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

Dilyara Bareeva[1*] **Galip Ümit Yolcu**[1*] **Anna Hedström**[1,2,3] **Niklas Schmolenski**[1]
Thomas Wiegand[1,3,4] **Wojciech Samek**[1,3,4†] **Sebastian Lapuschkin**[1†]
[1]Fraunhofer HHI, Berlin, Germany [2]UMI Lab, ATB Potsdam, Germany
[3]TU Berlin, Germany [4]BIFOLD, Berlin, Germany
* contributed equally
† corresponding authors:
`{wojciech.samek,sebastian.lapuschkin}@hhi.fraunhofer.de`

Abstract

In recent years, training data attribution (TDA) methods have emerged as a promising direction for the interpretability of neural networks. While research around TDA is thriving, limited effort has been dedicated to the evaluation of attributions. Similar to the development of evaluation metrics for the traditional feature attribution approaches, several standalone metrics have been proposed to evaluate the quality of TDA methods across various contexts. However, the lack of a unified framework that allows for systematic comparison limits the trust in TDA methods and stunts their widespread adoption. To address this research gap, we introduce **quanda**, a `Python` toolkit designed to facilitate the evaluation of TDA methods. Beyond offering a comprehensive set of evaluation metrics, **quanda** provides a uniform interface for seamless integration with existing TDA implementations across different repositories, thus enabling systematic benchmarking. The toolkit is user-friendly, thoroughly tested, well-documented, and available as an open-source library on PyPi and under `https://github.com/dilyabareeva/quanda`.

1 Introduction

As neural networks become increasingly complex and widely adopted, the field of *Explainable AI* (XAI) has emerged to address the need for elucidating the decision-making processes of these black-box models [Longo et al., 2024]. Initially, research in XAI predominantly focused on feature attribution methods, which highlight features in the input space that are responsible for a specific prediction [Simonyan et al., 2014, Bach et al., 2015, Lundberg and Lee, 2017, Sundararajan et al., 2017]. However, limitations in these methods' reliability [Adebayo et al., 2018, Ghorbani et al., 2019, Bilodeau et al., 2024] and their lack of informativeness [Rudin, 2019, Achtibat et al., 2023] have prompted a surge in alternative approaches. One category of such methods is *mechanistic interpretability* [Bereska and Gavves, 2024], which aims to reverse-engineer neural networks by identifying and understanding the features and their connections within the model. Similarly, *concept-based* explainability approaches seek to explain model predictions via high-level, human-understandable concepts that the model uses during inference [Poeta et al., 2023].

This study puts its focus on the interpretability approach known as *training data attribution* (TDA) [Hammoudeh and Lowd, 2024]. Broadly, TDA methods aim to relate a model's inference on a specific sample to its training data [Koh and Liang, 2017, Yeh et al., 2018, Pruthi et al., 2020, Park et al., 2023, Bae et al., 2024]. We denote by $\mathcal{D} = \{z_1, ..., z_n\} \in \mathcal{Z}^n$ an ordered set composed of n training samples, where each sample $z_i = (x_i, y_i) \in \mathcal{Z}$. Here, $x_i \in \mathbb{R}^d$ represents the model input and y_i represents the target variable. The space $\mathcal{Z}$ corresponds to the set of all possible input-target pairs and the targets' domain can vary depending on a task. For instance, in a classification setting, y_i

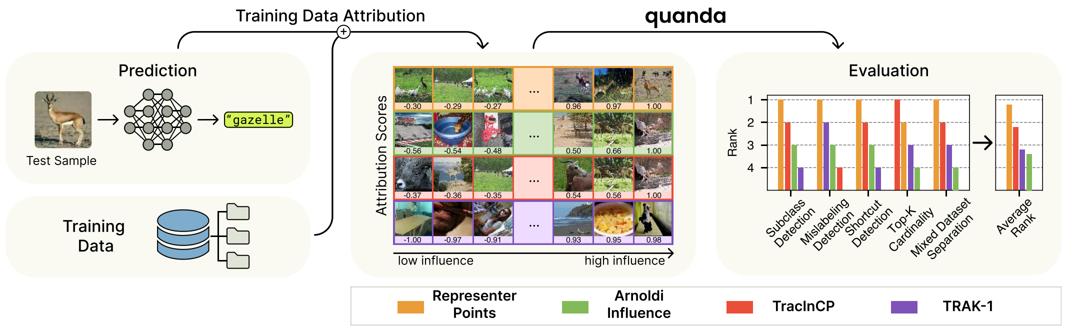

Figure 1: Overview of exemplary benchmarking results facilitated by **quanda**. The figure depicts the training data attribution (TDA) evaluation pipeline. Firstly, each TDA method generates attribution scores for test samples. Subsequently, **quanda** computes metric scores for each TDA method, assessing various aspects of attribution quality across different evaluation strategies. Each metric provides insight into a specific dimension of the attributor's performance, resulting in a comprehensive analysis of the strengths and weaknesses of each method. For a detailed experimental setup and a discussion of the results, please refer to Appendix E.

typically represents a label. Then, given a test sample $z = (x, y)$, a data attribution method defines a function $\tau : \mathcal{Z} \times \mathcal{Z}^n \to \mathbb{R}^n$ that assigns a real-valued *attribution score* to each training sample z_i. The attributed variable is typically the model output $f(z, \theta)$ or the model training loss $\mathcal{L}(z, \theta)$ on sample z. The attribution score, depending on the method, can be referred to as the influence [Koh and Liang, 2017, Pruthi et al., 2020], importance [Park et al., 2023, Bae et al., 2024], representer values [Yeh et al., 2018], or similarity [Caruana et al., 1999, Hanawa et al., 2021].

Some TDA methods aim to estimate the *counterfactual effects* of modifying the training dataset by reducing it to a subset of its samples [Koh and Liang, 2017, Koh et al., 2019, Park et al., 2023]. A classic approach for such estimation is leave-one-out (LOO) retraining [Cook and Weisberg, 1982]. LOO quantifies the influence of an individual sample by retraining the model after removing it from the training dataset and measuring the resulting change in the model's loss. Other methods leverage interpretable surrogates [Yeh et al., 2018, Yolcu et al., 2024] or kernel methods [Khanna et al., 2019], identify training samples that are deemed similar to the test sample by the model [Caruana et al., 1999, Hanawa et al., 2021], or track the influence of training data points throughout the training process [Pruthi et al., 2020].

