methods (ours and LIME) as most important in wrong predictions.

To this effect, we retrained and tested the accuracy of the *black box1* model on modified versions of the data sets after removing the features identified by each explanation method as being the most important for wrong predictions (see Table 4.2); feature removal affected only the input layer of the black-box model, but the rest of its structure was maintained. LIME identified just one clearly dominant feature (with importance above 10%). Thus, we removed the single most important feature identified by LIME on both data sets. In contrast, our method identified two features in Activity and four features in Adult with importance above 10%, which we removed. Afterwards, to confirm that the above feature removal was meaningful, we compared the accuracy values obtained by removing the above most important features with the accuracy values obtained by removing random subsets of features.

Table 4.3 reports the resulting accuracy values for the two data sets after removing: (i) the most important features detected by both methods and (ii) random subsets of features. For both data sets, removing features based on the explanations produced by our method resulted in better accuracy than removing features based on LIME explanations. We also see that removing the features we identified had a significant effect on the model accuracy (improvement from 96.55% in Table 4.1 to 99.67% in Table 4.3 for the Activity data set, and from 82.28% in Table 4.1 to 84.15% in Table 4.3 for the Adult data set).

In contrast, removing random subsets of features worsened

the model accuracy with respect to Table 4.1. These results suggest that our method yields explanations that are not only meaningful but more meaningful than those offered by LIME.

TABLE 4.3: Accuracy of the *black box1* model after removing the features with the highest importance on the wrong predictions, as identified by the proposed method and LIME, in comparison with the accuracy after removing random subsets of features. Subset1 and Subset2 are example random subsets. Subset1 consists of *(magnetometer_z_chest, magnetometer_z_ankle)* for the Activity data set, and *(native-country, workclass)* for the Adult data set. Subset2 consists of *(acceleration_16_z_ankle, acceleration_6_y_chest, acceleration_16_x_hand)* for the Activity data set, and *(relationship, occupation, marital-status)* for the Adult data set.

Data set	Activity data set	Adult data set
Proposed method	99.67%	84.15%
LIME	97.38%	80.37%
Subset 1	95.65%	79.72%
Subset 2	94.75%	78.39%
Average accuracy over 15 removed random subsets of features	95.22%	79.16%

Consistency

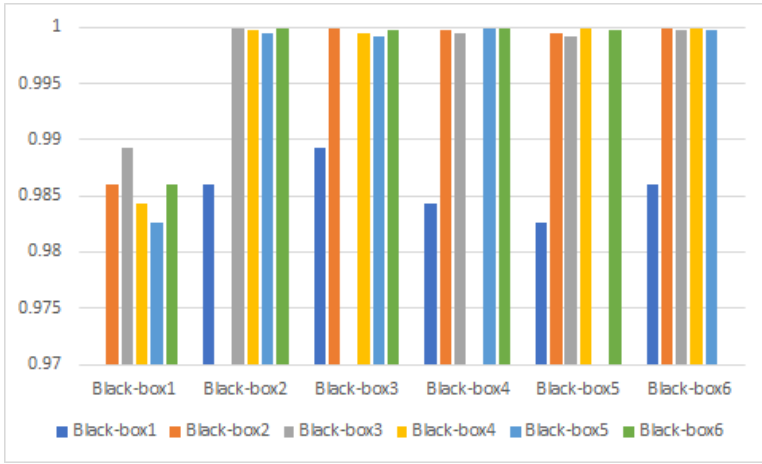
The surrogate model should consistently perform on any black-box model trained on the same data set. To measure the consistency of the surrogate model, we computed the Pearson correlation between the set of feature importances obtained by the surrogate for each pair of black-box models.

Figure 4.1 reports the correlation values for each pair of models. We can see that the correlation is very high (above 0.93) for all pairs and perfect in many of them, which indicates that the relative importances obtained by the surrogate are consistent across models.

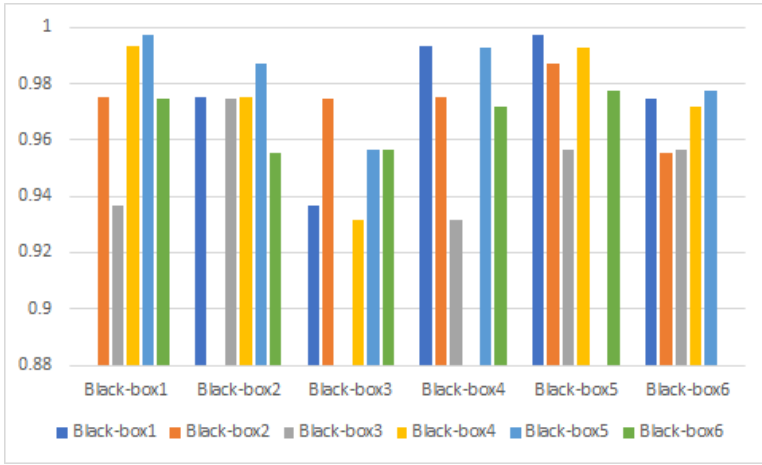
Comprehensibility

The outcomes of the surrogate model should be understandable by humans. In addition to the features' importance associated with wrong predictions, our method produces a collection of trees with limited depth that are easy to interpret. In particular, to obtain a visual representation of the behavior of the black-box model, we can select one or several trees that agree with the predictions of the black-box model. Figure 4.2 shows the structure of a tree selected from a random forest limited to the depth of 6 built on a synthetic data set with ten features. In contrast, if we build a surrogate consisting of a single tree covering the whole set of features in the data set, we obtain the very large structure shown in Figure 4.3, which has 20 as its maximum depth[4]. Notice that in this comparison, we are just interested in the size of the trees rather than their content. It is clear that trees in the random forest, being smaller and shallower, are easier to understand while still representing the behavior of the black-box model (because we can choose the trees that best agree with the black-box model).

[4]We have used a synthetic data set with ten features for this illustration because using the Activity or the Adult data sets would yield a standalone tree too large and deep to depict. *E.g.* for the Adult data set, we would get a tree with depth 41.



(A) Feature importance correlations for the Activity data set



(B) Feature importance correlations for the Adult data set

FIGURE 4.1: Pearson correlations between the feature importances obtained by the surrogate for each pair of black-box models on the Activity and Adult data sets

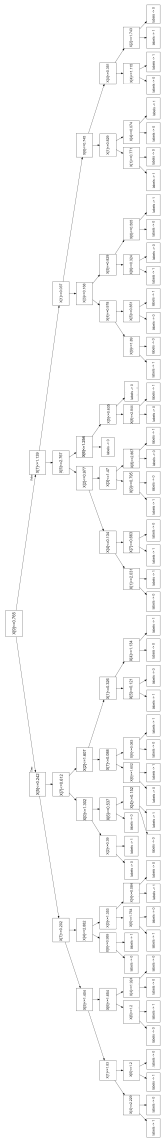


FIGURE 4.2: Structure of a tree from a random forest with depth limit 6 built on a synthetic data set



FIGURE 4.3: Standalone decision tree covering the whole set of features of the same synthetic data set used in Figure 4.2

4.3.3 Results on the attack detection performance

In this section, we evaluate the performance of our method at detecting security and privacy attacks on FL. We considered attacks conducted by peers participating in the training or the model manager himself. Moreover, we compared the performance of our method with other attack detection mechanisms described in Section 2.3.2.

Detection of security attacks

To test the effectiveness of our method at detecting security attacks initiated by malicious peers (or even by the model manager in case an attacker took control over her), we applied Algorithm 4 on the side of a peer during the training of the federated *black box1* model on the Adult data set. We took $\alpha = \frac{3}{2}$, which is neither too sensitive nor too insensitive an attack detection threshold. We considered the three types of security attacks described in Section 2.3.2: Byzantine attack, poisoning attack, and label-flipping attack. We ran several experiments with different numbers of participants: 30, 50, and 100 peers. In all cases, the attack was initiated by a single peer in the fourth epoch and ended in the seventh epoch. In Byzantine attacks, malicious peers replaced 40% of their data with random values in the same range as the original data. In a poisoning attack, the attacker targeted three data features, namely *relationship*, *marital-status* and *capital-gain*, by replacing the original values of these features in 60% of the data samples with random values in the same range. In a label-flipping attack, the attacker flipped label 0 to label 1 in 50% of the samples that had label 0.

Table 4.4 reports the difference between the average feature importances at each epoch and the importances of the features that were attacked. The proposed method detected attacks if there was a significant increase in such difference between two consecutive epochs. In most cases, this significant difference manifests one epoch after the attack begins. That is, the attack begins in the fourth epoch and is detected in the fifth epoch. However, in some cases, such as the Byzantine attack with 50 peers or the poisoning attack with 100 peers, the attack was detected in the sixth epoch. The reason is that the influence of the attack may not be immediately noticeable if a single peer conducts it in a large network. In other cases, the proposed method detects security attacks in post-attack epochs, such as the ninth or tenth epochs. This is because, once the attack is over, model training continues on the correct data, and the model weights undergo a big shift toward correct predictions. This facilitates post-attack detection.

Each time the proposed method detects an attack, it reports the features that are most suspicious of having been attacked. We considered that a feature had been attacked if the change in its importance accounts for more than 25% of the total changes in the feature importances. Thus, we used $\beta = \frac{1}{4}$ as the threshold for detecting the attacked features. Note that *Byzantine and label-flipping attacks* did not aim at specific features, but rather at impairing the overall system performance. Yet, these attacks were more effective on some features than on others, and our method reports the most perturbed features. In the case of the *poisoning attack*, however, three specific features were attacked, and our method

reported two out of the three every time: with 30 and 100 participating peers, *marital-status* and *capital-gain* were correctly detected, whereas, with 50 peers, *capital-gain* and *relationship* were correctly detected.

TABLE 4.4: Differences between average feature importances and importances of attacked features for different attacks on the Adult data set. Values in boldface correspond to the epoch in which the attack was detected.

