EVERYTHING IS

XAI

THIS PRESENTATION IS ABOUT

→ taxonomy

→ model (un)certainty

→ gradCAMs

→ the Nigerian prince.

TAXONOMY OF XAI

→ **by default**

1. linear models
2. tree-based models

→ **XAI**

→ global vs local methods

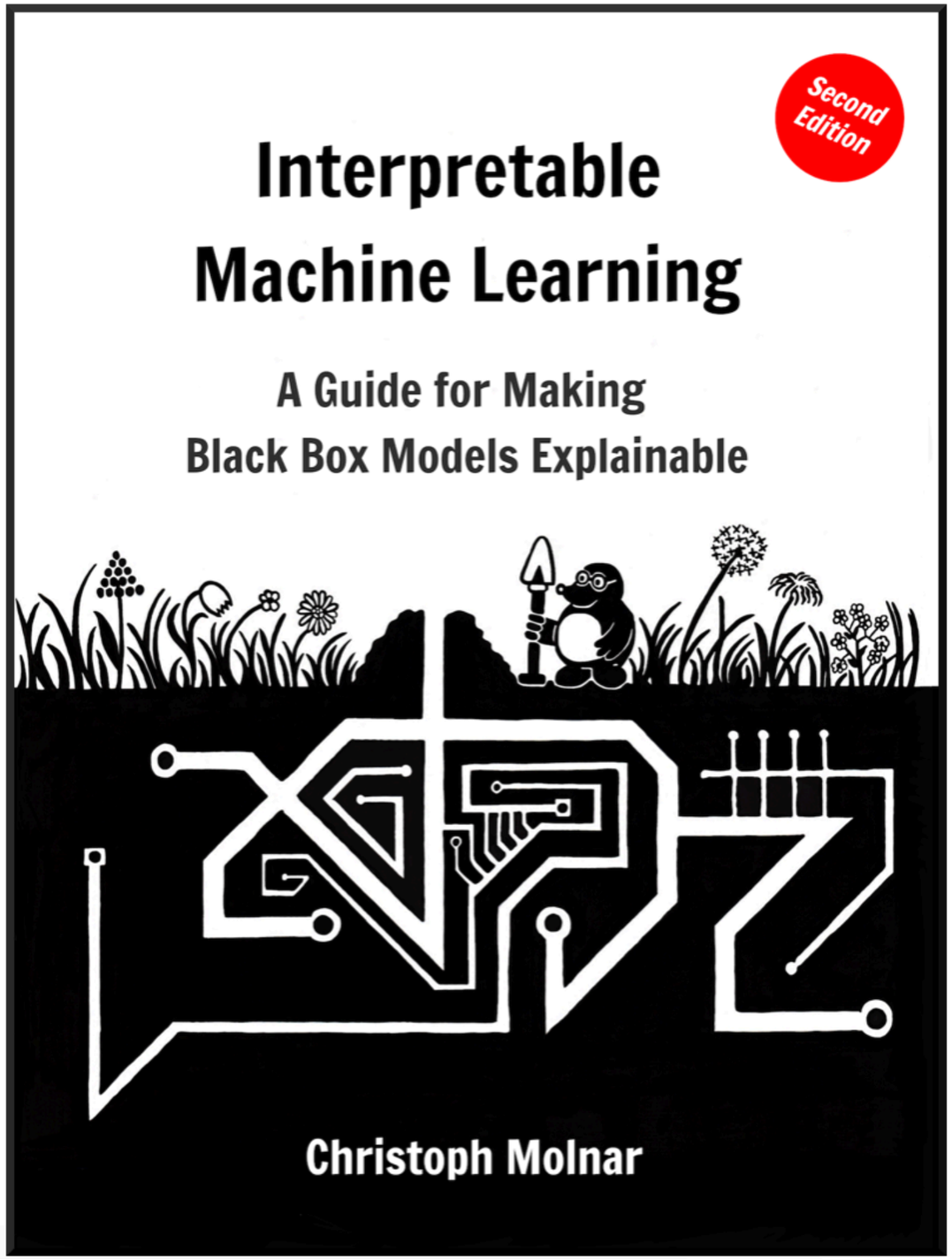
→ model-specific vs model-agnostic

classify gradCAM according to the taxonomy

TAXONOMY OF XAI

- → **by default**
  1. linear models
  2. tree-based models
- → **XAI**
  - → global vs **local** methods
  - → **model-specific** vs model-agnostic