Beyond facilitating model interpretability [Pezeshkpour et al., 2021, Naujoks et al., 2024], TDA has been used in a variety of applications, such as model debugging [Koh and Liang, 2017, Yeh et al., 2018, Guo et al., 2021], data pruning [Marion et al., 2023, Yang et al., 2023], data selection [Chhabra et al., 2024, Engstrom et al., 2024], subsampling [Ting and Brochu, 2018], machine unlearning [Warnecke et al., 2023], and fact tracing [Akyurek et al., 2022].

Although there are various demonstrations of TDA's potential for interpretability and practical applications, a comprehensive and standardized evaluation framework remains lacking. The lack of systematic evaluation procedures hinders a detailed understanding of the strengths and limitations of different TDA techniques across various contexts. Furthermore, the limited adoption and the lack of popularity of TDA methods [Nguyen et al., 2023] may in part be attributed to the absence of user-friendly, efficient, and ready-to-use tools for applying and evaluating different TDA techniques. To enhance the reliability of TDA-based explanations and task outcomes, researchers urgently need standardized implementations of various TDA explainers and unified, comprehensive evaluation tools. We address these research gaps with the release of the open-source Python package **quanda**, implemented for `PyTorch` [Paszke et al., 2019], offering the following contributions:

- A *standardized interface* for multiple TDA methods, which are currently scattered across different repositories, simplifying the application and comparison of various methods.
- Implementations of several *evaluation metrics* from the literature, enabling the assessment of explanation quality and downstream-task performance.
- Unified *benchmarking tools*, facilitating metrics-based evaluation of TDA methods in controlled environments.

- A set of standardized precomputed *evaluation benchmark suites* to ensure a reproducible, systematic, and reliable assessment of TDA methods, ready to use out-of-the-box. These benchmark suites include modified datasets and pre-trained models whenever applicable, allowing the user to skip the creation of controlled setups and directly initiate the evaluation.

2 Related Works

The need to assess the reliability of explanations and select the most suitable TDA method for a given intent and application raises the critical question of how TDA methods should be effectively evaluated. As some of the methods are designed to approximate LOO effects, *ground truth* can often be computed for TDA evaluation [e.g., Koh and Liang, 2017, Koh et al., 2019, Basu et al., 2021, Park et al., 2023]. However, this counterfactual ground truth approach requires retraining the model multiple times on different subsets of the training data, which quickly becomes computationally expensive. Furthermore, this ground truth is shown to be dominated by noise in practical deep learning settings, due to the inherent stochasticity of a typical training process [Nguyen et al., 2023]. The model parameters after optimization are often sensitive to hyperparameters and initialization, introducing further uncertainty, as discussed in Appendix A of Bae et al. [2024].

To address these challenges, alternative approaches have been introduced to make the evaluation of TDA methods practically feasible. *Downstream task evaluators* assess the utility of a TDA method within the context of an end-task, such as model debugging or data selection [e.g., Koh and Liang, 2017, Yeh et al., 2018, Pruthi et al., 2020]. *Heuristics* evaluate whether a TDA explanation meets certain expected properties, such as its dependence on model weights or the variance of the explanations for different test samples [e.g., Barshan et al., 2020, Hanawa et al., 2021, Singla et al., 2023]. However, these evaluation strategies, developed independently by various researchers, often rely on distinct assumptions and implementational nuances that can affect their outcomes. As a result, the evaluation results reported in independent studies are frequently not directly comparable.

There exist well-established libraries in the broader interpretability space, like `Captum` [Kokhlikyan et al., 2020], `Xplique` [Fel et al., 2022], `TransformerLens` [Nanda and Bloom, 2022], `InterpretML` [Nori et al., 2019], and `Alibi Explain` [Klaise et al., 2021]. However, TDA has not seen much attention in terms of dedicated software packages. Out of the libraries listed above, only Captum provides a small number of TDA methods, but it lacks dedicated TDA evaluators. The `Influenciæ` library [Picard et al., 2024] offers implementations of TDA methods in `TensorFlow` [Abadi, 2016]. However, unlike **quanda**, it has a limited focus on evaluation and includes only a single evaluation benchmark. Similarly, `PyDVL`[TransferLab, 2024] provides influence function implementations with a focus on data valuation [Sim et al., 2022]. **quanda** addresses the need for an open-source, standardized evaluation procedure for data attribution. In this respect, **quanda** draws significant inspiration from previous work on feature attribution evaluation, notably `Quantus` [Hedström et al., 2023] and `OpenXAI` [Agarwal et al., 2022]. **quanda**, in the vein of these two libraries, was designed to unify disparate evaluation strategies and to provide benchmarking tools for systematic comparison of different techniques.

3 Library Overview

quanda is designed to facilitate the evaluation of data attribution methods for practitioners, model developers, and researchers. Figure 1 illustrates the evaluation analysis made possible by **quanda**, using the Tiny ImageNet dataset [Le and Yang, 2015] and a ResNet-18 model [He et al., 2016]. For experimental details about the analysis, please refer to Appendix E and the repository, specifically the `/tutorials` folder.

The following sections describe the library's core components, key features, and API design. The first iterations of the library mainly focus on (but are not limited to) image classification and *post-hoc* data attribution. As such, **quanda** currently supports attributing decisions of a single model, as opposed to computing average attributions for different instantiations of the model architecture as sometimes done in the literature [K and Søgaard, 2021, Park et al., 2023].

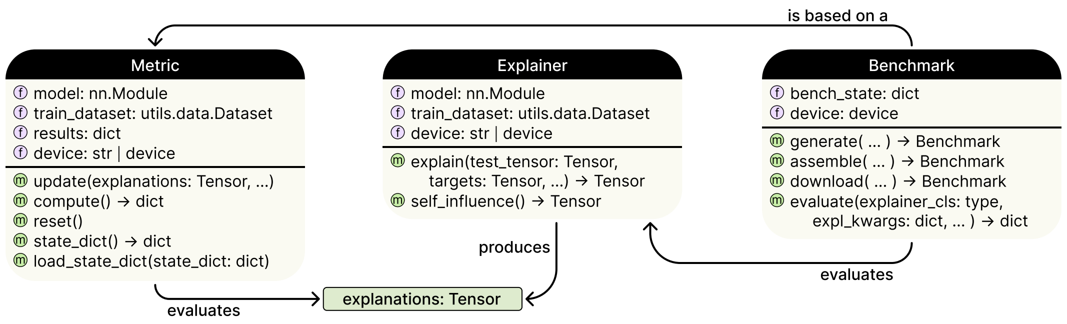
is based on a
Metric
ⓕ model: nn.Module
ⓕ train_dataset: utils.data.Dataset
ⓕ results: dict
ⓕ device: str | device
ⓜ update(explanations: Tensor, ...)
ⓜ compute() → dict
ⓜ reset()
ⓜ state_dict() → dict
ⓜ load_state_dict(state_dict: dict)
Explainer
ⓕ model: nn.Module
ⓕ train_dataset: utils.data.Dataset
ⓕ device: str | device
ⓜ explain(test_tensor: Tensor, targets: Tensor, ...) → Tensor
ⓜ self_influence() → Tensor
Benchmark
ⓕ bench_state: dict
ⓕ device: device
ⓜ generate(...) → Benchmark
ⓜ assemble(...) → Benchmark
ⓜ download(...) → Benchmark
ⓜ evaluate(explainer_cls: type, expl_kwargs: dict, ...) → dict
produces
evaluates
explanations: Tensor
evaluates

