**Pluribus** *One*

*seeing one in many*

# ML Explainability Techniques

***Maura Pintor***

AssureMOSS Research Meeting
15th October 2021

*www.pluribus-one.it*

 @pluribus_one

 @PluribusOne

 Pluribus One

# *Explainability and why it is important*

**Fairness**: Ensuring that predictions are unbiased.

**Privacy**: Ensuring that sensitive information in the data is protected.

**Reliability or Robustness**: Ensuring that small changes in the input do not lead to large changes in the prediction.

**Causality**: Check that only causal relationships are picked up.

**Trust**: It is easier for humans to trust a system that explains its decisions compared to a black box.

(a) Husky classified as wolf

(b) Explanation

**Figure 11: Raw data and explanation of a bad model's prediction in the "Husky vs Wolf" task.**

Ribeiro *et al.* "" *Why should i trust you?" Explaining the predictions of any classifier.*" *ACM SIGKDD* 2016.

# *Explainability in software vulnerability discovery*

**Case study**: VulDeePecker + Layerwise Relevance Propagation (LRP) explainability method

1) Dataset contains **artifact that occur only in one class**, but are not relevant for vulnerability detection (false causality)
2) The model gives importance to tokens that are **frequent in both classes**
3) A **linear classifier** on fewer features **achieves better performances**

| Model | # parameters | AUC | TPR |
|---|---|---|---|
| VulDeePecker | $1.2 \times 10^6$ | 0.980 | 0.968 |
| SVM (1-grams) | $5.8 \times 10^2$ | 0.805 | 0.505 |
| SVM (2-grams) | $1.3 \times 10^4$ | 0.958 | 0.942 |
| SVM (3-grams) | $6.6 \times 10^4$ | 0.984 | 0.978 |

Arp et al. "*Dos and Don'ts of Machine Learning in Computer Security.*" *preprint arXiv:2010.09470*, 2020.

# *Explainability in software vulnerability discovery*

Analyzes token-based and graph-based methods

Uses LEMNA for Token-based

Uses activation value of each vertex for Graph-based

Visualizes explanation for each feature, showing that graph-based methods better exploit the information about the **structure**

```
1  link_layer_show(struct ib_port *p,
2          struct port_attribute *unused, char * buf){
3      switch (rdma_port_get_link_layer(
4                  p->ibdev, p->port_num)) {
5          case IB_LINK_LAYER_INFINIBAND:
6              return sprintf(buf, "%s\n", "InfiniBand");
7          case IB_LINK_LAYER_ETHERNET:
8              return sprintf(buf, "%s\n", "Ethernet");
9          default:
10             return sprintf(buf, "%s\n", "Unknown");
11     }
12 }
```