gradCAM is a local, model-specific method to explain the predictions of a CNN



MODEL (UN)CERTAINTY

"It is remarkable that science, which originated in the consideration of games of chance, should have become the most important object of human knowledge."

— Pierre Simon Laplace

MODEL (UN)CERTAINTY I

aleatoric uncertainty

statistical in nature, refers to random variation

🔔 $P(Y|X;\theta)$

more data is not a solution

data quality, model averaging, and prediction intervals help.

MODEL (UN)CERTAINTY II

epistemic uncertainty

human in nature, refers to ignorance

🔔 $P(Y|X;\theta)$

more data is a solution

hyperparameter tuning, regularization etc. help

HOW IS THIS USEFUL?

if we train a classifier on digits [0,8], what happens when we run..?

! *never forget that the models you build are probabilistic in nature*

HOW IS THIS USEFUL?

if we train a classifier on digits [0,8], what happens when we run..?

→ model.predict([2])

! ***never forget that the models you build are probabilistic in nature***

HOW IS THIS USEFUL?

if we train a classifier on digits [0,8], what happens when we run..?

→ model.predict([2])

→ model.predict([7])

! ***never forget that the models you build are probabilistic in nature***

HOW IS THIS USEFUL?

if we train a classifier on digits [0,8], what happens when we run..?

→ model.predict([2])

→ model.predict([7])

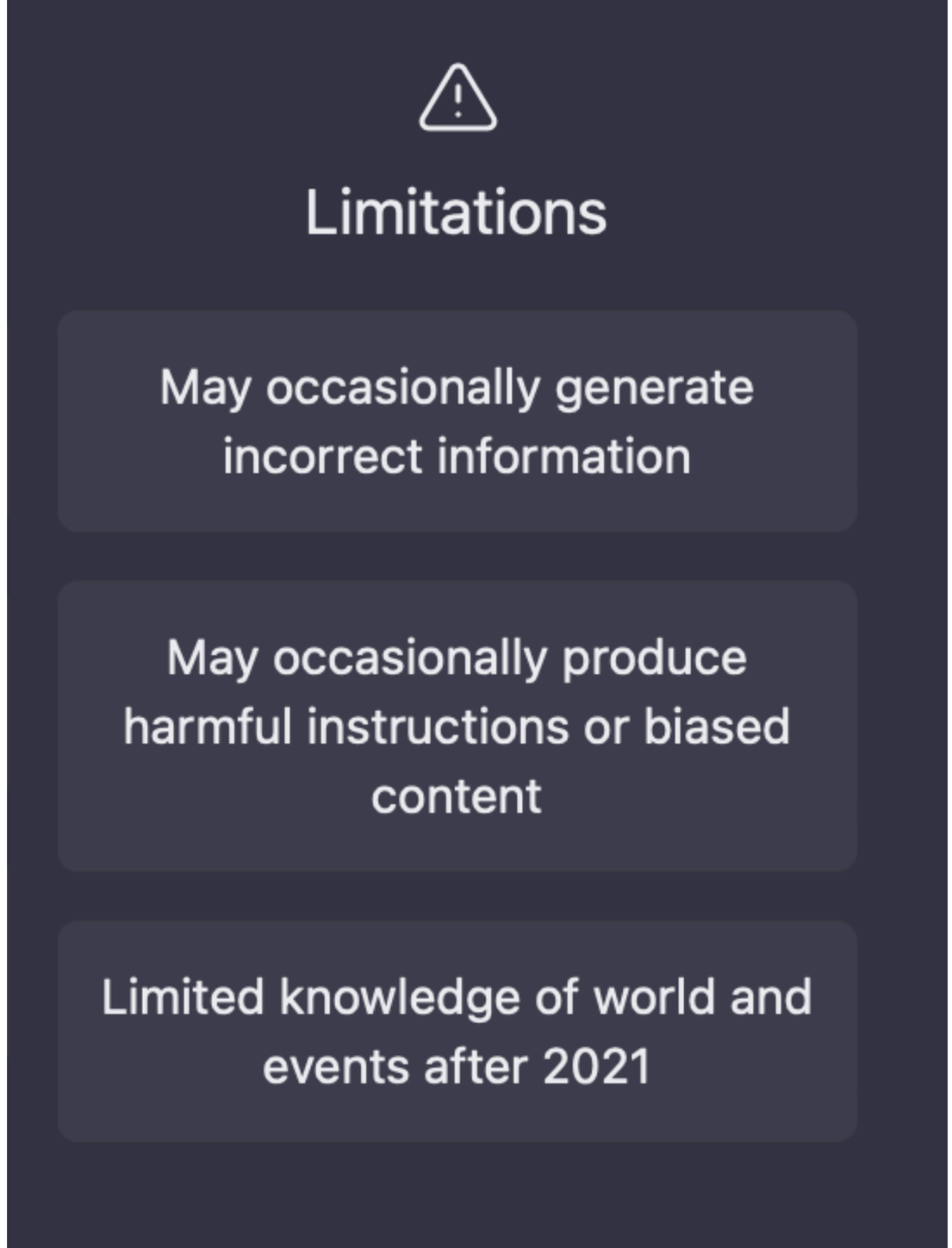
→ model.predict([9])

