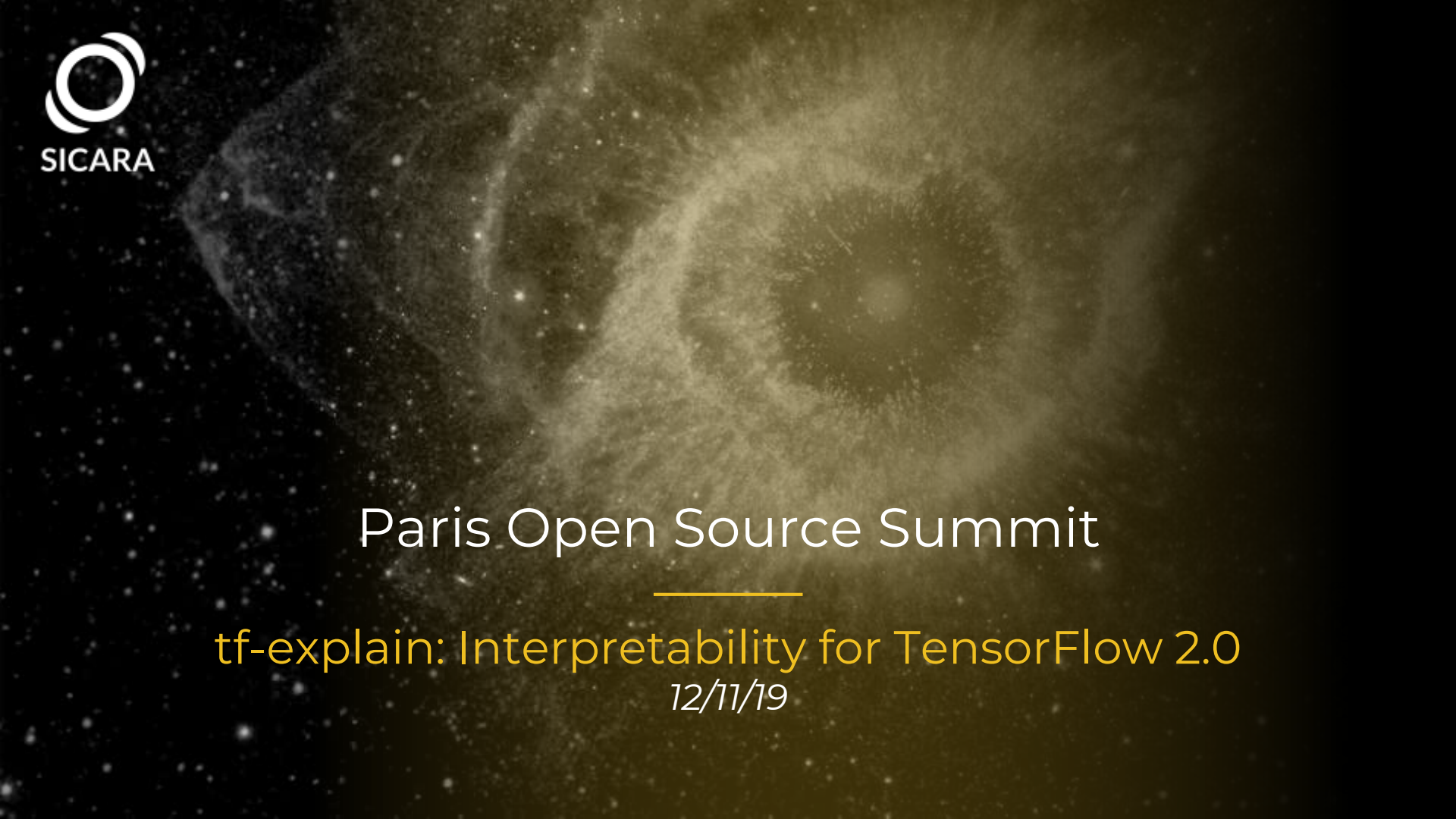


Paris Open Source Summit

tf-explain: Interpretability for TensorFlow 2.0

12/11/19

cofondateur & CTO

pierrehenric@sicara.com

@cpierrehenri

Thanks to

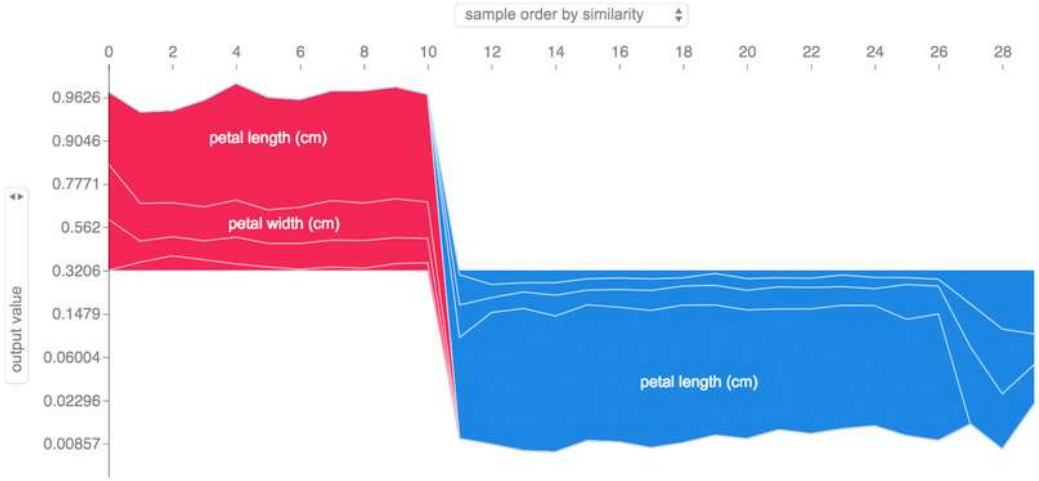
Lead Data Scientist Sicara
raphaelm@sicara.com
@raphaelmeudec

