DANCE: Differentiable Accelerator/Network Co-Exploration

Kanghyun Choi*, Deokki Hong*, Hojae Yoon*, Joonsang Yu, Youngsok Kim, and Jinho Lee

*equal contribution

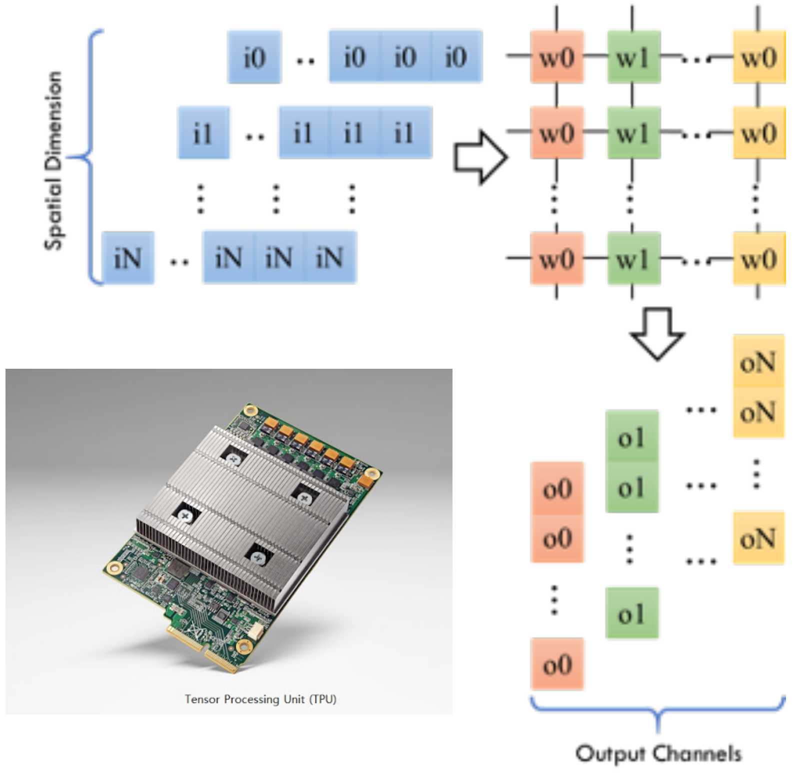
Current Trend

- Many interests in optimizing DNN
  - Hardware design
    - TPU[1]
  - Dataflow
    - Decisions about for-loop orders of DNN
    - Choose which type of data will be reused
  - Neural network design
    - Depthwise-Separable Convolution[2]
    - Neural Architecture search

[1] Norman P Jouppi et al. “In-datacenter performance analysis of a tensor processing unit”. In: ISCA. 2017.
[2] Andrew G Howard et al. “Mobilenets: Efficient convolutional neural networks for mobile vision applications”. In: arXiv preprint arXiv:1704.04861 (2017).

Current Trend

- Many interests in optimizing DNN
  - **Hardware design**
    - TPU[1]
  - Dataflow
    - Decisions about for-loop orders of DNN
    - Choose which type of data will be reused
  - Neural network design
    - Depthwise-Separable Convolution[2]
    - Neural Architecture search



TPU structure

[1] Norman P Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". In: ISCA. 2017.
[2] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: arXiv preprint arXiv:1704.04861 (2017).

Current Trend

- Many interests in optimizing DNN
  - Hardware design
    - TPU[1]
  - **Dataflow**
    - Decisions about for-loop orders of DNN
    - Choose which type of data will be reused
  - Neural network design
    - Depthwise-Separable Convolution[2]
    - Neural Architecture search

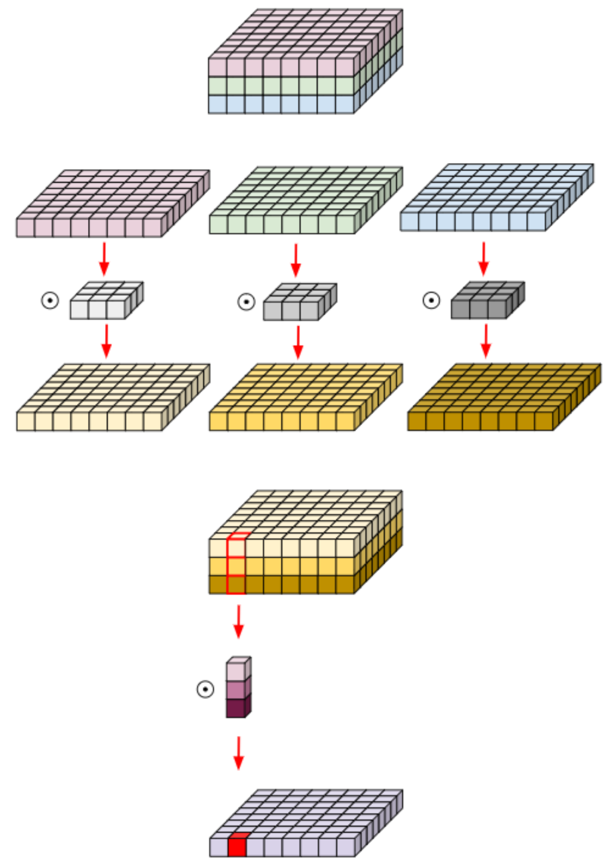
```
for n in (1,N)
  for k in (1,K)
    for c in (1,C)
      for w in (1,W)
        for h in (1,H)
          for r in (1,R)
            for s in (1,S)
              out[n,k,w,h] +=
                in[n,c,w+r-1,h+s-1]*
                weight[k,c,r,s]
```

Example of for-loop order for operating a DNN

[1] Norman P Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". In: ISCA. 2017.
[2] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: arXiv preprint arXiv:1704.04861 (2017).

Current Trend

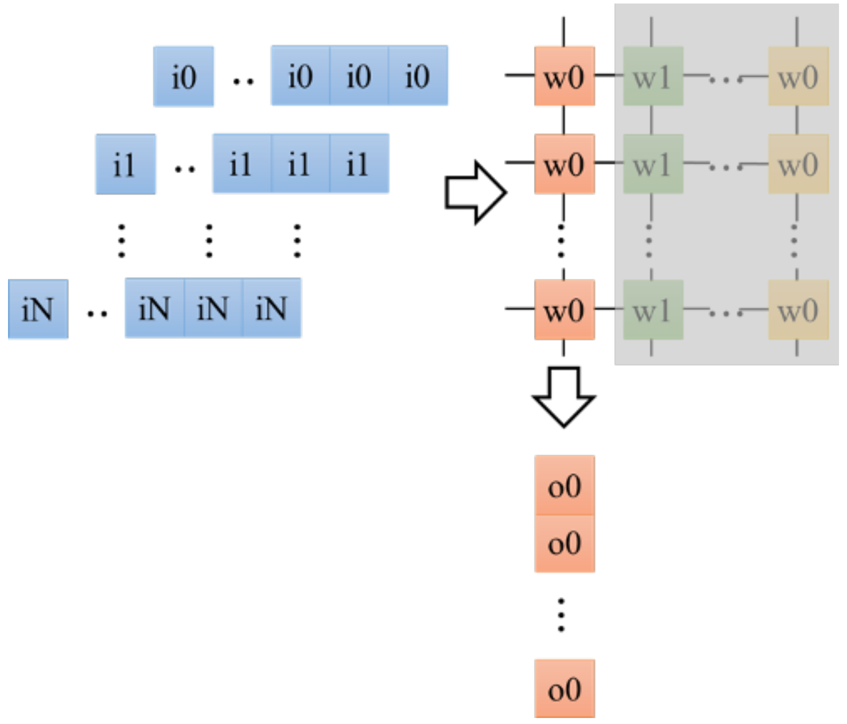
- Many interests in optimizing DNN
  - Hardware design
    - TPU[1]
  - Dataflow
    - Decisions about for-loop orders of DNN
    - Choose which type of data will be reused
  - **Neural network design**
    - Depthwise-Separable Convolution[2]
    - Neural Architecture Search