! ***never forget that the models you build are probabilistic in nature***

HOW IS THIS USEFUL?

even the best models can be wrong, and with horrible consequences

quantify the uncertainty of your model: a hot research topic in AI



may occasionally kill passengers

🎵 MIND YOUR STEP

⚠ neural networks tend to be overconfident when being completely wrong

think self-driving cars, medical diagnosis, or exploding rockets

🙋 XAI

EXPLAINING DEEP NEURAL NETWORKS

EXPLAINING DEEP NEURAL NETWORKS

feature visualization

visualize the activations of a deep neural network

feature attribution

visualize the features that contribute strongly to the activations of CNN layers for a given input image and class label.

adversarial examples

understand the robustness of a deep neural network by generating adversarial examples giving insight into the decision boundaries of the model.

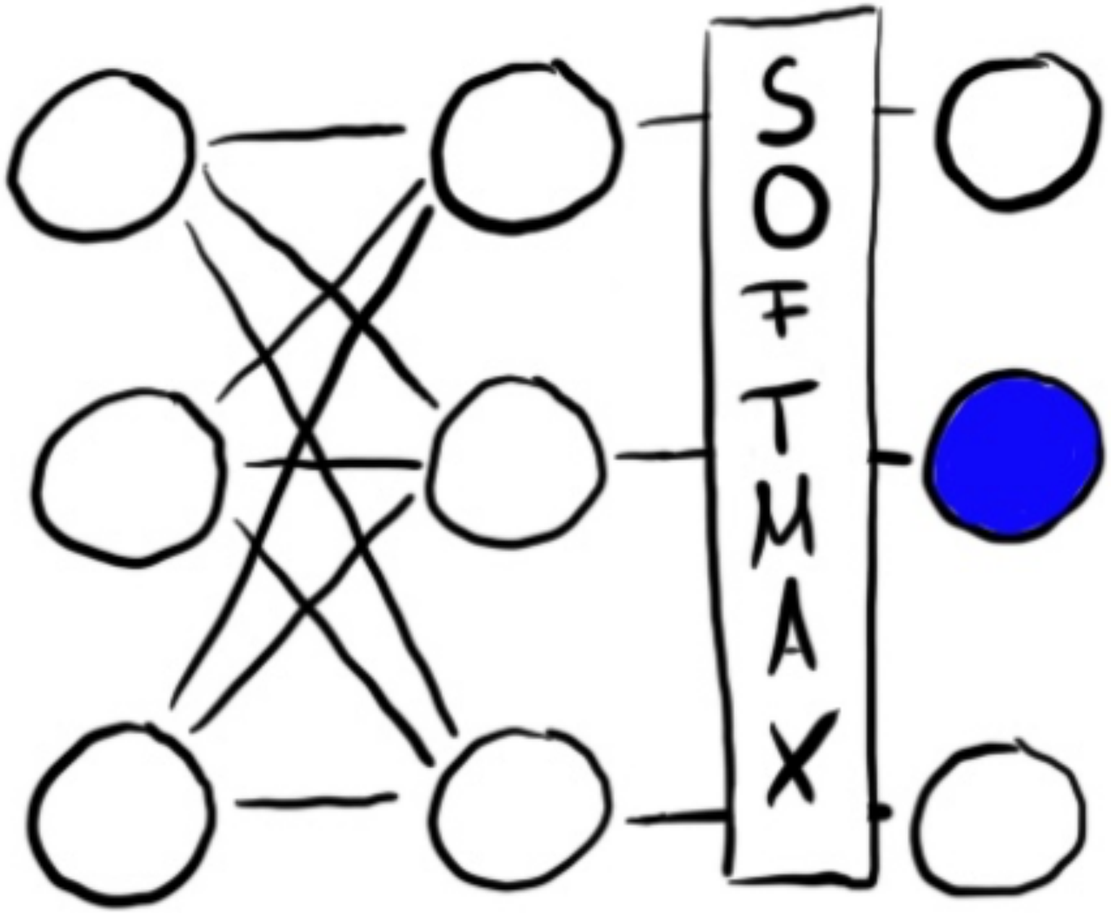
GradCAM

🔔 gradCAM is a feature attribution technique which visualizes the activations of a **CNN layer** by producing a heatmap that highlights the regions of an **input image** that are most relevant **to a particular output class.**

→ helps to understand how a CNN works

→ can be used to debug a CNN

can be used to create visual explanations for the predictions made by a CNN



HOW DOES GRADCAM WORK?

