tf-explain

Dec 27, 2019

Contents

tf-explain offers interpretability methods for Tensorflow 2.0 to ease neural network's understanding. With either its core API or its tf.keras callbacks, you can get a feedback on the training of your models.

CHAPTER 1

Overview

1.1 Installation

tf-explain is available on Pypi as an alpha release. To install it:

```
pip install tf-explain
```

1.2 Tensorflow Compatibility

tf-explain is compatible with Tensorflow 2. It is not declared as a dependency to let you choose between CPU and GPU versions. Additionally to the previous install, run:

```
# For CPU version
pip install tensorflow==2.0.0
# For GPU version
pip install tensorflow-gpu==2.0.0
```

CHAPTER 2

Usage

tf-explain implements methods you can use at different levels:

- either on a loaded model with the core API (which saves outputs to disk)
- either at training time with callbacks (which integrates into Tensorboard)

This section introduces both usages.

2.1 Core API

All methods implemented in tf-explain keep the same interface:

- a `explain` method which outputs the explaination (for instance, a heatmap)
- a `save` method compatible with its output

Usage of the core API should be the following:

```
# Import explainer
from tf_explain.core.grad_cam import GradCAM

# Instantiation of the explainer
explainer = GradCAM()

# Call to explain() method
output = explainer.explain(*explainer_args)

# Save output
explainer.save(output, output_dir, output_name)
```

Recurrent arguments contained in `explainer_args` are typically the data to use for the explanation, the model to inspect. Refer to each method docstring to know which elements are needed.

All methods are kept inside `tf_explain.core`.

2.2 Callbacks

To use those methods during trainings and inspect evolutions over the epochs, each one of them has its corresponding `tf.keras.Callback`.

Callback usage is coherent with Keras Callbacks:

```
from tf_explain.callbacks.grad_cam import GradCAMCallback

model = [...]

callbacks = [
    GradCAMCallback(
        validation_data=(x_val, y_val),
        layer_name="activation_1",
        class_index=0,
        output_dir=output_dir,
    )
]

model.fit(x_train, y_train, batch_size=2, epochs=2, callbacks=callbacks)
```

Then, launch Tensorboard and visualize the outputs in the Images section.

CHAPTER 3

Available Methods

3.1 Activations Visualization

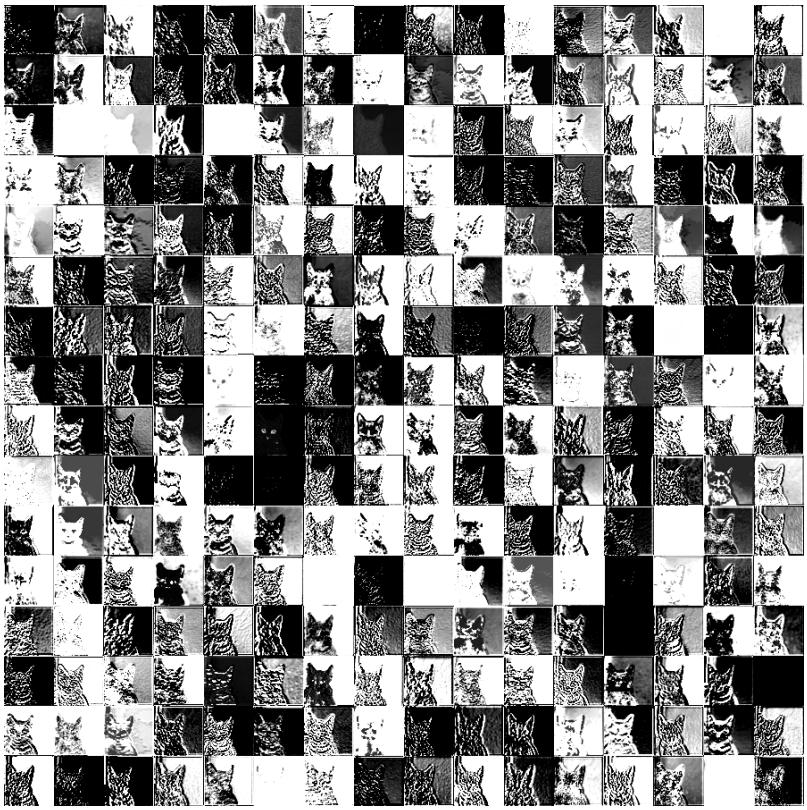
Visualize how a given input comes out of a specific activation layer

```
from tf_explain.callbacks.activations_visualization import␣
↪ActivationsVisualizationCallback

model = [...]

callbacks = [
    ActivationsVisualizationCallback(
        validation_data=(x_val, y_val),
        layers_name=["activation_1"],
        output_dir=output_dir,
    ),
]

model.fit(x_train, y_train, batch_size=2, epochs=2, callbacks=callbacks)
```



