Long-term Reproducibility for Neural Architecture Search

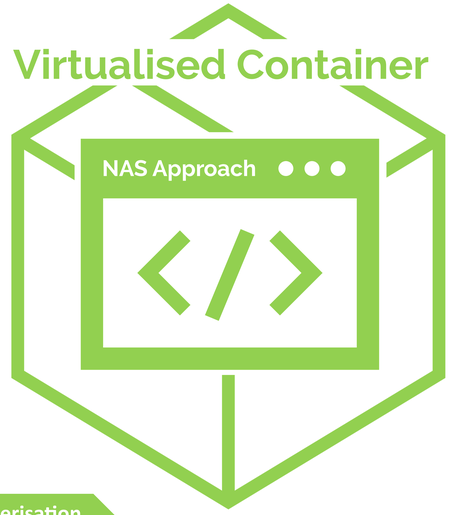
David Towers[1], Matthew Forshaw[1], A. Stephen McGough[1], Amir Atapour-Abarghouei[2]
[1]Newcastle University, UK — [2]Durham University, UK

Background

- Many NAS approaches are difficult to reproduce.
- With issues such as:
  - Missing/inaccurate documentation.
  - Major bugs in the code.
  - Missing dependencies.
- We created a checklist of five categories to encourage better reproducibility.
- We address the issue of unavailable libraries through containerisation tools such as Docker/Podman

Conclusions

- There is no standard IO for NAS, which means that each work has to be individually modified.
- Approaches as recent as 2021 have issues hindering reproducibility.
- Many Issues can be solved using containerisation techniques with the authors original environment.



Containerisation

By using containerisation tools such as Docker or Podman we can gain multiple features for Long-Term Reproducibility:

- All dependencies zipped up together.
- Automatically build the correct environment.
- Make an exact copy of the Author's original environment.

Criteria	Descriptor	ENAS	DARTS	PC-DARTS	DrNAS
Code Stability					
Seed reporting	The code should return the seed used in the logs/output.	✗	✓	✓	✓
Seed Setting	The seed should be able to be set.	✗	✓	✓	✓
Bug-Free	The code should be free of bugs which prevent sections of the code from working.	✗	✓	✓	✗[i]
Documentation					
Examples	There should be examples of all the high-level commands.	✓	✓	✓	✓
Argument Details	The documentation should include details regarding any additional arguments.	✗	✗	✗	✗
Dependency List	Environment Specifications indicating how to replicate the environment.	✗	✓	✓	✓
Pipeline Instructions	Instructions on how to Search, Train, and Test a model from scratch.	✗[ii]	✗[iii]	✗[iii]	✗[iii]
Ease of running					
Dependency Resilience	Dependencies should be installed with the code.	✗	✗	✗	✗
Executable	The code can be executed from an executable file, such as a bash script.	✓	✗	✗	✗
Intuitive Commands	The commands should be indicative of their function.	✓	✓	✓	✓
Standardisation					
Standard Phases	Search, Train, and Test.	✗	✓	✓	✗
Data Inclusive	Data is retrieved from a 'data' directory that accepts common data formats.	✗	✓	✓	✓
Standard output	The model is saved in an output folder in a form that is easily imported into other scripts.	✗	✓	✓	✓
Future Proofing					
Accessible Pseudocode	Pseudocode should be made available to allow recreation of the code.	✗	✗	✗	✗
Containerised Environment	A compressed file of the containerised image, which allows the entire environment to be saved.	✗	✗	✗	✗
Container Build File	A file containing the required info for a container manager to build and run the code.	✗	✗	✗	✗
Source Code	Source code should be publicly accessible.	✓	✓	✓	✓

Evaluated NAS algorithms using our proposed checklist. [i]Code is missing a 'hp', which needs to manually entered. [ii]Does not mention you need to change the default architecture in the training file. [iii]Does not explicitly explain how to train a found architecture.

Newcastle University

Durham University