gradCAM works by computing the gradients of the output class score with respect to the feature maps of the last convolutional layer in the CNN. These gradients are then used to compute a weight map for each feature map, which are then multiplied together and summed to produce the final heatmap [1].

[1] Molnar, C. (2020). Interpretable machine learning. Lulu. com. website.

TF_EXPLAIN

```
!pip install tf_explain
!pip install opencv-python

#load libraries
import numpy as np
import tensorflow as tf
import PIL

#load GradCAM
from tf_explain.core.grad_cam import GradCAM

IMAGE_PATH = "./assets/images/cat.jpg"
class_index = 281

img = tf.keras.preprocessing.image.load_img(IMAGE_PATH, target_size=(224, 224))
img = tf.keras.preprocessing.image.img_to_array(img)

model = tf.keras.applications.vgg16.VGG16(weights="imagenet", include_top=True)

#get model summary
model.summary()

#first create the input in a format that the explainer expects (a tuple)
input_img = (np.array([img]), None)

#initialize the explainer as an instance of the GradCAM object
explainer = GradCAM()

# Obtain explanations for your image using VGG 16 and GradCAM
grid = explainer.explain(input_img,
                         model,
                         class_index=class_index
                         )

#save the resulting image
explainer.save(grid, "./outputs/explain/", "grad_cam_cat.png")
```

