

AutoML: Neural Architecture Search (NAS)

NASLib: A Modular and Extensible NAS Library

Bernd Bischl Frank Hutter Lars Kotthoff
Marius Lindauer Joaquin Vanschoren

Motivation for NASLib [Zela et al., 2020]

NASLib is a **framework** for easily implementing different NAS methods, aiming to:

- Allow **fair comparisons without confounding factors**, which could be due to
 - Different codebases
 - Different search and evaluation pipelines
 - Different hyperparameter settings
 - Other confounding factors, e.g., library versions, GPU types, etc.

Motivation for NASLib [Zela et al., 2020]

NASLib is a **framework** for easily implementing different NAS methods, aiming to:

- Allow **fair comparisons without confounding factors**, which could be due to
 - Different codebases
 - Different search and evaluation pipelines
 - Different hyperparameter settings
 - Other confounding factors, e.g., library versions, GPU types, etc.
- **Modularize** different components of NAS optimizers to allow combining them

Motivation for NASLib [Zela et al., 2020]

NASLib is a **framework** for easily implementing different NAS methods, aiming to:

- Allow **fair comparisons without confounding factors**, which could be due to
 - Different codebases
 - Different search and evaluation pipelines
 - Different hyperparameter settings
 - Other confounding factors, e.g., library versions, GPU types, etc.
- **Modularize** different components of NAS optimizers to allow combining them
- Offer **researchers** a convenient way of prototyping new NAS methods

Motivation for NASLib [Zela et al., 2020]

NASLib is a **framework** for easily implementing different NAS methods, aiming to:

- Allow **fair comparisons without confounding factors**, which could be due to
 - Different codebases
 - Different search and evaluation pipelines
 - Different hyperparameter settings
 - Other confounding factors, e.g., library versions, GPU types, etc.
- **Modularize** different components of NAS optimizers to allow combining them
- Offer **researchers** a convenient way of prototyping new NAS methods
- Offer **users** reliable implementations of NAS methods
 - ▶ Facilitate the use of NAS for new search spaces
 - ▶ Develop a robust true AutoML framework

NASLib building blocks: Search Spaces, Optimizers, Evaluators

- NASLib implements a broad range of NAS optimizers
 - ▶ Blackbox NAS methods, e.g., Regularized Evolution
 - ▶ One-shot NAS methods, e.g., DARTS

NASLib building blocks: Search Spaces, Optimizers, Evaluators

- NASLib implements a broad range of NAS **optimizers**
 - ▶ **Blackbox NAS methods**, e.g., Regularized Evolution
 - ▶ **One-shot NAS methods**, e.g., DARTS
- The **optimizers are modularized**
 - ▶ This allows to switch from, e.g., DARTS to GDAS or PC-DARTS by just one method call

NASLib building blocks: Search Spaces, Optimizers, Evaluators

- NASLib implements a broad range of NAS **optimizers**
 - ▶ **Blackbox NAS methods**, e.g., Regularized Evolution
 - ▶ **One-shot NAS methods**, e.g., DARTS
- The **optimizers are modularized**
 - ▶ This allows to switch from, e.g., DARTS to GDAS or PC-DARTS by just one method call
- The **evaluators are agnostic to the origin of an architecture**
 - ▶ The final architecture is run using exactly the same object to evaluate on the test set

NASLib building blocks: Search Spaces, Optimizers, Evaluators

- NASLib implements a broad range of NAS **optimizers**
 - ▶ **Blackbox NAS methods**, e.g., Regularized Evolution
 - ▶ **One-shot NAS methods**, e.g., DARTS
- The **optimizers are modularized**
 - ▶ This allows to switch from, e.g., DARTS to GDAS or PC-DARTS by just one method call
- The **evaluators are agnostic to the origin of an architecture**
 - ▶ The final architecture is run using exactly the same object to evaluate on the test set
- NASLib's main building block is the graph object represented as a **NetworkX**¹ graph
 - **Easily manipulate the graph** by adding/removing nodes/edges
 - **Hide complexity** of dealing with the PyTorch computational graph
 - **Easy high-level way of creating complex structures**, e.g., hierarchical search spaces

