
Auto-PyTorch

Copyright (C) 2021 AutoML Groups Freiburg and Hannover

While early AutoML frameworks focused on optimizing traditional ML pipelines and their hyperparameters, another trend in AutoML is to focus on neural architecture search. To bring the best of these two worlds together, we developed **Auto-PyTorch**, which jointly and robustly optimizes the network architecture and the training hyperparameters to enable fully automated deep learning (AutoDL).

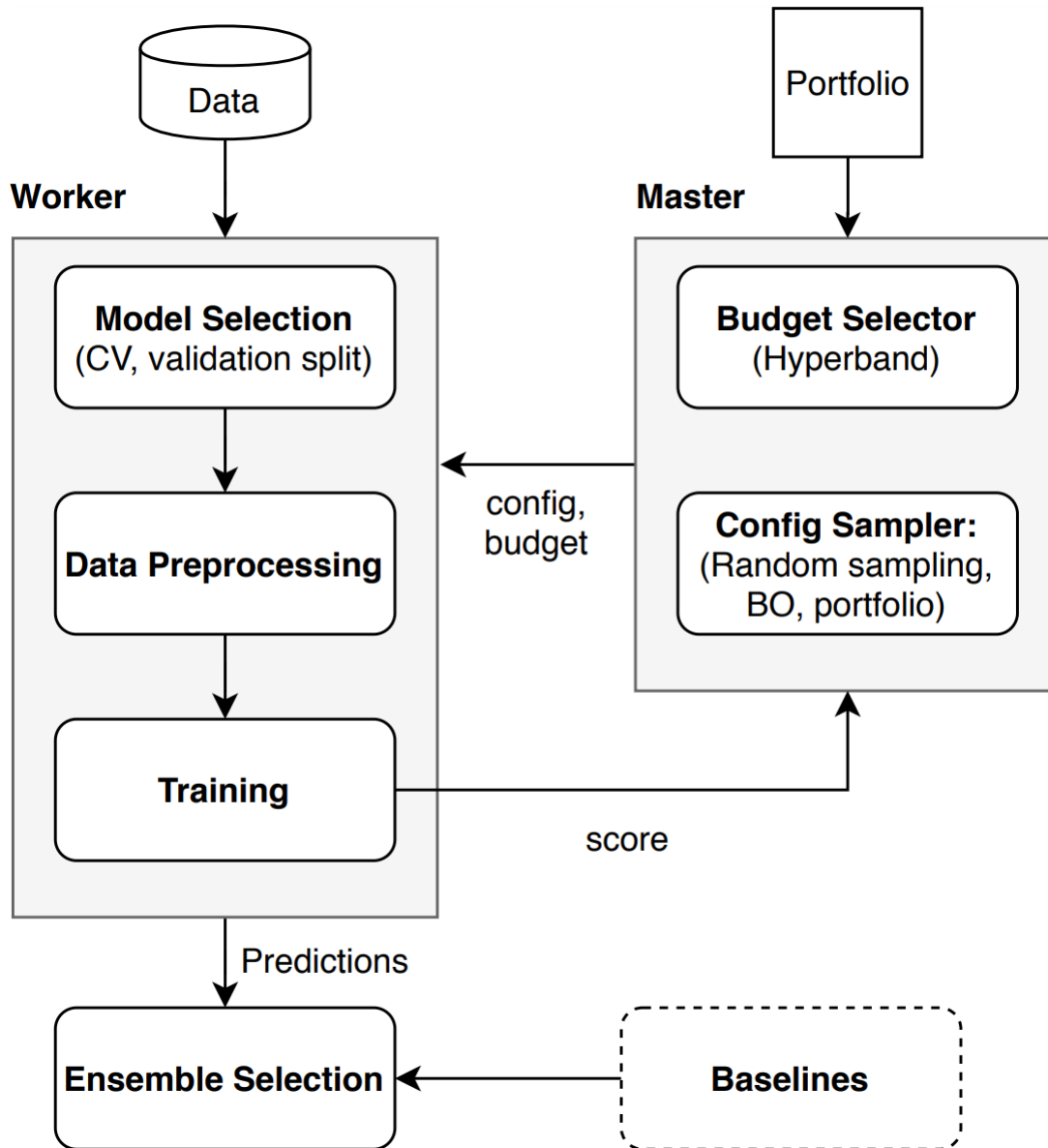
Auto-PyTorch is mainly developed to support tabular data (classification, regression) and time series data (forecasting). The newest features in Auto-PyTorch for tabular data are described in the paper “Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL” (see below for bibtex ref). Details about Auto-PyTorch for multi-horizontal time series forecasting tasks can be found in the paper “Efficient Automated Deep Learning for Time Series Forecasting” (also see below for bibtex ref).

