Towards modular and programmable architecture search

Renato Negrinho   Darshan Patil   Nghia Le   Daniel Ferreira   Matthew Gormley   Geoffrey Gordon

MACHINE LEARNING DEPARTMENT

Highlights

This work

- **Formal language to encode search spaces over architectures**
  - Decouples implementation of search space and search algorithm
  - Search algorithms interact with search spaces through a well-defined interface
- **Modular and programmable framework for architecture search over general domains**
  - Use architecture search for your use cases
  - Implement your search spaces and search algorithms
  - Mix and match implementations of search spaces and search algorithms without writing combinations from scratch
- **Code and documentation at github.com/negrinho/deep_architect.**
  - Contribute your search spaces, search algorithms, benchmarks, and more.
  - Easy wrapping of Pytorch, Tensorflow, and Keras layers.

Deep Architect

Motivation

- **Hyperparameter optimization not focused on architecture search**
- **No existing general architecture search systems**
  - Ad-hoc encodings for search spaces
  - Intertwined search space and search algorithm
  - Task-specific, e.g., image classification.
- **Programmable frameworks (e.g., deep learning) had transformative impact on machine learning**

Search space example

Description

- input convolutional layer
- optional dropout w/ rate 0.25 or 0.5
- two parallel convolutional chains:
  - one chain length 1, 2, or 4
  - other chain double the first
- chain outputs concatenated
- each convolution has 64 or 128 filters (chosen separately)

25008 possible architectures

Code

```
def search_space():
    h_n = D([1, 2, 4])
    h_ndep = DependentHyperparameter(
        lambda x: 2 * x, {"x": h_n})

    c_inputs, c_outputs = conv2d(D([64, 128]))
    o_inputs, o_outputs = siso_optional(
        lambda: dropout(D([0.25, 0.5])))
    fn = lambda: conv2d(D([64, 128]))
    r1_inputs, r1_outputs = siso_repeat(fn, h_n)
    r2_inputs, r2_outputs = siso_repeat(fn, h_ndep)
    cc_inputs, cc_outputs = concat(2)

    o_inputs["in"].connect(c_outputs["out"])
    r1_inputs["in"].connect(o_outputs["out"])
    r2_inputs["in"].connect(o_outputs["out"])
    cc_inputs["in0"].connect(r1_outputs["out"])
    cc_inputs["in1"].connect(r2_outputs["out"])
    return c_inputs, cc_outputs
```

Transitions

a) Search space encoded by code

a → b) Value assigned to *IH-1*

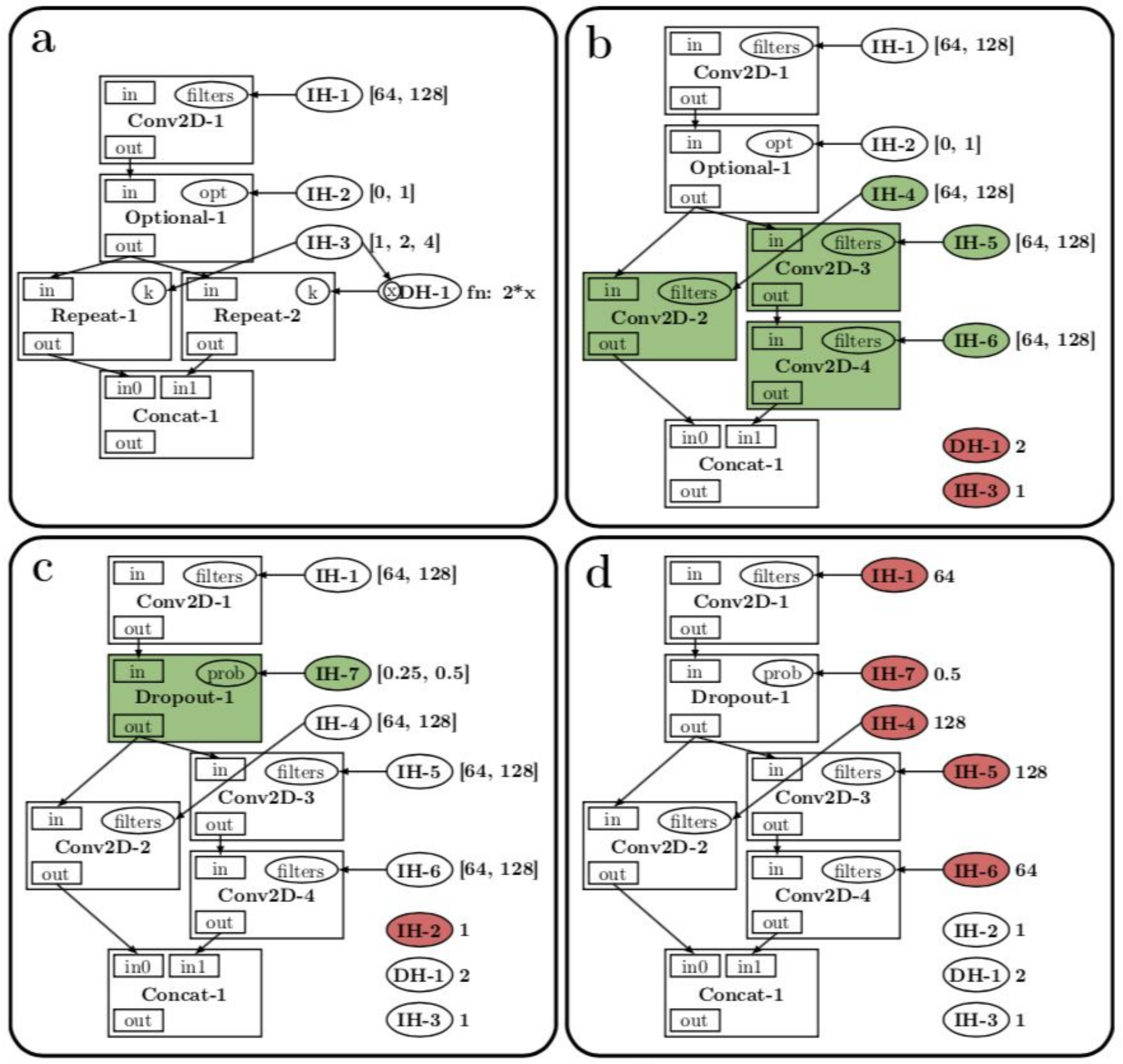
- Triggers value assignment to DH-1
- Triggers substitution for Repeats (1 and 2)