	Byzantine attack			*Poisoning attack*			*Label-flipping attack*		
	30 peers	50 peers	100 peers	30 peers	50 peers	100 peers	30 peers	50 peers	100 peers
Epoch 2	0.022	0.011	0.017	0.019	0.032	0.026	0.014	0.024	0.017
Epoch 3	0.005	0.017	0.012	0.004	0.003	0.004	0.003	0.001	0.006
Epoch 4	0.001	0.003	0.007	0.001	0.0003	0.001	0.004	0.004	0.002
Epoch 5	**0.02**	0.002	**0.021**	**0.018**	**0.029**	0.005	**0.042**	**0.038**	**0.056**
Epoch 6	**0.041**	**0.023**	**0.018**	**0.014**	0.001	**0.017**	0.02	0.004	0.002
Epoch 7	0.004	0.003	0.004	0.007	0.004	**0.032**	0.005	0.013	**0.056**
Epoch 8	0.002	0.005	0.004	0.001	0.0003	0.005	0.012	**0.035**	**0.048**
Epoch 9	0.018	**0.013**	0.009	0.004	**0.036**	0.008	0.003	**0.05**	0.005
Epoch 10	0.004	**0.032**	**0.025**	0.001	0.0007	**0.027**	**0.024**	0.006	0.002
Reported features	*educational-num age*	*capital-gain educational-num capital-loss*	*occupation race*	*capital-gain marital-status*	*capital-gain educational-num relationship*	*capital-gain marital-status occupation*	*age relationship*	*capital-gain age occupation*	*capital-gain workclass*

We then calculated the performance of our method at detecting attacks on the two data sets and for all types of attacks and network configurations described above. For each case studied, we considered different numbers of attacking peers; since the majority of the peers involved were regarded as honest, we took the number of attackers to be between 1 and $\frac{m}{2} - 1$, where m was the number of peers participating in the training. We used a threshold $\alpha = \frac{3}{2}$ for all the experiments. We compared the performance of our method against the performance of malicious peer detection via update statistics described in Section 2.3.2, even though this countermeasure can only be implemented on the server side. The reason for choosing the latter method is that it is also based on distances, and it does not require a lot of computing power. We

considered the detection to be correct if the attack was detected while it was taking place and one malicious peer was correctly reported as the attacker. The duration of the attacks was four epochs, starting at the fourth epoch, and ending at the seventh epoch. We calculated the percentage of the cases where the two methods being compared correctly detected suspicious activities.

Table 4.5 reports the detection rate of our proposed method and the method based on update statistics:

- For the Activity data set, the detection rate of our method was low for Byzantine attacks because these attacks did not affect the accuracy of the model on this data set, and hence were less noticeable. However, for poisoning and label-flipping attacks, the detection rate of our method was above 90%.
- For the Adult data set, the detection rate was over 90% for all numbers of peers and attacks, except for 100 peers under Byzantine attacks, where detection was only 75.51%.

In general, the detection performance of our method improved when there were fewer participating peers in the network. The reason is that the effect of a single altered update is more noticeable when fewer peers are involved in the training process.

Boldface numbers in Table 4.5 indicate the best detection rate for a given number of peers, data set, and attack configuration. We can see that our method outperformed the detection of malicious peers via the update statistics approach for all attacks and for all numbers of participants with one exception: Byzantine attacks with 100 peers in the Activity data set. This exception is due to the fact that Byzantine attacks with that number of peers, and that data set had little impact on the accuracy of the black box

model, in part because the number of attackers was small compared to the number of peers.

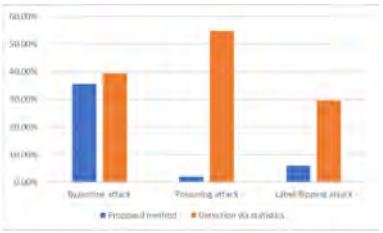
TABLE 4.5: Attack detection rate of the proposed method and the update statistics method on the two data sets. The detection rate of the best-performing method for a certain configuration of data set, attack type and number of peers is depicted in boldface.

	Data set	Attack type	30 peers	50 peers	100 peers
Proposed method	Activity	*Byzantine*	**71.42**%	**41.66**%	48.97%
		Poisoning	**100**%	**100**%	**95.91**%
		Label-flipping	**100**%	**91.66**%	**93.87**%
	Adult	*Byzantine*	**100**%	**95.83**%	**75.51**%
		Poisoning	**92.85**%	**91.66**%	**93.87**%
		Label-flipping	**100**%	**100**%	**97.95**%
Detection via statistics	Activity	*Byzantine*	64.29%	37.5%	**71.42**%
		Poisoning	28.57%	37.5%	48.95%
		Label-flipping	57.14%	66.66%	63.26%
	Adult	*Byzantine*	85.71%	62.5%	63.26%
		Poisoning	64.28%	50%	61.22%
		Label-flipping	71.42%	75%	77.55%

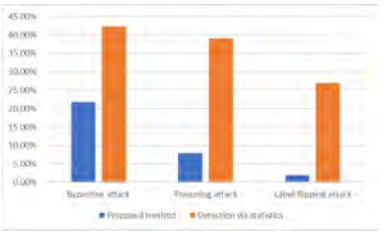
We also computed the false negative rate (FNR) and the false positive rate (FPR) of attack detection. A false negative occurs if the attack is not detected, or none of the attacking peers is reported by the attack detection via update statistics method, or none of the attacked features is reported by our method. To assess the FPR, we tested both methods while training the model without any attack; we repeated the test 50 times for each number of participating peers to calculate an accaurate error rate. Figures 4.4a and 4.4b depict the FNR for both detection methods under the three attacks mentioned above (Byzantine, poisoning, and label flipping) on the Activity and the Adult data sets, respectively. The results are consistent with those of Table 4.5. Even

though the FNR under the Byzantine attack was rather high for our detection method, it was lower than the detection method based on update statistics. On the other hand, under the poisoning and label-flipping attacks, our proposed method exhibited a very low FNR compared to the other method. Figures 4.4c and 4.4d depict the FPR for both detection methods, the three attacks, and the two data sets. We can see that the attack detection method via update statistics reports a list of malicious peers at every round of the training even if there were no attacks, while the FPR of the proposed method was less than 2% on the Activity data set and less than 18% on the Adult data set. Our FPR is lower on Activity than on Adult because the former data set has more features, resulting in smaller changes in the importance of each feature across the training epochs.

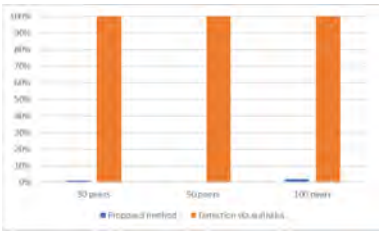
Although our proposed method was more efficient at detecting attacks, it might fail in front of attacks that continue non-stop from the first epoch to the end of training. In such a case, the features' importance would not vary significantly in any single epoch. Therefore, the attack would not be detected by the peers. However, since the updates that an attacker following this pattern would send during the whole training process would be very different from the updates generated by honest peers (especially in advanced stages of training), the server can detect and filter out those malicious updates with state-of-the-art server-side attack detection methods (mentioned in Section 2.3.2).



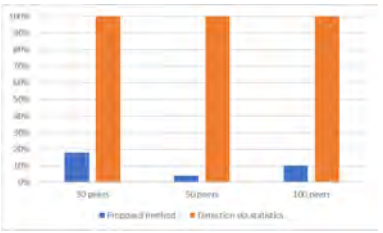
(A) FNR of attack detection on the Activity data set



(B) FNR of attack detection on the Adult data set



(C) FPR of attack detection on the Activity data set



(D) FPR of attack detection on the Adult data set

FIGURE 4.4: FNR and FPR of attack detection with the proposed method and with the method based on update statistics

Detection of privacy attacks

To test the ability of our method to detect attacks against privacy, we used the mGAN-AI attack described in Section 2.3.2. In this attack, the model manager tries to break the privacy of a targeted peer by generating data from the peer's data distribution. We considered the same three scenarios with 30, 50, and 100 peers by training *black box1* on the Adult data set and we used the same thresholds $\alpha = \frac{3}{2}$ and $\beta = \frac{1}{4}$ as in the above-mentioned experiments on security attacks.

In this case, the model manager initiates the attack in the sixth

epoch, when the network is close to converging. The attack generates data from the distribution of the two features *marital-status* and *relationship*.

In Table 4.6, we can see the performance of our method in detecting the mGAN-AI attack. For 50 and 100 participating peers, our method detected the attack one epoch after it was initiated (in the seventh epoch). For 30 peers, the attack was detected in the eighth epoch. Our method was able to detect this attack because the model manager modified the model before distributing it to the participants by including the data created by the generator G on the targeted peer. Because of this change, our method was able to detect the manipulated model and make the targeted peer aware of the attack so that she could decide to stop sharing further updates to avoid disclosing her data. Also, our method reported one valid target feature out of two in each case, which made the information provided about the attack very valuable in understanding the attacker's intent.

4.4 Conclusions

This chapter presents an approach based on random decision forests that explains the (mis)behavior of federated black-box models. Our method can detect and explain attacks orchestrated by the federated training participants. We have argued both conceptually and empirically that random forests inherently fulfill the main desirable properties of explainable surrogate models, namely fidelity, accuracy, consistency, degree of importance, and comprehensibility.

TABLE 4.6: Differences between average feature importances and importances of attacked features for the model manager's mGAN-AI privacy attack on the Adult data set. Values in boldface correspond to the epoch in which the attack was detected.

	30 peers	50 peers	100 peers
Epoch 2	0.037	0.014	0.046
Epoch 3	0.006	0.006	0.017
Epoch 4	0.018	0.004	0.008
Epoch 5	0.0003	0.004	0.005
Epoch 6	0.0001	0.002	0.003
Epoch 7	0.013	0.07	0.019
Epoch 8	0.002	**0.02**	**0.023**
Epoch 9	**0.015**	**0.022**	0.003
Epoch 10	0.002	0.004	0.015
Reported features	*marital-status* *age*	*marital-status* *hours-per-week* *educational-num*	*relationship* *occupation*

In particular, we have focused on investigating and explaining wrong predictions. To this end, we provide decision trees that explain such wrong predictions and report the features' importance associated with those predictions. We have shown that this information can be leveraged to detect security and privacy attacks in machine learning, specifically in federated learning. Our method is able to discover a variety of attacks with high detection rates, and it clearly outperforms specially tailored attack detection mechanisms based on update statistics. Moreover, unlike the countermeasures proposed in the literature, our method cannot only detect the attack but can also explain it by identifying the features that seem to have been affected by it.

Chapter 5

Explaining Image Misclassification in Deep Learning via Adversarial Examples

Whereas data features are explicit in structured data (*e.g.*, tabular data), for other data types such as images or text, those features are implicit. Since black-box models, such as convolutional neural networks (CNN), extract those features by themselves, it is hard for a surrogate model to explain the behavior of the black box since it may learn to make the decision based on a different set of features. To this end, in this chapter we propose a model-specific method to explain the predictions of image classification black-box models trained in centralized settings.

5.1 Contributions and plan of this chapter

In this chapter, we present two methods that explain CNN-based image classification by identifying the most influential features in the CNN predictions. The first method assumes access to the gradients of the CNN and is meant for model developers. The second method treats the model as a black box and, therefore, assumes that the party generating the explanations, such as a model end user, only has access to the model predictions.

Both methods leverage adversarial examples (Nguyen, Yosinski, and Clune, 2015) to generate explanations. While the first method computes adversarial examples by directly employing the CNN gradients, the second approach builds a simpler surrogate model to estimate the gradients of the original model. It then uses these estimated gradients to obtain adversarial examples. More specifically, for each image that the CNN incorrectly classified, we implemented an inverted adversarial attack consisting in modifying the input image as little as possible so that it becomes correctly classified. The changes made to the image to fix classification errors highlight the regions that had the greatest influence in the decisions and thus *explain* the causes of model misclassification. By identifying the causes of wrong predictions, one may tailor the model or the training data to improve the classification accuracy.

