ConfuciuX: Hardware Design-space Exploration via RL and Optimization

Sheng-Chun (Felix) Kao
Georgia Institute of Technology

S.-C. Kao, G Jeong, T Krishna, “ConfuciuX: Autonomous Hardware Resource Assignment for DNN Accelerators using Reinforcement Learning”, MICRO, 2020

Code available: https://github.com/maestro-project/confuciuX
For this work, we assume the dataflow is fixed.
Architecture of DNN Accelerator

DNN model

Platform

Deployment scenario

The model is mapped layer by layer.

The entire model (i.e., weights) are mapped to the chip.

Platform budget

DNN model

MobileNet-V2

Edge

IoT

What is the impact of HW resources on performance and energy?
The Impact of HW Resources Assignment

Different layers react to the PEs/ Buffers differently.

---

### PEs

<table>
<thead>
<tr>
<th>Latency (cy.)</th>
<th>Energy (1e3 Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>3</td>
</tr>
<tr>
<td>40</td>
<td>2</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
</tr>
</tbody>
</table>

### Buffers

<table>
<thead>
<tr>
<th>Area (um²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

### Layer 12 (CONV2D)

### Layer 34 (CONV2D)

### Layer 23 (DWCONV)

Selected layer from MobileNet-V2

https://github.com/maestro-project/maestro
Casting HW Resource Assignment as RL

**Goal:** Given power/area budget, find \{PE, Buf\} for each layer in a layer pipelined deployment scenario, such that objective (e.g., latency) is minimized

**RL algorithm:**

- Multiple episodes (until the algorithm converges)

**Each episode:**

- Multiple steps (until the environment terminates)
- Environment terminates when entire DNN is run or if environment runs out of area or power

**Each step:**

- Agent assigns \{PE, Buf\} to a layer
- Environment tracks the remaining power/area budget

*Layer Pipelined: The entire model (i.e., weights) are mapped to the chip.*
Template of Reinforcement Learning

Agent

Environment

Action

Reward

Update

HW resource scheduler

HW cost evaluation Env.

Accel cost

Resource cstr.

Resource model

Observation/ State

Dead

ConfuciuX | MICRO 2020

Felix Kao | Georgia Institute of Technology

October 20, 2020
Algorithm Details

Objective: Latency or energy

Platform constraint: Total chip area or power

---

Agent

The objective value (latency/energy)
Large penalty if the constraint is violated

Action

Reward

Environment

MAESTRO (DNN accelerator analytical model)

Observation/ State

(1) Current DNN layer description
(2) Last action
(3) Current status info. (step #, episode done signal)

---

RNN as underlying policy network, REINFORCE-based method

---

https://github.com/maestro-project/maestro

---

Objective: Latency or energy

Platform constraint: Total chip area or power

---

MAESTRO (DNN accelerator analytical model)
Action Space

The fine-grained search/action space is extremely large

- 52 layers MobileNet-V2: $2^{728} \rightarrow O(10^{211})$ years

Approach: Two-stage optimization

- **Stage 1:** Reinforcement Learning (RL) coarse-grained global search
  - We limit the actions of \{PE\}, \{Buf\} to 12 different values
    - PE sizes chosen empirically
    - Buf sizes increase by the unit of tile sizes

- **Stage 2:** Genetic Algorithm (GA) local fine-tuning
  - Fine-tune the stage-1 solution locally

<table>
<thead>
<tr>
<th>Action level values</th>
</tr>
</thead>
<tbody>
<tr>
<td>PEs</td>
</tr>
<tr>
<td>1, 2, 4, 8, 12, 16, 24, 32, 48, 64, 96, 128</td>
</tr>
<tr>
<td>Buffers (e.g., NVIDA-style)</td>
</tr>
</tbody>
</table>

---

ConfuciuX | MICRO 2020
Felix Kao | Georgia Institute of Technology
October 20, 2020
Overview of ConfuciuX

RL: Sample efficient

RL-based
Coarse-grained global search

GA-based
Local fine-tuning

Input
DNN model
Deployment
scenario (LP/LS)
Objective
(Latency/energy)
Constraint
(Power/area)
Dataflow
(NVDLA-style,
Eyeriss-style)

Action

Reward

State

ConfuciuX

Output
Fine-tuned sol.