Depthwise-Seperable Convolution

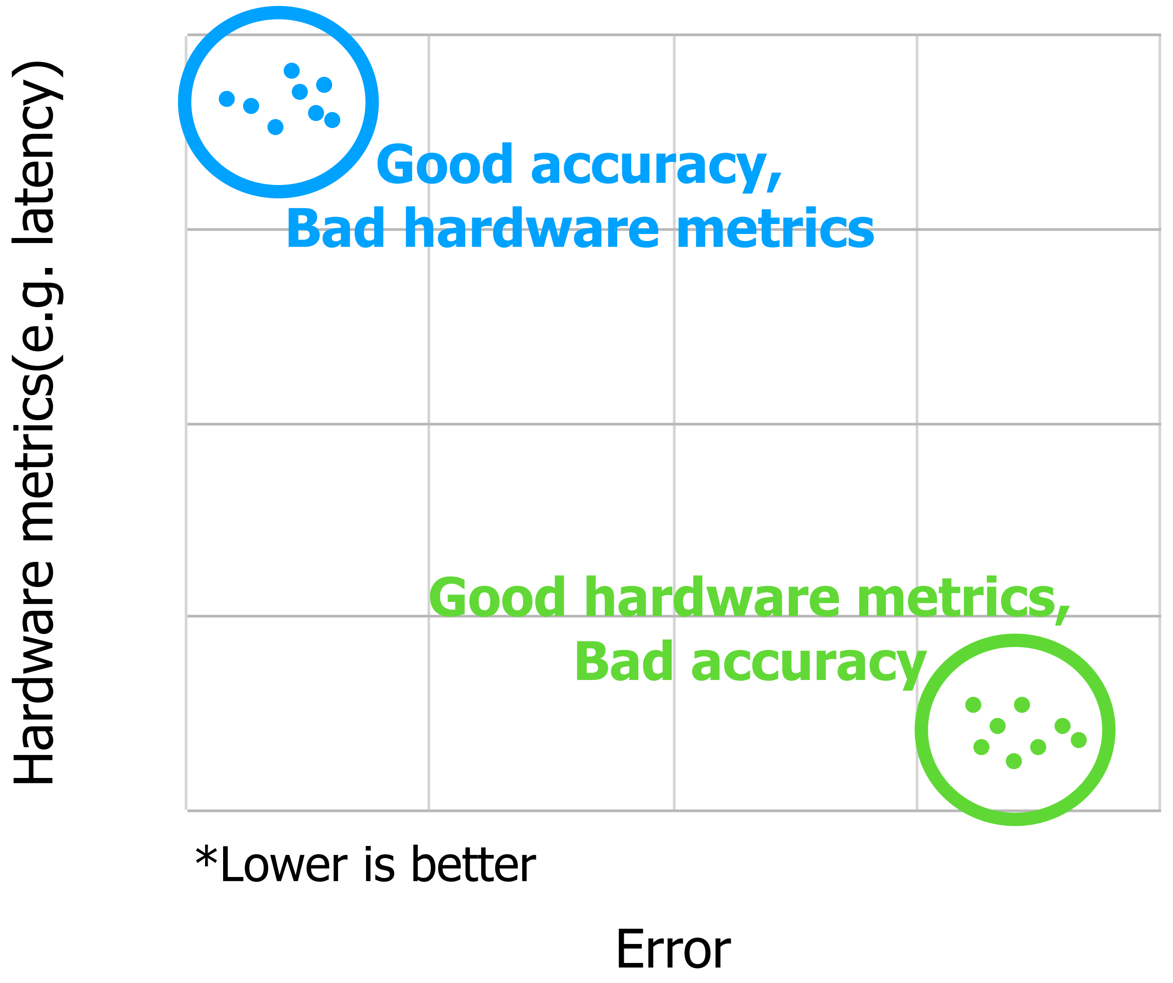
[1] Norman P Jouppi et al. "In-datacenter performance analysis of a tensor processing unit". In: ISCA. 2017.
[2] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: arXiv preprint arXiv:1704.04861 (2017).

Need for Co-search

- However…
  - TPU: Exploit channel-wise parallelism
  - DSConv: multiple Conv w/ 1 channel
  - With TPU, DSConv is slower than Conv due to **low hardware utilization**
- Therefore…
  - Accelerator and network have to be **co-optimized**



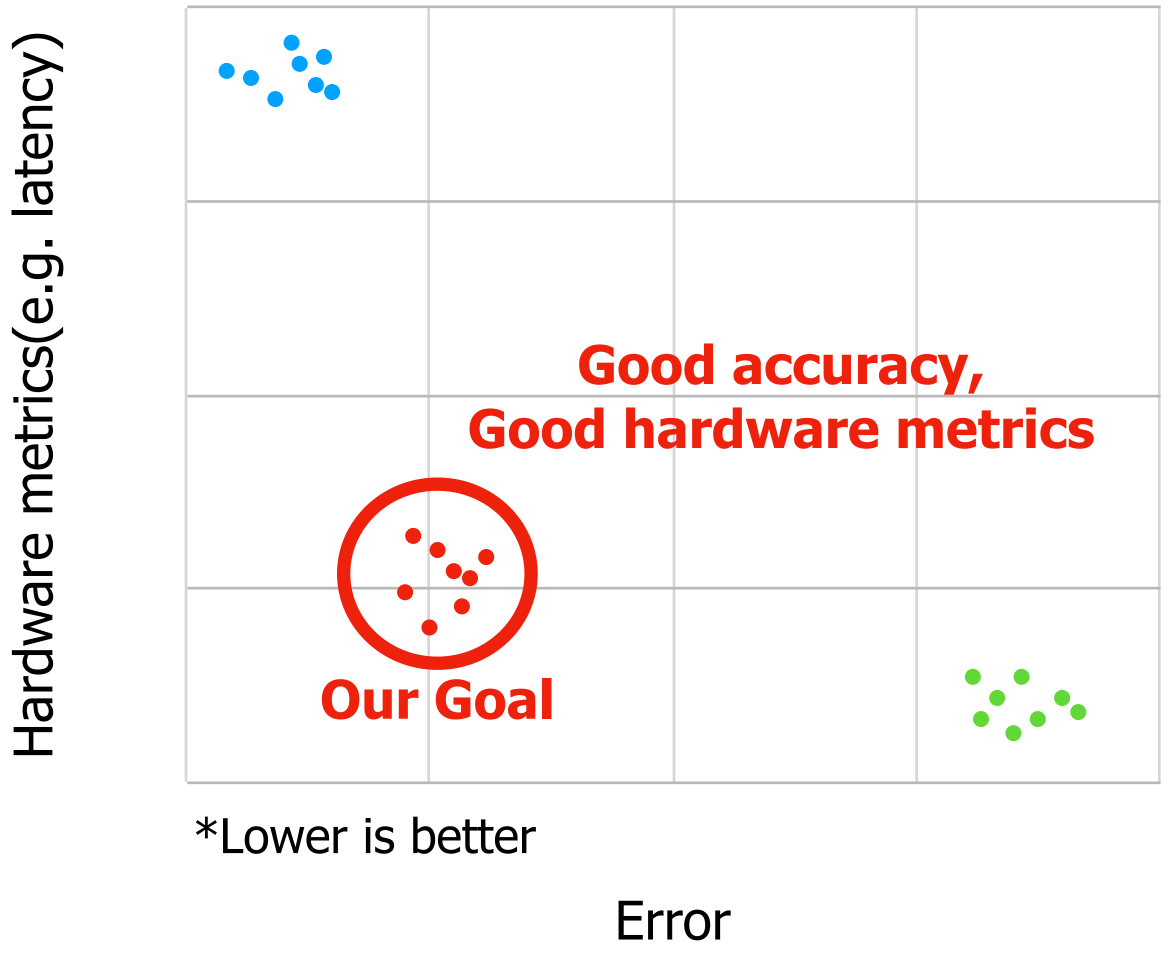
Need for Co-search



Separate search

- Fitting hardware to network
  - ➡ Degrades hardware metrics
- Fitting network to hardware
  - ➡ Degrades accuracy