Figure 2: Illustration of main **quanda** components. Each block represents a base class. The class fields are listed on the top section, indicated with the icon ⓕ, while the class methods are listed on the bottom, indicated with ⓜ. The leading `torch` is omitted for `PyTorch` types. The arrows explain the functions of and relations between individual components. Specifically, while `Explainer` and `Metric` classes relate indirectly through the generated explanations, the `Benchmark` class internally utilizes a `Metric` instance to evaluate the `Explainer` class, handling the generation of explanations.

3.1 Library Design

The design philosophy of **quanda** emphasizes creating modular interfaces that represent distinct functional units. The three main components of **quanda** are explainers, evaluation metrics, and benchmarks. Each component is implemented as a base class that defines the minimal functionalities required to implement a new instance. The design of the base classes prioritizes easy extension and integration with other components. This allows users, for instance, to evaluate a novel TDA method by simply wrapping their implementation to conform to the base explainer interface. Figure 2 illustrates the main functionality of different components and lists the base class fields and methods. Comprehensive details regarding the explainers, metrics, and benchmarks that are currently included in **quanda** are provided in Appendix B, C, and D, respectively.

Explainers An `Explainer` is a class representing a single TDA method. An `Explainer` instance maintains information about the specific model architecture, model weights, training dataset, and in some cases the training hyperparameters, such as the loss function. As the initialization of a TDA method can be computationally intense [e.g., Caruana et al., 1999, Koh and Liang, 2017, Schioppa et al., 2022], it is delegated to the initialization of `Explainer` instances. Attribution of a test batch is then performed on-demand with the `explain` method, while the `self_influence` method returns a vector of self-influences of training samples, following Koh and Liang [2017]. **quanda** also provides functional interfaces to the `explain` and `self_influence` methods, encapsulating the process of initializing and using the explainers in a single function.

Metrics A metric is a method that summarizes the performance and reliability of a TDA method in a compact representation, typically as a single number. Three categories of `Metric` classes can be found in **quanda**: `ground_truth`, `downstream_eval` and `heuristics`. To provide flexibility and efficiency, **quanda** adopts a *stateful* `Metric` design that supports the incremental addition of new test batches via the `update` method. This allows the user to directly evaluate precomputed explanations, or to conduct the evaluation process alongside the explanation process. The final metric score is obtained by calling the `compute` method after the `Metric` instance has been updated with the explanations.

Benchmarks A **quanda** benchmark is a combination of a specific model architecture, model weights, training and evaluation datasets, and a metric with its arguments, similar to the definition from Raji et al. [2021]. `Benchmark` classes enable standardized comparison of different TDA methods. The library interface supports the initialization of `Benchmark` instances by using user-provided assets with the `assemble` method, creating controlled settings with the `generate` method, and loading pre-configured benchmarks that are made available with the `download` method. The precomputed benchmark suites allow users to speed up the evaluation process by using datasets and models that are already properly set up for each specific metric. An `Explainer` class implementation with a set of hyperparameters can be evaluated with a call to the `evaluate` method to compute the associated metric score.

The following code snippet contains an example of how **quanda** can be used to evaluate concurrently-generated explanations. In this example, we assume to have a pre-trained model (`model`), a training dataset (`train_dataset`) and a test dataloader (`test_dataloader`).

```
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_datasetset)

for i, (data, labels) in test_dataloader:
    pred_labels = model(data).argmax(dim=1)
    tda =trak_explainer.explain(test_tensor=data, targets=pred_labels)
    class_detection.update(explanations=tda, test_labels=pred_labels)

print("Identical class metric output:", class_detection.compute())
```

3.2 Maintaining Library Quality and Accessibility

We ensure high code quality through thorough integration and unit testing with `pytest`, reaching 90% test coverage at submission time. To maintain style consistency and compliance with PEP8 conventions, the static type checker `mypy`, the automatic code formatter `black`, the styling tool `isort`, and the linter `flake8` are used. Each pull request triggers a full test run, coverage, type, and linting checks through a continuous integration (CI) pipeline on GitHub.

The **quanda** toolkit leverages widely-used libraries such as `PyTorch Lightning`, `HuggingFace Datasets`, `torchvision`, `torcheval` and `torchmetrics` to ensure seamless integration into users' pipelines while avoiding the reimplementation of functionality that is already available in established libraries. The `Explainer` classes provide wrappers for existing `PyTorch` implementations of TDA methods, including those from `Captum` [Kokhlikyan et al., 2020] and the official TRAK implementation [Park et al., 2023]. We plan to expand the number of supported TDA method wrappers in the future.

In designing the **quanda** library, the primary focus is to ensure ease of use by providing clear and consistent APIs. Comprehensive documentation, installation guides, and extensive tutorials are provided to help users navigate the library for attribution, evaluation, and benchmarking. The library documentation is available at the following URL: `https://quanda.readthedocs.io`.

The library source code is made public as a GitHub repository and a PyPi package under an open-source license. The benchmarks used in the library are also made available under this license and can be easily loaded using the provided tools. We encourage bug reports and contributions through GitHub and offer detailed guidelines to facilitate collaboration and community engagement.

4 Conclusion and Future Plans

Despite the rising interest in training data attribution (TDA) methods within the interpretability community, the lack of comprehensive evaluation tooling has hindered broader adoption. **quanda** directly addresses this gap by providing a versatile open-source Python library designed to streamline the evaluation of TDA methods. In addition to offering a suite of metrics and ready-to-use controlled setups to speed up complicated evaluation processes, **quanda** provides unified and consistent wrappers for existing implementations of major TDA methods, which are currently scattered across repositories.

With **quanda**, we aim to significantly lower the barrier to entry for researchers applying TDA methods, thereby promoting TDA methods as valuable tools within the interpretability community. In future releases, we plan to extend **quanda**'s capabilities to additional domains, including natural language processing. Our active development efforts will also continue to focus on integrating more TDA method wrappers, metrics, and tasks, releasing new benchmarks, as well as building our own implementations of TDA methods.