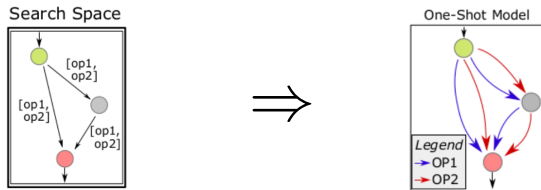
¹<https://networkx.github.io/>

NASLib building blocks: Search Spaces, Optimizers, Evaluators

- The optimizers are agnostic to the search space they are running on
 - ▶ This facilitates their use for new types of search spaces

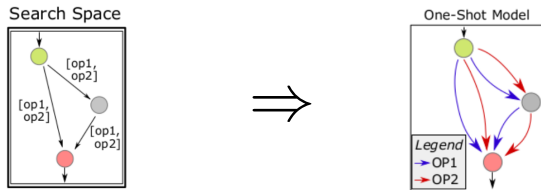
NASLib building blocks: Search Spaces, Optimizers, Evaluators

- The optimizers are agnostic to the search space they are running on
 - This facilitates their use for new types of search spaces
- An optimizer takes the search space as a NetworkX object and builds the PyTorch computational graph



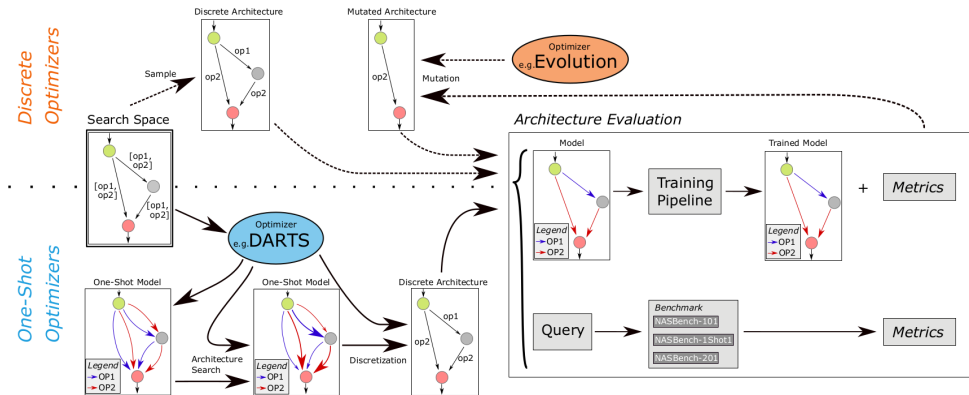
NASLib building blocks: Search Spaces, Optimizers, Evaluators

- The **optimizers** are **agnostic** to the **search space** they are running on
 - ▶ This facilitates their use for new types of search spaces
- An optimizer takes the search space as a **NetworkX** object and builds the **PyTorch** computational graph



- Depending on the optimizer, each operation choice in the NetworkX object becomes:
 - a **MixedOp** – for one-shot NAS optimizers, e.g. DARTS
 - a **CategoricalOp** – for black-box optimizers, e.g. Regularized Evolution

NASLib: Overview



Tabular benchmarks for one-shot NAS

- NAS-Bench-101 [Ying et al. 2019] is not directly compatible with one-shot NAS methods
 - ▶ Mainly due to the constraint of at most 9 edges in the cell

Tabular benchmarks for one-shot NAS

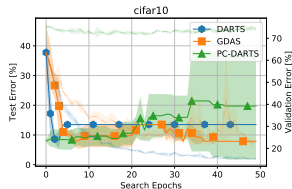
- NAS-Bench-101 [Ying et al. 2019] is not directly compatible with one-shot NAS methods
 - ▶ Mainly due to the constraint of at most 9 edges in the cell
- NAS-Bench-1Shot1 [Zela et al. 2020]
 - 3 sub-spaces of NAS-Bench-101 that are compatible with one-shot methods
 - ★ 6 240, 29 160, and 363 648 architectures, respectively
 - Currently the largest one-shot NAS tabular benchmark

Tabular benchmarks for one-shot NAS

- NAS-Bench-101 [Ying et al. 2019] is not directly compatible with one-shot NAS methods
 - ▶ Mainly due to the constraint of at most 9 edges in the cell
- NAS-Bench-1Shot1 [Zela et al. 2020]
 - 3 sub-spaces of NAS-Bench-101 that are compatible with one-shot methods
 - ★ 6 240, 29 160, and 363 648 architectures, respectively
 - Currently the largest one-shot NAS tabular benchmark
- NAS-Bench-201 [Dong and Yang. 2020]
 - Much smaller than NAS-Bench-101 and largest NAS-Bench-1Shot1 subspace
 - ★ 15 625 architectures
 - Every architecture in the search space evaluated on 3 image classification datasets