TF_EXPLAIN

```
!pip install tf_explain
!pip install opencv-python

#load libraries
import numpy as np
import tensorflow as tf
import PIL

#load GradCAM
from tf_explain.core.grad_cam import GradCAM

IMAGE_PATH = "./assets/images/cat.jpg"
class_index = 281

img = tf.keras.preprocessing.image.load_img(IMAGE_PATH, target_size=(224, 224))
img = tf.keras.preprocessing.image.img_to_array(img)

model = tf.keras.applications.vgg16.VGG16(weights="imagenet", include_top=True)

#get model summary
model.summary()

#first create the input in a format that the explainer expects (a tuple)
input_img = (np.array([img]), None)

#initialize the explainer as an instance of the GradCAM object
explainer = GradCAM()

# Obtain explanations for your image using VGG 16 and GradCAM
grid = explainer.explain(input_img,
                         model,
                         class_index=class_index
                         )

#save the resulting image
explainer.save(grid, "./outputs/explain/", "grad_cam_cat.png")
```

```
(150 sloc)  |  6.14 KB

e Module for Grad CAM Algorithm

ort numpy as np
ort tensorflow as tf
ort cv2

m tf_explain.utils.display import grid_display, heatmap_display
m tf_explain.utils.saver import save_rgb


ss GradCAM:

 """
 Perform Grad CAM algorithm for a given input

 Paper: [Grad-CAM: Visual Explanations from Deep Networks
        via Gradient-based Localization](https://arxiv.org/abs/1610.
 """

 def explain(
     self,
     validation_data,
     model,
     class_index,
     layer_name=None,
```

SOURCE CODE

💻 https://github.com/sicara/tf-explain

- → [x] gradCAM
- → [x] smoothGrad
- → [x] guided smoothgrad
- → [x] integrated gradients
- → [x] occlusion sensitivity
- → [x] activations visualization

this is an active area of research, so expect more methods to be added in the future

THE NIGERIAN PRINCE

MOVING BEYOND FEATURE ATTRIBUTION

- → deep neural networks are notoriously sensitive to small perturbations in the input
- → by intentionally introducing small perturbations in the input, we can fool the model into making a different prediction
- → the nigerian prince scam is a classic example of this, where the scammer intentionally introduces small perturbations in the email to fool the spam filter into predicting that the email is not spam

ADVERSARIAL TRAINING vs ADVERSARIAL ATTACKS

→ Using **complete knowledge** of the model architecture, it's sources of uncertainty, it's parameters, and the training data, we can intentionally introduce adversarial examples to the model to test the robustness and reliability of the model. **adversarial training**. Used to improve the network's ability to make accurate predictions in real-world scenarios.

An example of adversarial training is training a neural network to recognize handwritten digits by incorporating adversarial examples of slightly modified digits to the training data.

→ Using **incomplete knowledge** of the model architecture, it's sources of uncertainty, it's parameters, and the training data, we can intentionally try to manipulate the input data to cause it to make incorrect predictions. **adversarial attacks**. Used to identify vulnerabilities in the network and to improve security.

An example of an adversarial attack is adding a small amount of noise to an image of a stop sign in order to make it appear as a go sign to an autonomous vehicle's image recognition system.


ADVERSARIAL ATTACKS
99.9% CAT
PAPERS
WITH KÁROLY ZSOLNAI-FEHÉR

SUMMARY

→ **XAI** is a field of research that aims to make machine learning models more interpretable and explainable

→ **linear models** are intrinsically interpretable because they are linear functions of the input features

→ **decision trees** are intrinsically interpretable because they are a series of if-then-else statements

→ **neural networks** are not intrinsically interpretable because they are non-linear functions of the input features. They need to be **interpreted** for high-stakes decision making applications.

→ **quantifying uncertainty** is a key component of XAI regardless of the model architecture

→ **gradCAM** is a technique for visualizing the activations of a CNN layer by producing a heatmap that highlights the regions of an input image that are most relevant to a particular output class

→ **tf_explain** is one among several python library that makes it easy to use XAI techniques in your machine learning projects

→ **adversarial attacks** and **adversarial training** are two different approaches to testing the reliability and robustness of machine learning models

YOU CAN CHOOSE TO GO DEEPER IN A TOPIC OF YOUR CHOICE

THANK YOU!

@nbhushan