5 Acknowledgements

This work was supported by the Fraunhofer Internal Programs as grant PREPARE (40-08394); the Federal Ministry of Education and Research (BMBF) as grant BIFOLD (01IS18025A, 01IS180371I); the German Research Foundation (DFG) as research unit DeSBi (KI-FOR 5363); the European Union's Horizon Europe research and innovation programme (EU Horizon Europe) as grant TEMA (101093003); and the European Union's Horizon 2020 research and innovation programme (EU Horizon 2020) as grant iToBoS (965221).

References

Luca Longo, Mario Brcic, Federico Cabitza, Jaesik Choi, Roberto Confalonieri, Javier Del Ser, Riccardo Guidotti, Yoichi Hayashi, Francisco Herrera, Andreas Holzinger, Richard Jiang, Hassan Khosravi, Freddy Lecue, Gianclaudio Malgieri, Andrés Páez, Wojciech Samek, Johannes Schneider, Timo Speith, and Simone Stumpf. Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion*, 106:102301, 2024. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2024.102301. URL https://www.sciencedirect.com/science/article/pii/S1566253524000794.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL https://arxiv.org/abs/1312.6034.

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. *PLOS ONE*, 10(7):e0130140, July 2015. ISSN 1932-6203. doi: 10.1371/journal.pone.0130140. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0130140. Publisher: Public Library of Science.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/sundararajan17a.html.

Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf.

Amirata Ghorbani, Abubakar Abid, and James Zou. Interpretation of neural networks is fragile. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3681–3688, Jul. 2019. doi: 10.1609/aaai.v33i01.33013681. URL https://ojs.aaai.org/index.php/AAAI/article/view/4252.

Blair Bilodeau, Natasha Jaques, Pang Wei Koh, and Been Kim. Impossibility theorems for feature attribution. *Proceedings of the National Academy of Sciences*, 121(2):e2304406120, 2024. doi: 10.1073/pnas.2304406120. URL https://www.pnas.org/doi/abs/10.1073/pnas.2304406120.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019. doi: 10.1038/s42256-019-0048-x. URL https://doi.org/10.1038/s42256-019-0048-x.

Reduan Achtibat, Maximilian Dreyer, Ilona Eisenbraun, Sebastian Bosse, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. From attribution maps to human-understandable explanations through concept relevance propagation. *Nature Machine Intelligence*, 5(9):1006–1019, 2023. doi: 10.1038/s42256-023-00711-8. URL https://doi.org/10.1038/s42256-023-00711-8.

Leonard Bereska and Stratis Gavves. Mechanistic interpretability for AI safety - a review. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ePUVetPKu6.

Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence: A survey, 2023. URL https://arxiv.org/abs/2312.12936.

Zayd Hammoudeh and Daniel Lowd. Training data influence analysis and estimation: a survey. *Machine Learning*, 113(5):2351–2403, May 2024. ISSN 1573-0565. doi: 10.1007/s10994-023-06495-7. URL `https://doi.org/10.1007/s10994-023-06495-7`.

Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1885–1894. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/koh17a.html`.

Chih-Kuan Yeh, Joon Kim, Ian En-Hsu Yen, and Pradeep K Ravikumar. Representer point selection for explaining deep neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/8a7129b8f3edd95b7d969dfc2c8e9d9d-Paper.pdf`.

Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 19920–19930. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper_files/paper/2020/file/e6385d39ec9394f2f3a354d9d2b88eec-Paper.pdf`.

Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. TRAK: Attributing model behavior at scale. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 27074–27113. PMLR, 23–29 Jul 2023. URL `https://proceedings.mlr.press/v202/park23c.html`.

Juhan Bae, Wu Lin, Jonathan Lorraine, and Roger Grosse. Training data attribution via approximate unrolled differentiation, 2024. URL `https://arxiv.org/abs/2405.12186`.

Rich Caruana, Hooshang Kangarloo, John David Dionisio, Usha Sinha, and David Johnson. Case-based explanation of non-case-based learning methods. *Proceedings AMIA Symposium*, pages 212–215, January 1999. ISSN 1531-605X. URL `https://europepmc.org/articles/PMC2232607`.

Kazuaki Hanawa, Sho Yokoi, Satoshi Hara, and Kentaro Inui. Evaluation of similarity-based explanations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=9uvhpyQwzM_`.

Pang Wei W Koh, Kai-Siang Ang, Hubert Teo, and Percy S Liang. On the accuracy of influence functions for measuring group effects. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/a78482ce76496fcf49085f2190e675b4-Paper.pdf`.

R Dennis Cook and Sanford Weisberg. Residuals and influence in regression, 1982.

Galip Ümit Yolcu, Thomas Wiegand, Wojciech Samek, and Sebastian Lapuschkin. Dualview: Data attribution from the dual perspective. *arXiv e-prints*, pages arXiv–2402, 2024.

Rajiv Khanna, Been Kim, Joydeep Ghosh, and Sanmi Koyejo. Interpreting black box predictions using fisher kernels. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 3382–3390. PMLR, 16–18 Apr 2019. URL `https://proceedings.mlr.press/v89/khanna19a.html`.

Pouya Pezeshkpour, Sarthak Jain, Byron Wallace, and Sameer Singh. An empirical comparison of instance attribution methods for NLP. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 967–975, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.75. URL `https://aclanthology.org/2021.naacl-main.75`.

Jonas R. Naujoks, Aleksander Krasowski, Moritz Weckbecker, Thomas Wiegand, Sebastian Lapuschkin, Wojciech Samek, and René P. Klausen. Pinnfluence: Influence functions for physics-informed neural networks, 2024. URL `https://arxiv.org/abs/2409.08958`.

Han Guo, Nazneen Rajani, Peter Hase, Mohit Bansal, and Caiming Xiong. FastIF: Scalable influence functions for efficient model interpretation and debugging. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10333–10350, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.808. URL https://aclanthology.org/2021.emnlp-main.808.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining LLMs at scale, 2023. URL https://openreview.net/forum?id=XUIYn3jo5T.

Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=4wZiAXD29TQ.

Anshuman Chhabra, Peizhao Li, Prasant Mohapatra, and Hongfu Liu. "What Data Benefits My Classifier?" Enhancing Model Performance and Interpretability through Influence-Based Data Selection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HE9eUQlAvo.

Logan Engstrom, Axel Feldmann, and Aleksander Madry. Dsdm: Model-aware dataset selection with datamodels. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=GC8HkKeH8s.