The remainder of this chapter is organized as follows. Section 5.2 describes our methods for explaining CNN-based image misclassification from adversarial examples. Experimental results are reported in Section 5.3. Finally, in Section 5.4 we gather the conclusions.

5.2 Our proposals

We focus on generating explanations for the wrong predictions made by CNNs by identifying the regions of the input images that had the highest influence on those predictions. To this end, we need a way to modify the input images towards the correct classification while keeping the number of required queries and the computational cost at a minimum. Our choice is to use gradient-based adversarial examples (Szegedy et al., 2014). Specifically, we add minimal perturbations to incorrectly classified images to create correctly classified adversarial examples. Then, by comparing the original image with the modified image, we can find the regions that had the highest impact on the wrong black-box predictions.

5.2.1 Adversarial examples

An adversarial example is a sample from the same distribution as the original data in which small, intentional perturbations of its features cause an AI model to change its prediction (Molnar, 2020). Adversarial examples can be used to alter predictions of various machine learning models, including state-of-the-art neural networks (Szegedy et al., 2014). Even though adversarial examples are usually employed to cause the AI models to produce wrong predictions, in this chapter we use them the other way around: to correct wrongly classified samples.

To create adversarial examples, we used the gradient-based optimization approach of Szegedy et al., 2014, in which we set the target to be the correct label for the wrongly predicted samples.

The adversarial examples are created by minimizing the following function with respect to r:

$$loss(f(x+r), l) + \epsilon \cdot |r|, \tag{5.1}$$

where f is the AI classifier, x is the original image, r is the perturbation added to the pixels of x to create the perturbed image that constitutes the adversarial example, l is the target class label, and ϵ is used to balance the distance between images and the distance between predictions. The smaller ϵ, the more similar is the created perturbed image to the original image. To minimize the loss function in Equation (5.1), the party that computes it needs access to the model gradients.

5.2.2 Explaining model predictions on the developer's side

To explain the wrong predictions made by the model on the developer's side, we consider that the developer has full access to the CNN and, more specifically, to the gradients of the model.

As shown in Algorithm 5, first, the developer splits the input images into training and testing sets and trains the model with the training images. Then, in the testing phase, the developer keeps track of all the wrongly classified images. For each of these images, she tries to find the closest adversarial example that is correctly classified. The developer does this in the following way: i) she calculates the value of the loss function between the model prediction and the correct prediction; ii) she calculates the gradients of the model according to the image and the loss value; 3) she modifies the image according to the gradients and the perturbation ratio ϵ. These steps are repeated until the adversarial example

is obtained, or ϵ exceeds the α value signaling the termination condition (in the latter case, the image misclassification cannot be explained). The final step consists in comparing each original image with its corresponding adversarial example. To draw a saliency map that identifies the features that caused wrong predictions, Algorithm 6 prescribes that pixels in perturbations with values smaller than $q3 + iqr \cdot \tau$, where $q3$ is the third quartile of perturbations, iqr is their interquartile range, and $\tau > 0$ is the relaxation parameter, are neglected because they do not identify regions of interest, whereas the remaining pixels are multiplied by $\beta > 1$ to boost them in the saliency map.

Algorithm 5 Explaining the model predictions on the developer's side

input: Data set X, CNN model *model*
$Train_X, Test_X \leftarrow Split_Train_Test(X)$
$model \leftarrow Train_Model(Train_X)$
$perturbations \leftarrow \{\}$
for *i in Test_X* **do**
  $model_prediction \leftarrow model.predict(Test_X[i])$
  $\epsilon \leftarrow 0.1$
  while $model_prediction \neq correct_prediction$ OR $\epsilon < \alpha$ **do**
    $loss \leftarrow loss_function(model_prediction, correct_prediction)$
    $gradients \leftarrow get_model_gradients(Test_X[i], loss)$
    $perturbed_image \leftarrow Test_X[i] - \epsilon \cdot gradients$
    $model_prediction \leftarrow model.predict(perturbed_image)$
    $\epsilon \leftarrow \epsilon + 0.1$
  end while
  if $model.prediction = correct_prediction$ **then**
    $perturbations[i] \leftarrow perturbed_image - Test_X[i]$
  else
    $perturbations[i] \leftarrow NIL$
  end if
end for
return *perturbations*

5.2.3 Explaining model predictions on the user's side

End users should have the right to obtain explanations about predictions made by the AI models that concern them. However, for

Algorithm 6 Drawing the saliency maps

```
1: input: perturbations
2: q1, q3 = get_quartiles_of_non_NIL_perturbations(perturbations)
3: iqr ← q3 − q1
4: for i such that perturbations[i] ≠ NIL do
5:    for pixel in perturbations[i] do
6:       if perturbations[i][pixel] < q3 + iqr · τ then
7:          perturbations[i][pixel] ← 0
8:       else
9:          perturbations[i][pixel] ← perturbations[i][pixel] · β
10:      end if
11:   end for
12:   Draw(perturbations[i])
13: end for
```

end users, the model is a black box, and they only have access to the model predictions. Therefore, they must create their own local explanations.

In this chapter, we considered that the user who wants to generate explanations of an AI model must have enough data to train a simpler CNN model, a.k.a. a surrogate model. It is shown in Hinton, Vinyals, and Dean, 2015 that knowledge of one or more models can be compressed into another, less complex model, which allows us to estimate the gradations of the original model using a surrogate model.

The method we propose is formalized in Algorithm 7. First, the user splits her data into training and testing data sets. Then she builds a surrogate model by using the local training data set. Afterward, she uses the test data set to identify the wrong predictions to be explained. Finally, she generates the adversarial examples in a similar way as in Algorithm 5. The only difference is that

the gradients from the surrogate model will be used instead of the original model's.

Algorithm 7 Explaining the model predictions on the user's side

input: local data X_local , black-box model $black_box$, surrogate model $local_surrogate$
$Train_X_local, Test_X_local \leftarrow Split_Train_Test(X_local)$
$local_surrogate \leftarrow Train_Black_Box(Train_X_local)$
for i *in* $Test_X_local$ **do**
  $black_box_prediction \leftarrow black_box.predict(Test_X_local[i])$
  $\epsilon \leftarrow 0.1$
  while $black_box_prediction \neq correct_prediction$ OR $\epsilon < \alpha$ **do**
    $local_prediction \leftarrow local_surrogate.predict(Test_X_local[i])$
    $loss \leftarrow loss_function(local_prediction, correct_prediction)$
    $gradients \leftarrow get_surrogate_gradients(Test_X_local[i], loss)$
    $perturbed_image \leftarrow Test_X_local[i] - \epsilon \cdot gradients$
    $black_box_prediction \leftarrow black_box.predict(perturbed_image)$
    $\epsilon \leftarrow \epsilon + 0.1$
  end while
  if $black_box_prediction = correct_prediction$ **then**
    $perturbations[i] \leftarrow perturbed_image - Test_X_local[i]$
  else
    $perturbations[i] \leftarrow NIL$
  end if
end for
return $perturbations$

5.3 Experimental results

We tested the two proposed methods introduced above on the gender classification data set[1] from the Kaggle website. This data set consists of cropped RGB images of male and female faces. The training data set contains 23,200 female images and 23,800 male images. The validation data set contains 5,800 images in each class. The images are rectangular, but not all of them are of the same size. Thus, we first resized all the images to 100 x 100 pixels.

5.3.1 Explanations for the developer

To test Algorithm 5 we used the CNN shown in Figure 5.1 with four Conv blocks followed by six fully connected layers. Each Conv block contained two convolutional layers and a max-pooling layer. The output depths for Conv blocks were, respectively, 64, 128, 256, and 512. The numbers of nodes of fully connected layers were, respectively, 2048, 1024, 512, 128, 32, and 2. We trained the model for 20 epochs, with a batch size 64 and a learning rate 0.001. The test accuracy was 96.3%.

The number of misclassified images in the test data set was 301, for which our method created adversarial examples. As a result, 300 of the 301 adversarial examples were classified into the correct labels, corresponding to a success rate 99.67%, with an average of 1.96 queries per image. Figure 5.2 shows four examples of the explanations created by Algorithm 5, in the form of saliency maps highlighting the differences between original images and adversarial examples.

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

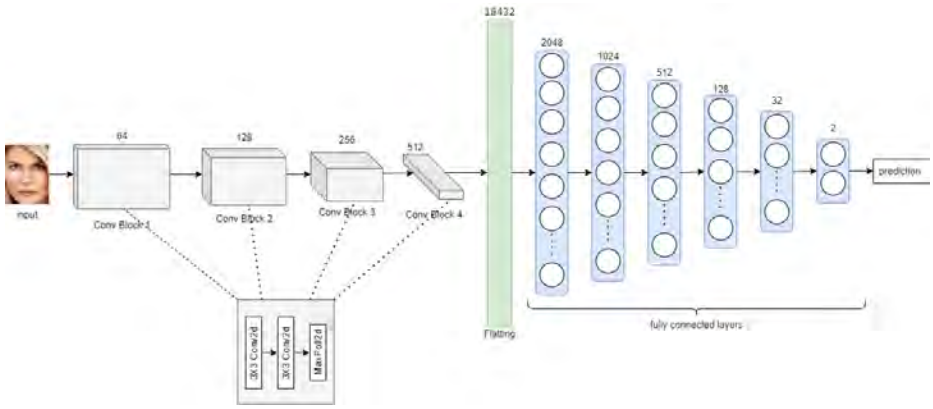


FIGURE 5.1: Architecture of the CNN used as original model in the experiments

In Figure 5.2, we can see that perturbations added to the original images to create the adversarial examples are not noticeable to the naked eye. Nevertheless, the explanations resulting from Algorithm 5 in the form of saliency maps shed light on the most important regions that caused the wrong predictions. For example, in Image 1, the important pixels were those around the eyes and the edge of the nose. In Image 2, the causes of the wrong classification were also found in the eyes, nose, and left cheek. In Image 3, the causes of the misclassification were mainly the left eye and the edge of the right eye, in addition to parts of the covered forehead. Finally, the most relevant regions for Image 4 were the left eye, the edge of the nose, and the left cheek.