tf-explain

Introduction

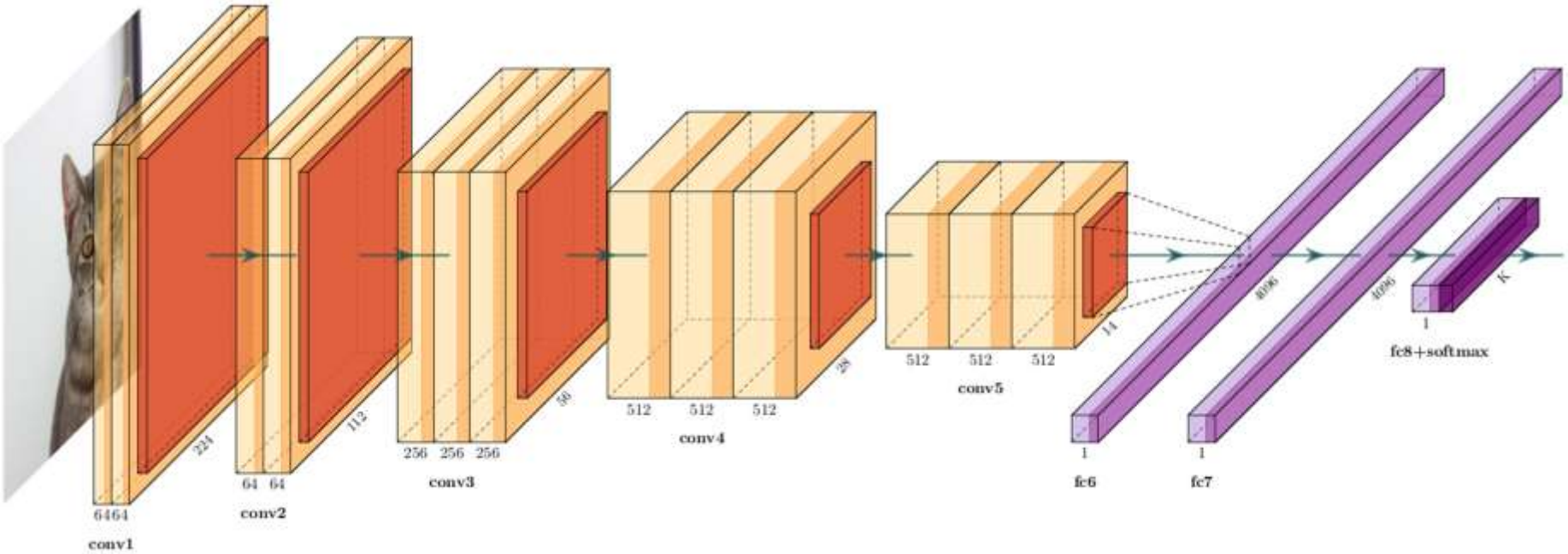
Interpretability
Linear models

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8



SICARA

Interpretability
Deep networks



SICARA

Interpretability

Provide explanations on a neural network's decisions

SICARA


TF2.0

TF 2.0
Keras as the main entrypoint

Stable release: 10/01/2019

Sequential API

Functional API

Subclassing API

SICARA

Gradient Descent

2.0 - Record operations in tf.GradientTape context

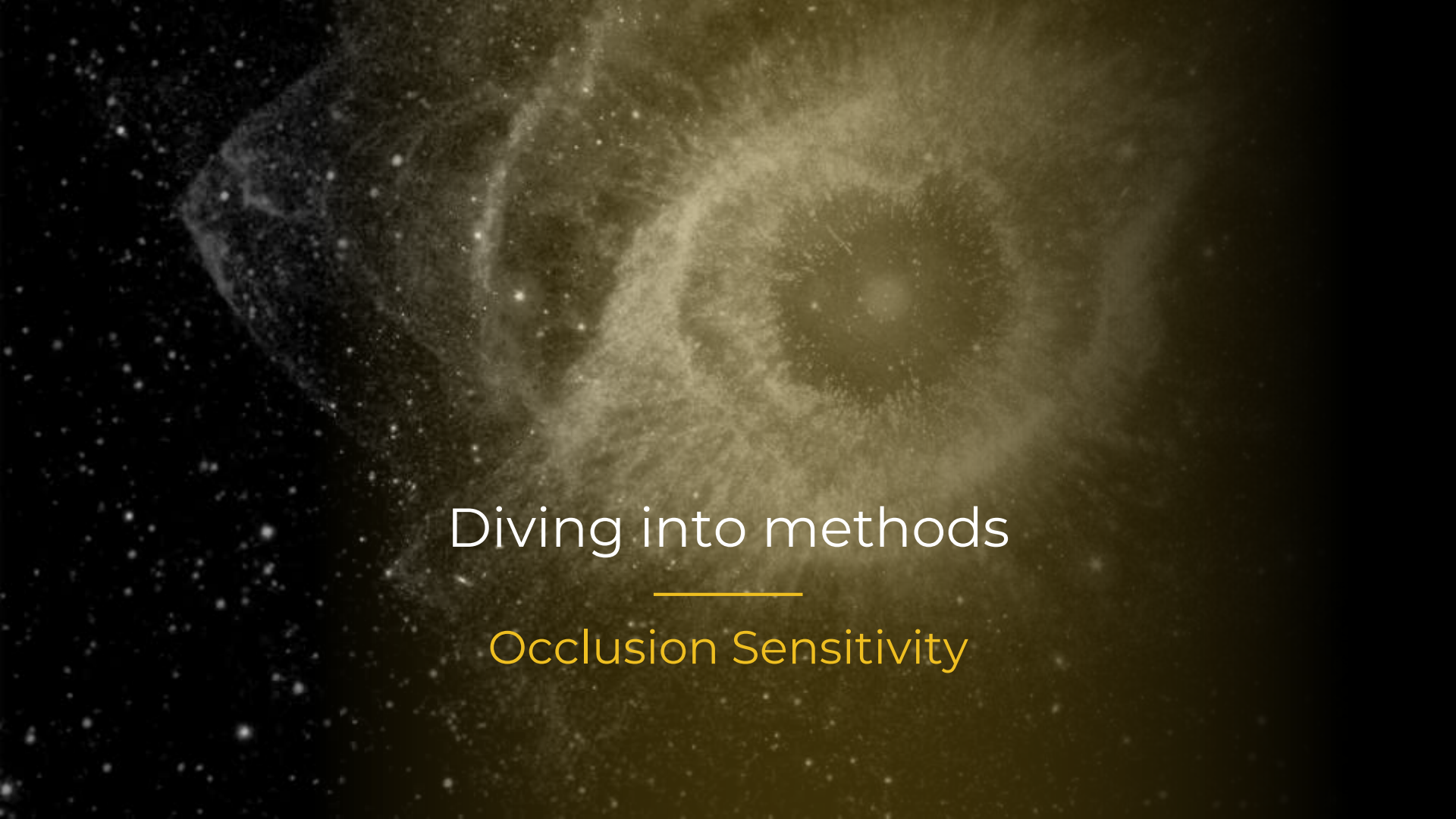
```python
# TensorFlow 2.0
import numpy as np
import tensorflow as tf

num_samples = 4
dimension = 3

x = tf.convert_to_tensor(np.random.random((num_samples, dimension)), dtype=tf.float32)
y = tf.one_hot(np.random.randint(2, size=(num_samples,)), depth=2)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(2, activation='softmax', input_shape=(dimension,))