Correct prediction with token-based methods, but for the wrong reason

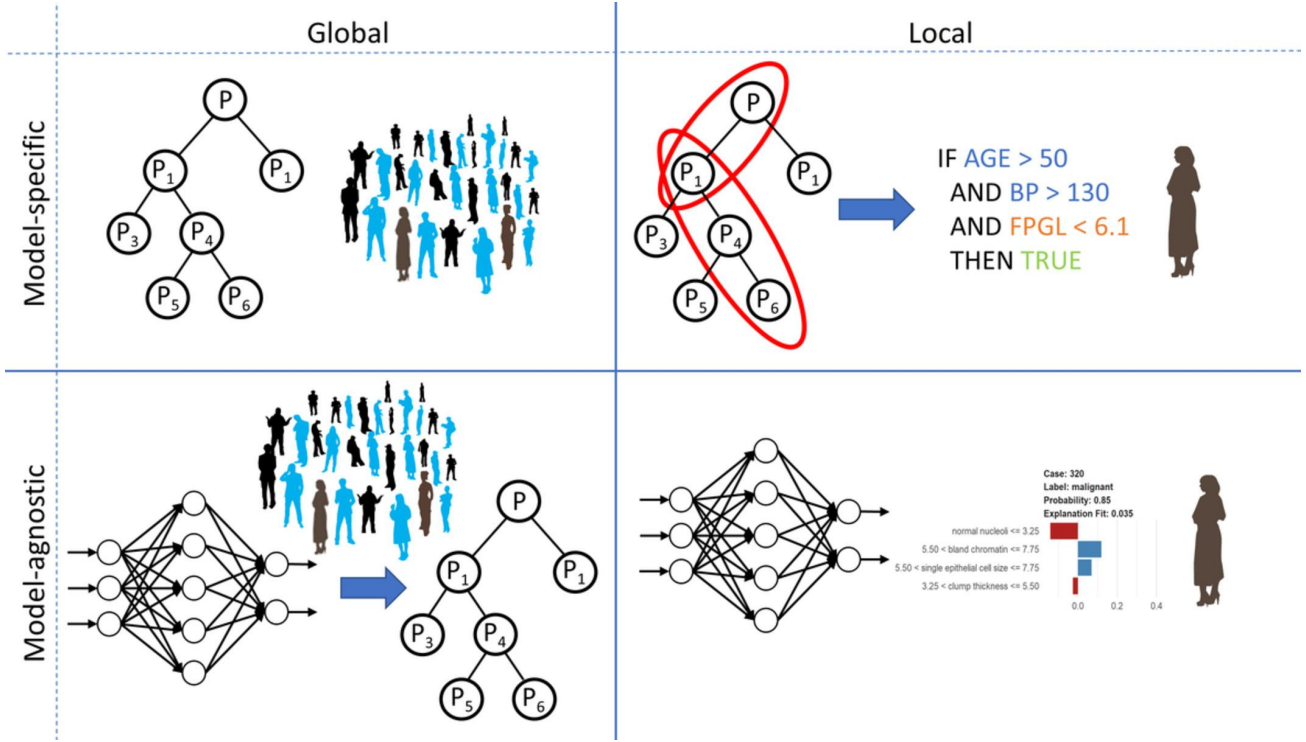
```
1   static int mov_read_dvc1(MOVContext *c,
2           AVIOContext *pb, MOVAtom atom) {
3       AVStream *st;
4       uint8_t profile_level;
5       if (c->fc->nb_streams < 1)
6           return 0;
7       st = c->fc->streams[c->fc->nb_streams-1];
8       if (atom.size >= (1<<28) || atom.size < 7)
9           return AVERROR_INVALIDDATA;
10      profile_level = avio_r8(pb);
11      if ((profile_level & 0xf0) != 0xc0)
12          return 0;
...     ...
18      st->codec->extradata_size = atom.size - 7;
19      avio_seek(pb, 6, SEEK_CUR);
20      avio_read(
21          pb, st->codec->extradata,
22              st->codec->extradata_size);
23      return 0;
24  }
```

Correct but only when using graphs

Chakraborty et al. "*Deep learning based vulnerability detection: Are we there yet.*" *IEEE TSE* (2021).

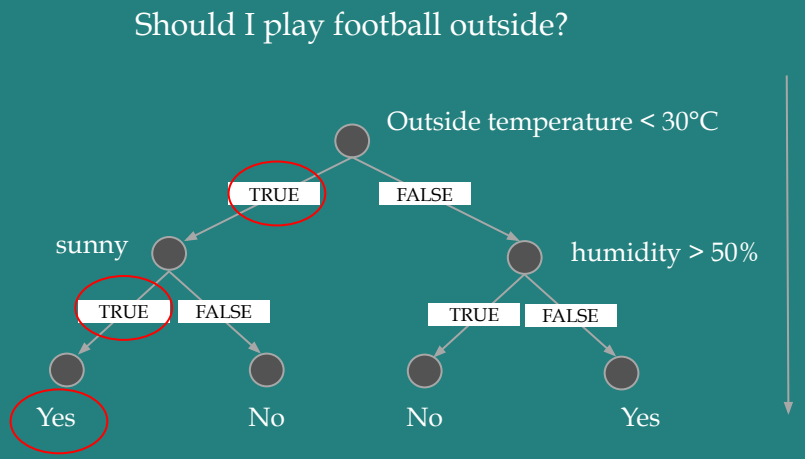
# *xAI methods*

# *Taxonomy of explainability methods*




Stiglic et al. "*Interpretability of machine learning-based prediction models in healthcare". WIREs Data Mining Knowl Discov*, 2020.