Daniel Ting and Eric Brochu. Optimal subsampling with influence functions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/57c0531e13f40b91b3b0f1a30b529a1d-Paper.pdf.

Alexander Warnecke, Lukas Pirch, Christian Wressnegger, and Konrad Rieck. Machine unlearning of features and labels. In *Proc. of the 30th Network and Distributed System Security (NDSS)*, 2023.

Ekin Akyurek, Tolga Bolukbasi, Frederick Liu, Binbin Xiong, Ian Tenney, Jacob Andreas, and Kelvin Guu. Towards tracing knowledge in language models back to the training data. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2429–2446, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.180. URL https://aclanthology.org/2022.findings-emnlp.180.

Elisa Nguyen, Evgenii Kortukov, Jean Song, and Seong Joon Oh. Exploring Practitioner Perspectives On Training Data Attribution Explanations. In *XAI in Action: Past, Present, and Future Applications*, 2023. URL https://openreview.net/forum?id=x9H6lNez5b.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Samyadeep Basu, Phil Pope, and Soheil Feizi. Influence functions in deep learning are fragile. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xHKVVHGDOEk.

Elnaz Barshan, Marc-Etienne Brunet, and Gintare Karolina Dziugaite. Relatif: Identifying explanatory training samples via relative influence. In Silvia Chiappa and Roberto Calandra, editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1899–1909. PMLR, 26–28 Aug 2020. URL https://proceedings.mlr.press/v108/barshan20a.html.

Vasu Singla, Pedro Sandoval-Segura, Micah Goldblum, Jonas Geiping, and Tom Goldstein. A simple and efficient baseline for data attribution on images, 2023. URL https://arxiv.org/abs/2311.03386.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. Captum: A unified and generic model interpretability library for PyTorch, 2020. URL https://arxiv.org/abs/2009.07896. _eprint: 2009.07896.

Thomas Fel, Lucas Hervier, David Vigouroux, Antonin Poche, Justin Plakoo, Remi Cadene, Mathieu Chalvidal, Julien Colin, Thibaut Boissin, Louis Bethune, Agustin Picard, Claire Nicodeme, Laurent Gardes, Gregory Flandin, and Thomas Serre. Xplique: A Deep Learning Explainability Toolbox. In *The Conference on Computer Vision and Pattern Recognition*, Nouvelle-Orléans, United States, June 2022. Workshop: Explainable Artificial Intelligence for Computer Vision (XAI4CV). URL `https://hal.science/hal-03696248`.

Neel Nanda and Joseph Bloom. TransformerLens. `https://github.com/TransformerLensOrg/TransformerLens`, 2022.

Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. InterpretML: A Unified Framework for Machine Learning Interpretability. *arXiv preprint arXiv:1909.09223*, 2019. URL `https://arxiv.org/abs/1909.09223`.

Janis Klaise, Arnaud Van Looveren, Giovanni Vacanti, and Alexandru Coca. Alibi Explain: Algorithms for Explaining Machine Learning Models. *Journal of Machine Learning Research*, 22(181):1–7, 2021. URL `http://jmlr.org/papers/v22/21-0017.html`.

Agustin Picard, Lucas Hervier, Thomas Fel, and David Vigouroux. Influenciæ: A library for tracing the influence back to the data-points. In Luca Longo, Sebastian Lapuschkin, and Christin Seifert, editors, *Explainable Artificial Intelligence*, pages 193–204, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-63803-9.

Martín Abadi. TensorFlow: learning functions at scale. In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, ICFP 2016, page 1, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 978-1-4503-4219-3. doi: 10.1145/2951913.2976746. URL `https://doi.org/10.1145/2951913.2976746`. event-place: Nara, Japan.

TransferLab. pyDVL, 2024. URL `https://zenodo.org/doi/10.5281/zenodo.8311582`.

Rachael Hwee Ling Sim, Xinyi Xu, and Bryan Kian Hsiang Low. Data valuation in machine learning: "ingredients", strategies, and open challenges. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 5607–5614. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/782. URL `https://doi.org/10.24963/ijcai.2022/782`. Survey Track.

Anna Hedström, Leander Weber, Daniel Krakowczyk, Dilyara Bareeva, Franz Motzkus, Wojciech Samek, Sebastian Lapuschkin, and Marina M.-C. Höhne. Quantus: An explainable ai toolkit for responsible evaluation of neural network explanations and beyond. *Journal of Machine Learning Research*, 24(34):1–11, 2023. URL `http://jmlr.org/papers/v24/22-0142.html`.

Chirag Agarwal, Satyapriya Krishna, Eshika Saxena, Martin Pawelczyk, Nari Johnson, Isha Puri, Marinka Zitnik, and Himabindu Lakkaraju. Openxai: Towards a transparent evaluation of model explanations. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 15784–15799. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/file/65398a0eba88c9b4a1c38ae405b125ef-Paper-Datasets_and_Benchmarks.pdf`.

Ya Le and Xuan S. Yang. Tiny imagenet visual recognition challenge. CS231N, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

Karthikeyan K and Anders Søgaard. Revisiting methods for finding influential examples, 2021. URL `https://arxiv.org/abs/2111.04683`.

Andrea Schioppa, Polina Zablotskaia, David Vilar, and Artem Sokolov. Scaling up influence functions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8179–8186, Jun. 2022. doi: 10.1609/aaai.v36i8.20791. URL `https://ojs.aaai.org/index.php/AAAI/article/view/20791`.

Inioluwa Deborah Raji, Emily Denton, Emily M. Bender, Alex Hanna, and Amandalynne Paullada. AI and the everything in the whole wide world benchmark. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL `https://openreview.net/forum?id=j6NxpQbREA1`.

Guillaume Charpiat, Nicolas Girard, Loris Felardos, and Yuliya Tarabalka. Input similarity from the neural network perspective. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper_files/paper/2019/file/c61f571dbd2fb949d3fe5ae1608dd48b-Paper.pdf`.

Walter Edwin Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL `https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf`.

Yongchan Kwon, Eric Wu, Kevin Wu, and James Zou. Datainf: Efficiently estimating data influence in loRA-tuned LLMs and diffusion models. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=9m02ib92Wz`.

Zayd Hammoudeh and Daniel Lowd. Identifying a training-set attack's target using renormalized influence estimation. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, CCS '22, page 1367–1381, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450394505. doi: 10.1145/3548606.3559335. URL `https://doi.org/10.1145/3548606.3559335`.

Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *Advances in Neural Information Processing Systems*, pages 10506–10518, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

A Notation

We consider an ordered training dataset of size n, denoted as $\mathcal{D} = \{z_i\}_{i=1}^n \in \mathcal{Z}^n$, where each training sample $z_i \in \mathcal{Z}$ represents an individual datapoint. In the context of classification tasks, a datapoint z_i consists of a pair $z_i = (x_i, y_i)$, where $x_i \in \mathbb{R}^d$ is the input sample and $y_i \in [C]$ is the corresponding label. Here, $C \in \mathbb{Z}$ denotes the number of classes, and $[C]$ represents the set $\{1, 2, \ldots, C\}$. For test samples $z = (x, y) \in \mathcal{Z}$, the label y indicates the output of the network that was used for attributions.

The model we aim to explain is assumed to be a trained neural network, which implements the function $f(\,\cdot\,;\theta)$, where θ denotes the set of learned parameters of the model. We denote the empirical risk:

$$\mathcal{J}(\theta, \mathcal{D}) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(z_i, \theta),$$

where $\mathcal{L}(\cdot,\cdot)$ is a sample-wise loss function, such as the cross entropy loss.

According to the assumptions of certain TDA methods [Caruana et al., 1999, Yeh et al., 2018], we consider the penultimate layer of the model as a feature extractor, denoted by $h(\,\cdot\,;\vartheta) : \mathbb{R}^d \to \mathbb{R}^p$, with parameters ϑ. This is followed by a linear classifier characterized by a weight matrix $W \in \mathbb{R}^{p\times C}$ and a bias term $b \in \mathbb{R}^C$. Consequently, we can express the model output as $f(z;\theta) = f(z;\vartheta, W) = W^\top h(z;\vartheta) + b$. For brevity, we will denote the final hidden features as $h(z)$ and the model output as $f(z)$ in the following sections. In the context of classification tasks, the model output corresponds to the value associated with the correct class. The attribution for a test sample z, computed by a TDA method, is denoted as $\tau(z, \mathcal{D})_i$ for the training sample z_i.

B Training Data Attribution Methods

In this section, we present the details on TDA method implementations currently included in **quanda**. We provide a theoretical definition for each method, along with insights into their specific implementations and the original codebases that `quanda` wraps in its `Explainer` interface.

B.1 Similarity of Representations

Gaining insights into what makes two samples similar from a model's perspective can aid practitioners in understanding the model's underlying reasoning and potentially detecting unwanted behaviours [Charpiat et al., 2019, Hanawa et al., 2021]. The similarity between hidden representations of a given sample and those of training samples can be framed as an attribution method [Caruana et al., 1999]:

$$\tau_{\text{SIM}}(z, \mathcal{D})_i = \sigma(h(z), h(z_i)), \tag{1}$$

where $\sigma(\cdot,\cdot)$ is a similarity measure such as the dot product or the cosine distance.

This explanation method is implemented as `SimilarityExplainer` in `Captum`[1]. In **quanda**, we provide a wrapper around this `Captum` implementation.

B.2 Influence Functions

Influence Functions (IF) [Koh and Liang, 2017] is a TDA method that approximates the counterfactual effect of retraining the model after removing a single training sample from the dataset. The approximation is computed by performing a single Newton optimization step on the counterfactual loss landscape, which results in the following attribution:

$$\tau_{\text{IF}}(z, \mathcal{D})_i = \nabla_\theta \mathcal{L}(z, \theta)^\top H_\theta^{-1} \nabla_\theta \mathcal{L}(z_i, \theta), \tag{2}$$

where $H_\theta = \nabla_\theta^2 \mathcal{J}(\theta, \mathcal{D})$ is the Hessian of the total loss.

Computing the inverse Hessian is a computationally demanding task, and the approach introduced in the original paper [Koh and Liang, 2017] incurs a substantial memory footprint. To mitigate these challenges, recent work [Schioppa et al., 2022] has proposed to speed up the inverse Hessian calculation using Arnoldi iterations [Arnoldi, 1951]. This method is named `Arnoldi Influence Function` in `Captum`. **quanda** provides a wrapper around this implementation[2].

To generate a global ranking of training samples, such as for detecting mislabeled examples, [Koh and Liang, 2017] proposes evaluating the influence of each sample on itself, following Equation 2. In **quanda**, we also provide a wrapper around the `Captum` implementation to compute these self-influences.

[1]https://github.com/pytorch/captum/blob/master/captum/influence/_core/similarity_influence.py

[2]https://github.com/pytorch/captum/blob/master/captum/influence/_core/arnoldi_influence_function.py

B.3 TracIn

TracIn [Pruthi et al., 2020] builds on the idea of tracking how the loss on a test point evolves throughout the training process. The change in the loss for a test sample z caused by a training sample z_i is approximated via a linear approximation:

$$\tau_{\text{TracIn}}(z, \mathcal{D})_i = \sum_t \eta_t \nabla_{\theta_t} \mathcal{L}(z, \theta_t)^\top \nabla_{\theta_t} \mathcal{L}(z_i, \theta_t), \tag{3}$$

where t indexes the training epochs where the training sample z_i was used, θ_t represents the parameters and η_t the learning rate at epoch t.

The full TracIn attributor as defined in Equation 3 is prohibitively computationally expensive, as it requires storing the parameters at each training step. To address this, attributions are computed using only a selected set of checkpoints, a method known as `TracInCP`. `Captum` implements this approach[3] with two separate extensions, as described in the original paper [Pruthi et al., 2020]:

- `TracInCPFast`: This variant simplifies the process further by considering only the final layer parameters when computing the gradients.
- `TracInCPFastRandProj`: This version additionally uses random projections to reduce the dimensionality of the gradients, thereby speeding up the computations.

quanda provides wrappers for all the aforementioned variants[4].

B.4 Representer Points

The Representer Points method [Yeh et al., 2018] leverages a representer theorem for gradient-descent-based training of deep neural networks. Under the assumption that the model is trained to convergence with L_2 regularization, they demonstrate that the final layer parameters W can be written as a linear combination of final layer features $h(z_i)$. This formulation reduces the model to a kernel machine, enabling a decomposition of the model output to the contributions of each training point. This results in the following attribution function:

$$\tau_{\text{RP}}(z, \mathcal{D})_i = -\frac{\partial \mathcal{L}(z_i; \theta)}{\partial f(z_i)} h(z_i)^\top h(z). \tag{4}$$

Given a model that is not originally trained with L_2 regularization, the Representer Points approach suggests training the final layer parameters W with L_2 regularization, using backtracking line search to ensure convergence.

quanda provides a wrapper around the official code release by the original paper's authors[5].

B.5 TRAK

TRAK [Park et al., 2023] is a method that approximates the counterfactual effect of retraining on a subset of training points. It achieves this by linearizing the model around the test point using a first-order Taylor decomposition of the model output. This approach corresponds to using the empirical Neural Tangent Kernel [Jacot et al., 2018] to define a surrogate for the model.