Need for Co-search



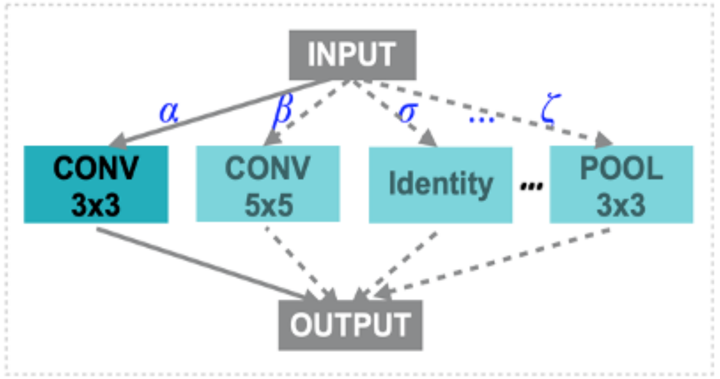
Co-search

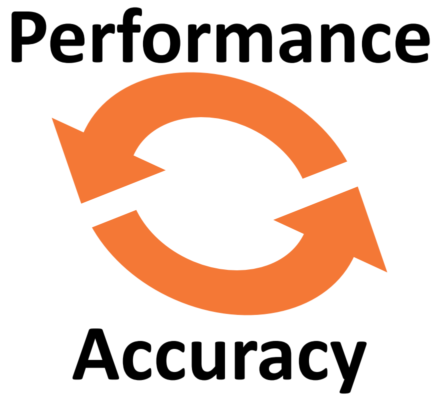
- Fitting hardware and network **simultaneously**
  - ✓Good hardware metrics
  - ✓Good accuracy

Our Target Problem

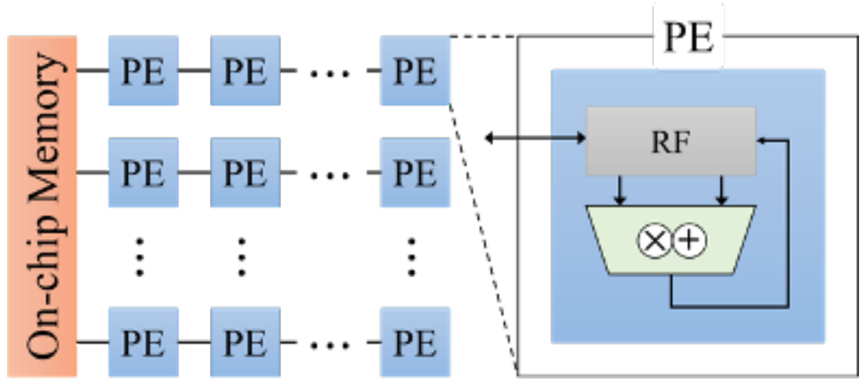
Integrating hardware search with NAS method

Differentiable NAS





Accelerator Architecture Search



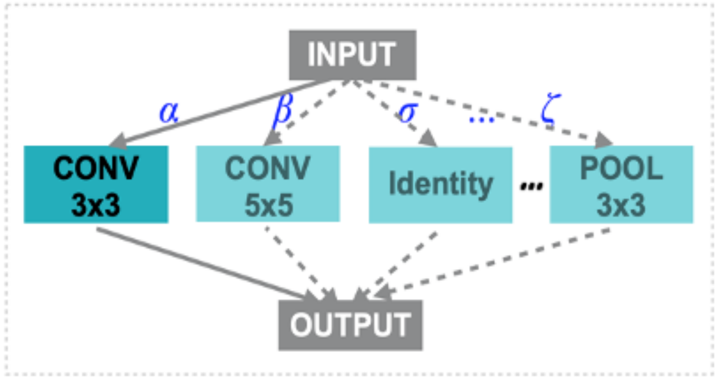
- Automatically find neural architecture
- Find and evaluate all candidates at once
- All loss function need to be **differentiable**

- Dataflow : loop order and data reuse scheme
- Number of PEs (PEx, PEy)
- Size of Register File(RF)

Our Target Problem

Integrating hardware search with NAS method

Differentiable NAS

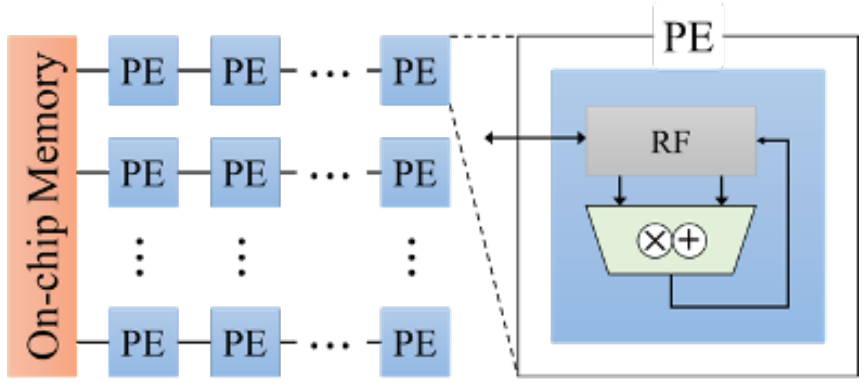


Not Differentiable

Performance

Accuracy

Accelerator Architecture Search



- Automatically find neural architecture
- Find and evaluate all candidates at once
- All loss function need to be **differentiable**

- Dataflow : loop order and data reuse scheme
- Number of PEs (PEx, PEy)
- Size of Register File(RF)

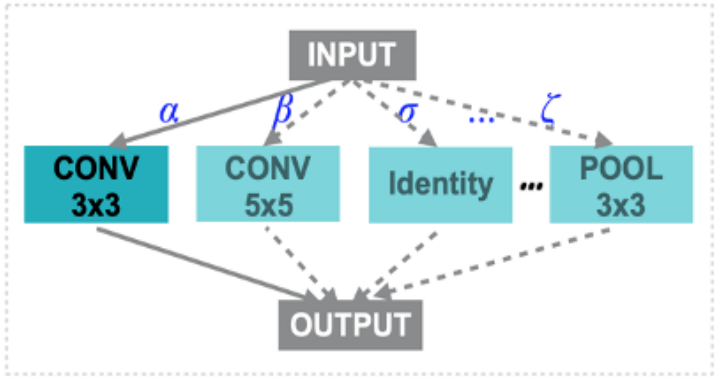
Our Target Problem

Integrating hardware search with NAS method

Differentiable NAS

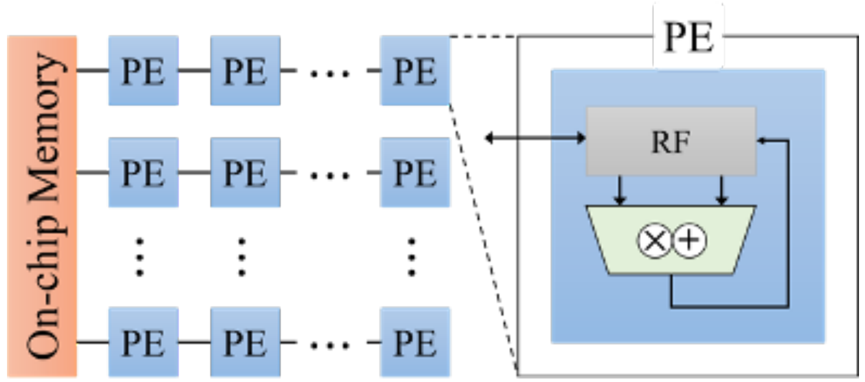
Differentiable NN approximation

Accelerator Architecture Search



Performance

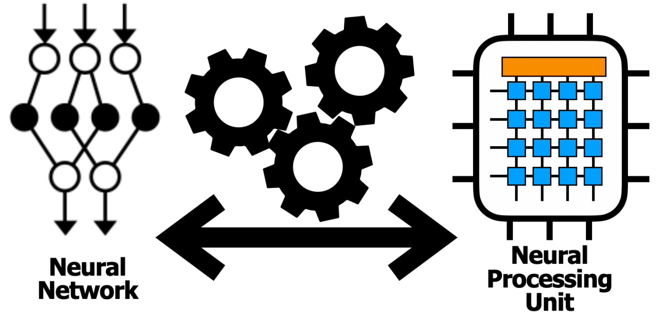
Accuracy



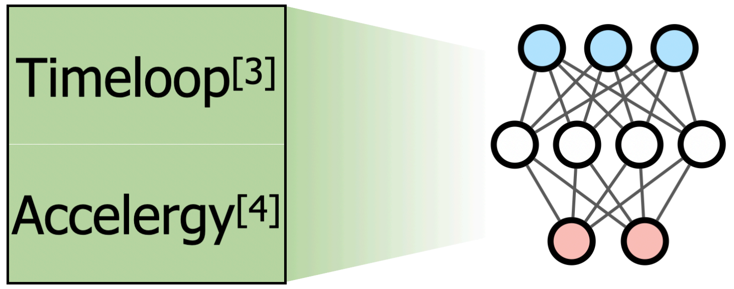
- Automatically find neural architecture
- Find and evaluate all candidates at once
- All loss function need to be **differentiable**