Also, find the documentation [here](#).

From v0.1.0, AutoPyTorch has been updated to further improve usability, robustness and efficiency by using SMAC as the underlying optimization package as well as changing the code structure. Therefore, moving from v0.0.2 to v0.1.0 will break compatibility. In case you would like to use the old API, you can find it at [master_old](#).

Workflow

The rough description of the workflow of Auto-Pytorch is drawn in the following figure.



In the figure, **Data** is provided by user and **Portfolio** is a set of configurations of neural networks that work well on diverse datasets. The current version only supports the *greedy portfolio* as described in the paper *Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL*. This portfolio is used to warm-start the optimization of SMAC. In other words, we evaluate the portfolio on a provided data as initial configurations. Then API starts the following procedures: 1. **Validate input data**: Process each data type, e.g. encoding categorical data, so that Auto-Pytorch can handled. 2. **Create dataset**: Create a dataset that can be handled in this API with a choice of cross validation or holdout splits. 3. **Evaluate baselines** * **Tabular dataset** 1: Train each algorithm in the predefined pool with a fixed hyperparameter configuration and dummy model from *sklearn.dummy* that represents the worst possible performance. **Time Series Forecasting dataset** : Train a dummy predictor that

repeats the last observed value in each series 4. **Search by SMAC:**

- a. Determine budget and cut-off rules by Hyperband
- b. Sample a pipeline hyperparameter configuration *2 by SMAC
- c. Update the observations by obtained results
- d. Repeat a. – c. until the budget runs out 5. Build the best ensemble for the provided dataset from the observations and model selection of the ensemble.

*1: Baselines are a predefined pool of machine learning algorithms, e.g. LightGBM and support vector machine, to solve either regression or classification task on the provided dataset

*2: A pipeline hyperparameter configuration specifies the choice of components, e.g. target algorithm, the shape of neural networks, in each step and (which specifies the choice of components in each step and their corresponding hyperparameters).

Installation

PyPI Installation

```
1
2 pip install autoPyTorch
```

Auto-PyTorch for Time Series Forecasting requires additional dependencies

```
1
2 pip install autoPyTorch[forecasting]
```

Manual Installation

We recommend using Anaconda for developing as follows:

```
1 # Following commands assume the user is in a cloned directory of Auto-
  Pytorch
2
3 # We also need to initialize the automl_common repository as follows
4 # You can find more information about this here:
5 # https://github.com/automl/automl_common/
6 git submodule update --init --recursive
7
8 # Create the environment
9 conda create -n auto-pytorch python=3.8
10 conda activate auto-pytorch
11 conda install swig
12 python setup.py install
```

Similarly, to install all the dependencies for Auto-PyTorch-TimeSeriesForecasting:

```
1
2 git submodule update --init --recursive
3
4 conda create -n auto-pytorch python=3.8
5 conda activate auto-pytorch
6 conda install swig
7 pip install -e[forecasting]
```

Examples

In a nutshell:

```
1 from autoPyTorch.api.tabular_classification import
   TabularClassificationTask
2
3 # data and metric imports
4 import sklearn.model_selection
5 import sklearn.datasets
6 import sklearn.metrics
7 X, y = sklearn.datasets.load_digits(return_X_y=True)
8 X_train, X_test, y_train, y_test = \
9     sklearn.model_selection.train_test_split(X, y, random_state=1)
10
11 # initialise Auto-PyTorch api
12 api = TabularClassificationTask()
13
14 # Search for an ensemble of machine learning algorithms
15 api.search(
16     X_train=X_train,
17     y_train=y_train,
18     X_test=X_test,
19     y_test=y_test,
20     optimize_metric='accuracy',
21     total_walltime_limit=300,
22     func_eval_time_limit_secs=50
23 )
24
25 # Calculate test accuracy
26 y_pred = api.predict(X_test)
27 score = api.score(y_pred, y_test)
28 print("Accuracy score", score)
```