# Model-specific methods and models that are interpretable by-design




**Depth** = how many levels of decision

Too much depth makes the model **not interpretable**

# *Model-agnostic methods*

**Black-box**: work by observing only input-output pairs


**White-box**: access to model's internals (usually gradients)

# *Black-box models*

# LIME

Local linear approximation, weighting perturbed points by **proximity**

**Additive attribution** (each feature contributes additively to the outcome)

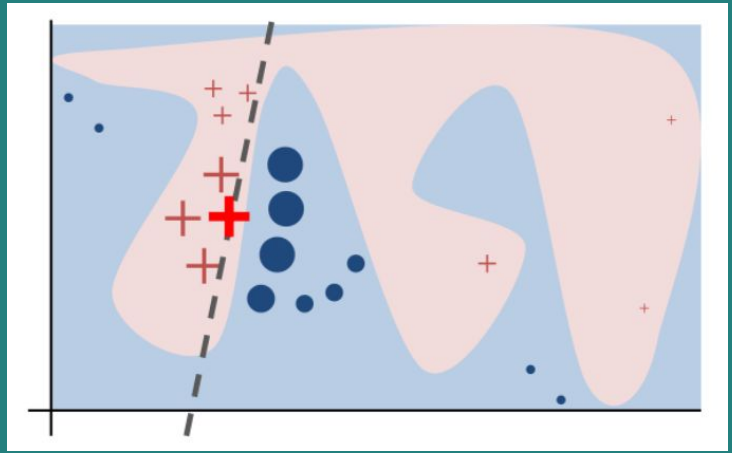
**Local fidelity**, i.e. explained features might differ from one sample to the other (as opposed to global explanations)

local approximation

$$\xi(x) = \underset{g \in G}{\operatorname{argmin}} \mathcal{L}(f, g, \pi_x) + \Omega(g)$$

family of interpretable models

regularization



Ribeiro et al. "" *Why should i trust you?" Explaining the predictions of any classifier*." *ACM SIGKDD* 2016.

# LEMNA

**Fused lasso** (penalty that forces relevant/adjacent features to be grouped together to give meaningful explanations)

**Mixture regression model** (combines different linear models, allowing to approximate more complex functions)

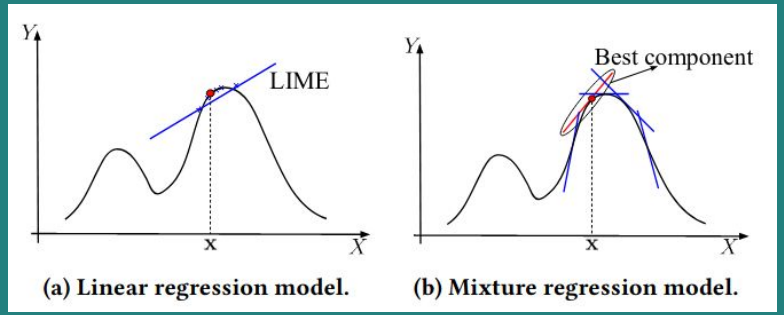
$$L(f(\mathbf{x}), y) = \sum_{i=1}^{N} \|f(\mathbf{x}_i) - y_i\|$$

$$\text{subject to } \sum_{j=2}^{M} \left\|\beta_{kj} - \beta_{k(j-1)}\right\| \leq S, k = 1, \ldots, K$$

fused lasso regularization

$$f(x) = \sum_{j=1}^{K} \pi_j \left(\beta_j \cdot x + \epsilon_j\right)$$

weighted sum of K linear models




(a) Linear regression model. (b) Mixture regression model.

Guo et al. "*Lemna: Explaining deep learning based security applications.*" *ACM SIGSAC* 2018.

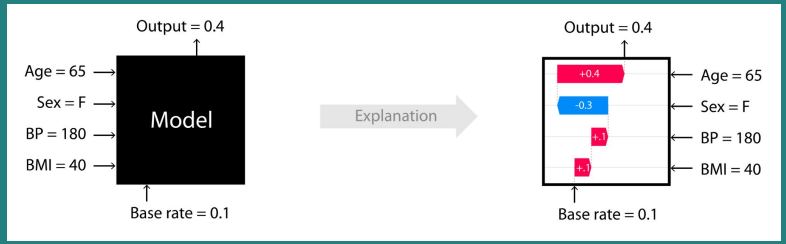
# SHAP

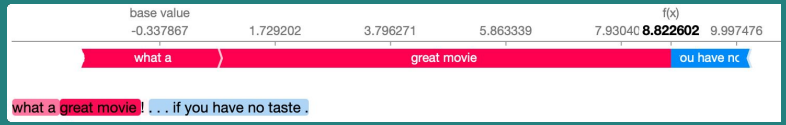
**Additive attribution** method (like LIME)

Trains a model with and without subsets of features, compares the difference in performance (and then weight features based on all differences observed)

Finds out the **marginal contribution** of each feature and feature sets

Weights the features by the **information they contain**, rather than the proximity




Lundberg et al. "*A unified approach to interpreting model predictions.*" *NeurIPS* 2017.

$$\phi_i(f,x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M-|z'|-1)!}{M!}\left[f_x(z') - f_x(z' \backslash i)\right]$$

difference in outcome

subset of features

weighting term
(how many features are in the subset)