5.3.2 Explanations for the user

Testing the performance of Algorithm 7 tells whether the user is capable of creating model explanations locally. We assumed

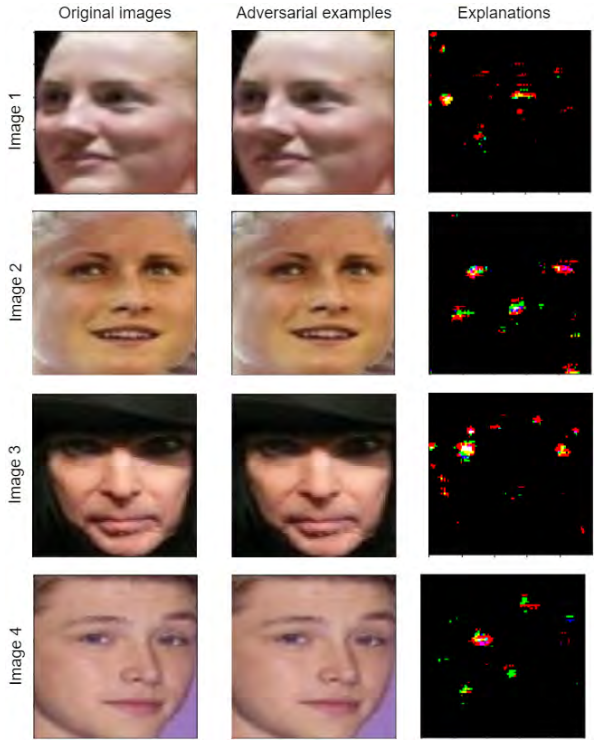


FIGURE 5.2: Four examples of explanations generated on the developer's side

the user's local data consisted of a random 10% sample of the training data described in the previous section. With these local data, the user trained her surrogate model. The black-box model was the same CNN described in the previous section, whereas the local surrogate model built by the user consisted of a CNN with three Conv blocks followed by four fully connected layers. Each Conv block contained two convolutional layers and a max-pooling layer. The output depths for Conv blocks were, respectively, 64, 256, and 512. The numbers of nodes of fully connected

layers were, respectively, 1024, 256, 32, and 2. We trained the model for 50 epochs, with a batch size 64 and a learning rate 0.001. The test accuracy for the surrogate model was 87.78%.

The complete test data set was used to generate explanations. Therefore, we got the same 301 misclassified images. Following Algorithm 7, explanations were obtained as follows: i) obtain the prediction and gradients of the local surrogate model; ii) create the adversarial example using the gradients of the surrogate model; iii) test whether the adversarial example was correctly predicted by using the original black-box model; iv) draw the saliency map. Out of the 301 adversarial examples, the original black box model correctly classified 220 images, corresponding to a 73.08% success rate. The average number of queries to the original CNN model per image required to create the adversarial example was 8.73.

Figure 5.3 shows the same four samples of Figure 5.2 but with saliency maps that were locally generated using the gradients of the surrogate model. As in the previous test, the differences between the adversarial examples and the original images are not noticeable to the naked eye. However, the most relevant regions of the images are similar to those obtained with Algorithm 5: in the four images, the same regions highlighted by Algorithm 5 are also highlighted here, even though in a less focused way due to the less accurate surrogate model.

The number of queries per image needed to create adversarial examples with Algorithm 5 was lower than with Algorithm 7: 1.96 vs 8.73. The reason is that the former algorithm uses the gradients from the original model, whereas the latter uses the surrogate model's gradients. Hence, the second algorithm's generation of adversarial examples is less accurate. However, both

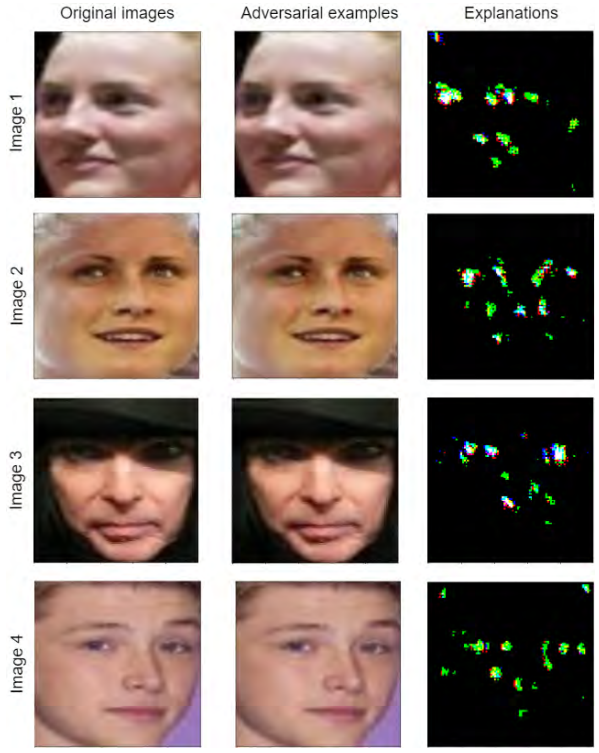


FIGURE 5.3: Four examples of explanations generated on the user's side

algorithms successfully generated explanations for the original model's wrong predictions, and both highlighted the same regions as important.

The explanations provided by our methods can help model developers to identify the weaknesses of the data sets used to train the model. Specifically, for the gender classification data set,

the eye regions are highlighted in most saliency maps as important regions. This suggests that classification accuracy may be improved by training on images where the eye region is clear. For model users, it is important to know which features influenced the predictions since some of the black boxes may be artificially biased or employ features that may discriminate against some minorities (Azad et al., 2021). Beyond face classification, an even more crucial application could be to help understand medical diagnoses (Singh et al., 2019).

5.4 Conclusions

We have presented two methods employing gradient-based adversarial examples to obtain explanations of the predictions of CNNs in image classification.

We have reduced the number of queries needed to create the adversarial examples by adding targeted perturbations to change the predictions for each image. In the experiments, developer-side Algorithm 5 required only 1.96 queries per image, whereas user-side Algorithm 7 needed 8.73 queries per image.

The two proposed algorithms showed promising results to explain misclassification by CNNs. Moreover, both produced similar explanations on the same samples. Algorithm 5 had a higher success rate (99.67%) thanks to using the gradients of the original model, whereas Algorithm 7 had a lower success rate (73.08%) due to using a surrogate.

Chapter 6

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

Building highly accurate DL classification models requires a large amount of training data, whose collection and labeling involve a significant effort (Jebreel et al., 2021). Therefore, small businesses and ordinary users that 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 (Ribeiro, Grolinger, and Capretz, 2015), as shown in Figure 6.1. These users then query those models with their (small) data and obtain the final classification predictions. In this chapter, 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. However, 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 (Chazette, Karras, and Schneider, 2019; Blanco-Justicia et al., 2020). 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 model's input and output. 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, no work enables an end user with only API access to a remote DL classification model to generate explanations using DL model-specific explanation methods locally.

6.1 Contributions and plan of this chapter

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

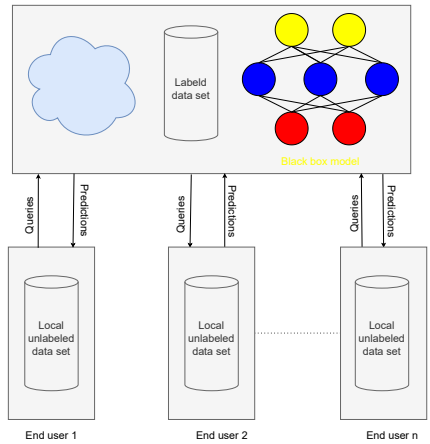


FIGURE 6.1: Architecture of the MLaaS

prior knowledge of the provider's model architecture or training hyper-parameters.

First, the (small) unlabeled data available to the end user is augmented (Simard et al., 1996) for it to be more representative of the input data distribution. Then, knowledge distillation (KD) (Hinton, Vinyals, and Dean, 2015) 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 (Nguyen, Yosinski, and Clune, 2015) 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 to the smallest number of features to generate understandable counterfactual examples (CE).

We demonstrate the accuracy and 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 gives end users an independent and reliable way to determine if the provider's predictions and explanations are trustworthy.

The remainder of this chapter is organized as follows. Section 6.2 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 6.3 describes our method for generating DL model-specific explanations at the end user's side. Section 6.4 details the experimental setup and reports and discusses empirical results. Finally, in Section 6.5 we gather the conclusions.

6.2 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 (Simard et al., 1996). 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 (Zhang et al., 2018). 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 (Hinton, Vinyals, and Dean, 2015; Chen et al., 2017a) 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) (Hinton, Vinyals, and Dean, 2015), feature maps of the provider's model intermediate layers (Wang et al., 2020) or the relationships between different layers or data samples (Gou et al., 2021). 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 (Shen et al., 2021).

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 hyperparameters. 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.

6.3 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 (Hinton, Vinyals, and Dean, 2015). 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.

6.3.1 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 (Zhang et al., 2018) to augment D_u and obtain more representative training data. Mixup constructs a virtual input example

$$\hat{x} = \lambda x^i + (1 - \lambda)x^j, \tag{6.1}$$

$$\hat{y} = \lambda y^i + (1 - \lambda)y^j, \tag{6.2}$$

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 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

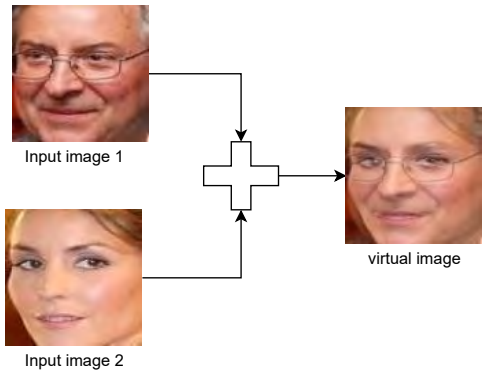


FIGURE 6.2: Virtual input image generated by mixing two original input images with $\lambda = 0.5$

data that help train more robust and accurate models (Zhang et al., 2018; Cubuk et al., 2018).

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. 6.2 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

$$m_u + \binom{m_u}{2} \tag{6.3}$$

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.

Algorithm 8 describes the methodology we use to augment the small unlabeled data set and then label it using the provider's model f_p.

Algorithm 8 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 train 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 (Srivastava et al., 2021). 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. First, 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 (Snoek, Larochelle, and Adams, 2012) 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 with 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[1] or Microsoft AutoML[2]. These platforms have advanced optimization algorithms determining 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.

6.3.2 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 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 this thesis, we explain the provider's model by generating counterfactual explanations (Wachter, Mittelstadt, and Russell, 2017; Mothilal, Sharma, and Tan, 2020) 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 (Molnar, 2020;

[1] `https://cloud.google.com/automl`

[2] `https://www.microsoft.com/en-us/research/project/automl/`

Jeyakumar et al., 2020). We use adversarial training (Szegedy et al., 2014; Goodfellow, Shlens, and Szegedy, 2015) described in Section 5.2.1 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 (Goodfellow, Shlens, and Szegedy, 2015). 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 (Molnar, 2020).

We use the following expression to generate adversarial examples:

$$x_* = x - \epsilon \cdot \frac{\partial}{\partial x}\mathcal{L}(f(x), y_*), \tag{6.4}$$

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 (6.4) 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 (6.4) 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 9 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 and surrogate models 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,, 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 (6.4) 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 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_*.

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 (Molnar, 2020): instead of overwhelming the user by changing many attributes, 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

Algorithm 9 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

choose the c attributes, we start by computing the loss gradient 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 (Molnar, 2020; Mothilal, Sharma,

and Tan, 2020).

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 (Slack et al., 2021), some authors (Artelt et al., 2021) propose that plausible tabular CEs can be used instead of closest CEs to improve the robustness and consequently the individual fairness of counterfactual explanations.