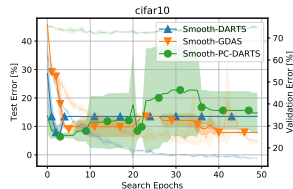
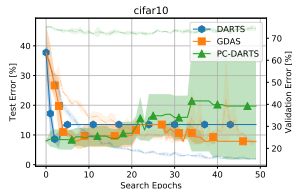
NASLib case study: Results on NAS-Bench-201

- NAS-Bench-201 is already integrated in NASLib and we can run any one-shot optimizer on it



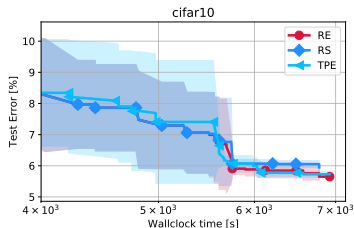
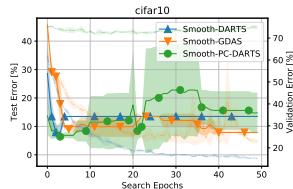
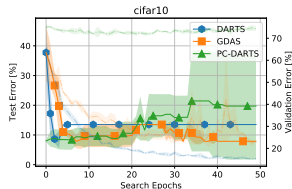
NASLib case study: Results on NAS-Bench-201

- NAS-Bench-201 is already integrated in NASLib and we can run any one-shot optimizer on it
- We can also combine random perturbations [Chen and Hsieh, 2020] with any one-shot optimizer



NASLib case study: Results on NAS-Bench-201

- NAS-Bench-201 is already integrated in NASLib and we can run any one-shot optimizer on it
- We can also combine random perturbations [Chen and Hsieh, 2020] with any one-shot optimizer
- We can also evaluate black-box optimizers cheaply with a tabular benchmark



Opportunities with NASLib

Room for many interesting projects and theses

- Applications of NAS to *your problem of interest*, with interesting search spaces
 - ▶ NASLib is the first library that separates the NAS method from the search spaces
 - ★ Therefore, *no changes are required in the NAS methods*
 - ★ This should make new applications much easier

Opportunities with NASLib

Room for many interesting projects and theses

- Applications of NAS to **your problem of interest**, with interesting search spaces
 - ▶ NASLib is the first library that separates the NAS method from the search spaces
 - ★ Therefore, **no changes are required in the NAS methods**
 - ★ This should make new applications much easier
- Studying **hierarchical search spaces** in detail

Opportunities with NASLib

Room for many interesting projects and theses

- Applications of NAS to **your problem of interest**, with interesting search spaces
 - ▶ NASLib is the first library that separates the NAS method from the search spaces
 - ★ Therefore, **no changes are required in the NAS methods**
 - ★ This should make new applications much easier
- Studying **hierarchical search spaces** in detail
- **Combining different components** of existing NAS methods
 - ▶ So far, it has been very hard on a code level to mix and match components
 - ▶ It ought to be possible to design the world's best NAS method by combining the right components

Opportunities with NASLib

Room for many interesting projects and theses

- Applications of NAS to **your problem of interest**, with interesting search spaces
 - ▶ NASLib is the first library that separates the NAS method from the search spaces
 - ★ Therefore, **no changes are required in the NAS methods**
 - ★ This should make new applications much easier
- Studying **hierarchical search spaces** in detail
- **Combining different components** of existing NAS methods
 - ▶ So far, it has been very hard on a code level to mix and match components
 - ▶ It ought to be possible to design the world's best NAS method by combining the right components

Room for interesting Hiwi projects

- Not everything is perfect yet, we can use lots of support by great programmers

Questions to Answer for Yourself / Discuss with Friends

- Repetition:
What would one have to do in order to apply the methods in NASLib to a new search space?
- Discussion:
Is there a problem of your interest that you would you like to apply the methods in NASLib to?
- Discussion:
Given that NASLib's modular design allows mixing and matching components of one-shot NAS methods, which of the methods we discussed might make sense to combine?

NASLib: A Modular and Extensible NAS Library