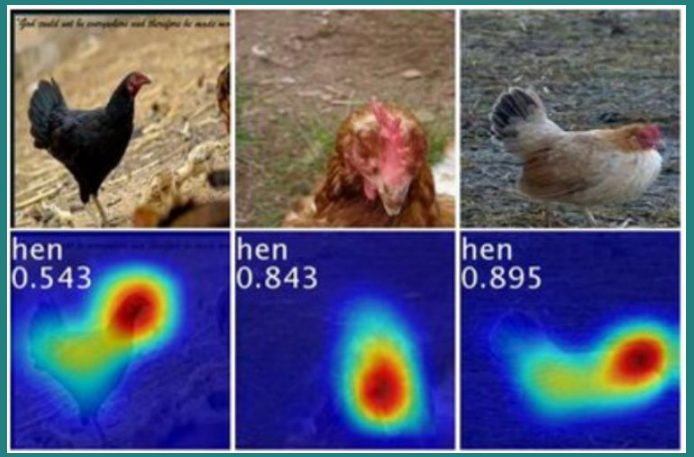
Pluribus One
seeing one in many

# *White-box models*

# Class Activation Maps (CAM)

Scales features from the last hidden layer with the weight connecting them to the desired output node

Simple method, but often saturates and creates useless maps




Zhou et al. "*Learning Deep Features for Discriminative Localization". CVPR* 2016

# Layer-wise relevance propagation (LRP)

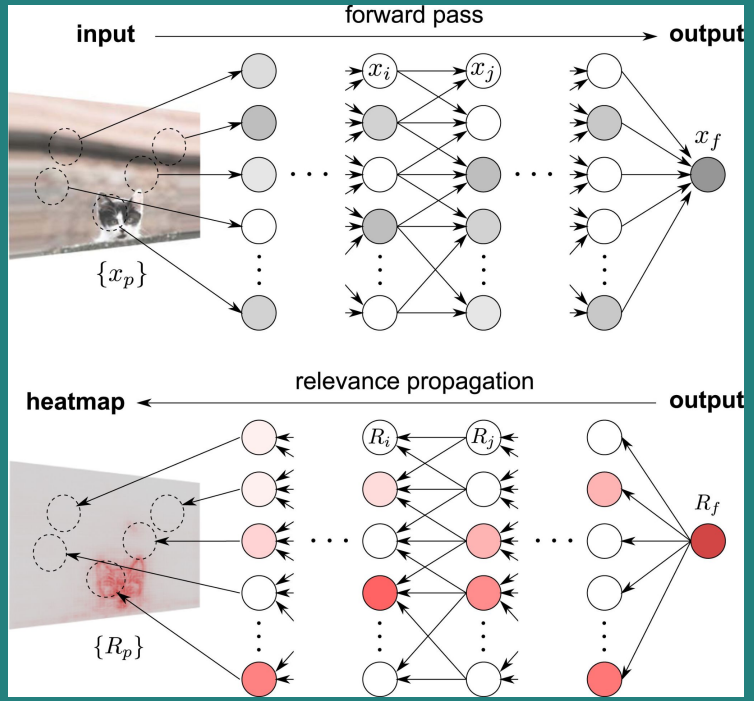
Uses a map that assigns a value to each feature, representing the effect of that input being set to a reference value (usually zero), as opposed to its original value

$$R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j}} R_j^{(l+1)} \quad \text{with} \quad \mathrm{z_{ij}} = \mathrm{x_i^{(l)}} \mathrm{w_{ij}^{(1,l+1)}}$$

relevance at next layer

influence of the single neuron w.r.t. the sum of layer neurons

relevance at current layer




Bach et al. "*On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation*." *PloS one* 2015.
Image source: *Montavon et al. "Explaining nonlinear classification decisions with deep taylor decomposition." Pattern Recognition,* 2017.

# Explaining using gradients

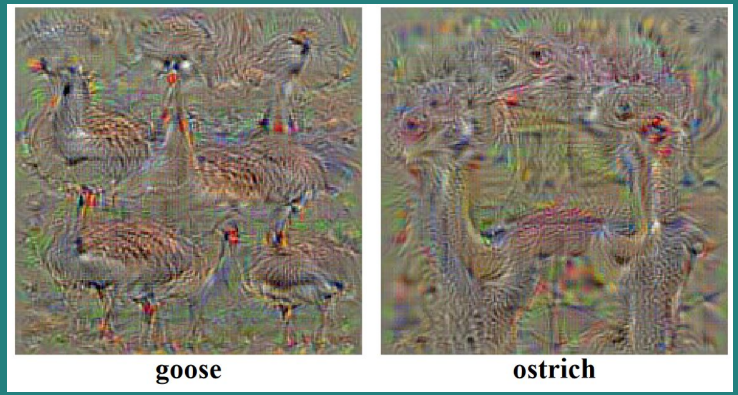
Highlights simple **gradients of the output class** w.r.t. the input