Algorithm 10 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

6.4 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 (Ribeiro, Singh, and Guestrin, 2016)), which shows that our explanations are more accurate and understandable.

Our code is available for reproducibility purposes[3].

6.4.1 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.

Data sets and provider models.

We evaluated the proposed approach on three data sets:

- **Gender** described in Section 5.3. The provider's model was the deep CNN used in a previous Chapter 5. The model was trained for 10 epochs with a batch size 64, the binary cross-entropy loss function, and the Adam optimizer (Kingma and Ba, 2015) with a learning rate 0.001.

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

- **MNIST** contains 70K handwritten images corresponding to digits 0 to 9 (LeCun et al., 1999). The images are divided into a training set (60K examples) and a validation set (10K examples). We used the LeNet model (LeCun et al., 2015) 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.

- **Adult** described in section 3.3.3 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. These data were augmented following Algorithm 8. 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 6.1 reports the number of samples the user owns before and after augmenting the data.

We used four surrogate models with different architectures for the Gender and the MNIST data sets. Surrogate 1 was shallower

TABLE 6.1: 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

TABLE 6.2: 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 0, and a padding 1; $MP(2, 2)$ denotes a max-pooling layer with a kernel of size 2×2 and a stride 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

than the provider's model, Surrogate 2 had the same architecture, and Surrogates 3 and 4 were larger than the provider's model.

Tables 6.2 and 6.3 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: the 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 tasks. Table 6.4 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 (Snoek, Larochelle, and Adams, 2012) 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.

TABLE 6.3: 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 (LeCun et al., 2015)	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 (LeCun et al., 2015)	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 6.4: 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

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

  $$c_1 = (K_1 L)^2, c_2 = (K_2 L)^2,$$

  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.

6.4.2 Results and discussion

Accuracy of surrogate models

Table 6.5 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 for 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 (Hinton, Vinyals, and Dean, 2015).

TABLE 6.5: 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%	-	-	-	-	-	-

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 numerically measure their similarity. Table 6.6 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 6.6: Similarity between the adversarial examples generated by the surrogate models and those 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 9. We then overlaid them as heat maps on the original images. Figure 6.3a 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 of 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 of male.

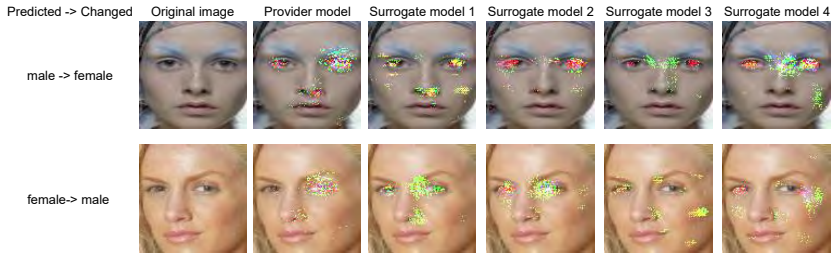
By looking at the pixels that caused the prediction to change, we can see that, in general, the explanations generated by the surrogates were consistent with those generated by the provider's model: All the models, including the provider's, identified the pixels corresponding to the regions of the eyes, 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 (Brown and Perrett, 1993; Fellous, 1997), 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 6.3b shows two examples of the explanations generated for MNIST.

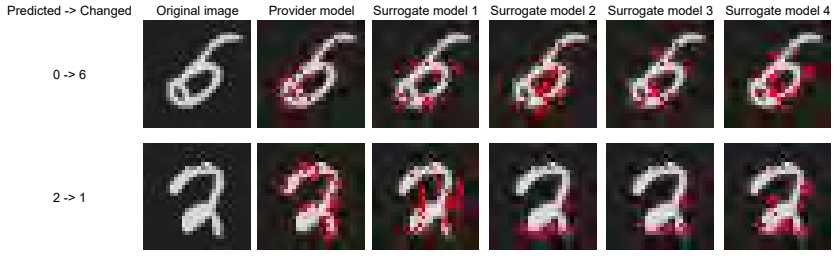
In the first example (row 1), the provider's model wrongly predicted the original image as 0, and the added pixel perturbations changed its prediction to the true prediction of 6. In the example (row 2), the provider's model correctly predicted the original image as 2, and the added pixel perturbations changed its prediction

to the wrong prediction of 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 (Liu et al., 2003).

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

FIGURE 6.3: 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 10 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 10 with $c \in \{1, \ldots, 12\}$. Table 6.7 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 level of education, 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, aside 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.

TABLE 6.7: 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**	-	$\leq 50K$

Comparison with LIME

To illustrate the advantages of our DL model-specific method over model-agnostic methods, we compared it with LIME (Ribeiro, Singh, and Guestrin, 2016) in terms of runtime and quality of the explanations.

Table 6.8 reports the time required to train the surrogates, to generate the explanation 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

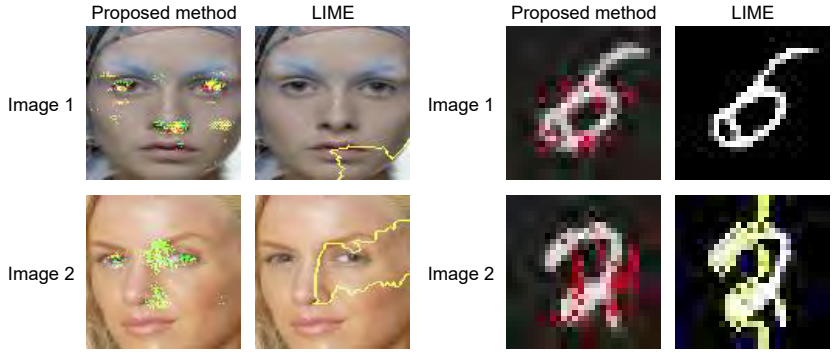
predictions $y_i = f(x_i)$. Finally, it builds a local linear interpretable SVM model described in Section 2.1.1 in fitting the y_is to the presence or absence of superpixels. Since each coefficient of the built model is associated with a superpixel of the image, the more positive the coefficient is, 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.

TABLE 6.8: 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

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 6.4a 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 (Brown and Perrett, 1993; Fellous, 1997). 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 Figure 6.4b.



(A) Two examples from the Gender data set (B) Two examples from the MNIST data set

FIGURE 6.4: Comparison of the explanations generated by the proposed method with those generated by LIME

6.5 Conclusions

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 a 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 could 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.

Chapter 7

Conclusion and Future Work

This thesis has focused on explaining the predictions of black-box DL models, and on using these explanations to protect the models. First, we exploited random forests as surrogates to explain the prediction of the centralized DL black-box models in Chapter 3. The explanations provided by the forests enabled us to detect if the data were attacked or wrongly clustered, which might harm the model. Even though the explanations were understandable and helped with the attack detection, the surrogate model could only be built by the model owner.

We also tested random forests' surrogates in a federated learning environment in Chapter 4, by which participants in the training process could build the surrogate model on their own using only their local data. The generated explanations provided a numerical importance to each attribute according to its influence on the prediction. Also, the trees in the random forest surrogate provided further understanding on the reasoning of the black-box

predictions. At the same time, the surrogate model was able to detect security and privacy attacks that might target the training of the federated learning model. Nevertheless, the surrogate model was able to explain only the predictions of the black-box models that take structured data as input, such as tabular data.

In Chapter 5, we proposed a more general method to obtain explanations regardless of the input data type. This model-specific explanation method is based on the concept of counterfactual examples using adversarial examples. Therefore, it can provide explanations when using it on the model owner's side. Additionally, end users can use it to obtain explanations by leveraging a local surrogate model that mimics the behavior and performance of the original black-box model. The drawback is that the end user should hold around 10% of the original training data to train the surrogate model.

To overcome the limitation of our previous method on the data availability, in Chapter 6, we used the Mixup data augmentation method, which allowed end users to train their surrogates by using only a 0.5% of the original data.

7.1 Contributions and publications

Chapter 3 focuses on explaining the centralized DL predictions by using random forests as a surrogates, and by detecting if the training data were corrupted or wrongly labeled. This work resulted in the following publication:

- Rami Haffar, Josep Domingo-Ferrer, and David Sánchez. "Explaining misclassification and attacks in deep learning via

random forests." In *Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence – MDAI 2020*, pp. 273-285. Springer, 2020. CORE ranking: B.

Chapter 4 presents our work on explaining predictions of federated learning models on the participant's side. In addition, the explanation provided by the surrogates was used to detect security and privacy attacks. This work resulted in the following publication:

- Rami Haffar, David Sánchez, and Josep Domingo-Ferrer. "Explaining predictions and attacks in federated learning via random forests." In *Applied Intelligence* (2022): 1-17. Impact Factor: 5.086 (2nd quartile).

Chapter 5 discusses the use of counterfactual examples to create explanations for image classification. First, the counterfactual examples were created by adding small targeted perturbations to alter the model predictions. Then, by measuring the difference between the original data sample and the counterfactual example, we were able to identify the regions with the largest influence on the model prediction. This work resulted in the following publication:

- Rami Haffar, Najeeb Jebreel, Josep Domingo-Ferrer, and David Sánchez. "Explaining Image Misclassification in Deep Learning via Adversarial Examples." In *Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence – MDAI 2021*, pp. 323-334. Springer, 2021. CORE ranking: B.

Chapter 6 focuses on user-end explanations for any model accessible via MLaaS. We used adversarial examples to create counterfactual examples to explain the important features in the predictions. This work resulted in the following publication:

- Rami Haffar, Najeeb Jebreel, David Sánchez,and Josep Domingo-Ferrer. "Generating Deep Learning Model-Specific Explanations at the End User's Side." In *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems.* 30(Supplement-2): 255-278 (2022) Impact Factor: 1.518 (3rd quartile).

7.2 Future work

This thesis opens new ground for research on the following lines:

- Regarding the use of random forests as surrogates (Chapters 3 and 4) we plan to:
  - Test the performance of the random forest surrogates in detecting other types of attacks initiated by any participant in decentralized training, such as multi-feature random attacks (Liu et al., 2020) and backdoor attacks (Yao et al., 2019).
  - Try to satisfy more properties among those described in Section 2.2.1, such as certainty –by testing the proposed method on a black-box model that outputs a degree of certainty alongside its final predictions–, and novelty –by testing the explanations of the surrogate model on data from the same domain of the training data set but with different distributions of features–.

  - Since the proposed surrogate model is trained with a partition of the data held by one of the participants, we would like to test the performance of our approach on data that are not identically and independently distributed (non-iid) (Zhao et al., 2018).
  - Apply the proposed methods to fully decentralized machine learning scenarios (Lian et al., 2017), where there is no single model manager that aggregates the updates.

- Regarding the use of counterfactual examples in Chapters 5 and 6 we plan to:
  - Test the performance of the proposed approaches in explaining the predictions of black-box models trained on non-iid data sets. Also, evaluate them when the end user holds non-iid data to be used to train the local surrogate model.
  - Test the performance of our approaches on other computer vision tasks, such as object detection and image segmentation, as well as natural language processing tasks (Petsiuk et al., 2021; Liu, Yin, and Wang, 2019).
  - Experiment with different methodologies to create CEs, such as plausible CE explanations (Artelt et al., 2021) and CE creation by generative adversarial networks (GANs) (Creswell et al., 2018), in order to achieve more representative explanations on the end user side.