])

model.compile(loss='binary_crossentropy', optimizer='adam')

optimizer = tf.keras.optimizers.Adam()

with tf.GradientTape() as tape:
    output = model(x)
    loss = tf.losses.binary_crossentropy(output, y)
    print(loss)
    # >> tf.Tensor([ 9.7544365  7.3376875 10.41608   10.882192 ], shape=(4,), dtype=float32)

grads = tape.gradient(loss, model.trainable_variables)

optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

Create model

Compute loss in tf.GradientTape context

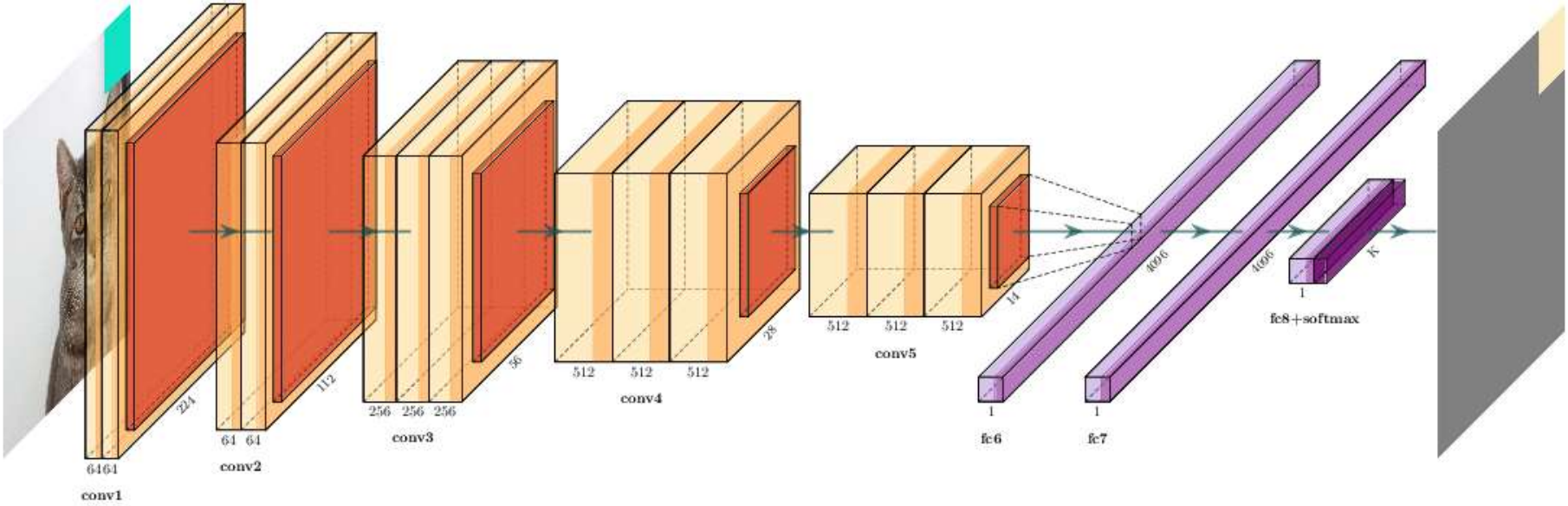
Run optimizer with recorded gradients

SICARA


Diving into methods
Occlusion Sensitivity

Method #1: Occlusion Sensitivity

Hide parts iteratively to determine its importance

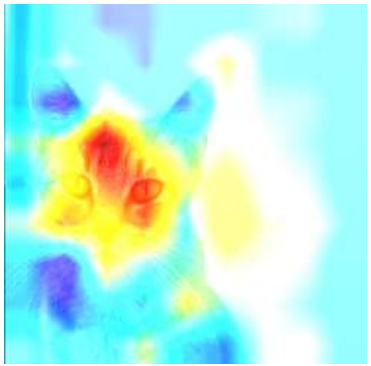


SICARA

Method #1: Occlusion Sensitivity

Hide parts iteratively to determine its importance