The method allows also to maximize the gradients of a class to see meaningful features (global explanation)

$$r_i = \frac{\partial y}{\partial x_i}$$

local explanation

global explanation

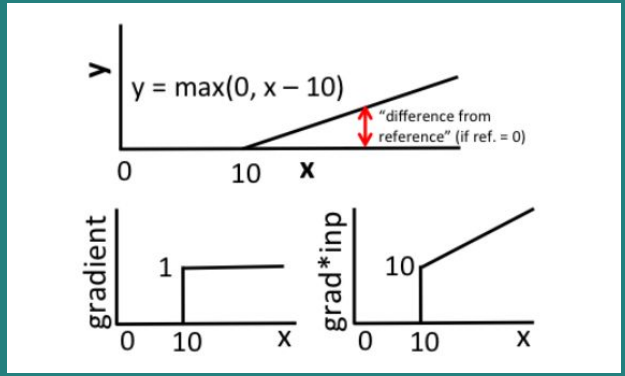



Simonyan et al. "*Deep inside convolutional networks: Visualising image classification models and saliency maps*." *preprint arXiv:1312.6034* (2013).

# *Gradients x input, a.k.a. linear approximation*

Decomposes the output on a specific input by backpropagating the contributions of all neurons to every feature

$$r_i = \frac{\partial y}{\partial x_i} x_i$$




Shrikumar et al. "*Learning important features through propagating activation differences*." *ICML*, 2017.

# Integrated gradients

Improves the linear approximation by referring to a **counterfactual baseline input**

Accumulates the gradients along the path

$$r_i = (x_i - x_i') \int_0^1 \frac{\partial f_N\left(x' + \alpha\left(x - x'\right)\right)}{\partial x_i} \, \mathrm{d}\alpha$$

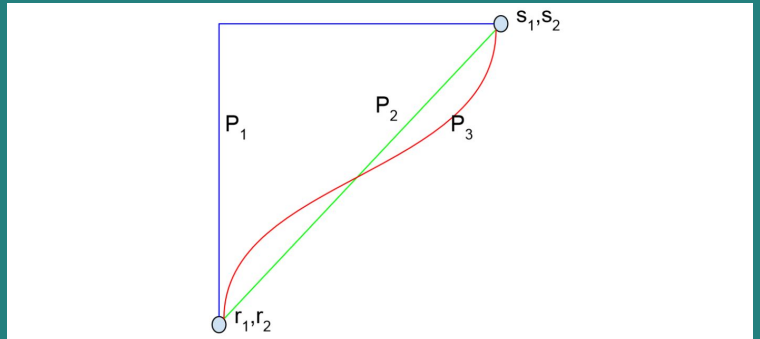
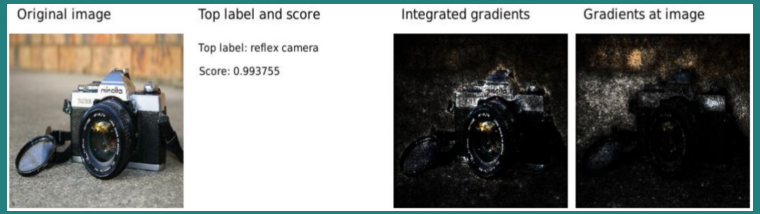



*Figure 1.* Three paths between an a baseline $(r_1, r_2)$ and an input $(s_1, s_2)$. Each path corresponds to a different attribution method. The path $P_2$ corresponds to the path used by integrated gradients.



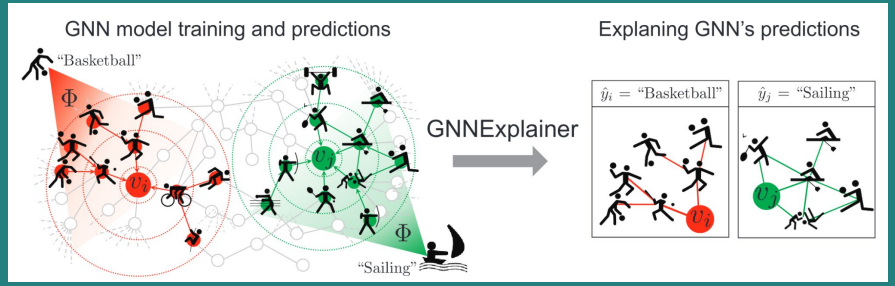

Sundararajan et al. "*Axiomatic attribution for deep networks.*" *ICML*, 2017.