- Test the use of the explanations provided by the proposed method to identify biases in the model w.r.t. discriminatory features. This could satisfy the counterfactual fairness (individual fairness) of deep learning models (Kusner et al., 2017).

Bibliography

Adankon, Mathias M. and Mohamed Cheriet (2009). "Support Vector Machine". In: *Encyclopedia of Biometrics*. Ed. by Stan Z. Li and Anil Jain. Boston, MA: Springer US, pp. 1303–1308. ISBN: 978-0-387-73003-5. DOI: 10.1007/978-0-387-73003-5_299. URL: https://doi.org/10.1007/978-0-387-73003-5_299.

Agarap, Abien Fred (2018). "Deep learning using rectified linear units (relu)". In: *arXiv preprint arXiv:1803.08375*.

Alvarez-Melis, David and Tommi S. Jaakkola (2018). "On the Robustness of Interpretability Methods". In: *CoRR* abs/1806.08049. arXiv: 1806.08049. URL: http://arxiv.org/abs/1806.08049.

Artelt, André, Valerie Vaquet, Riza Velioglu, Fabian Hinder, Johannes Brinkrolf, Malte Schilling, and Barbara Hammer (2021). "Evaluating Robustness of Counterfactual Explanations". In: *IEEE Symposium Series on Computational Intelligence, SSCI 2021, Orlando, FL, USA, December 5-7, 2021*. IEEE, pp. 1–9. DOI: 10.1109/SSCI50451.2021.9660058. URL: https://doi.org/10.1109/SSCI50451.2021.9660058.

Azad, Reza, Abdur R. Fayjie, Claude Kauffmann, Ismail Ben Ayed, Marco Pedersoli, and Jose Dolz (2021). "On the Texture Bias for Few-Shot CNN Segmentation". In: *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*. IEEE, pp. 2673–2682. DOI: 10.1109/WACV486

30.2021.00272. URL: https://doi.org/10.1109/WACV48630.2021.00272.

Bach, Sebastian, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek (2015). "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS One* 10.7, e0130140.

Bini, Stefano A (2018). "Artificial intelligence, machine learning, deep learning, and cognitive computing: what do these terms mean and how will they impact health care?" In: *The Journal of arthroplasty* 33.8, pp. 2358–2361.

Blanchard, Peva, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer (2017). "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 119–129. URL: https://proceedings.neurips.cc/paper/2017/hash/f4b9ec30ad9f68f89b29639786cb62ef-Abstract.html.

Blanco-Justicia, Alberto, Josep Domingo-Ferrer, Sergio Martínez, and David Sánchez (2020). "Machine learning explainability via microaggregation and shallow decision trees". In: *Knowl. Based Syst.* 194, p. 105532. DOI: 10.1016/j.knosys.2020.105532. URL: https://doi.org/10.1016/j.knosys.2020.105532.

Blanco-Justicia, Alberto, Josep Domingo-Ferrer, Sergio Martínez, David Sánchez, Adrian Flanagan, and Kuan Eeik Tan (2021). "Achieving security and privacy in federated learning systems: Survey, research challenges and future directions". In: *Eng. Appl.*

Artif. Intell. 106, p. 104468. DOI: 10.1016/j.engappai.2021.104468. URL: https://doi.org/10.1016/j.engappai.2021.104468.

Blanco-Justicia, Alberto, David Sánchez, Josep Domingo-Ferrer, and Krishnamurty Muralidhar (2022). "A Critical Review on the Use (and Misuse) of Differential Privacy in Machine Learning". In: *CoRR* abs/2206.04621. DOI: 10.48550/arXiv.2206.04621. arXiv: 2206.04621. URL: https://doi.org/10.48550/arXiv.2206.04621.

Bonawitz, Kallista A., Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth (2017). "Practical Secure Aggregation for Privacy-Preserving Machine Learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, pp. 1175–1191. DOI: 10.1145/3133956.3133982. URL: https://doi.org/10.1145/3133956.3133982.

Brown, Elizabeth and David I Perrett (1993). "What gives a face its gender?" In: *Perception* 22.7, pp. 829–840.

Cao, Di, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun (2019). "Understanding Distributed Poisoning Attack in Federated Learning". In: *25th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2019, Tianjin, China, December 4-6, 2019*. IEEE, pp. 233–239. DOI: 10.1109/ICPADS47876.2019.00042. URL: https://doi.org/10.1109/ICPADS47876.2019.00042.

Chattopadhyay, Aditya, Anirban Sarkar, Prantik Howlader, and Vineeth N. Balasubramanian (2018). "Grad-CAM++: Generalized Gradient-Based Visual Explanations for Deep Convolutional Networks". In: *2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018*. IEEE Computer Society, pp. 839–847. DOI: 10.1109/WACV.2018.00097. URL: https://doi.org/10.1109/WACV.2018.00097.

Chazette, Larissa, Oliver Karras, and Kurt Schneider (2019). "Do End-Users Want Explanations? Analyzing the Role of Explainability as an Emerging Aspect of Non-Functional Requirements". In: *27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019*. Ed. by Daniela E. Damian, Anna Perini, and Seok-Won Lee. IEEE, pp. 223–233. DOI: 10.1109/RE.2019.00032. URL: https://doi.org/10.1109/RE.2019.00032.

Chen, Guobin, Wongun Choi, Xiang Yu, Tony X. Han, and Manmohan Chandraker (2017a). "Learning Efficient Object Detection Models with Knowledge Distillation". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 742–751. URL: https://proceedings.neurips.cc/paper/2017/hash/e1e32e235eee1f970470a3a6658dfdd5-Abstract.html.

Chen, Mingzhe, Ursula Challita, Walid Saad, Changchuan Yin, and Mérouane Debbah (2017b). "Machine learning for wireless networks with artificial intelligence: A tutorial on neural networks". In: *arXiv preprint arXiv:1710.02913* 9.

Chen, Yu, Fang Luo, Tong Li, Tao Xiang, Zheli Liu, and Jin Li (2020). "A training-integrity privacy-preserving federated learning scheme with trusted execution environment". In: *Inf. Sci.* 522, pp. 69–79. DOI: 10.1016/j.ins.2020.02.037. URL: https://doi.org/10.1016/j.ins.2020.02.037.

Creswell, Antonia, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A. Bharath (2018). "Generative Adversarial Networks: An Overview". In: *IEEE Signal Process. Mag.* 35.1, pp. 53–65. DOI: 10.1109/MSP.2017.2765202. URL: https://doi.org/10.1109/MSP.2017.2765202.

Cubuk, Ekin Dogus, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le (2018). "AutoAugment: Learning Augmentation Policies from Data". In: *CoRR* abs/1805.09501. arXiv: 1805.09501. URL: http://arxiv.org/abs/1805.09501.

Danks, David and Alex John London (2017). "Regulating autonomous systems: Beyond standards". In: *IEEE Intelligent Systems* 32.1, pp. 88–91.

Deng, Li and Dong Yu (2014). "Deep Learning: Methods and Applications". In: *Found. Trends Signal Process.* 7.3-4, pp. 197–387. DOI: 10.1561/2000000039. URL: https://doi.org/10.1561/2000000039.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805*.

Domingo-Ferrer, Josep, Alberto Blanco-Justicia, David Sánchez, and Najeeb Jebreel (2020). "Co-Utile Peer-to-Peer Decentralized Computing". In: *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGRID 2020, Melbourne, Australia, May 11-14, 2020*. IEEE, pp. 31–40. DOI: 10.1109/CCGrid49817.2020.00-90. URL: https://doi.org/10.1109/CCGrid49817.2020.00-90.

Domingo-Ferrer, Josep, David Sánchez, and Alberto Blanco-Justicia (2021). "The limits of differential privacy (and its misuse in data release and machine learning)". In: *Commun. ACM* 64.7, pp. 33–35. DOI: 10.1145/3433638. URL: https://doi.org/10.1145/3433638.

European Commission's High-Level Expert Group on Artificial Intelligence (2019). *Ethics Guidelines for Trustworthy AI*. URL: https://ec.europa.eu/futurium/en/ai-alliance-consultation.

Fang, Minghong, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong (2020). "Local Model Poisoning Attacks to Byzantine-Robust Federated Learning". In: *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*. Ed. by Srdjan Capkun and Franziska Roesner. USENIX Association, pp. 1605–1622. URL: https://www.usenix.org/conference/usenixsecurity20/presentation/fang.

Fellous, Jean-Marc (1997). "Gender discrimination and prediction on the basis of facial metric information". In: *Vision Research* 37.14, pp. 1961–1973.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio (2020). "Generative adversarial networks". In:

Commun. ACM 63.11, pp. 139–144. DOI: 10.1145/3422622. URL: https://doi.org/10.1145/3422622.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy (2015). "Explaining and Harnessing Adversarial Examples". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6572.

Gou, Jianping, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao (2021). "Knowledge Distillation: A Survey". In: *Int. J. Comput. Vis.* 129.6, pp. 1789–1819. DOI: 10.1007/s11263-021-01453-z. URL: https://doi.org/10.1007/s11263-021-01453-z.

Guidotti, Riccardo, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi (2019). "A Survey of Methods for Explaining Black Box Models". In: *ACM Comput. Surv.* 51.5, 93:1–93:42. DOI: 10.1145/3236009. URL: https://doi.org/10.1145/3236009.

Gulli, Antonio and Sujit Pal (2017). *Deep learning with Keras*. Packt Publishing Ltd.

Gunn, Steve R et al. (1998). "Support vector machines for classification and regression". In: *ISIS technical report* 14.1, pp. 5–16.

Haeberle, Heather S, James M Helm, Sergio M Navarro, Jaret M Karnuta, Jonathan L Schaffer, John J Callaghan, Michael A Mont, Atul F Kamath, Viktor E Krebs, and Prem N Ramkumar (2019). "Artificial intelligence and machine learning in lower extremity arthroplasty: a review". In: *The Journal of arthroplasty* 34.10, pp. 2201–2203.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778.

Helm, J Matthew, Andrew M Swiergosz, Heather S Haeberle, Jaret M Karnuta, Jonathan L Schaffer, Viktor E Krebs, Andrew I Spitzer, and Prem N Ramkumar (2020). "Machine learning and artificial intelligence: definitions, applications, and future directions". In: *Current reviews in musculoskeletal medicine* 13.1, pp. 69–76.

Hinton, Geoffrey E., Oriol Vinyals, and Jeffrey Dean (2015). "Distilling the Knowledge in a Neural Network". In: *CoRR* abs/1503.02531. arXiv: 1503.02531. URL: http://arxiv.org/abs/1503.02531.

