AIMobilityAuditPrep

Overview Toolbox
1 Development Approach

The goal of the toolbox is to combine common functionalities for training neural networks and performing audit experiments on the trained neural networks. It should be applicable to different use cases in mobility applications and not limited to only a single output type like segmentation maps or 2D bounding boxes. Hence, it is designed to be as extendable as possible while providing generic components as the basis for more concrete and specific functionalities.

Currently, two different use cases in mobility applications are already implemented to serve as a starting point for experiments. These use cases include the traffic sign assistant and the road user detection. Both are based on the task of image-based 2D bounding box detection and classification. Other sensor modalities and input types are currently not implemented but can be added in the future. Similarly, it is possible to add the support for additional use cases (e.g. free space detection) and additional prediction tasks (e.g. segmentation) in the future.

For all use cases two different modules are implemented. One the one hand, this includes the pure neural network that is trained from data. On the other hand, this includes the entire system where a neural network forms the basis but is extended with other components. For example, an entire system can be composed of multiple neural networks and traditional algorithm-based functionalities.

For both discussed module types (neural network vs. entire system) different experiments are implemented. This includes the creation/training of the neural networks and exemplary tests for audit requirements. The available experiments depend on whether only a neural network or an entire system should be audited. In principle all experiments take in a configuration file which links the different functionalities of the toolbox together. For example, in each configuration file the data source, model/system source, experiment type or hyperparameters need to be specified.

The current toolbox is a first implementation which provides the main functionalities and allows first audit experiments. Over time the goal is to include further experiments mainly to increase the range of potential audit experiments that are available. Also, more use cases and data sources should be added to increase the overall applicability of the toolbox.
2 Structure

The current directory structure of the toolbox is shown in Figure 1. This shows the schematic structure of the components in this toolbox. The datasets directory serves as a placeholder for a directory that holds the respective dataset. It can be located at any location and does not necessarily have to be in the toolbox directory. The docs directory contains all information to build the documentation of the toolbox by extracting relevant information from docstrings in the source code. Next, the examples directory contains exemplary config files that can be used to perform different experiments. These are saved as .json files which enable the readability for humans. They serve as examples and can be adapted to the concrete goals of an experiment and the given local conditions. The models directory functions similar to the datasets directory and serves as a placeholder for a directory that holds trained neural networks and where the intermediate model versions during training are stored. Most important is the source_code directory which holds the actual toolbox implementation and all functionalities. In this directory there are different subdirectories that structure the source code. Here, the configs directory holds different functionalities to read and write configuration files. It also contains the logic to handle the configurations of different experiments, models or datasets. In the data directory all functionalities are located that provide data which can be used for neural networks or also for entire systems. It is subdivided in the dataset and simulation directories which contain the respective functionalities. For datasets this is further divided into the annotation style a dataset supports. For simulation there are different functionalities to connect to different external simulation tools. Next, the models directory is structured after the deep learning framework that is used to implement the included functionality. One level deeper, a similar structure is used as in the dataset directory where the tasks a model solves serve as subdirectories. Next, we use a systems directory containing all functionalities that extend neural networks to entire systems. In addition, the tools directory is structured to differentiate between auditing tools for neural networks and entire systems. The last subdirectory in the source code is the utils directory which contains all kinds of utility methods and classes to support the different functionalities. For example, these can be specific data structures, common functionalities for file interaction, functionalities for enabling reproducibility, etc. Lastly, there is a test directory, which holds the code used for testing the source code of the toolbox and also files that are required for certain tests to run.

Figure 1 Schematic overview of the components in the toolbox.
3 Used Libraries

The described toolbox is implemented using the Python programming language. This is chosen, since it provides a vast ecosystem of relevant libraries for AI development, allows a high productivity for the developers and is used in the majority of state-of-the-art publications in this area. To implement specific functionalities, we use the following libraries:

- Albumentations\(^1\): Library that is used for data augmentation and allows to augment images and the associated annotations for different use cases, like classification, segmentation or detection.
- Argparse\(^2\): Library that is used for reading arguments from command line interfaces.
- Dataclasses\(^3\): Library that is used for handling of configuration values for different experiments.
- FastAI\(^4\): Library that is used for training neural networks.
- Grad-Cam\(^5\): Library that is used for generating heatmaps to explain neural networks.
- Matplotlib\(^6\): Library that is used for visualizing data in plots.
- MLFlow\(^7\): Library that is used for tracking and logging experiments with neural networks.
- Numpy\(^8\): Library that is used for all scientific computations unrelated to neural networks.
- ONNX\(^9\): Library that is used for exchanging the format of neural networks between different deep learning libraries.
- Pandas\(^10\): Library that is used for reading tabular-like data and especially annotations.
- Pathlib\(^11\): Library that is used for handling all file paths.
- Pillow\(^12\): Library that is used for loading and saving images from/to disk.
- Pytest\(^13\): Library that is used for testing of the source code.
- Sphinx\(^14\): Library that is used for generating a documentation from docstrings in the source code.
- Pytorch\(^15\): Deep learning library that is used for all features related to neural networks.
- TorchAttacks\(^16\): Library that is used for executing adversarial attacks on neural networks.
- TorchMetrics\(^17\): Library that is used for evaluating neural networks on standardized metrics.
- TorchVision\(^18\): Library that is used for vision related functionalities.
- Typing\(^19\): Library that is used for type annotations in the source code.

\(^1\) [https://albumentations.ai/](https://albumentations.ai/)
\(^2\) [https://docs.python.org/3/library/argparse.html](https://docs.python.org/3/library/argparse.html)
\(^3\) [https://docs.python.org/3/library/dataclasses.html](https://docs.python.org/3/library/dataclasses.html)
\(^4\) [https://www.fast.ai/](https://www.fast.ai/)
\(^5\) [https://github.com/jacobgil/pytorch-grad-cam](https://github.com/jacobgil/pytorch-grad-cam)
\(^6\) [https://matplotlib.org/](https://matplotlib.org/)
\(^7\) [https://mlflow.org/](https://mlflow.org/)
\(^8\) [https://numpy.org/](https://numpy.org/)
\(^9\) [https://onnx.ai/](https://onnx.ai/)
\(^10\) [https://pandas.pydata.org/](https://pandas.pydata.org/)
\(^11\) [https://docs.python.org/3/library/pathlib.html](https://docs.python.org/3/library/pathlib.html)
\(^12\) [https://pillow.readthedocs.io/en/stable/](https://pillow.readthedocs.io/en/stable/)
\(^13\) [https://docs.pytest.org/en/7.2.x/](https://docs.pytest.org/en/7.2.x/)
\(^15\) [https://pytorch.org/](https://pytorch.org/)
\(^16\) [https://github.com/Harry24k/adversarial-attacks-pytorch](https://github.com/Harry24k/adversarial-attacks-pytorch)
\(^19\) [https://docs.python.org/3/library/typing.html](https://docs.python.org/3/library/typing.html)