Coarse-grained sol.

Hardware resource assignment strategy

Genetic encoding

Selection

Mutation

Crossover

PE
Buf
96
19
32
47
29
96


Detail in the next two slides
Algorithm Flow of ConfuciuX RL-based Search

Episode 1

1. Action: PE 4, Buf 6
2. Reward: -100
3. State: PE 9, Buf 7

Constraint is checked at every step, if violated, it early terminate with large penalty.

Accumulated Reward: -55

Episode 2

1. Action: PE 5, Buf 3
2. Reward: 15
3. State: PE 12, Buf 9

Output HW assignment strategy

Accumulated Reward: 57
Local Fine-tuning using GA

GA action space
- ± step size
- E.g., step size=4, PEs=64, action space=[60, 68]

Genetic Operators

Local mutation

Coarse-grained sol.

<table>
<thead>
<tr>
<th>PE</th>
<th>Buf</th>
</tr>
</thead>
<tbody>
<tr>
<td>64</td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>49</td>
</tr>
<tr>
<td>49</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>96</td>
</tr>
<tr>
<td>96</td>
<td>99</td>
</tr>
<tr>
<td>99</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>19</td>
</tr>
</tbody>
</table>

Local-mutated sol.

<table>
<thead>
<tr>
<th>PE</th>
<th>Buf</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>47</td>
</tr>
<tr>
<td>47</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>96</td>
</tr>
<tr>
<td>96</td>
<td>99</td>
</tr>
<tr>
<td>99</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>19</td>
</tr>
</tbody>
</table>

Local crossover

Coarse-grained sol.

<table>
<thead>
<tr>
<th>PE</th>
<th>Buf</th>
</tr>
</thead>
<tbody>
<tr>
<td>66</td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>47</td>
</tr>
<tr>
<td>47</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>96</td>
</tr>
<tr>
<td>96</td>
<td>99</td>
</tr>
<tr>
<td>99</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>19</td>
</tr>
</tbody>
</table>

Local-crossovered sol.

<table>
<thead>
<tr>
<th>PE</th>
<th>Buf</th>
</tr>
</thead>
<tbody>
<tr>
<td>96</td>
<td>19</td>
</tr>
<tr>
<td>19</td>
<td>32</td>
</tr>
<tr>
<td>32</td>
<td>47</td>
</tr>
<tr>
<td>47</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>66</td>
</tr>
<tr>
<td>66</td>
<td>99</td>
</tr>
<tr>
<td>99</td>
<td>16</td>
</tr>
<tr>
<td>16</td>
<td>19</td>
</tr>
</tbody>
</table>

More details in the paper.
ConfuciuX Setup

• Setup
  • git clone https://github.com/maestro-project/confuciuX.git
  • conda activate gammaEnv
  • pip install -r requirements.txt
  • python build.py

• Run ConfuciuX
  • sh ./run_ConfX.sh

• Run optimization methods
  • sh ./run_otherOpts.sh

• Run RL methods
  • sh ./run_otherRLs.sh
ConfuciuX Code-base

User will select one of them from the argument

Supported Mapping/dataflow

Supported model descriptions

Result directory

Main program

Change the buffer size by changing tile size of K. (Here is where your action for Buffer apply.)

(However, it is not required to change by the K dim., you can use your own dataflow file to explore Buffer with different tile dimension.)