- Dataflow : loop order and data reuse scheme
- Number of PEs (PEx, PEy)
- Size of Register File(RF)

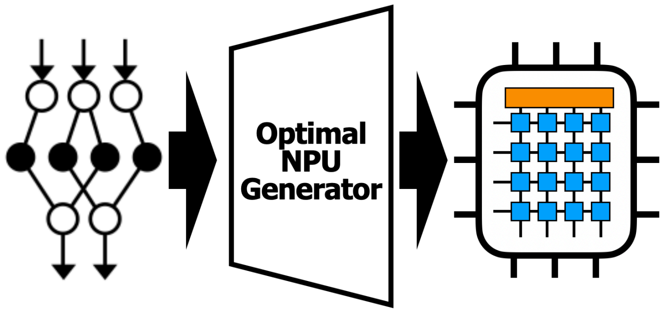
DANCE:
Differentiable Accelerator/Network Co-Exploration



Differentiable co-exploration



Differentiable Cost-estimation Network

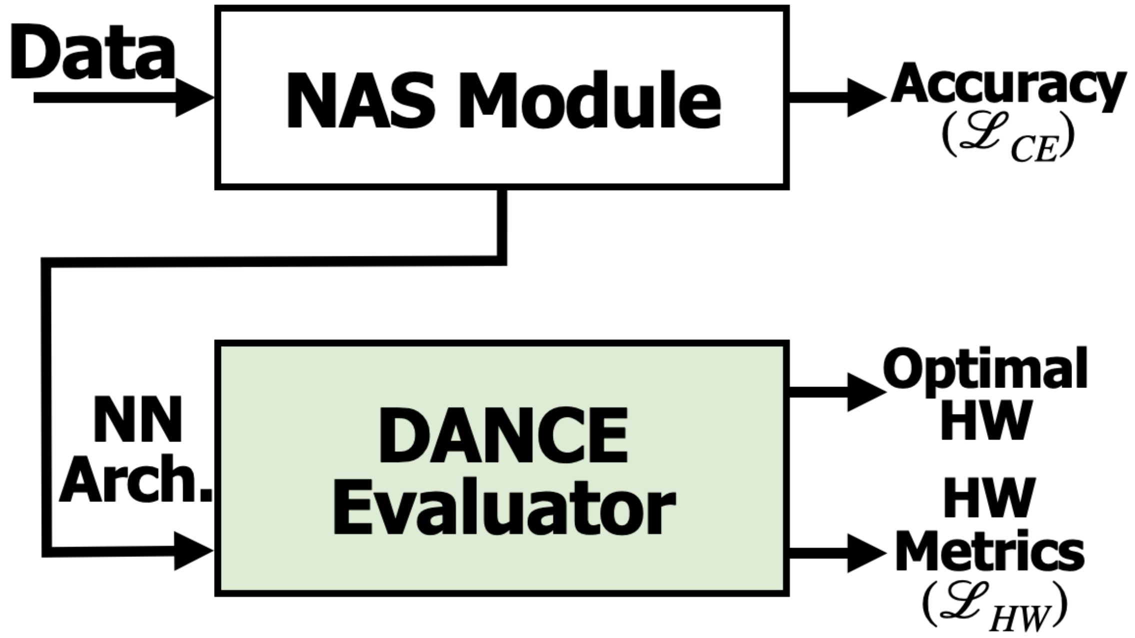


Differentiable
Optimal HW-generation
Network

[3] Angshuman Parashar et al. "Timeloop: A Systematic Approach to DNN Accelerator Evaluation". In: ISPASS. 2019.
[4] Yannan Nellie Wu et al. "Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs". In: ICCAD. 2019.

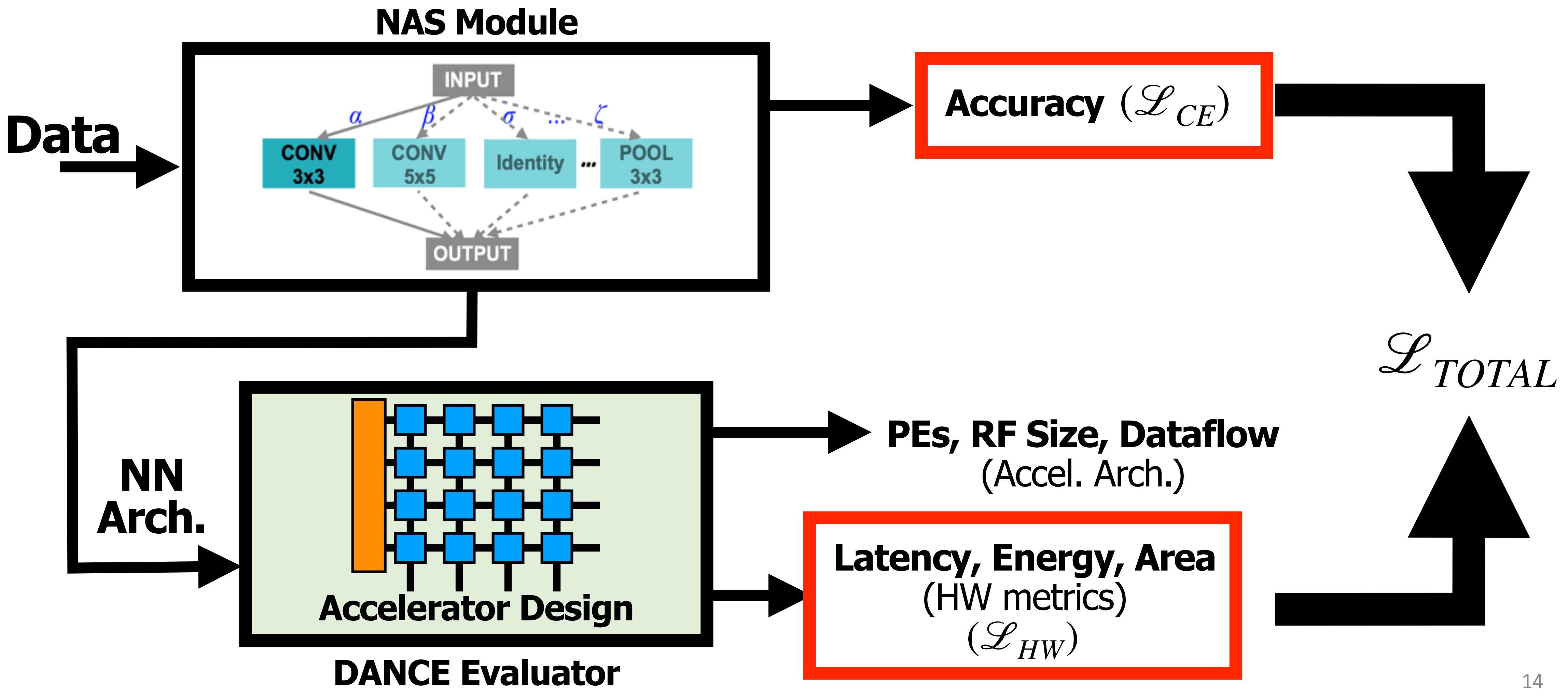
Overall Architecture

- NAS module determines neural network architecture
- DANCE evaluator determines accelerator architecture and predicts hardware metrics