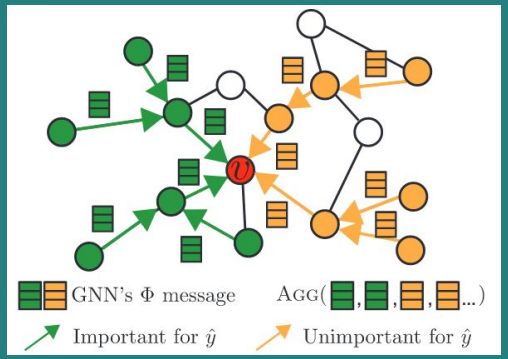
# Explaining Graphs

Highlights **subgraph** with the nodes that are relevant for the prediction on the input point

For the given node, it quantifies the change in probability of the prediction, if only a subgraph is used

Formulated as an optimization task that maximizes the **mutual information** between a GNN's prediction and distribution of possible subgraph structures




Ying et al. "*Gnnexplainer: Generating explanations for graph neural networks.*" *NeurIPS* 2019.

# Graph-LRP

Uses higher order Taylor expansions to identify relevant walks over multiple layers of a GNN

The relevance per walk is computed using a backpropagation similar to Layer-wise relevance propagation (**LRP**) for each node in the walk

The relevance score is associated to a **walk**

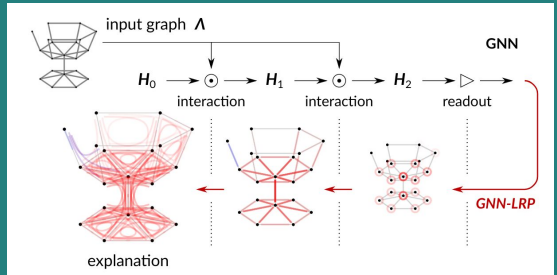
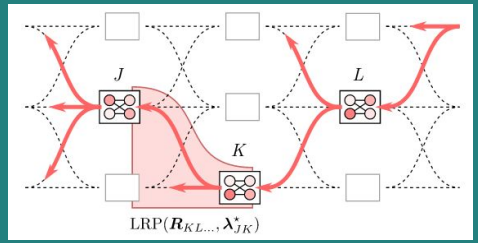



Fig. 1. High-level illustration of GNN-LRP. The explanation procedure starts at the GNN output, and proceeds backwards to progressively uncover the walks that are relevant for the prediction.




Schnake et al. "*Higher-Order Explanations of Graph Neural Networks via Relevant Walks*". arXiv:2006.03589

# *Prototype-based methods*

# *Prototype-based methods*

Aim to identify training points most responsible for a given prediction

Uses **influence functions**: how would the model's predictions change if we did not have this training point?

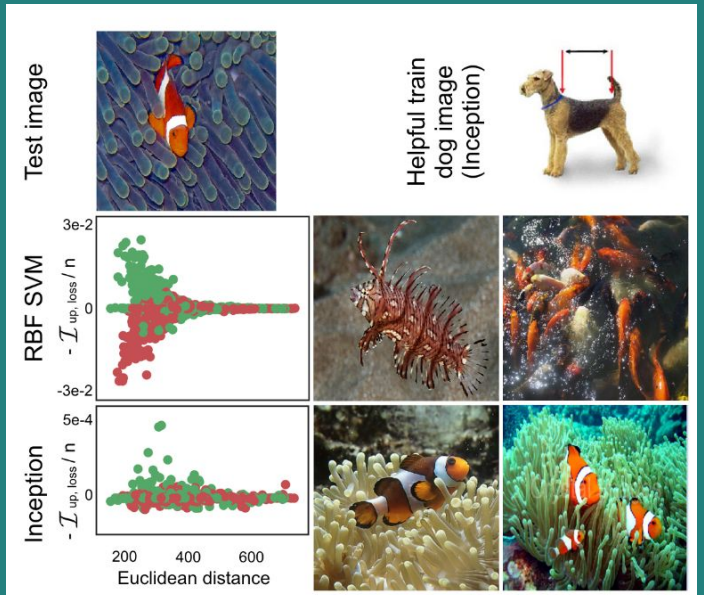



*Figure 4.* **Inception vs. RBF SVM. Bottom left:** $-\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}})$ vs. $\|z - z_{\text{test}}\|_2^2$. Green dots are fish and red dots are dogs. **Bottom right:** The two most helpful training images, for each model, on the test. **Top right:** An image of a dog in the training set that helped the Inception model correctly classify the test image as a fish.