3.2 Vanilla Gradients

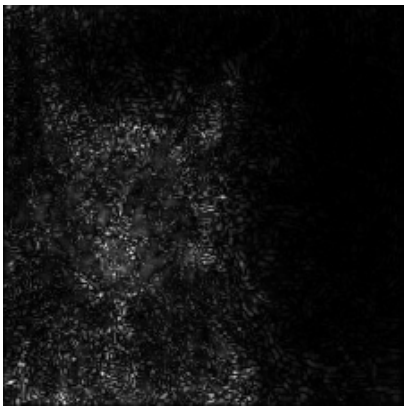
Visualize gradients on the inputs towards the decision.

From Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps

```
from tf_explain.callbacks.gradients import VanillaGradients

model = [...]
callbacks = [
    VanillaGradients(
        validation_data=(x_val, y_val),
        class_index=0,
        output_dir=output_dir,
    )
]

model.fit(x_train, y_train, batch_size=2, epochs=2, callbacks=callbacks)
```



3.3 Occlusion Sensitivity

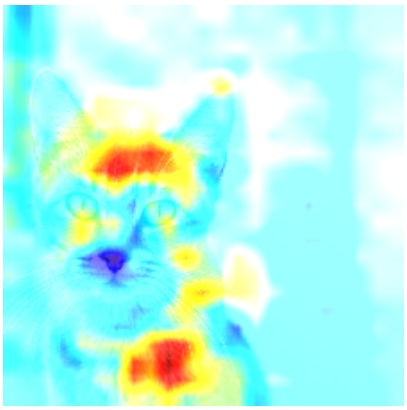
Visualize how parts of the image affects neural network's confidence by occluding parts iteratively

```
from tf_explain.callbacks.occlusion_sensitivity import OcclusionSensitivityCallback

model = [...]

callbacks = [
    OcclusionSensitivityCallback(
        validation_data=(x_val, y_val),
        class_index=0,
        patch_size=4,
        output_dir=output_dir,
    ),
]

model.fit(x_train, y_train, batch_size=2, epochs=2, callbacks=callbacks)
```



3.4 Grad CAM

Visualize how parts of the image affects neural network's output by looking into the activation maps

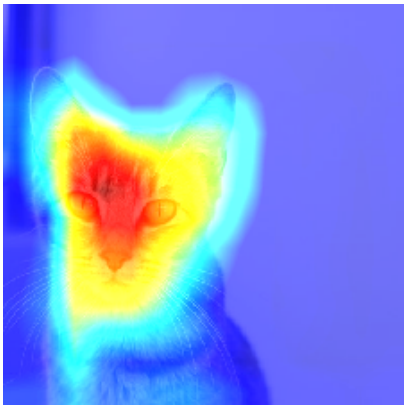
From Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

```
from tf_explain.callbacks.grad_cam import GradCAMCallback

model = [...]

callbacks = [
    GradCAMCallback(
        validation_data=(x_val, y_val),
        layer_name="activation_1",
        class_index=0,
        output_dir=output_dir,
    )
]

model.fit(x_train, y_train, batch_size=2, epochs=2, callbacks=callbacks)
```



3.5 SmoothGrad

Visualize stabilized gradients on the inputs towards the decision.