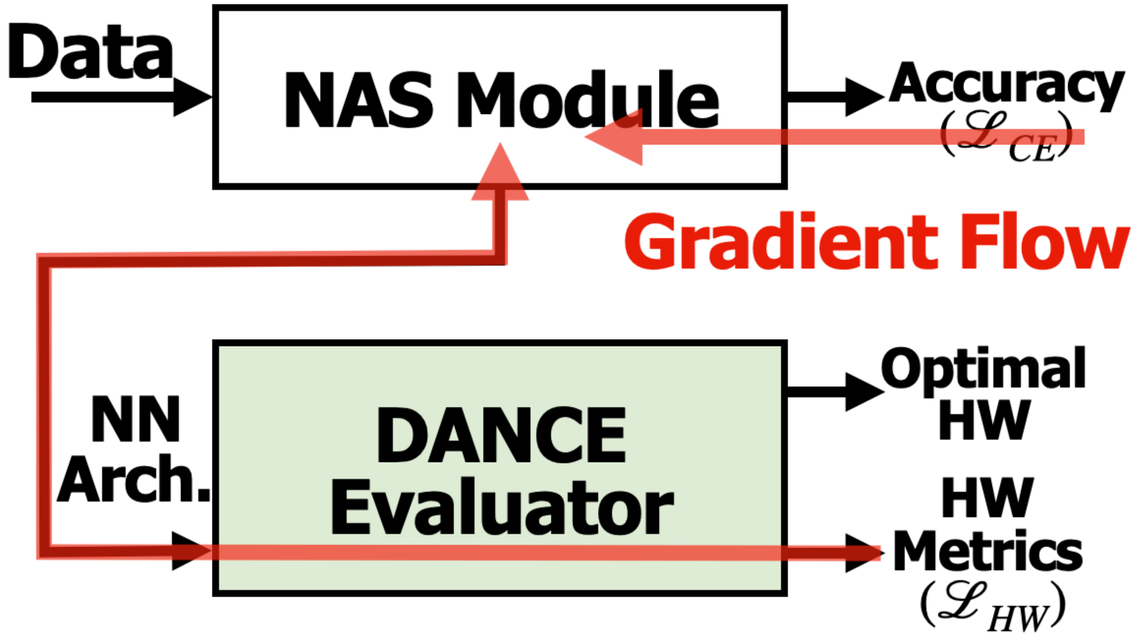
Data
NAS Module
Accuracy ($\mathscr{L}_{CE}$)
NN Arch.
DANCE Evaluator
Optimal HW
HW Metrics ($\mathscr{L}_{HW}$)

Overall Architecture



Overall Architecture

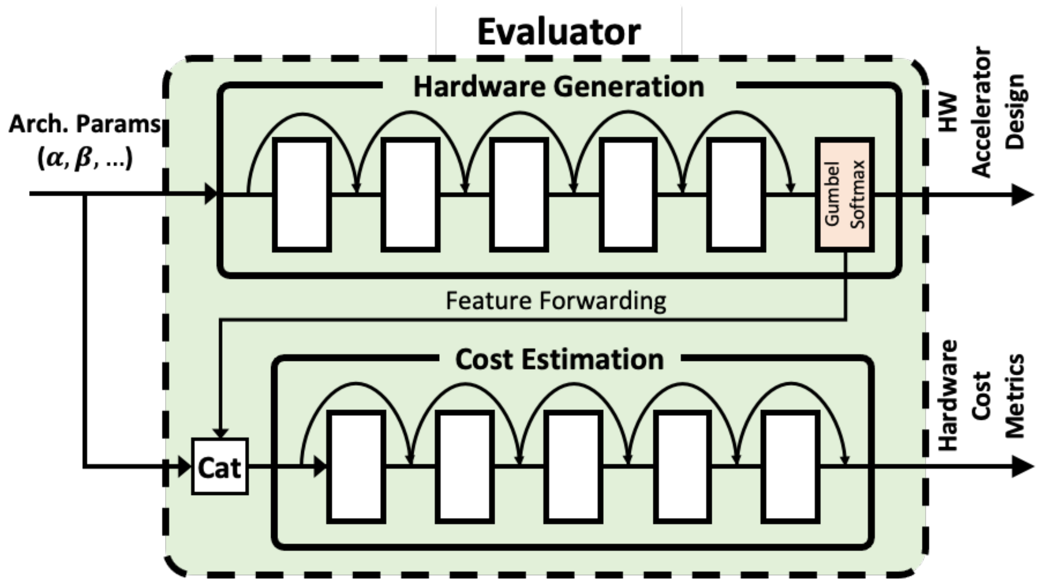
- NAS module determines neural network architecture
- DANCE evaluator determines accelerator architecture and predicts hardware metrics
- HW metrics of accelerator directly affect NN architecture by gradient flow



Evaluator Module

Consists of Two Neural Network

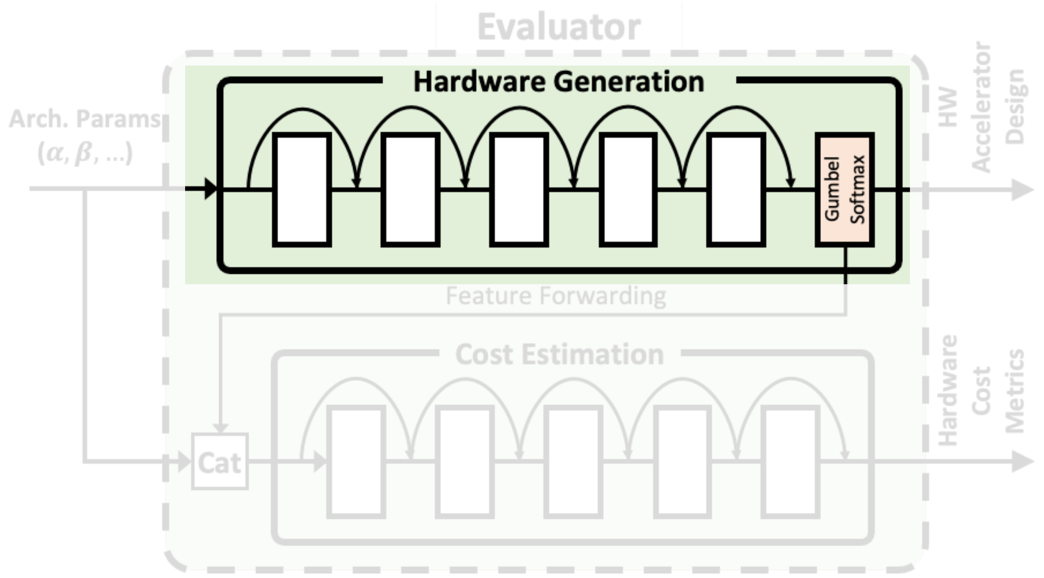
1. Hardware generation network
2. Cost estimation network



Evaluator Module

Hardware Generation Network

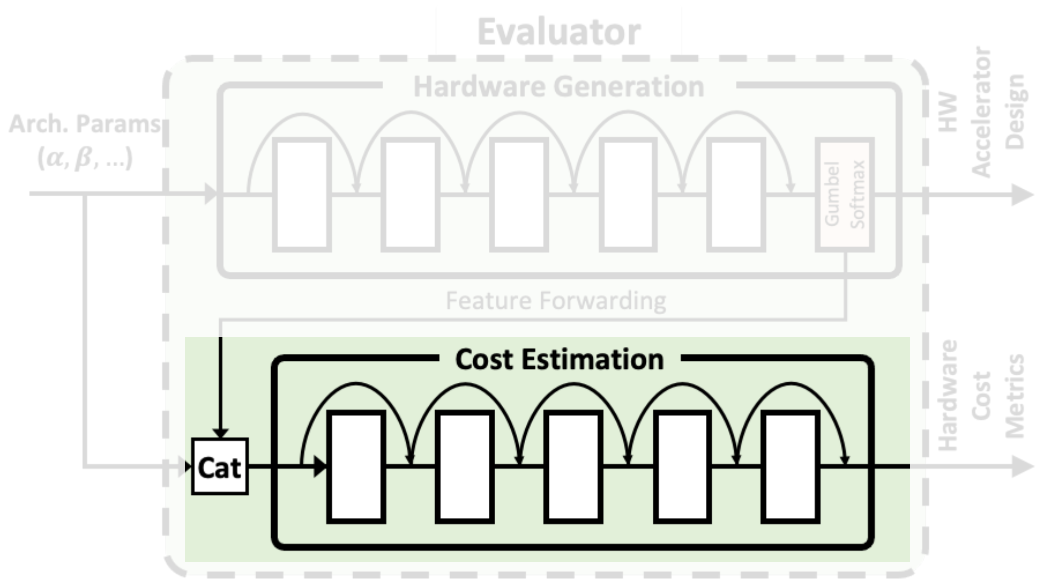
- Pre-trained to find **optimal HW architecture**
- Choose optimal **PEx, PEy, RF, DF** configuration