```python
import numpy as np
import tensorflow as tf
from tf_explain.utils.image import apply_grey_patch

IMAGE_PATH = "./img/cat.jpg"
PATCH_SIZE = 10
TABBY_CAT_CLASS_INDEX = 281

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

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

patches = [
    apply_grey_patch(img, top_left_x, top_left_y, PATCH_SIZE)
    for index_x, top_left_x in enumerate(range(0, img.shape[0], PATCH_SIZE))
    for index_y, top_left_y in enumerate(range(0, img.shape[1], PATCH_SIZE))
]

predictions = model.predict(np.array(patches), batch_size=16)
target_class_predictions = [
    prediction[TABBY_CAT_CLASS_INDEX] for prediction in predictions
]

coordinates = [
    (index_y, index_x)
    for index_x, _ in enumerate(range(0, img.shape[0], PATCH_SIZE))
    for index_y, _ in enumerate(range(0, img.shape[1], PATCH_SIZE))
]

sensitivity_map = np.zeros((img.shape[0] // PATCH_SIZE, img.shape[1] // PATCH_SIZE))
for (index_y, index_x), confidence in zip(
        coordinates, target_class_predictions
):
    sensitivity_map[index_y, index_x] = 1 - confidence
```

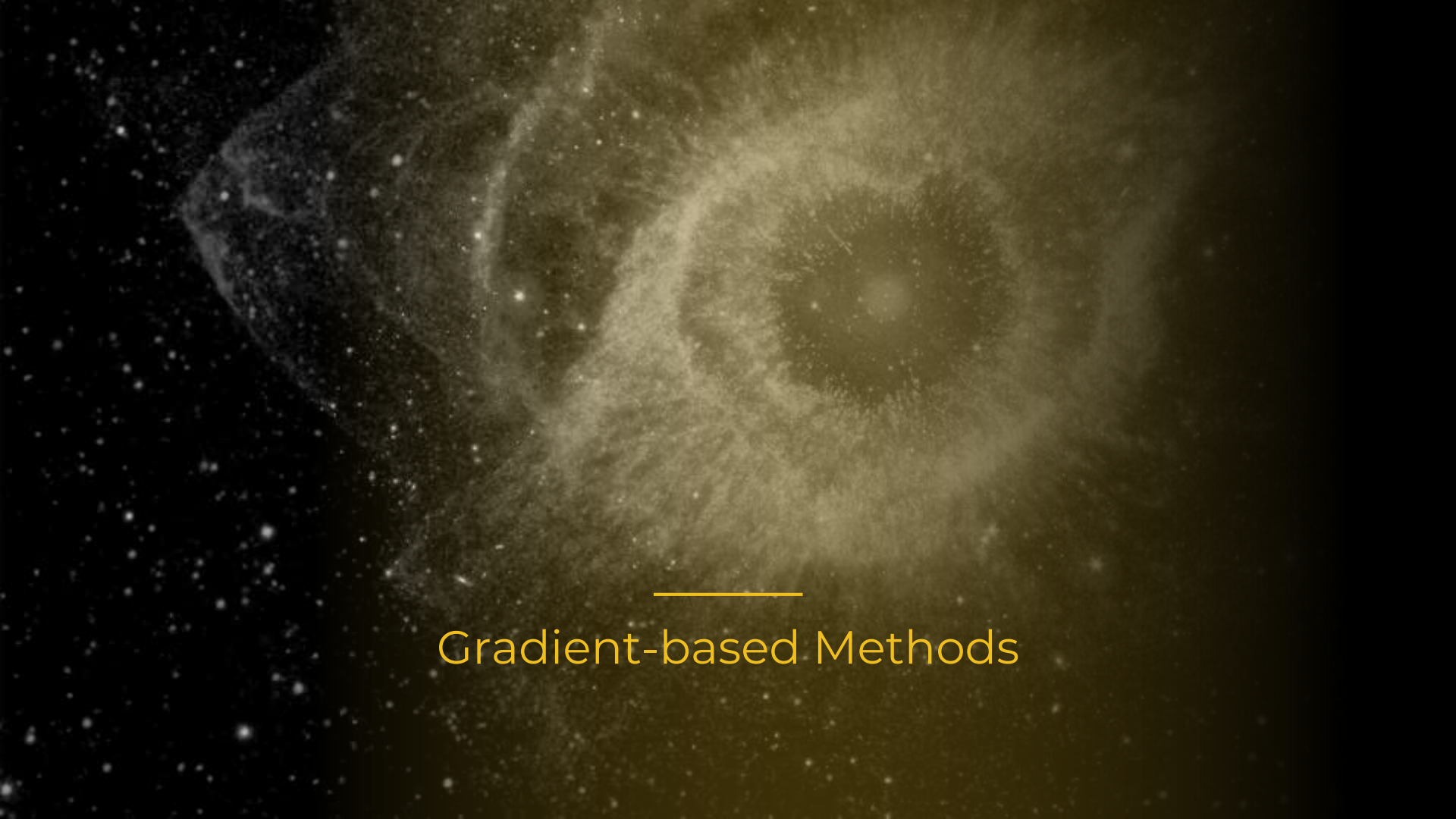


Create patched images

Predict patches

Create output map

SICARA

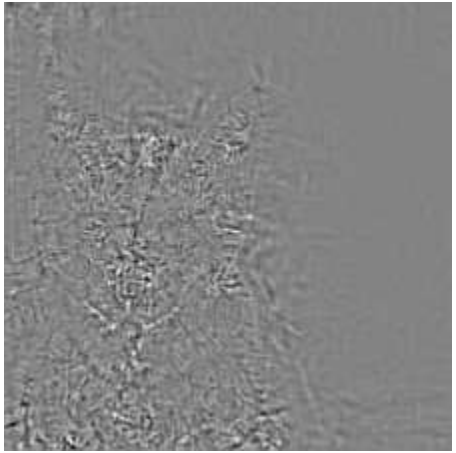


Gradient-based Methods

Method #2: Vanilla Gradients

Input image

Gradients for
“Cat” class



Understanding Neural Networks Through Deep Visualization | Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, Hod Lipson

SICARA

Method #2: Vanilla Gradients

