Anomalib in 15 Minutes

Contents

This section will walk you through the steps to train a model and use it to detect anomalies in a dataset.

Installation

Installation is simple and can be done in two ways. The first is through PyPI, and the second is through a local installation. PyPI installation is recommended if you want to use the library without making any changes to the source code. If you want to make changes to the library, then a local installation is recommended.

Installing the Installer

Anomalib comes with a CLI installer that can be used to install the full package. The installer can be installed using the following commands:

API **Source**

```
pip install anomalib
```

The main reason why PyPI and source installer does not install the full package is to keep the installation wheel small. The CLI installer also automates the installation such as finding the torch version with the right CUDA/CUDNN version.

The next section demonstrates how to install the full package using the CLI installer.

Installing the Full Package

After installing anomalib, you can install the full package using the following commands:

```
❯ anomalib -h
To use other subcommand using `anomalib install`
To use any logger install it using `anomalib install -v`
╭─ Arguments ──────────────────────────────────────────────────────────────────╮
│ Usage: anomalib [-h] [-c CONFIG] [--print_config [=flags]] {install} ...     │
│                                                                              │
│                                                                              │
│ Options:                                                                     │
│   -h, --help            Show this help message and exit.                     │
│   -c, --config CONFIG   Path to a configuration file in json or yaml format. │
│   --print_config [=flags]                                                    │
│                         Print the configuration after applying all other     │
│                         arguments and exit. The optional flags customizes the│
│                         output and are one or more keywords separated b comma│
│                         The supported flags are: comments, skip_default,     │
│                         skip_null.                                           │
│                                                                              │
│ Subcommands:                                                                 │
│   For more details of each subcommand, add it as an argument followed by     │
│                         --help.                                              │
│                                                                              │
│                                                                              │
│   Available subcommands:                                                     │
│     install             Install the full-package for anomalib.               │
│                                                                              │
╰──────────────────────────────────────────────────────────────────────────────╯
```

As can be seen above, the only available sub-command is `install` at the moment. The `install` sub-command has options to install either the full package or the specific

components of the package.

```
❯ anomalib install -h
To use other subcommand using `anomalib install`
To use any logger install it using `anomalib install -v`
╭─ Arguments ──────────────────────────────────────────────────────────────────╮
│ Usage: anomalib [options] install [-h]                                       │
│                         [--option {full,core,dev,loggers,notebooks,openvino}]
│                         [-v]                                                 │
│                                                                              │
│                                                                              │
│ Options:                                                                     │
│   -h, --help              Show this help message and exit.                   │
│   --option {full,core,dev,loggers,notebooks,openvino}                        │
│                           Install the full or optional-dependencies.         │
│                           (type: None, default: full)                        │
│   -v, --verbose           Set Logger level to INFO (default: False)          │
│                                                                              │
╰──────────────────────────────────────────────────────────────────────────────╯
```

By default the `install` sub-command installs the full package. If you want to install only the specific components of the package, you can use the `--option` flag.

```
# Get help for the installation arguments
anomalib install -h

# Install the full package
anomalib install

# Install with verbose output
anomalib install -v

# Install the core package option only to train and evaluate models via Torch a
anomalib install --option core

# Install with OpenVINO option only. This is useful for edge deployment as the
anomalib install --option openvino
```

After following these steps, your environment will be ready to use anomalib!

🎓 Training

Anomalib supports both API and CLI-based training. The API is more flexible and allows for more customization, while the CLI training utilizes command line interfaces, and might be

easier for those who would like to use anomalib off-the-shelf.

API CLI

```python
# Import the required modules
from anomalib.data import MVTec
from anomalib.models import Patchcore
from anomalib.engine import Engine

# Initialize the datamodule, model and engine
datamodule = MVTec()
model = Patchcore()
engine = Engine()

# Train the model
engine.fit(datamodule=datamodule, model=model)
```

Inference

Anomalib includes multiple inferencing scripts, including Torch, Lightning, Gradio, and OpenVINO inferencers to perform inference using the trained/exported model. Here we show an inference example using the Lightning inferencer.

Lightning Inference

API CLI

```python
# Assuming the datamodule, model and engine is initialized from the previous st
# a prediction via a checkpoint file can be performed as follows:
predictions = engine.predict(
    datamodule=datamodule,
    model=model,
    ckpt_path="path/to/checkpoint.ckpt",
)
```

Torch Inference

API CLI

```
Python code here.
```

OpenVINO Inference

API CLI

```
Python code here.
```

Gradio Inference

API CLI

```
Python code here.
```

Hyper-Parameter Optimization

Anomalib supports hyper-parameter optimization using wandb and comet.ml. Here we show an example of hyper-parameter optimization using both comet and wandb.

CLI API

```
# To perform hpo using wandb sweep
anomalib hpo --backend WANDB  --sweep_config tools/hpo/configs/wandb.yaml

# To perform hpo using comet.ml sweep
anomalib hpo --backend COMET  --sweep_config tools/hpo/configs/comet.yaml
```

Experiment Management

Anomalib is integrated with various libraries for experiment tracking such as comet, tensorboard, and wandb through lighting loggers.

CLI **API**

To run a training experiment with experiment tracking, you will need the following configuration file:

```
# Place the experiment management config here.
```

By using the configuration file above, you can run the experiment with the following command:

```
# Place the Experiment Management CLI command here.
```

Benchmarking

Anomalib provides a benchmarking tool to evaluate the performance of the anomaly detection models on a given dataset. The benchmarking tool can be used to evaluate the performance of the models on a given dataset, or to compare the performance of multiple models on a given dataset.

Each model in anomalib is benchmarked on a set of datasets, and the results are available in `src/anomalib/models/<model_name>README.md`. For example, the MVTec AD results for the Patchcore model are available in the corresponding README.md file.

CLI API

To run the benchmarking tool, run the following command:

```
anomalib benchmark --config tools/benchmarking/benchmark_params.yaml
```

Reference

If you use this library and love it, use this to cite it:

```
@inproceedings{akcay2022anomalib,
  title={Anomalib: A deep learning library for anomaly detection},
  author={
    Akcay, Samet and
    Ameln, Dick and
    Vaidya, Ashwin and
    Lakshmanan, Barath
    and Ahuja, Nilesh
    and Genc, Utku
  },
  booktitle={2022 IEEE International Conference on Image Processing (ICIP)},
  pages={1706--1710},
  year={2022},
  organization={IEEE}
}
```

Previous
Anomalib Documentation

Next
Migrating from 0.* to 1.0