Hitaj, Briland, Giuseppe Ateniese, and Fernando Pérez-Cruz (2017). "Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*. Ed. by Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu. ACM, pp. 603–618. DOI: 10.1145/3133956.3134012. URL: https://doi.org/10.1145/3133956.3134012.

Ho, Tin Kam (1995). "Random decision forests". In: *Third International Conference on Document Analysis and Recognition, ICDAR 1995, August 14 - 15, 1995, Montreal, Canada. Volume I*. IEEE Computer Society, pp. 278–282. DOI: 10.1109/ICDAR.1995.598994. URL: https://doi.org/10.1109/ICDAR.1995.598994.

Janiesch, Christian, Patrick Zschech, and Kai Heinrich (2021). "Machine learning and deep learning". In: *Electronic Markets* 31.3, pp. 685–695.

Jebreel, Najeeb, Alberto Blanco-Justicia, David Sánchez, and Josep Domingo-Ferrer (2020). "Efficient Detection of Byzantine Attacks in Federated Learning Using Last Layer Biases". In: *Modeling Decisions for Artificial Intelligence - 17th International Conference, MDAI 2020, Sant Cugat, Spain, September 2-4, 2020, Proceedings*. Ed. by Vicenç Torra, Yasuo Narukawa, Jordi Nin, and Núria Agell. Vol. 12256. Lecture Notes in Computer Science. Springer, pp. 154–165. DOI: 10.1007/978-3-030-57524-3_13. URL: https://doi.org/10.1007/978-3-030-57524-3_13.

Jebreel, Najeeb Moharram, Josep Domingo-Ferrer, David Sánchez, and Alberto Blanco-Justicia (2021). "KeyNet: An Asymmetric Key-Style Framework for Watermarking Deep Learning Models". In: *Applied Sciences* 11.3, p. 999.

Jeyakumar, Jeya Vikranth, Joseph Noor, Yu-Hsi Cheng, Luis Garcia, and Mani B. Srivastava (2020). "How Can I Explain This to You? An Empirical Study of Deep Neural Network Explanation Methods". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/2c29d89cc56cdb191c60db2f0bae796b-Abstract.html.

Joshua, Kroll, Huey Joanna, Barocas Solon, Felten Edward, Reidenberg Joel, Robinson David, and Yu Harlan (2017). "Accountable Algorithms". In: *University of Pennsylvania Law Review* 165, p. 633.

Kairouz, Peter, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz,

Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao (2021). "Advances and Open Problems in Federated Learning". In: *Found. Trends Mach. Learn.* 14.1-2, pp. 1–210. DOI: 10.1561/2200000083. URL: https://doi.org/10.1561/2200000083.

Karlik, Bekir and A Vehbi Olgac (2011). "Performance analysis of various activation functions in generalized MLP architectures of neural networks". In: *International Journal of Artificial Intelligence and Expert Systems* 1.4, pp. 111–122.

Ke, Qiuhong, Jun Liu, Mohammed Bennamoun, Senjian An, Ferdous Sohel, and Farid Boussaid (2018). "Computer vision for human–machine interaction". In: *Computer Vision for Assistive Healthcare*. Elsevier, pp. 127–145.

Kim, Hyesung, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim (2020). "Blockchained On-Device Federated Learning". In: *IEEE Commun. Lett.* 24.6, pp. 1279–1283. DOI: 10.1109/LCOMM.2019

.2921755. URL: https://doi.org/10.1109/LCOMM.2019.2921755.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1412.6980.

Kingston, John K. C. (2016). "Artificial Intelligence and Legal Liability". In: *Research and Development in Intelligent Systems XXXIII - Incorporating Applications and Innovations in Intelligent Systems XXIV. Proceedings of AI-2016, The Thirty-Sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, December 13-15, 2016*. Ed. by Max Bramer and Miltos Petridis. Springer, pp. 269–279. DOI: 10.1007/978-3-319-47175-4_20. URL: https://doi.org/10.1007/978-3-319-47175-4_20.

Konečný, Jakub, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon (2016). "Federated Learning: Strategies for Improving Communication Efficiency". In: *CoRR* abs/1610.05492. arXiv: 1610.05492. URL: http://arxiv.org/abs/1610.05492.

Kusner, Matt J., Joshua R. Loftus, Chris Russell, and Ricardo Silva (2017). "Counterfactual Fairness". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 4066–4076. URL: https:

//proceedings.neurips.cc/paper/2017/hash/a486cd07e4ac3d270571622f4f316ec5-Abstract.html.

Lamport, Leslie, Robert E. Shostak, and Marshall C. Pease (2019). "The Byzantine generals problem". In: *Concurrency: the Works of Leslie Lamport*. Ed. by Dahlia Malkhi. ACM, pp. 203–226. DOI: 10.1145/3335772.3335936. URL: https://doi.org/10.1145/3335772.3335936.

Lash, Michael T., Qihang Lin, W. Nick Street, Jennifer G. Robinson, and Jeffrey W. Ohlmann (2017). "Generalized Inverse Classification". In: *Proceedings of the 2017 SIAM International Conference on Data Mining, Houston, Texas, USA, April 27-29, 2017*. Ed. by Nitesh V. Chawla and Wei Wang. SIAM, pp. 162–170. DOI: 10.1137/1.9781611974973.19. URL: https://doi.org/10.1137/1.9781611974973.19.

Laugel, Thibault, Marie-Jeanne Lesot, Christophe Marsala, Xavier Renard, and Marcin Detyniecki (2018). "Comparison-Based Inverse Classification for Interpretability in Machine Learning". In: *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations - 17th International Conference, IPMU 2018, Cádiz, Spain, June 11-15, 2018, Proceedings, Part I*. Ed. by Jesús Medina, Manuel Ojeda-Aciego, José Luis Verdegay Galdeano, David A. Pelta, Inma P. Cabrera, Bernadette Bouchon-Meunier, and Ronald R. Yager. Vol. 853. Communications in Computer and Information Science. Springer, pp. 100–111. DOI: 10.1007/978-3-319-91473-2_9. URL: https://doi.org/10.1007/978-3-319-91473-2_9.

LeCun, Yann, Yoshua Bengio, and Geoffrey E. Hinton (2015). "Deep learning". In: *Nat.* 521.7553, pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

LeCun, Yann, Patrick Haffner, Léon Bottou, and Yoshua Bengio (1999). "Object Recognition with Gradient-Based Learning". In: *Shape, Contour and Grouping in Computer Vision*. Ed. by David A. Forsyth, Joseph L. Mundy, Vito Di Gesù, and Roberto Cipolla. Vol. 1681. Lecture Notes in Computer Science. Springer, p. 319. DOI: 10.1007/3-540-46805-6_19. URL: https://doi.org/10.1007/3-540-46805-6_19.

LeCun, Yann et al. (2015). "LeNet-5, convolutional neural networks". In: *URL: http://yann. lecun. com/exdb/lenet* 20.5, p. 14.

Lian, Xiangru, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu (2017). "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 5330–5340. URL: https://proceedings.neurips.cc/paper/2017/hash/f75526659f31040afeb61cb7133e4e6d-Abstract.html.

Liu, Cheng-Lin, Kazuki Nakashima, Hiroshi Sako, and Hiromichi Fujisawa (2003). "Handwritten digit recognition: benchmarking of state-of-the-art techniques". In: *Pattern Recognit.* 36.10, pp. 2271–2285. DOI: 10.1016/S0031-3203(03)00085-2. URL: https://doi.org/10.1016/S0031-3203(03)00085-2.

Liu, Hui, Qingyu Yin, and William Yang Wang (2019). "Towards Explainable NLP: A Generative Explanation Framework for Text Classification". In: *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy,*

July 28- August 2, 2019, Volume 1: Long Papers. Ed. by Anna Korhonen, David R. Traum, and Lluís Màrquez. Association for Computational Linguistics, pp. 5570–5581. DOI: 10.18653/v1/p19-1560. URL: https://doi.org/10.18653/v1/p19-1560.

Liu, Liang, Huaiyuan Wang, Zhijun Wu, and Meng Yue (2020). "The detection method of low-rate DoS attack based on multi-feature fusion". In: *Digit. Commun. Networks* 6.4, pp. 504–513. DOI: 10.1016/j.dcan.2020.04.002. URL: https://doi.org/10.1016/j.dcan.2020.04.002.

Ljung, Lennart (2001). "Black-box models from input-output measurements". In: *IMTC 2001. Proceedings of the 18th IEEE instrumentation and measurement technology conference. Rediscovering measurement in the age of informatics (Cat. No. 01CH 37188)*. Vol. 1. IEEE, pp. 138–146.

Lundberg, Scott M. and Su-In Lee (2017). "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, pp. 4765–4774. URL: https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.

Mahajan, Divyat, Chenhao Tan, and Amit Sharma (2019). "Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers". In: *CoRR* abs/1912.03277. arXiv: 1912.03277. URL: http://arxiv.org/abs/1912.03277.

McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas (2017). "Communication-Efficient

Learning of Deep Networks from Decentralized Data". In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*. Ed. by Aarti Singh and Xiaojin (Jerry) Zhu. Vol. 54. Proceedings of Machine Learning Research. PMLR, pp. 1273–1282. URL: http://proceedings.mlr.press/v54/mcmahan17a.html.

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

Mothilal, Ramaravind Kommiya, Amit Sharma, and Chenhao Tan (2020). "Explaining machine learning classifiers through diverse counterfactual explanations". In: *FAT* '20: Conference on Fairness, Accountability, and Transparency, Barcelona, Spain, January 27-30, 2020*. Ed. by Mireille Hildebrandt, Carlos Castillo, L. Elisa Celis, Salvatore Ruggieri, Linnet Taylor, and Gabriela Zanfir-Fortuna. ACM, pp. 607–617. DOI: 10.1145/3351095.3372850. URL: https://doi.org/10.1145/3351095.3372850.

Naylor, C David (2018). "On the prospects for a (deep) learning health care system". In: *Jama* 320.11, pp. 1099–1100.

Nguyen, Anh Mai, Jason Yosinski, and Jeff Clune (2015). "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, pp. 427–436. DOI: 10.1109/CVPR.2015.7298640. URL: https://doi.org/10.1109/CVPR.2015.7298640.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher,

Matthieu Perrot, and Edouard Duchesnay (2012). "Scikit-learn: Machine Learning in Python". In: *CoRR* abs/1201.0490. arXiv: 1201.0490. URL: http://arxiv.org/abs/1201.0490.

Petsiuk, Vitali, Rajiv Jain, Varun Manjunatha, Vlad I. Morariu, Ashutosh Mehra, Vicente Ordonez, and Kate Saenko (2021). "Black-Box Explanation of Object Detectors via Saliency Maps". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, pp. 11443–11452. DOI: 10.1109/CVPR46437.2021.01128. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Petsiuk_Black-Box_Explanation_of_Object_Detectors_via_Saliency_Maps_CVPR_2021_paper.html.