```python
import tensorflow as tf

IMAGE_PATH = './img/cat.jpg'
TABBY_CAT_CLASS_INDEX = 281

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

# Load model
model = tf.keras.applications.vgg16.VGG16(weights='imagenet', include_top=True)

num_classes = model.output.shape[1]
expected_output = tf.one_hot([TABBY_CAT_CLASS_INDEX], num_classes)

# Compute Gradients
with tf.GradientTape() as tape:
    inputs = tf.cast([img], tf.float32)
    tape.watch(inputs)
    predictions = model(inputs)
    loss = tf.keras.losses.categorical_crossentropy(
        expected_output, predictions
    )

gradients = tape.gradient(loss, inputs)[0]
```

Load sample data

Define expected output as one hot vector

Compute gradients with respect to the expected class

SICARA

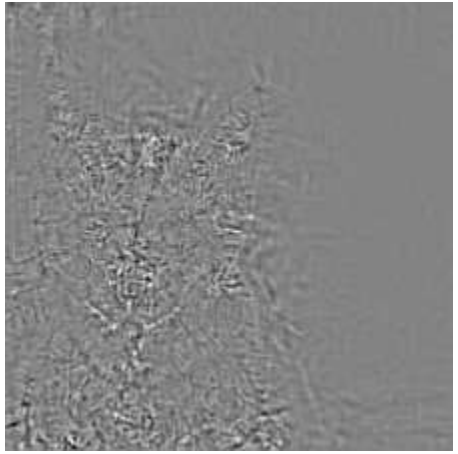

Flaws in Interpretability

What's the point?

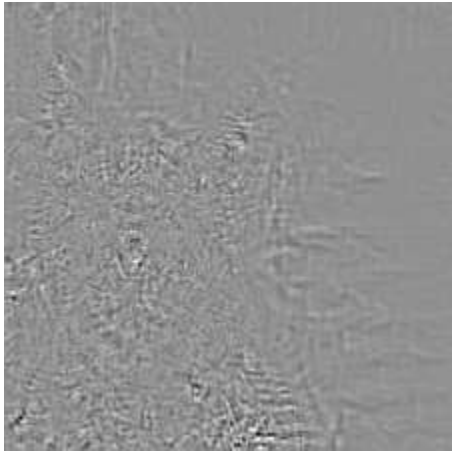
Gradients are open to interpretation