/data/dataflow/dla.m

Dataflow {
  SpatialMap(KTileSz,KTileSz) K;
  TemporalMap(ClusterSz,ClusterSz) C;
  TemporalMap(Sz(R),Sz(R)) R;
  TemporalMap(Sz(S),Sz(S)) S;
  TemporalMap(Sz(R),1) Y;
  TemporalMap(Sz(S),1) X;
  Cluster(ClusterSz, P);
  SpatialMap(1,1) C;
  TemporalMap(Sz(R),1) Y;
  TemporalMap(Sz(S),1) X;
  TemporalMap(Sz(R),Sz(R)) R;
  TemporalMap(Sz(S),Sz(S)) S;
}
python main.py --outdir outdir --model example --fitness latency --cstr area --mul 0.5 --epochs 500 --df shi --alg RL_GA

**Objective**
- *fitness*: The fitness objective (latency/energy)
- *df*: The dataflow strategy
- *model*: The model to run (available model in model dir)
- *cstr*: Constraint (area/power)

**Constraint**
- *mul*: Resource multiplier. The resource ratio, the design is allowed to use. 

*The system under design is only allowed to use mul * power_max or mul * area_max.*
- *epochs*: Number of generation for the optimization
- *alg*: The algorithm to run
  - For ConX, choose from [RL, RL_GA]
  - For RL, choose from [PPO2, A2C, ACKTR, SAC, TD3, DDPG]
  - For optimization methods, choose from [genetic, random, bayesian, anneal, exhaustive]
- *outdir*: The output result directory

**Hyper parameters**
- Learning rate, discount ratio, mutation rate, etc., can be set at codebase
User-defined Action Space

Inside ./src/utils/get_action_space.py

Define your action space here

[[The choice for # of PEs]
[The choice for buffer sizes]]
[Advanced User] ConfuciuX-specific Tuning

RL: In ./src/ConfX/rl_confx.py

```python
LR_ACTOR = 1e-3  # learning rate of the actor
GAMMA = 0.9  # discount factor
CLIPPING_LSTM = 10
CLIPPING_MODEL = 100
EPISIOLON = 2**(-12)
```

RL-specific hyper-parameters

GA: In ./src/ConfX/main.py

```python
def genetic_search(best_sol, best_reward, action_bound, action_bottom, num_layers=None, num_generations=100, num_pop=20):
    ...  # genetic search
```

Tune number of generation/populations

In ./src/ConfX/ga_confx.py

```python
def self_crossover(pop, eps=0):
    ...
```

Mutation ratio.

In ./src/Other_Opts/main.py

```
if method == "random":
    env.random_search(opt.epochs, chkpt_file)
elif method == "exhaustive":
    env.exhaustive_search(opt.epochs, chkpt_file, stride=opt.stride)
elif method == "genetic":
    env.genetic_search(epochs=opt.epochs, chkpt_file=chkpt_file)
elif method == "bayesian":
    env.bayesian_search(opt.epochs, chkpt_file)
elif method == "anneal":
    env.annealing_search(opt.epochs, chkpt_file=chkpt_file)
else:
    print("Please choose from [genetic, random, bayesian, anneal, exhaustive]")
    exit(-1)
```

Add your algorithm here

In ./src/Other_Opts/other_opt_env.py

```
Random search method

for epoch in range(max_epoch):
    self.epoch = epoch
    guess_action = []
    for _ in range(n_layer):
        self.start_range = start_range
        self.end_range = end_range
        pe = action_space[0][random.randint(self.start_range, self.end_range)]
        bf = action_space[1][random.randint(self.start_range, self.end_range)]
        action = [pe, bf]
        guess_action.append(action)
    reward, total_used_constraint = self.exterior_search(guess_action)
```

Add your algorithm here

Random search method

You could use the same code structure.
The only work for your algorithm is to suggest [PE, Buffer] for each layer.
Resources

Paper:
S.-C. Kao, G Jeong, T Krishna, “Confuciux: Autonomous Hardware Resource Assignment for DNN Accelerators using Reinforcement Learning”, MICRO, 2020

Repo:
https://github.com/maestro-project/confuciux