TRAK is defined to use multiple independent instantiations of the model, trained on the same dataset, or different subsets of the dataset. Since **quanda** currently considers post-hoc attribution of a single model, we consider the case of using only a single model, whose behavior we want to attribute to the training data.

Let $G = \left[g_1; g_2; \ldots; g_N\right]$ be a matrix of gradients, where each column $g_i = \nabla_\theta f(z_i)$ represents the gradient of the model output corresponding to a training point. Additionally, let Q be the diagonal matrix such that $Q_{i,i} = 1 - p_i$, where p_i is the probability corresponding to the ground truth label of data point z_i. In the binary classification scenario, TRAK can then be formulated as follows:

$$\tau_{\text{TRAK}}(z, \mathcal{D})_i = \nabla_\theta f(z)^\top (G^\top G)^{-1} G^\top Q. \tag{5}$$

To extend the formulation to the multiclass case, instead of using the model output $f(\cdot)$, we consider the gradients of the function $r(z) = \log\left(\frac{p(z;\theta)}{1-p(z;\theta)}\right)$, where $p(z;\theta)$ denotes the softmax probability corresponding to the ground truth label. For test points, this function represents the probability of the output that is chosen for explanation.

[3]https://github.com/pytorch/captum/blob/master/captum/influence/_core/tracincp.py

[4]https://github.com/pytorch/captum/blob/master/captum/influence/_core/tracincp_fast_rand_proj.py

[5]https://github.com/chihkuanyeh/Representer_Point_Selection

Finally, given that contemporary architectures contain millions of parameters, computing the above-mentioned attribution exactly becomes too computationally expensive to be practically feasible. Hence, similar to TracIn, TRAK employs random projections on the gradients to enable efficient attribution calculation, while preserving the inner products [Park et al., 2023].

TRAK offers an official code release[6], which can be conveniently used with its **quanda** wrapper.

C Evaluation Metrics

In this section, we summarize the evaluation metrics that are currently implemented in **quanda** and provide references for related work.

C.1 Ground Truth Metrics

Ground truth metrics evaluate the attributions against the ground truth values that the respective TDA methods aim to approximate, e.g., the counterfactual effects of modifying the training dataset.

C.1.1 Linear Datamodeling Score

Proposed in [Park et al., 2023], the Linear Datamodeling Score (LDS) evaluates the ability of a TDA method to make accurate counterfactual predictions about the model's output when trained on a subset of training data points. Methods like Influence Functions (IF) [Koh and Liang, 2017] and TRAK [Park et al., 2023] explicitly aim to approximate these counterfactual values, making LDS a ground truth metric for their evaluation. The metric relies on the assumption that the attributions are linear, implying that the attribution for a subset of training samples can be represented as the sum of the individual attributions from those samples.

Let $\mathcal{D}' \subset \mathcal{D}$ be a training dataset, and $g_\tau(z, \mathcal{D}')$ be the *attribution-based output prediction* of the model trained on $\mathcal{D}'$:

$$g_\tau(z, \mathcal{D}') = \sum_{i : z_i \in \mathcal{D}'} \tau(z, \mathcal{D})_i, \tag{6}$$

Let $f(z; S)$ denote the output of a model trained on the dataset S.

We randomly sample m subsets of the training data: $\{\mathcal{D}'_1, \mathcal{D}'_2, \ldots \mathcal{D}'_m : \mathcal{D}'_j \subset \mathcal{D}\, \forall j \in [m]\}$. For each subset, we compute the counterfactual prediction from the original attributions. We then compute the Spearman rank correlation of these predictions with the actual outputs after retraining models on each subset. This gives the LDS score for a single test point z:

$$LDS(\tau, z) = \rho(\{g_\tau(z, \mathcal{D}'_j) : j \in [m]\}, \{f(z; \mathcal{D}'_j) : j \in [m]\}, \tag{7}$$

where ρ denotes the Spearman rank correlation function.

The final LDS score is the average LDS score over the test samples.

C.2 Downstream Evaluation Tasks

Downstream tasks assess the effectiveness of attributions in addressing a specific end-task.

C.2.1 Class Detection

Class detection is the task of inferring a test sample's label based on the labels of the training data points that have the highest attributions in the corresponding TDA explanations. As defined in [Hanawa et al., 2021, Kwon et al., 2024], the class detection task is grounded in the intuition that same-class training data points are more likely to assist the model in making a correct decision than those of differing classes.

Following [Hanawa et al., 2021], the **quanda** implementation of `ClassDetectionMetric` computes the ratio of test samples for which the training sample with the highest attribution corresponds to a datapoint of the correct class among the supplied attributions.

C.2.2 Subclass Detection

Built on the assumption that a model learns distinct representations for different sub-groups within the same class, [Hanawa et al., 2021] introduces a subclass detection test. The original formulation of the test involves creating a modified training dataset where labels are randomly grouped to form new labels, followed by training

[6]https://github.com/MadryLab/trak

a model. In **quanda**, this functionality is handled by the respective `Benchmark` class, while the `Metric` requires ground-truth labels for the sub-groups. It is assumed that the model develops separate mechanisms for classifying data points from different sub-groups, which should be reflected in the attributions. Particularly, in evaluating similarity-based attributions, Hanawa et al. [2021] recommend calculating the ratio of test data points for which the highest attributed training sample belongs to the correct (original, sub-) class.

quanda implements this metric as defined in the paper, allowing for random grouping of classes, as well as user-defined groupings.

C.2.3 Mislabeling Detection

Mislabeling Detection is widely used as an evaluation strategy for TDA methods [Koh and Liang, 2017, Yeh et al., 2018, Pruthi et al., 2020, Khanna et al., 2019]. This approach measures the effectiveness of TDA methods in identifying training samples that have been incorrectly labeled, also referred to as noisy labels.

Under the assumption that mislabeled samples will be strong evidence for their changed label for the model [Koh and Liang, 2017, Pruthi et al., 2020], the metric calculation procedure involves ordering the training samples by their self-influence ranking and assessing each label for potential mislabeling one-by-one based on the ground-truth labels. The resulting cumulative mislabeling detection curve is expected to rise sharply for more effective TDA methods. The metric scores are derived from the corresponding AUC score. As an additional feature, **quanda** enables users to generate a global ranking of the training samples from TDA attributions for test samples using *aggregators*.