Pratiwi, Heny, Agus Perdana Windarto, S Susliansyah, Ririn Restu Aria, Susi Susilowati, Luci Kanti Rahayu, Yuni Fitriani, Agustiena Merdekawati, and Indra Riyana Rahadjeng (2020). "Sigmoid activation function in selecting the best model of artificial neural networks". In: *Journal of Physics: Conference Series*. Vol. 1471. 1. IOP Publishing, p. 012010.

Proposal for a Regulation of the European Parliament and of the Council laying down harmonised rules on artificial intelligence (Artificial Intelligence Act) and amending certain Union legislative acts (2021). URL: https://digital-strategy.ec.europa.eu/en/library/proposal-regulation-laying-down-harmonised-rules-artificial-intelligence.

Quinlan, J. Ross (1986). "Induction of Decision Trees". In: *Mach. Learn.* 1.1, pp. 81–106. DOI: 10.1023/A:1022643204877. URL: https://doi.org/10.1023/A:1022643204877.

Regulation, General Data Protection (2016). "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016

on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46". In: *Official Journal of the European Union (OJ)* 59.1-88, p. 294.

Reiss, Attila and Didier Stricker (2012). "Introducing a New Benchmarked Dataset for Activity Monitoring". In: *16th International Symposium on Wearable Computers, ISWC 2012, Newcastle, United Kingdom, June 18-22, 2012*. IEEE Computer Society, pp. 108–109. DOI: 10.1109/ISWC.2012.13. URL: https://doi.org/10.1109/ISWC.2012.13.

Ribeiro, Marco Túlio, Sameer Singh, and Carlos Guestrin (2016). ""Why Should I Trust You?": Explaining the Predictions of Any Classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. Ed. by Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi. ACM, pp. 1135–1144. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

Ribeiro, Marco Túlio, Sameer Singh, and Carlos Guestrin (2018). "Anchors: High-Precision Model-Agnostic Explanations". In: *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. Ed. by Sheila A. McIlraith and Kilian Q. Weinberger. AAAI Press, pp. 1527–1535. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16982.

Ribeiro, Mauro, Katarina Grolinger, and Miriam A. M. Capretz (2015). "MLaaS: Machine Learning as a Service". In: *14th IEEE International Conference on Machine Learning and Applications, ICMLA 2015, Miami, FL, USA, December 9-11, 2015*. Ed. by Tao Li, Lukasz A. Kurgan, Vasile Palade, Randy Goebel, Andreas Holzinger, Karin Verspoor, and M. Arif Wani. IEEE, pp. 896–902. DOI: 10.1109/ICMLA.2015.152. URL: https://doi.org/10.1109/ICMLA.2015.152.

Salah, Khaled, Muhammad Habib Ur Rehman, Nishara Nizamuddin, and Ala I. Al-Fuqaha (2019). "Blockchain for AI: Review and Open Research Challenges". In: *IEEE Access* 7, pp. 10127–10149. DOI: 10.1109/ACCESS.2018.2890507. URL: https://doi.org/10.1109/ACCESS.2018.2890507.

Shahriari, Kyarash and Mana Shahriari (2017). "IEEE standard review - Ethically aligned design: A vision for prioritizing human wellbeing with artificial intelligence and autonomous systems". In: *IEEE Canada International Humanitarian Technology Conference, IHTC 2017, Toronto, ON, Canada, July 21-22, 2017*. IEEE, pp. 197–201. DOI: 10.1109/IHTC.2017.8058187. URL: https://doi.org/10.1109/IHTC.2017.8058187.

Shen, Zhiqiang, Zechun Liu, Dejia Xu, Zitian Chen, Kwang-Ting Cheng, and Marios Savvides (2021). "Is Label Smoothing Truly Incompatible with Knowledge Distillation: An Empirical Study". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net. URL: https://openreview.net/forum?id=PObuuGVrGaZ.

Simard, Patrice Y., Yann LeCun, John S. Denker, and Bernard Victorri (1996). "Transformation Invariance in Pattern Recognition-Tangent Distance and Tangent Propagation". In: *Neural Networks: Tricks of the Trade*. Ed. by Genevieve B. Orr and Klaus-Robert Müller. Vol. 1524. Lecture Notes in Computer Science. Springer, pp. 239–27. DOI: 10.1007/3-540-49430-8_13. URL: https://doi.org/10.1007/3-540-49430-8_13.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman (2014). "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Workshop Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1312.6034.

Singh, Sameer, Marco Túlio Ribeiro, and Carlos Guestrin (2016). "Programs as Black-Box Explanations". In: *CoRR* abs/1611.07579. arXiv: 1611.07579. URL: http://arxiv.org/abs/1611.07579.

Singh, Vivek Kumar, Mohamed Abdel-Nasser, Hatem A. Rashwan, Farhan Akram, Rami Haffar, Nidhi Pandey, Md. Mostafa Kamal Sarker, Sebastian Kohan, Josep Guma, Santiago Romani, and Domenec Puig (2019). "Mass Detection in Mammograms Using a Robust Deep Learning Model". In: *Artificial Intelligence Research and Development - Proceedings of the 22nd International Conference of the Catalan Association for Artificial Intelligence, CCIA 2019, Mallorca, Spain, 23-25 October 2019*. Ed. by Jordi Sabater-Mir, Vicenç Torra, Isabel Aguiló, and Manuel González Hidalgo. Vol. 319. Frontiers in Artificial Intelligence and Applications. IOS Press, pp. 365–372. DOI: 10.3233/FAIA190147. URL: https://doi.org/10.3233/FAIA190147.

Slack, Dylan, Anna Hilgard, Himabindu Lakkaraju, and Sameer Singh (2021). "Counterfactual Explanations Can Be Manipulated". In: *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*. Ed. by Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, pp. 62–75. URL: https://proceedings.neurips.cc/paper/2021/hash/009c434cab57de48a31f6b669e7ba266-Abstract.html.

Snoek, Jasper, Hugo Larochelle, and Ryan P. Adams (2012). "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*. Ed. by Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, pp. 2960–2968. URL: https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.

Srivastava, Aditya, Tanvi Shinde, Raj Joshi, Sameer Ahmed Ansari, and Nupur Giri (2021). "Auto-DL: A Platform to Generate Deep Learning Models". In: *International Conference on Soft Computing in Data Science*. Springer, pp. 89–103.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus (2014). "Intriguing properties of neural networks". In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. Ed. by

Yoshua Bengio and Yann LeCun. URL: http://arxiv.org/abs/1312.6199.

Taheri, Rahim, Reza Javidan, Mohammad Shojafar, Zahra Pooranian, Ali Miri, and Mauro Conti (2020). "On defending against label flipping attacks on malware detection systems". In: *Neural Comput. Appl.* 32.18, pp. 14781–14800. DOI: 10.1007/s00521-020-04831-9. URL: https://doi.org/10.1007/s00521-020-04831-9.

Tong, Joo Chuan and Shoba Ranganathan (2013). "5 - Computational T cell vaccine design". In: *Computer-Aided Vaccine Design*. Ed. by Joo Chuan Tong and Shoba Ranganathan. Woodhead Publishing Series in Biomedicine. Woodhead Publishing, pp. 59–86. ISBN: 978-1-907568-41-1. DOI: https://doi.org/10.1533/9781908818416.59. URL: https://www.sciencedirect.com/science/article/pii/B9781907568411500052.

Tukey, John W. (1977). *Exploratory data analysis*. Addison-Wesley series in behavioral science : quantitative methods. Addison-Wesley. ISBN: 0201076160. URL: https://www.worldcat.org/oclc/03058187.

Valueva, Maria V, NN Nagornov, Pavel A Lyakhov, Georgii V Valuev, and Nikolay I Chervyakov (2020). "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". In: *Mathematics and computers in simulation* 177, pp. 232–243.

Verma, Sahil, John P. Dickerson, and Keegan Hines (2020). "Counterfactual Explanations for Machine Learning: A Review". In: *CoRR* abs/2010.10596. arXiv: 2010.10596. URL: https://arxiv.org/abs/2010.10596.

Vermeire, Tom, Dieter Brughmans, Sofie Goethals, Raphael Mazzine Barbossa de Oliveira, and David Martens (2022). "Explainable image classification with evidence counterfactual". In: *Pattern Anal. Appl.* 25.2, pp. 315–335. DOI: 10.1007/s10044-021-01055-y. URL: https://doi.org/10.1007/s10044-021-01055-y.

Wachter, Sandra, Brent Mittelstadt, and Chris Russell (2017). "Counterfactual explanations without opening the black box: Automated decisions and the GDPR". In: *Harvard Journal of Law & Technology* 31, p. 841.

Wang, Xiaobo, Tianyu Fu, Shengcai Liao, Shuo Wang, Zhen Lei, and Tao Mei (2020). "Exclusivity-Consistency Regularized Knowledge Distillation for Face Recognition". In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXIV*. Ed. by Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm. Vol. 12369. Lecture Notes in Computer Science. Springer, pp. 325–342. DOI: 10.1007/978-3-030-58586-0_20. URL: https://doi.org/10.1007/978-3-030-58586-0_20.

Wei, Kang, Jun Li, Ming Ding, Chuan Ma, Howard H. Yang, Farhad Farokhi, Shi Jin, Tony Q. S. Quek, and H. Vincent Poor (2020). "Federated Learning With Differential Privacy: Algorithms and Performance Analysis". In: *IEEE Trans. Inf. Forensics Secur.* 15, pp. 3454–3469. DOI: 10.1109/TIFS.2020.2988575. URL: https://doi.org/10.1109/TIFS.2020.2988575.

Xiong, Wayne, Lingfeng Wu, Fil Alleva, Jasha Droppo, Xuedong Huang, and Andreas Stolcke (2018). "The Microsoft 2017 conversational speech recognition system". In: *2018 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp. 5934–5938.

Yao, Yuanshun, Huiying Li, Haitao Zheng, and Ben Y. Zhao (2019). "Latent Backdoor Attacks on Deep Neural Networks". In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*. Ed. by Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz. ACM, pp. 2041–2055. DOI: 10.1145/3319535.3354209. URL: https://doi.org/10.1145/3319535.3354209.

Yin, Dong, Yudong Chen, Kannan Ramchandran, and Peter L. Bartlett (2018). "Byzantine-Robust Distributed Learning: Towards Optimal Statistical Rates". In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Ed. by Jennifer G. Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 5636–5645. URL: http://proceedings.mlr.press/v80/yin18a.html.

Zhang, Hongyi, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz (2018). "mixup: Beyond Empirical Risk Minimization". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=r1Ddp1-Rb.

Zhao, Yue, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra (2018). "Federated Learning with Non-IID

Data". In: *CoRR* abs/1806.00582. arXiv: 1806.00582. URL: http://arxiv.org/abs/1806.00582.

Zhong, Yaoyao and Weihong Deng (2021). "Towards Transferable Adversarial Attack Against Deep Face Recognition". In: *IEEE Trans. Inf. Forensics Secur.* 16, pp. 1452–1466. DOI: 10.1109/TIFS.2020.3036801. URL: https://doi.org/10.1109/TIFS.2020.3036801.



UNIVERSITAT ROVIRA i VIRGILI