For Time Series Forecasting Tasks

```
1
2 from autoPyTorch.api.time_series_forecasting import
   TimeSeriesForecastingTask
```

```

3
4 # data and metric imports
5 from sktime.datasets import load_longley
6 targets, features = load_longley()
7
8 # define the forecasting horizon
9 forecasting_horizon = 3
10
11 # Dataset optimized by APT-TS can be a list of np.ndarray/ pd.DataFrame
    where each series represents an element in the
12 # list, or a single pd.DataFrame that records the series
13 # index information: to which series the timestep belongs? This id can
    be stored as the DataFrame's index or a separate
14 # column
15 # Within each series, we take the last forecasting_horizon as test
    targets. The items before that as training targets
16 # Normally the value to be forecasted should follow the training sets
17 y_train = [targets[: -forecasting_horizon]]
18 y_test = [targets[-forecasting_horizon:]]
19
20 # same for features. For uni-variant models, X_train, X_test can be
    omitted and set as None
21 X_train = [features[: -forecasting_horizon]]
22 # Here x_test indicates the 'known future features': they are the
    features known previously, features that are unknown
23 # could be replaced with NAN or zeros (which will not be used by our
    networks). If no feature is known beforehand,
24 # we could also omit X_test
25 known_future_features = list(features.columns)
26 X_test = [features[-forecasting_horizon:]]
27
28 start_times = [targets.index.to_timestamp()[0]]
29 freq = '1Y'
30
31 # initialise Auto-PyTorch api
32 api = TimeSeriesForecastingTask()
33
34 # Search for an ensemble of machine learning algorithms
35 api.search(
36     X_train=X_train,
37     y_train=y_train,
38     X_test=X_test,
39     optimize_metric='mean_MAPE_forecasting',
40     n_prediction_steps=forecasting_horizon,
41     memory_limit=16 * 1024, # Currently, forecasting models use much
        more memories
42     freq=freq,
43     start_times=start_times,
44     func_eval_time_limit_secs=50,
45     total_walltime_limit=60,
46     min_num_test_instances=1000, # proxy validation sets. This only

```

```
        works for the tasks with more than 1000 series
47     known_future_features=known_future_features,
48 )
49
50 # our dataset could directly generate sequences for new datasets
51 test_sets = api.dataset.generate_test_seqs()
52
53 # Calculate test accuracy
54 y_pred = api.predict(test_sets)
55 score = api.score(y_pred, y_test)
56 print("Forecasting score", score)
```

For more examples including customising the search space, parellising the code, etc, checkout the [examples](#) folder

```
1 $ cd examples/
```

Code for the paper is available under [examples/ensemble](#) in the TPAMI.2021.3067763 branch.

Contributing

If you want to contribute to Auto-PyTorch, clone the repository and checkout our current development branch

```
1 $ git checkout development
```

License

This program is free software: you can redistribute it and/or modify it under the terms of the Apache license 2.0 (please see the LICENSE file).

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

You should have received a copy of the Apache license 2.0 along with this program (see LICENSE file).

Reference

Please refer to the branch [TPAMI.2021.3067763](#) to reproduce the paper *Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for Efficient and Robust AutoDL*.

```
1 @article{zimmer-tpami21a,
2   author = {Lucas Zimmer and Marius Lindauer and Frank Hutter},
```

```
3 title = {Auto-PyTorch Tabular: Multi-Fidelity MetaLearning for
4         Efficient and Robust AutoDL},
5 journal = {IEEE Transactions on Pattern Analysis and Machine
6             Intelligence},
7 year = {2021},
8 note = {also available under https://arxiv.org/abs/2006.13799},
9 pages = {3079 - 3090}
10 }
```

```
1 @incollection{mendoza-automlbook18a,
2   author    = {Hector Mendoza and Aaron Klein and Matthias Feurer and
3               Jost Tobias Springenberg and Matthias Urban and Michael Burkart
4               and Max Dippel and Marius Lindauer and Frank Hutter},
5   title     = {Towards Automatically-Tuned Deep Neural Networks},
6   year      = {2018},
7   month     = dec,
8   editor    = {Hutter, Frank and Kotthoff, Lars and Vanschoren, Joaquin
9               },
10  booktitle = {AutoML: Methods, Sytems, Challenges},
11  publisher = {Springer},
12  chapter   = {7},
13  pages      = {141--156}
14 }
```

```
1 @article{deng-ecml22,
2   author    = {Difan Deng and Florian Karl and Frank Hutter and Bernd
3               Bischl and Marius Lindauer},
4   title     = {Efficient Automated Deep Learning for Time Series
5               Forecasting},
6   year      = {2022},
7   booktitle = {Machine Learning and Knowledge Discovery in Databases.
8               Research Track
9               - European Conference, {ECML} {PKDD} 2022},
10  url       = {https://doi.org/10.48550/arXiv.2205.05511},
11 }
```

Contact

Auto-PyTorch is developed by the AutoML Groups of the University of Freiburg and Hannover.