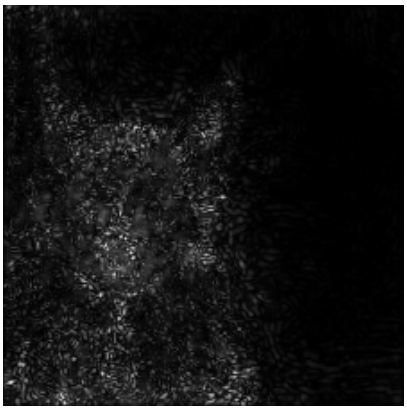
From SmoothGrad: removing noise by adding noise

```
from tf_explain.callbacks.smoothgrad import SmoothGradCallback

model = [...]

callbacks = [
    SmoothGradCallback(
        validation_data=(x_val, y_val),
        class_index=0,
        num_samples=20,
        noise=1.,
        output_dir=output_dir,
    )
]

model.fit(x_train, y_train, batch_size=2, epochs=2, callbacks=callbacks)
```



3.6 Integrated Gradients

Visualize an average of the gradients along the construction of the input towards the decision.

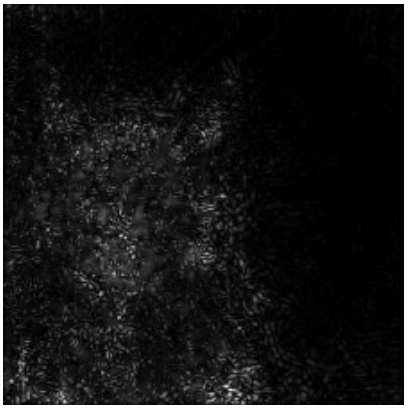
From Axiomatic Attribution for Deep Networks

```
from tf_explain.callbacks.integrated_gradients import IntegratedGradientsCallback

model = [...]

callbacks = [
    IntegratedGradientsCallback(
        validation_data=(x_val, y_val),
        class_index=0,
        n_steps=20,
        output_dir=output_dir,
    )
]

model.fit(x_train, y_train, batch_size=2, epochs=2, callbacks=callbacks)
```



CHAPTER 4

API

4.1 tf_explain.callbacks package

4.1.1 Submodules

4.1.2 tf_explain.callbacks.activations_visualization module

4.1.3 tf_explain.callbacks.grad_cam module

4.1.4 tf_explain.callbacks.integrated_gradients module

4.1.5 tf_explain.callbacks.occlusion_sensitivity module

4.1.6 tf_explain.callbacks.smoothgrad module

4.1.7 Module contents

4.2 tf_explain.core package

4.2.1 Submodules

4.2.2 tf_explain.core.activations module

4.2.3 tf_explain.core.grad_cam module

4.2.4 tf_explain.core.integrated_gradients module

4.2.5 tf_explain.core.occlusion_sensitivity module

4.2.6 tf_explain.core.smoothgrad module

4.2.7 Module contents

4.3 tf_explain.utils package

CHAPTER 5

Contributing

Contributions are welcome on this repo! Follow this guide to see how you can help.

5.1 What can I do?

There are multiple ways to give a hand on this repo:

- resolve issues already opened
- tackle new features from the roadmap
- fix typos, improve code quality, code coverage

5.2 Guidelines

5.2.1 Tests

tf-explain is run against Python 3.6 and 3.7, for Tensorflow beta. We use tox (available with `pip`) to perform the tests. All the submitted code should be unit tested (we use pytest).

To run all the tests, run `tox` in a terminal.

5.2.2 Code Format

All code is formatted with Black (available with `pip`). When opening your PR, make sure your code is formatted or Travis will fail. To format your code, simply call `make black`.

CHAPTER 6

Roadmap

Next features are listed as issues with the `roadmap` label.