Evaluator Module

Cost Estimation Network

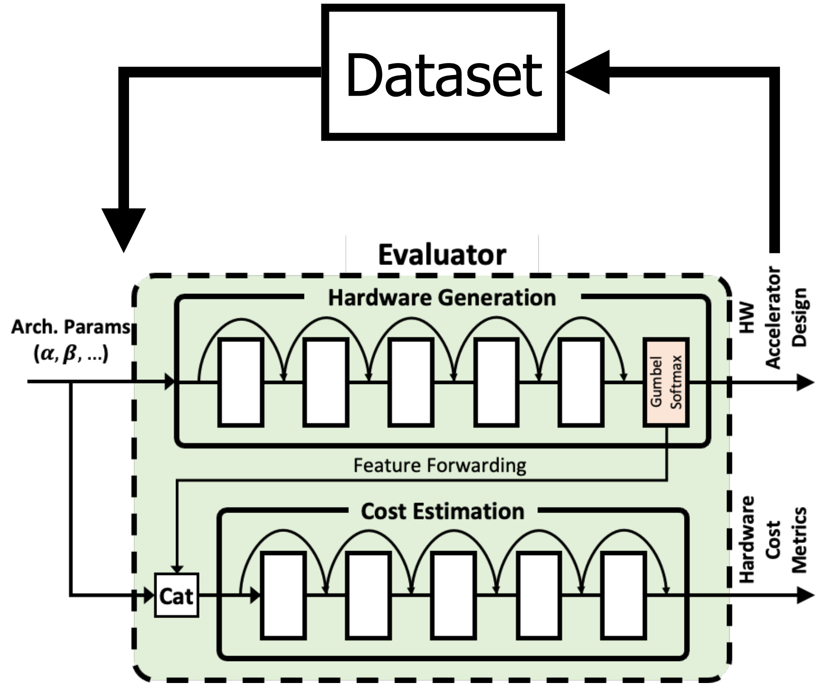
- Neural Net approximation of Hardware Cost Model
- Timeloop, Accelergy
- Any Cost Model can be used



Training Evaluator Module

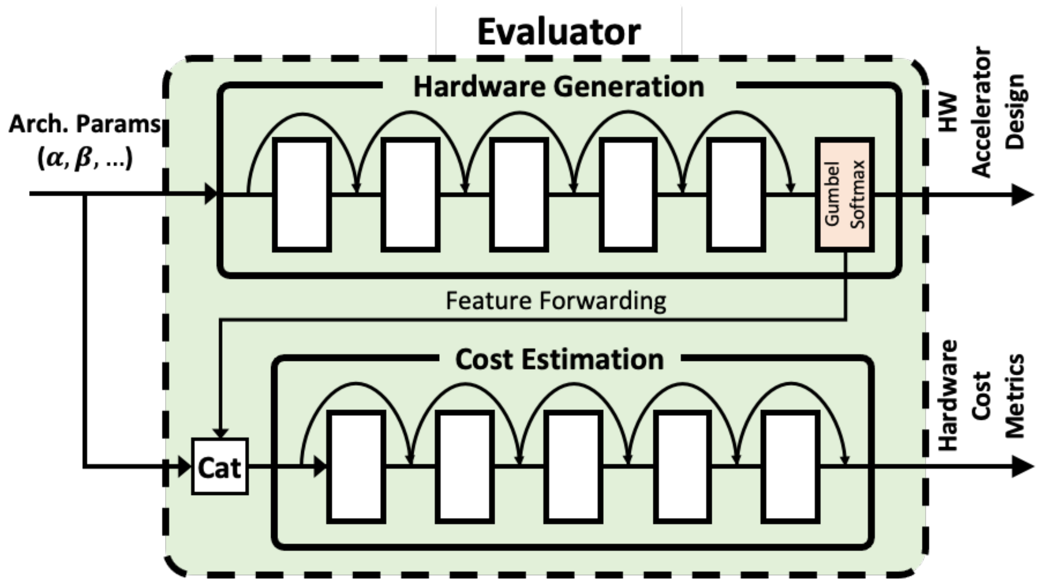
Dataset for training Evaluator Module

- Hardware generation network
    - input: Neural network architecture
    - output: Optimal accelerator architecture
    - 50K networks sampled
- Cost estimation network
    - input: NN-HW pair
    - output: HW metrics for given NN-HW pair
    - 1.8 million pairs sampled



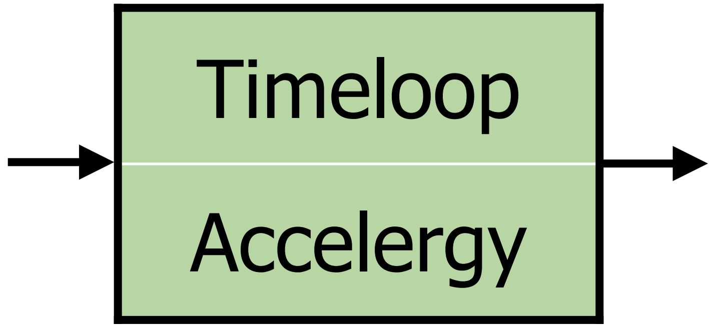
Evaluator Module

Network	**Accuracy**			
Hardware Generator	PE_x 98.9%	PE_y 98.3%	RF size 98.3%	Dataflow 98.8%
Network	**MSRE(Mean Square Relative Error)**			
Cost Estimator	Latency 99.6%	Energy 99.7%	Area 99.9%	
Network	**MSRE(Mean Square Relative Error)**			
Overal Evaluator	Latency 98.3%	Energy 98.3%	Area 99.2%	

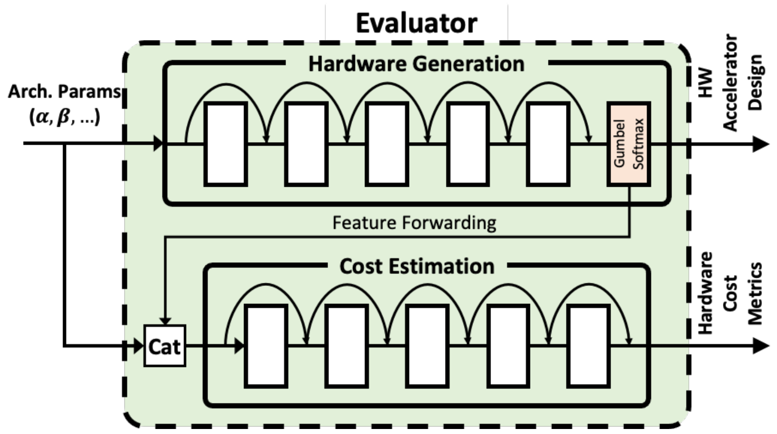


Less than 2 percent error estimation compared with Timeloop cost model

Comparison with Naive approach



112s of exhaustive search
on 48 threads
of two Xeon Silver-4214 CPUs



0.5ms for a single inference
on a single 2080Ti GPU

Experimental Settings

$$Cost_{HW\_EDAP} = Energy \cdot Latency \cdot Area$$
$$Cost_{HW\_linear} = \lambda_E Energy + \lambda_L Latency + \lambda_A Area$$