Input image

Gradients for "Cat" class

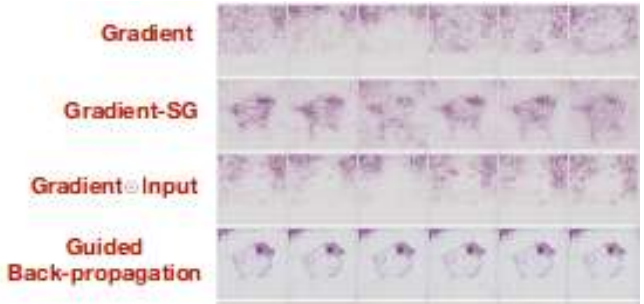


Gradients for "Kite" class

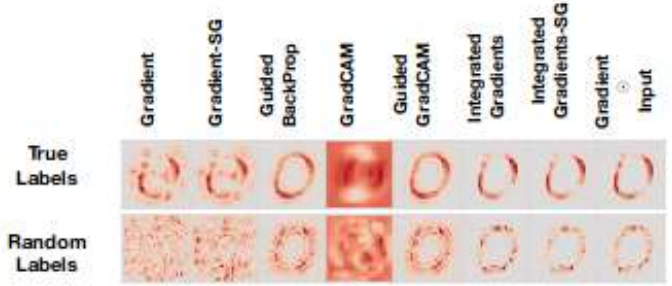


SICARA

Checks for saliency maps



Parameters Randomization



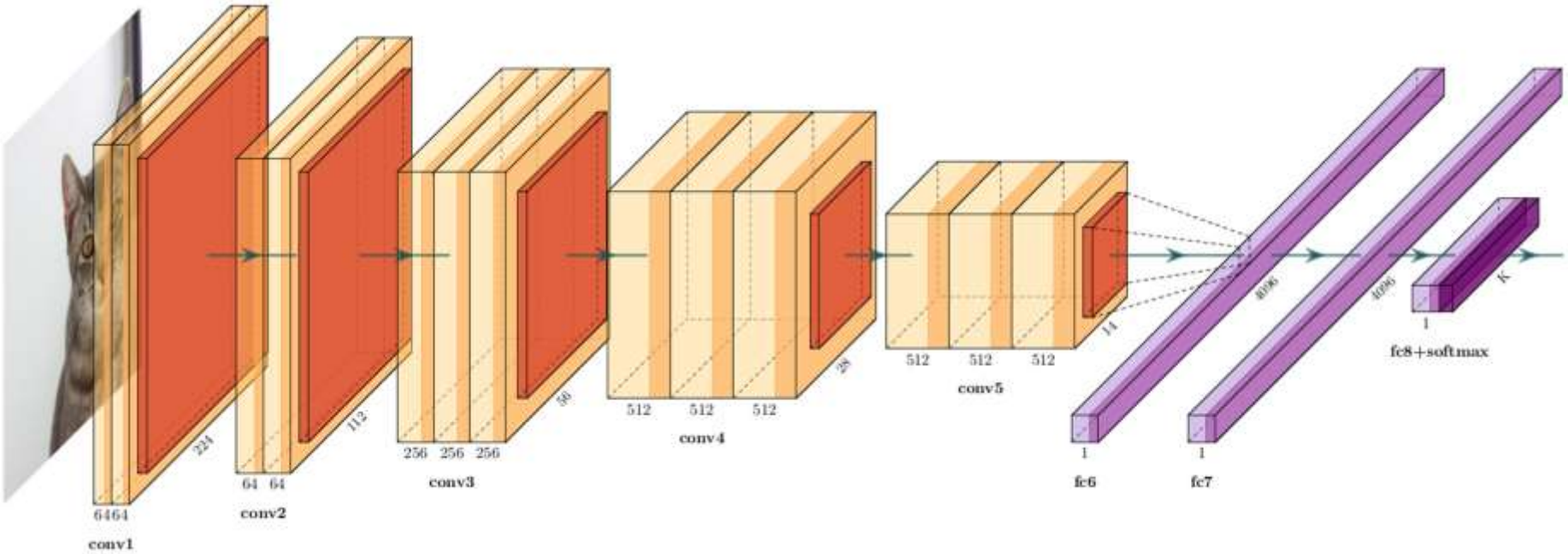
Data Randomization

[Sanity Checks for Saliency Maps] | Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, Been Kim

SICARA


Grad CAM

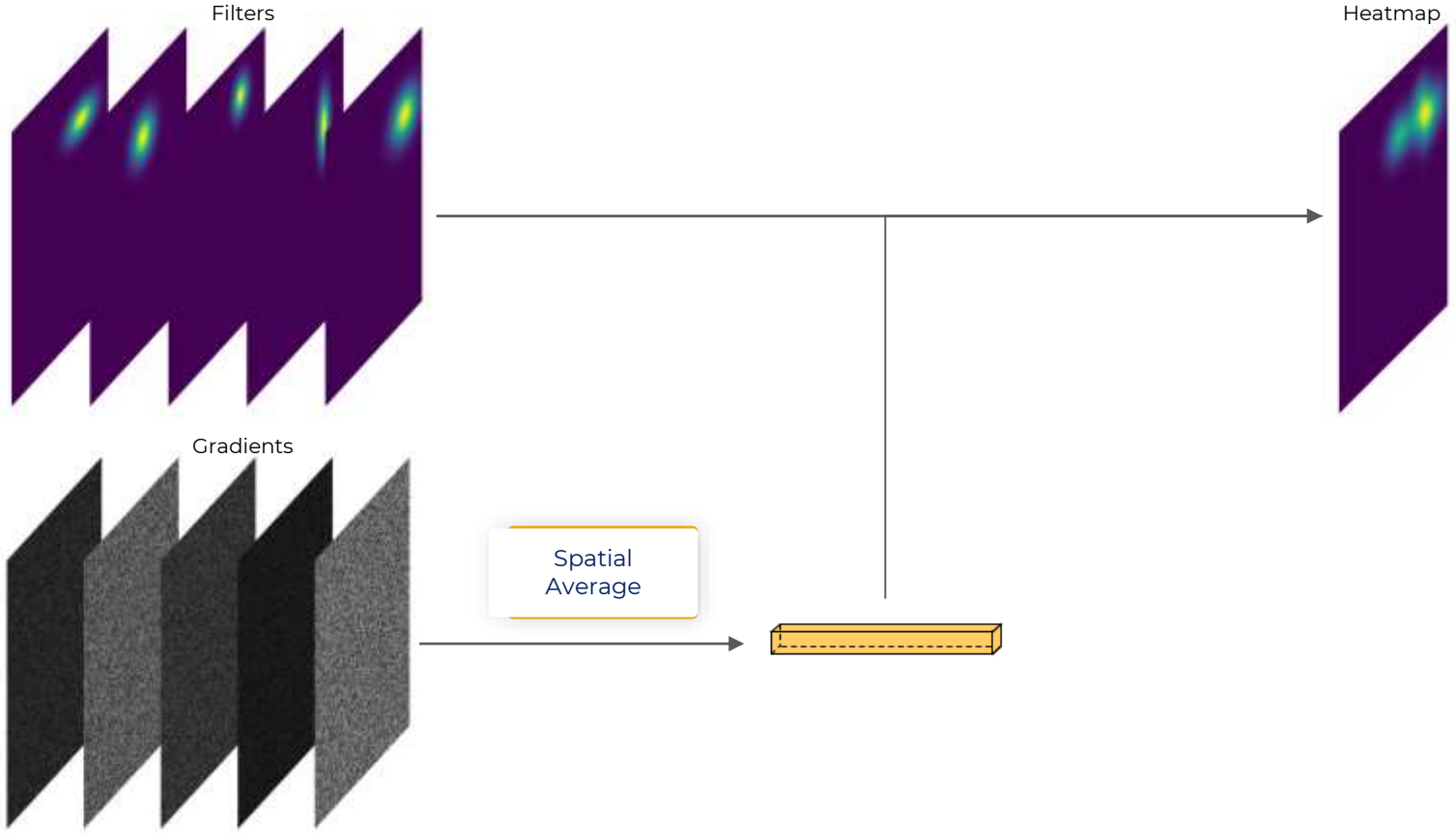
Method #3: Grad CAM



Grad-CAM: Gradient-weighted Class Activation Mapping | Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra

SICARA

Method #3: Grad CAM



Grad-CAM: Gradient-weighted Class Activation Mapping | Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, Dhruv Batra

SICARA

Method #3: Grad CAM

```python
import cv2
import tensorflow as tf


IMAGE_PATH = "../img/cat.jpg"
TABBY_CAT_CLASS_INDEX = 281
NON_TABBY_CAT_CLASS_INDEX = 21  # Class "Kite"