Koh et al. "*Understanding black-box predictions via influence functions*." *ICML*, 2017.

# Counterfactuals

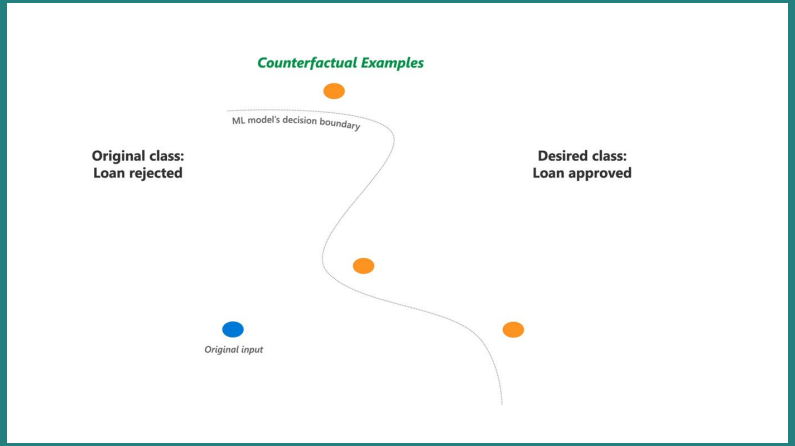
Hypothetical examples that show how to obtain a different prediction (using adversarial ML)

Found with adversarial techniques

**Feasibility** of the counterfactual actions given user context and constraints

**Diversity** among the counterfactuals presented (different solutions)

$$\boldsymbol{c} = \arg\min_{\boldsymbol{c}} y\,\text{loss}(f(\boldsymbol{c}), y) + |\boldsymbol{x} - \boldsymbol{c}|$$




Wachter et al. "*Counterfactual explanations without opening the black box: Automated decisions and the GDPR*".
Image source: Mothilal et al. "*Explaining machine learning classifiers through diverse counterfactual explanations.*" ACM FaccT. 2020.

# *Explaining GNNs for Vulnerability Discovery*

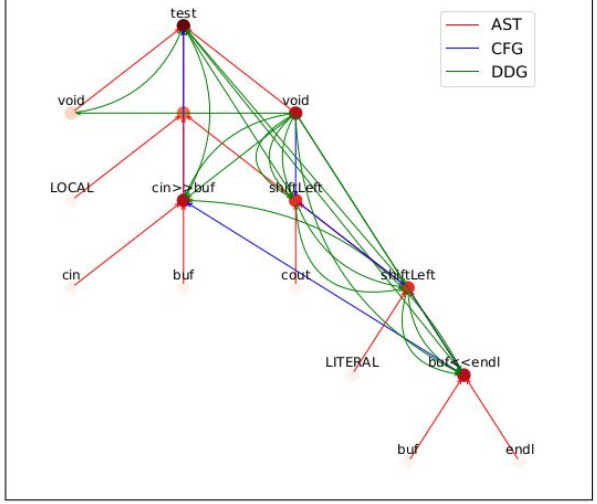
# *Explaining Graph Neural Networks for Vulnerability Discovery*

Comparison between **Graph Neural Networks** explanation methods
- 9 graph-agnostic
- 3 graph-specific

Evaluation based on **6 proposed metrics**

**Spoiler**: none of them provide an optimal solution, the two approaches are complementary

```
#define MAXSIZE    40
void test(void)
{
        char buf[MAXSIZE];
        cin>>buf;
        cout<<"result:"<<buf<<endl;
}
```




*Ganz et al. "Explaining Graph Neural Networks for Vulnerability Discovery". AISec '21.*

# Descriptive Accuracy

Measures the accuracy of the explanation

It is obtained by removing the *k*% of the **most relevant nodes** of the input graph, and then calculating another forward pass

The more the accuracy drops the more the nodes **identified** by the explanation were relevant

Graph-specific methods are inferior to the graph-agnostic ones under this criterion

# *Structural robustness*

Measures the robustness of explanations against input graph **perturbations** (e.g. noise and adversarial manipulations)

It is computed by comparing the *10%* most relevant nodes before and after perturbing the input graph

Integrated Gradients is by far the best

Graph-LRP is the worst

The other methods are comparable

# Contrastivity

Provide a measure of the suitability of the explanation method to code analysis and vulnerability discovery

It is calculated by **comparing relevant statements** for the **vulnerable** and **non-vulnerable** class ($\chi 2$-distance of the histograms)