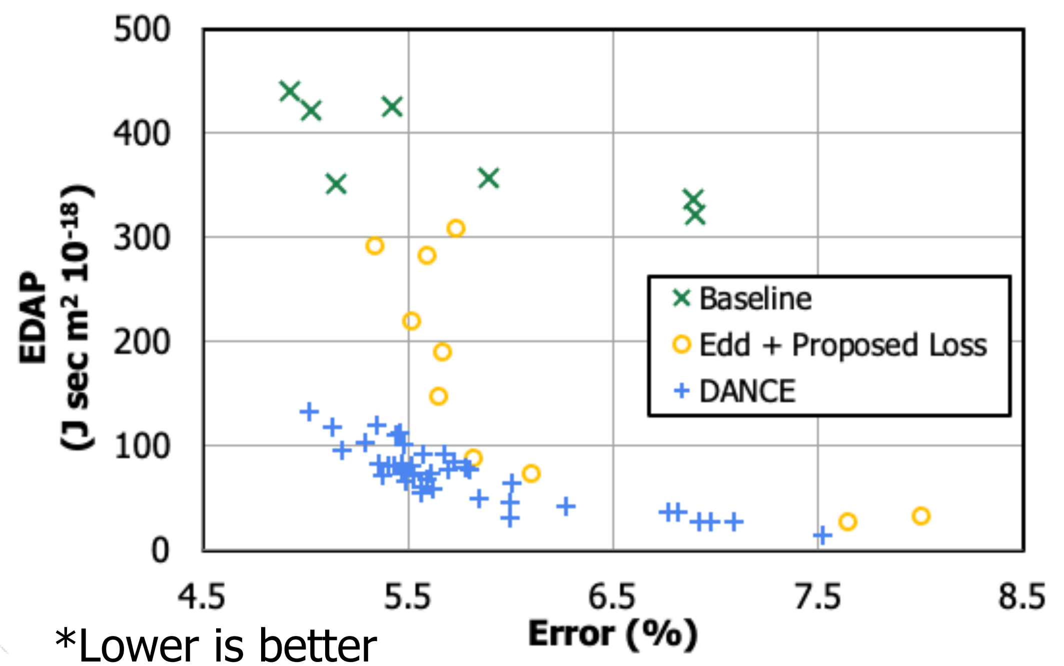
Select one equation to optimize HW metrics

- EDAP cost
  - Optimize three aspects at the same time
- {Latency, Energy}-oriented loss
  - Give more weights by optimization priority

Experimental Results

- CIFAR-10 Dataset
- Compared to baseline,
  - **10x** better EDAP
  - **3x** better latency

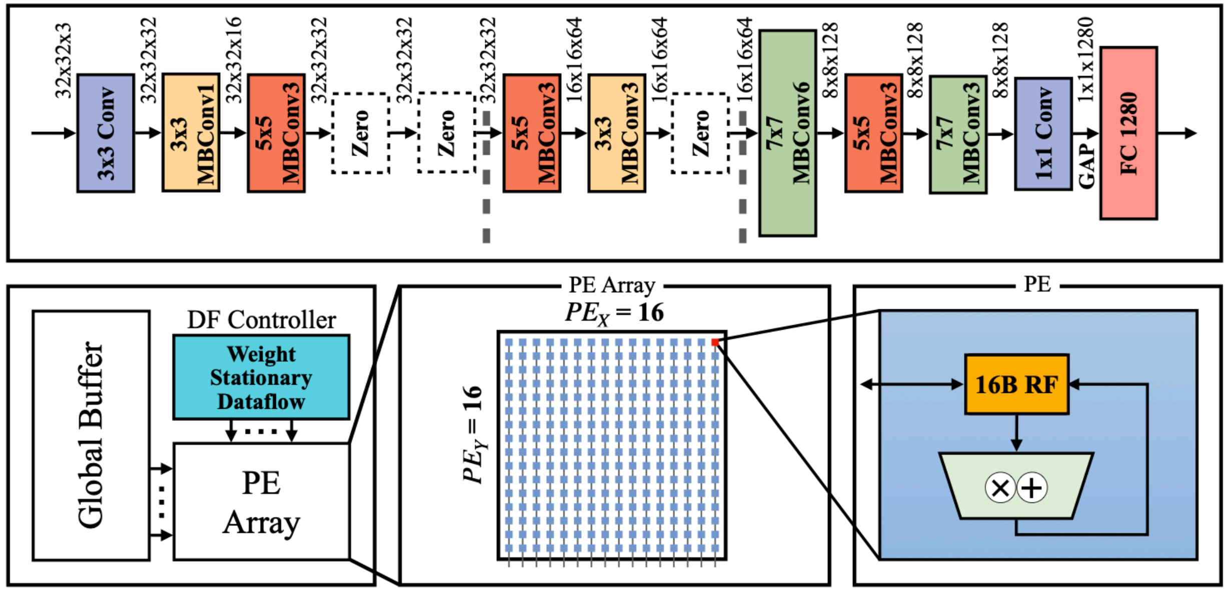
  with similar accuracy



*Lower is better

Experimental Results: Latency-oriented

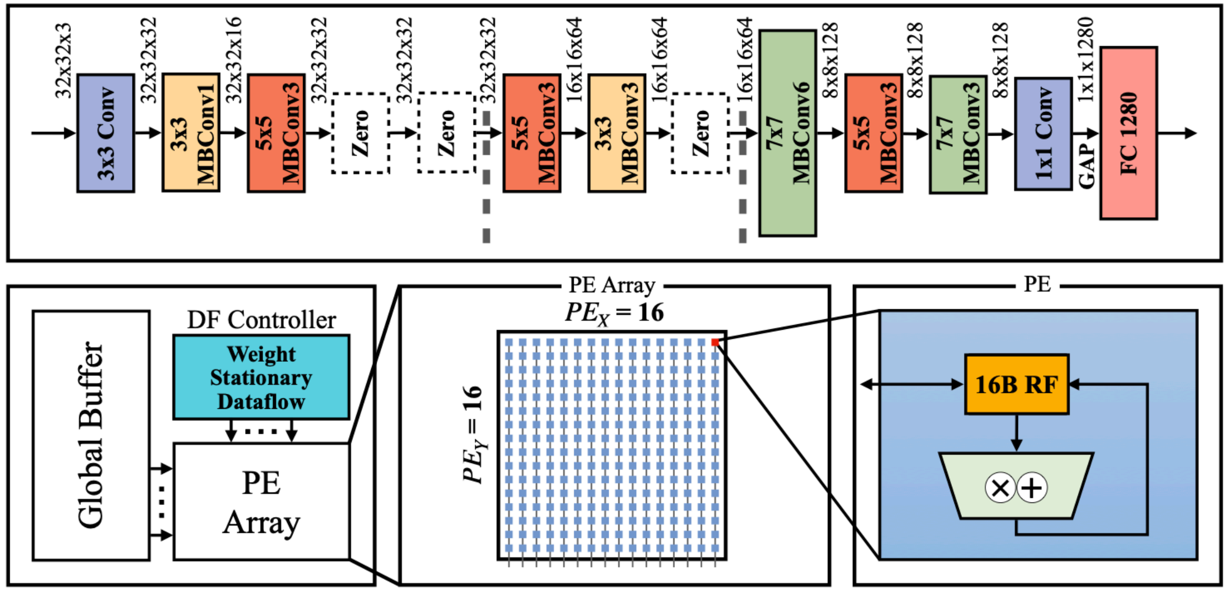
- Accelerators can utilize channel-level parallelism
- To achieve low latency,
  - Get benefit from channel-level parallelism