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

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

grad_cam_model = tf.keras.Model(
    [model.inputs], [model.get_layer("block5_conv3").output, model.output]
)

# Compute Gradients
with tf.GradientTape() as tape:
    inputs = tf.cast([img], tf.float32)
    conv_output, predictions = grad_cam_model(inputs)
    loss = predictions[:, NON_TABBY_CAT_CLASS_INDEX]

# Get Gradients
grads = tape.gradient(loss, conv_output)[0]

# Compute Class Activation Maps
weights = tf.reduce_sum(grads, axis=(0, 1))
gradcam = tf.reduce_sum(tf.multiply(weights, conv_output), axis=-1)[0].numpy()
```

Create subgraph to also extract convolutional outputs

Get gradients for target class

Compute gradients

Ponderate Class Activation Maps

SICARA

Method #3: Grad CAM

Input image

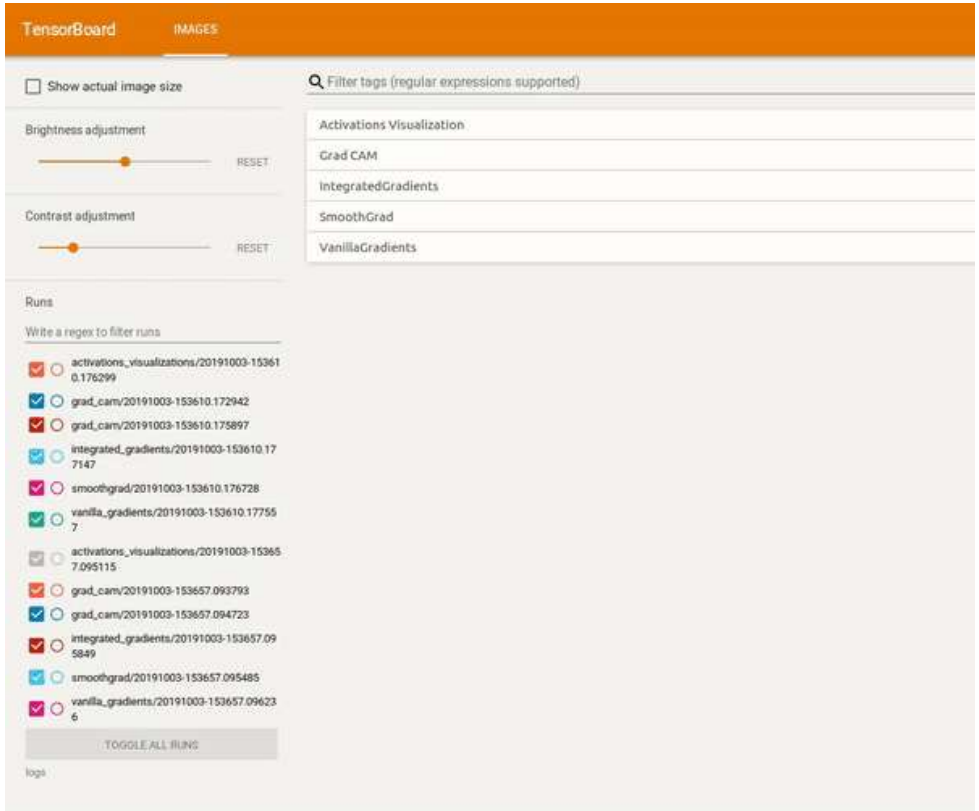
GradCAM for
“Cat” class

GradCAM for
“Kite” class

SICARA

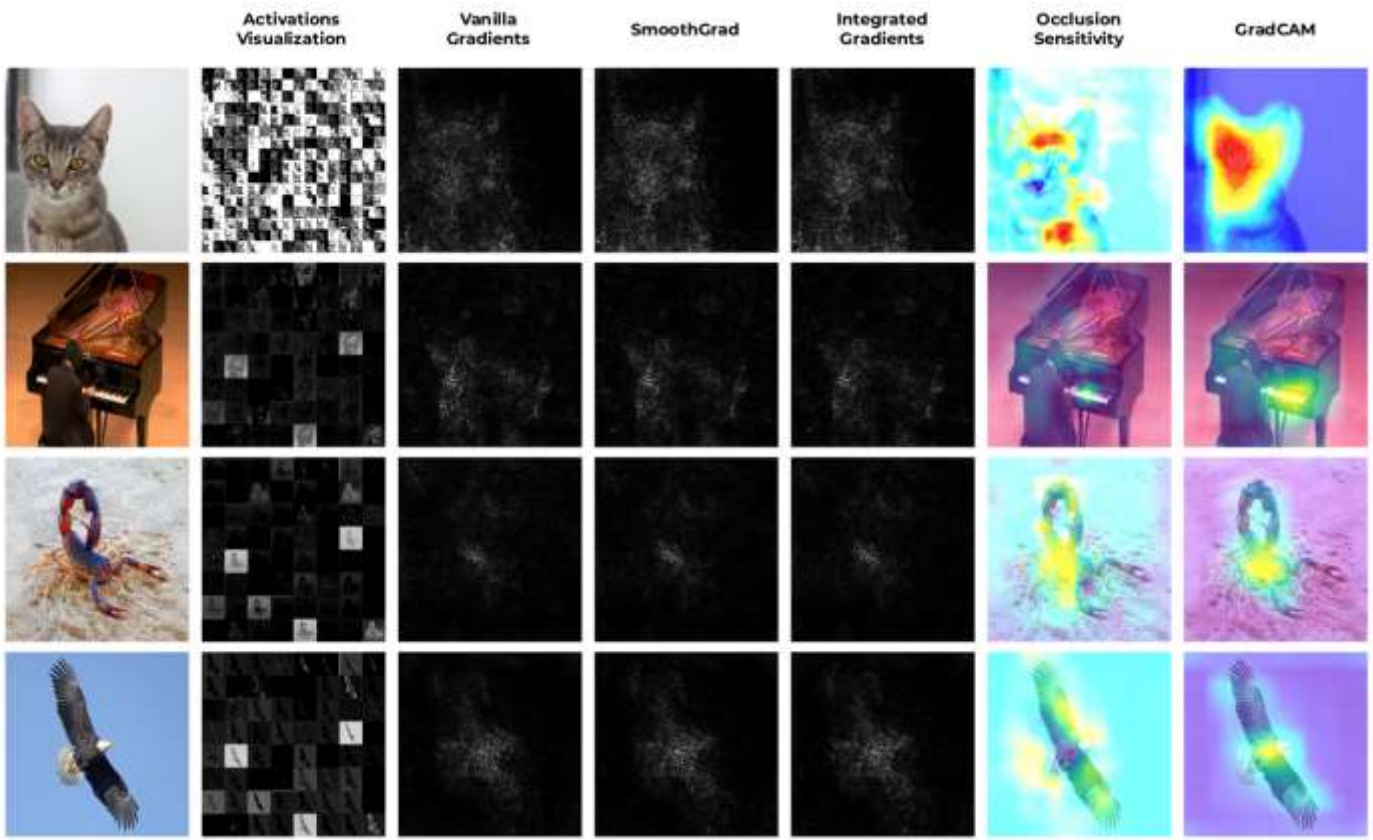

tf-explain

tf-explain
An interpretability library for TF2.0



SICARA

Methods implemented



SICARA

Uses
tf_explain.core