C.2.4 Shortcut Detection

A shortcut, or a Clever-Hans effect, refers to decision rules learned by a model that allows it to perform well on a specific test data distribution while failing to generalize to more challenging testing conditions. The Shortcut Detection metric assesses the ability of TDA methods to identify test samples for which the model relies on shortcut features for its predictions. This metric is referred to as *domain mismatch debugging* in [Koh and Liang, 2017] and [Yolcu et al., 2024].

The metric evaluates the explanations of test samples that trigger the shortcut effect in a model. To confirm that the model is relying on shortcut features, the **quanda** implementation of the metric enables users to filter out samples where the shortcut effect is unlikely to have occurred during inference following [Yolcu et al., 2024]. Assuming the indices of the training samples exhibiting a shortcut are known, this metric quantifies the ranking of the attributions of “shortcutted" samples relative to clean samples for the predictions of “shortcutted" test samples. The **quanda** implementation utilizes the area under the precision-recall curve (AUPRC) for calculation, as per [Hammoudeh and Lowd, 2022]. AUPRC is chosen because it provides better insight into performance in highly skewed classification tasks where false positives are common.

C.3 Heuristics

Heuristics are metrics designed to estimate desirable properties of explanations or serve as sanity checks for their validity.

C.3.1 Mixed Datasets

In scenarios where a model is trained on multiple datasets, this metric assesses the effectiveness of a given TDA method in identifying samples from the correct dataset as the most influential for a specific prediction, as outlined by [Hammoudeh and Lowd, 2022].

The metric assumes that a model has been trained on two distinct datasets: a base dataset and an adversarial dataset. These datasets are assumed to have substantially distinct underlying data distributions. The number of samples in the base dataset is significantly larger than the number of samples in the adversarial dataset. All “adversarial" samples are assigned a single “adversarial” label from the base dataset. The evaluation score is calculated based on explanations of “adversarial” test samples where the model correctly predicts the “adversarial” label. The AUPRC score quantifies the ranking of the attributions of “adversarial" samples in relation to other training samples.

C.3.2 Model Randomization

This metric, proposed in [Hanawa et al., 2021], draws inspiration from a sanity check for feature attributions introduced by [Adebayo et al., 2018]. The underlying intuition is that attributions should be sensitive to changes in model parameters. If the attributions remain unchanged despite randomizing the model parameters, this suggests a weak connection between the attributions and the model behavior. Accordingly, the metric procedure involves randomizing the model parameters and generating explanations for each test sample using the modified model. The average Spearman correlation is then computed between the attributions obtained from the randomized model and those from the original model over the test set. A lower correlation indicates a better score, reflecting that the attributions are indeed linked to the model's parameters.

C.3.3 Top-K Cardinality

The attributions should depend on the test samples used as input. However, a suboptimal TDA method may yield the same top influential samples for a given explanation target, regardless of the test sample being explained. The Top-K Cardinality metric, proposed in [Barshan et al., 2020], quantifies this dependence by calculating the ratio of the cardinality of the set of top-k attributed training samples across all test sample attributions to the maximum possible cardinality of this set (the product of the number of test samples and k). A higher ratio indicates better performance of the TDA method, reflecting a greater dependence of attributions on the specific test samples being examined.

D Benchmarks

All metrics in **quanda** are associated with corresponding benchmarks. One key use case for these benchmarks is the generation of controlled environments that metric definitions often require. While `Metric` objects necessitate that users provide all components—such as modified datasets, models trained on these datasets, and attributions from these models—`Benchmark` objects facilitate the creation of these components. They take vanilla components and manage dataset manipulation, model training, and the generation of explanations as needed by the associated metrics. They utilize the respective `Metric` objects to handle the entire evaluation process.

Additionally, **quanda** enables users to download pre-computed benchmarks, allowing them to bypass the setup process for controlled environments. By utilizing the `download` method to initialize a `Benchmark`, users can immediately begin their evaluation. This feature also offers a standardized benchmark environment for researchers, facilitating consistent assessments of their methods.

E Experimental Details

In this section, we outline the experimental setup used for the evaluation displayed in Figure 1. Following this, we highlight important caveats regarding the implementation of the TDA methods employed, as well as the evaluation metrics used, to provide better context and clarity for the results.

E.1 Experimental Setup

To achieve the controlled conditions required for various evaluation strategies, we modified the original Tiny ImageNet dataset [Le and Yang, 2015], incorporating several special features to create two distinct datasets.

The first dataset includes the following modifications:

- For the subclass detection task, all cat subclasses are merged into a single `cat` class, and similarly, all dog subclasses are grouped into a single `dog` class.
- For the shortcut detection test, a yellow box is added to 20% of the input images in the `pomegranate` class.
- For the mixed dataset test, we introduced 200 images of panda sketches from the ImageNet-Sketch [Wang et al., 2019] dataset, all labeled as `basketball`.

In the second dataset:

- For the mislabeling detection test, 30% of images are intentionally mislabeled.

We fine-tuned ResNet18 models [He et al., 2016] pre-trained on ImageNet using these datasets. Both models were trained for 10 epochs using AdamW [Loshchilov and Hutter, 2019] optimizer with a learning rate of 0.0003, weight decay $w = 0.01$, and a CosineAnnealingLR learning rate scheduler. The model trained on the noisy-label dataset achieved a top-1 accuracy of 43%, while the model trained on the transformed dataset achieved a top-1 accuracy of 56%.

These modified datasets and the trained models were then used to generate attributions using the following methods: Representer Points, Influence Functions with Arnoldi Iterations, TracIn (performed only on the last layer), TRAK, and a baseline random explainer. The displayed attributions are normalized by dividing the attribution score vector of each TDA method by its maximum absolute value. The first dataset, along with the model trained on it, is used in the Mixed Datasets, Shortcut Detection, Subclass Detection, and Top-K Cardinality metrics. The dataset with noisy labels and its respective model are used for calculating the Mislabeling Detection metric.

E.2 Discussion of the Results

In the final subsection, we aim to provide further explanations and potential reasoning for the evaluation outcomes depicted in Figure 1.

Firstly, TRAK is a TDA method designed to leverage multiple trained models to mitigate the noisiness of attributions [Park et al., 2023]. However, **quanda** currently emphasizes *post-hoc attribution* of model decisions. Consequently, our implementation of TRAK utilizes only a single model instance, which reduces the quality of attributions, as noted in the original paper [Park et al., 2023].

Furthermore, we focus solely on the parameters of the final linear layer for Arnoldi Influence Function and TracIn attributions. This choice is made to ensure feasible computation times, following the recommendations from the original papers [Koh and Liang, 2017, Pruthi et al., 2020]. This means that the TDA methods have less information about the model behaviour, which results in suboptimal attributions.