# quanda: An Interpretability Toolkit for Training Data Attribution Evaluation and Beyond

Dilyara Bareeva, Galip Ümit Yolcu, Anna Hedström, Niklas Schmolenski, Thomas Wiegand, Wojciech Samek, Sebastian Lapuschkin

Fraunhofer HHI · berlin · BIFOLD · UMI LAB

NEURAL INFORMATION PROCESSING SYSTEMS

## Training Data Attribution (TDA)

**TDA methods aim to relate a model's decision on a specific sample to its training data.** Given a dataset $\mathcal{D} = \{z_1, ..., z_n\} \in \mathcal{Z}^n$, and a test sample $z \in \mathcal{Z}$, a TDA method is a function $\tau : \mathcal{Z} \times \mathcal{Z}^n \to \mathbb{R}^n$ which assigns a real valued attribution score to each training sample.

**Several approaches to TDA exist.** Major approaches include approximating the effect of Leave-one-out (LOO) training [1], and using interpretable kernel surrogates [2].

## Evaluation of TDA Methods

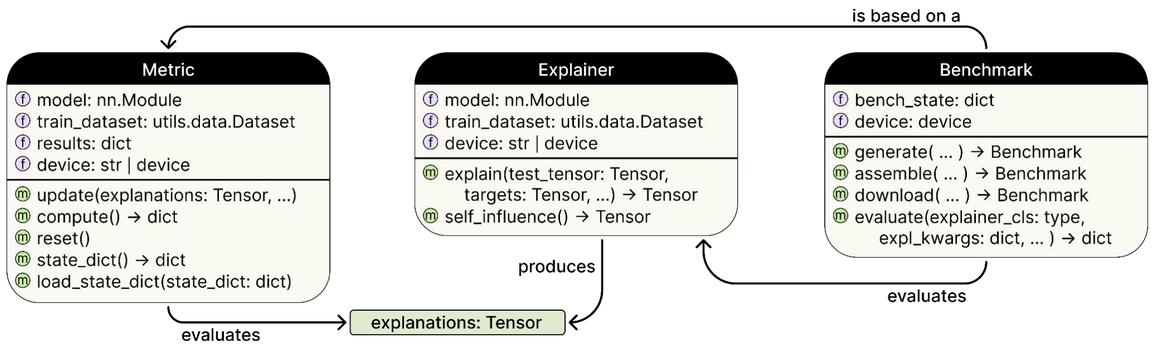
**Evaluation of attributions is a challenging task,** due to the absence of a *reliable* ground truth. Existing approaches can be summarized by the following groups:

- **Ground Truths:** Directly comparing attributions with a ground truth, (e.g. LOO retraining [1] ). This ground truth is expensive to compute, and dominated by noise due to the stochastic nature of the standard training procedures [3] . Therefore, alternative approaches were proposed.
- **Heuristics:** Measuring for desirable properties and applying sanity checks. One example metric measures the dependence of the attributions on the model parameters, by measuring the correlation between the attributions of the original model and a randomly initialized model [4] .
- **Downstream Tasks:** Evaluate the effectiveness of attributions in achieving a task in a controlled setup. The most prominent example of this kind of evaluation is mislabeling detection. This is done by changing some of the training labels randomly, training a model on this modified dataset, and finally using attributions to detect the mislabeled samples in the training dataset [1,2] .

## Library Features

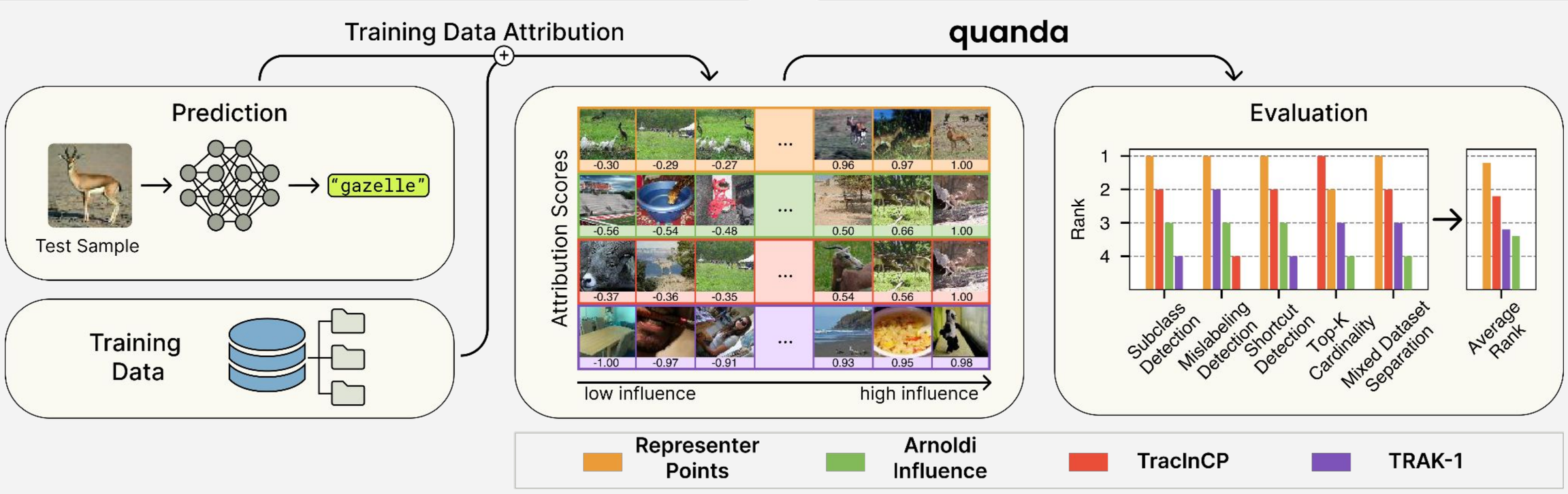
**The community lacks a comprehensive and standardized evaluation framework**, despite TDA's potential for interpretability and applications.