```python
import tensorflow as tf

from tf_explain.core.grad_cam import GradCAM

IMAGE_PATH = './cat.jpg'

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

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

model.summary()
data = ([img], None)

tabby_cat_class_index = 281

explainer = GradCAM()
grid = explainer.explain(data, model, 'block5_conv3', tabby_cat_class_index)
explainer.save(grid, '.', 'grad_cam.png')
```

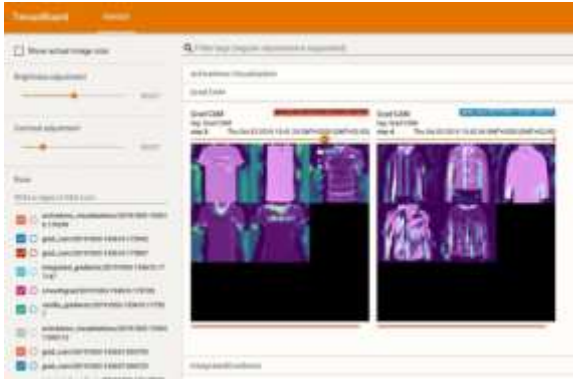
Load model and validation data

Instantiate explainer
Call .explain() method

SICARA

Uses
tf_explain.callbacks

```python
import numpy as np
import tensorflow as tf
import tf_explain

INPUT_SHAPE = (28, 28, 1)
NUM_CLASSES = 10

(train_images, train_labels), (test_images, test_labels) = tf.keras.datasets.mnist.load_data()

model = ...

# Select a subset of the validation data to examine
# Here, we choose 5 elements with label "0" == [1, 0, 0, ..., 0]
validation_class_zero = (np.array([
    el for el, label in zip(test_images, test_labels)
    if np.all(label == np.array([1] + [0] * 9))
][0:5]), None)

# Instantiate callbacks
# class_index value should match the validation_data selected above
callbacks = [tf_explain.callbacks.GradCAMCallback(validation_class_zero, 'target_layer', class_index=0)]

# Start training
model.fit(train_images, train_labels, epochs=5, callbacks=callbacks)
```



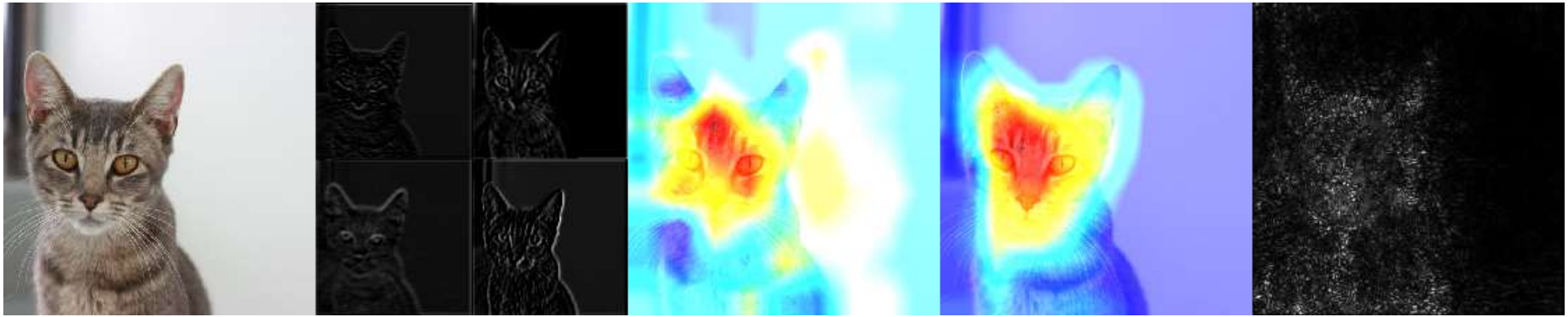
Load dataset of a specific class index

Create callback

Call .fit() with callback

SICARA

Available on PyPi



```
>> pip install tf-explain
```


SICARA

Github: **github.com/sicara/tf-explain**

Twitter: **@cpierrehenri / @raphaelmeudec**

Thank you