Higher **difference** indicates that the model takes more different nodes for the two classes

Low for most methods, but graph-specific methods seem to be better

# Sparsity

Measures the **conciseness** of the provided explanation

It is computed by normalizing the relevance values of the nodes in [-1,1] and analyzing the distribution of their absolute values

The more relevance values of nodes are close to 0, the more the provided explanation is **sparse**

Graph-specific methods yield the sparsest scores

# Stability

Evaluates the stability of provided explanations with respect to **randomness** (some methods are not deterministic)

It is measured in terms of **standard deviation** of the descriptive accuracy and sparsity over five runs

Graph-specific methods yield an uncertainty that differs extremely from model to model

Graph-agnostic explanation methods do not vary at all

# Efficiency

Measures the **runtime performance** of the explanation method

It is obtained computing the average runtime of an EM per single graph

Graph-specific methods are almost always slower

# *Summarizing results…*

All explanation methods have shortcomings in at least two criteria

Graph-agnostics methods often lack sparse explanations and tend to mark more nodes as relevant than needed

Graph-specific methods have limited stability and accuracy in the relevant features

| Category | DA | Sparsity | Robustness | Contrastivity | Stability | Efficiency |
|---|---|---|---|---|---|---|
| Graph-agnostic | ● ● ● | ○ ○ ○ | ● ● ● | ● ○ ○ | ● ● ● | ● ● ● |
| Graph-specific | ○ ○ ○ | ● ● ● | ○ ○ ○ | ● ● ○ | ○ ○ ○ | ○ ○ ○ |

# *Final remarks*

# Summary

There is a great variety of explainability methods

They have been tested **predominantly on images and text**

There is some emerging study on explaining code vulnerability predictions

There are useful **metrics** available for evaluating explainability methods and helping choosing the best one depending on the requirements of the application

A few pointers to related topics in the next slides

# *Adversarial attacks against explanations*

Explanations are not robust to adversarial attacks

The sample can be manipulated in a way that creates an **arbitrary explanation**

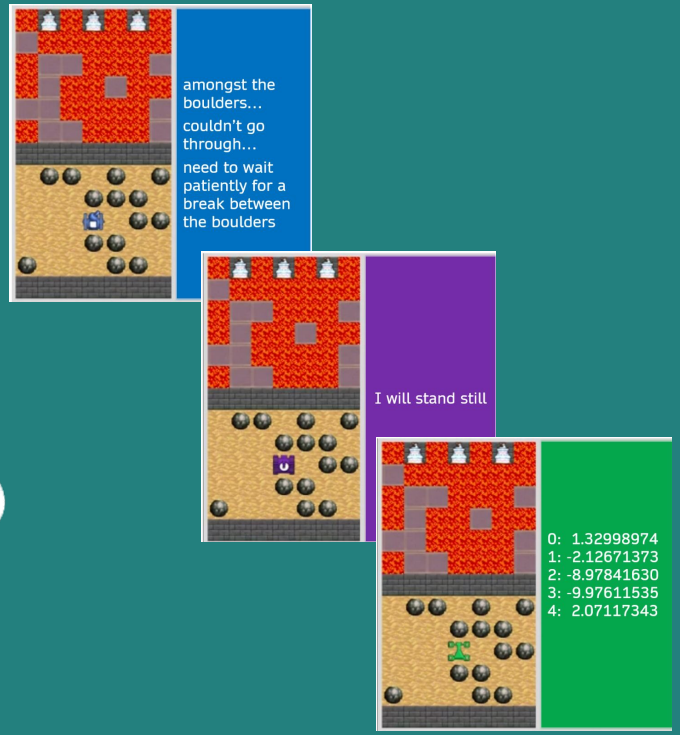

Dombrowski et al. "*Explanations can be manipulated and geometry is to blame.*" *NeurIPS 2019*.

# *Human-centric xAI*

Study on how the explanations provided by AI are perceived by who opens the “black box”

Studies how two different groups, with and without background in AI, **perceive** the explanations

Aims towards **tailoring** the explanations to the public that is using them




Ehsan et al. "*The who in explainable ai: How ai background shapes perceptions of ai explanations.*" *preprint arXiv:2107.13509*, 2021.

# Pluribus One

*seeing one in many*


Pluribus One S.r.l.

Via Vincenzo Bellini 9, Cagliari (CA), Italy

Via Emilio Segrè, 17, Elmas (CA), Italy

*info@.pluribus-one.it*

*www.pluribus-one.it*


*Thank you for your attention*

 @pluribus_one

 @PluribusOne

 Pluribus One