**quanda** strives to fill this research gap and provides an *easily extendable*, *unified* and *user-friendly* interface to practitioners on Python using PyTorch. This allows users to easily obtain a bird's eye view on the performance of attributors in different setups.




**quanda** presents the following components to its users:

- **Explainers:** Unified interface to easily wrap a separate TDA implementation. Ready to use wrappers for existing implementations of major methods.
- **Metrics:** Implementations of several evaluation metrics from the literature, which directly evaluate generated attributions.
- **Benchmarks:** Implementations of benchmarking tools, providing a seamless evaluation process. They encapsulate creation of controlled setups, model training, attribution and evaluation. Furthermore, **quanda** provides *precomputed benchmarks* for easy and standardized evaluation of different TDA methods. These benchmarks include pretrained models as required by the evaluation criteria and are currently being worked on to include new datasets.




## Basic Usage

Below is an example usage of **quanda** that generates attributions and evaluates them using a **Metric**:

```
import quanda
trak_explainer = quanda.explainers.wrappers.TRAK(
                    model=model, train_dataset=train_dataset, model_id=model_id,
                    cache_dir=cache_dir, proj_dim=2048
                )
class_detection = quanda.metrics.downstream_eval.ClassDetectionMetric(
                        model=model, train_dataset=train_dataset
                 )
for (samples, labels) in test_dataloader:
      pred_labels = model(samples).argmax(dim=1)
      tda = trak_explainer.explain(test_tensor=data, targets=pred_labels)
      class_detection.update(explanations=tda, test_labels=pred_labels)
print("Class Detection metric output: ", class_detection.compute())
```

We can also use **Benchmarks** to use precomputed assets for evaluation:

```
import quanda
benchmark = quanda.benchmarks.downstream_eval.ClassDetection.download(
                name="mnist_class_detection", cache_dir=cache_dir, device=device
            )
trak_args = {
      "model_id": "mnist_class_detection",
      "cache_dir": cache_dir,
      "proj_dim": 2048,
}
score = benchmark.evaluate(
                explainer_cls=quanda.explainers.wrappers.TRAK,
                expl_kwargs=trak_args
        )["score"]
```

## Library Maintenance and Code Quality

ruff checked · mypy checked · docs passing · codecov 92% · license MIT · pypi v0.0.2

**quanda** is completely open-sourced, thoroughly tested. We apply linting and type checking to ensure code quality and functionality. Detailed documentation is available in quanda.readthedocs.io along with guides and tutorials on **quanda**'s different use cases.

## Learn More

- **Correspondence**:
  sebastian.lapuschkin@hhi.fraunhofer.de wojciech.samek@hhi.fraunhofer.de
  dilyara.bareeva@hhi.fraunhofer.de galip.uemit.yolcu@hhi.fraunhofer.de
- **Documentation**
- **Repository**
- **Paper**

**References**

[1] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Proceedings of the 34th ICML, vol. 70 of Proceedings of Machine Learning Research, p. 1885–1894. PMLR, 06–11 Aug 2017.

[2] Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. In Advances in NeurIPS, vol. 31. 2018.

[3] Elisa Nguyen, Seo Minjoon and Seong Joon Oh. A Bayesian approach to analysing training data attribution in deep learning. In Advances in NeurIPS, vol 36., 2024

[4] Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. Evaluation of similarity-based explanations. In International Conference on Learning Representations, 2021.