b → c) Value assigned to IH-2

- Triggers substitution for Optional-1

c → d) Assignments to IH-1 (64), IH-4 (128), IH-5 (128), IH-6 (64), and IH-7 (0.5)

d) can be mapped to implementation



Newly created modules and hyperparameters

Newly assigned hyperparameters

Language

Constructs

- Independent hyperparameters
  - Value picked from set (e.g., IH-1)
- Dependent hyperparameters
  - Value computed as function of other hypers (e.g., DH-2)
- Basic modules
  - Deep learning operation (Conv2D-1)
- Substitution modules
  - Lazy transformations (e.g., Repeat-1) to computation graph through substitutions (replace and reroute)
- Auxiliary functions
  - Helps compose search spaces into larger search spaces (e.g., rnn_cell)

```
def rnn_cell(hidden_fn, output_fn):
    h_inputs, h_outputs = hidden_fn()
    y_inputs, y_outputs = output_fn()
    h_outputs["out"].connect(y_inputs["in"])
    return h_inputs, y_outputs
```

Mechanics

- Search algorithms interface with search spaces by assigning values to independent hyperparameters
- After all hyperparameters have values assigned, architecture mapped automatically to implementation

Experiments

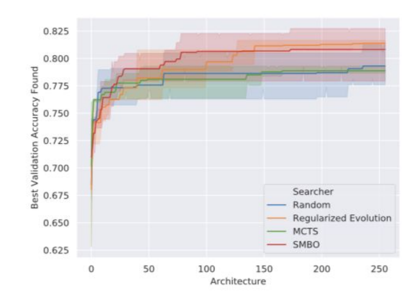
Mix and match search spaces and search algorithms without implementing each combination from scratch

One search algorithm (random), many search spaces

Table 1: Test results for search space experiments.

Search Space	Test Accuracy
Genetic [23]	90.07
Flat [15]	93.58
Nasbench [24]	94.59
Nasnet [25]	93.77

One search space (genetic), many search algorithms



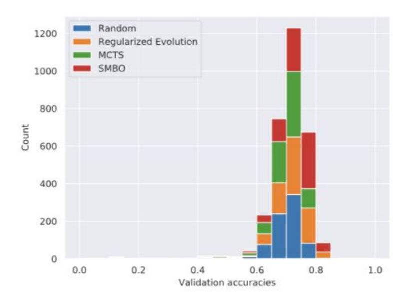


Table 2: Test results for search algorithm experiments.

Search algorithm	Test Accuracy
Random	91.61 ± 0.67
MCTS [26]	91.45 ± 0.11
SMBO [16]	91.93 ± 1.03
Evolution [14]	91.32 ± 0.50