Search NN-HW with latency-oriented cost function

Experimental Results: Latency-oriented

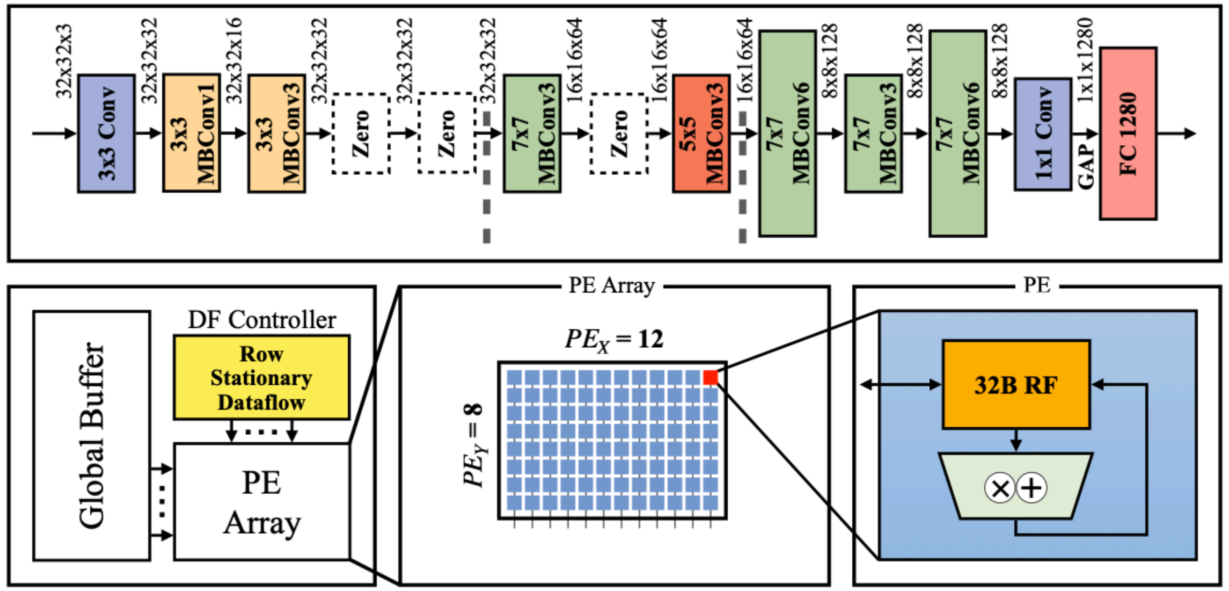
- Neural network
  - Small kernel sizes
- Hardware accelerator
  - Big PE array
- Weight Stationary dataflow
  - ➡ low latency



Search NN-HW with latency-oriented cost function

Experimental Results: Energy-oriented

- Energy consumption
  - Memory accesses
  - MAC ops
- To achieve low energy
  - ➡ Reduce memory accesses

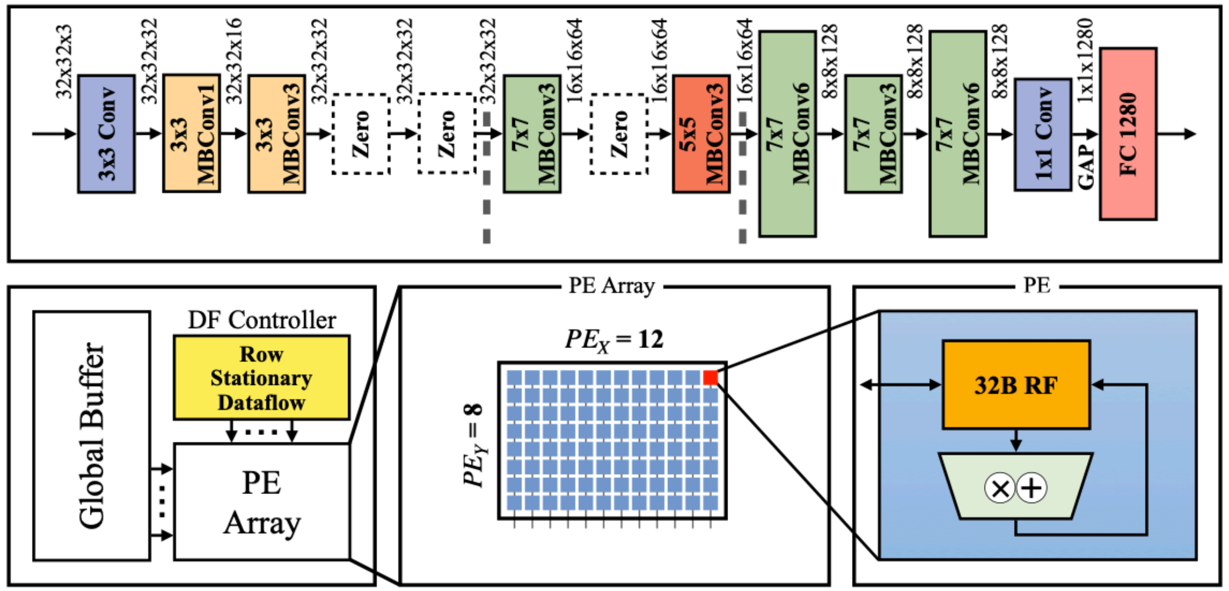


Search NN-HW with energy-oriented cost function

Experimental Results: Energy-oriented

- Neural network
  - Large kernel sizes
  - Narrow channels
- Hardware accelerator
  - Large RF for more local data reuse
- Row Stationary dataflow

  ➡good energy efficiency



Search NN-HW with energy-oriented cost function

Comparison with Prior Works

Algorithm	Backbone	Dataset	Acc.(%)	GPU-hours	Candidates	Method	Net-HW Relation
FPGA/DNN co-design: An efficient design methodology for IoT intelligence on the edge	Custom	DAC-SDC	68.6%	N/A	68	CD	O
On neural architecture search for resource-constrained hardware platforms	Custom	CIFAR-10	89.7%	N/A	N/A	RL	O
Co-Exploration of neural architectures and heterogeneous ASIC accelerator designs Targeting Multiple Tasks	ResNet-9	CIFAR-10	93.2%	3.5h	~160	RL	O
Best of both worlds: AutoML code sign of a CNN and its hardware accelerator	NASBench	CIFAR-100	74.2%	2300h	2300	RL	O
Hardware/software co-exploration of neural architectures	ProxylessNAS	CIFAR-10	85.2%	103.9h	308	RL	O
EDD: Efficient Differentiable DNN Architecture and Implementation Co-search for embedded AI solutions	ProxylessNAS	CIFAR-10	94.4%	3h	1	gradient	X
DANCE: Differentiable Accelerator/ Network Co-Exploration	ProxylessNAS	CIFAR-10	**95.0%**	**3h**	**1**	**gradient**	O

Comparison with Prior Works

Algorithm	Backbone	Dataset	Acc.(%)	GPU-hours	Candidates	Method	Net-HW Relation
FPGA/DNN co-design: An efficient design methodology for IoT intelligence on the edge	Custom	DAC-SDC	68.6%	N/A	68	CD	O
On neural architecture search for resource-constrained hardware platforms	Custom	CIFAR-10	89.7%	N/A	N/A	RL	O
Co-Exploration of neural architectures and heterogeneous ASIC accelerator designs Targeting Multiple Tasks	ResNet-9	CIFAR-10	93.2%	3.5h	~160	RL	O
Best of both worlds: AutoML code sign of a CNN and its hardware accelerator	NASBench	CIFAR-100	74.2%	2300h	2300	RL	O
Hardware/software co-exploration of neural architectures	ProxylessNAS	CIFAR-10	85.2%	103.9h	308	RL	O
EDD: Efficient Differentiable DNN Architecture and Implementation Co-search for embedded AI solutions	ProxylessNAS	CIFAR-10	94.4%	3h	1	gradient	X
DANCE: Differentiable Accelerator/ Network Co-Exploration	ProxylessNAS	CIFAR-10	**95.0%**	**3h**	**1**	**gradient**	O

Comparison with Prior Works

Algorithm	Backbone	Dataset	Acc.(%)	GPU-hours	Candidat es	Method	Net-HW Relation
FPGA/DNN co-design: An efficient design methodology for IoT intelligence on the edge	Custom	DAC-SDC	68.6%	N/A	68	CD	O
On neural architecture search for resource-constrained hardware platforms	Custom	CIFAR-10	89.7%	N/A	N/A	RL	O
Co-Exploration of neural architectures and heterogeneous ASIC accelerator designs Targeting Multiple Tasks	ResNet-9	CIFAR-10	93.2%	3.5h	~160	RL	O
Best of both worlds: AutoML code sign of a CNN and its hardware accelerator	NASBench	CIFAR-100	74.2%	2300h	2300	RL	O
Hardware/software co-exploration of neural architectures	ProxylessNAS	CIFAR-10	85.2%	103.9h	308	RL	O
EDD: Efficient Differentiable DNN Architecture and Implementation Co-search for embedded AI solutions	ProxylessNAS	CIFAR-10	94.4%	3h	1	gradient	X
DANCE: Differentiable Accelerator/ Network Co-Exploration	ProxylessNAS	CIFAR-10	**95.0%**	**3h**	**1**	**gradient**	O

Conclusion

- Novel differentiable method for co-optimizing DNN & accelerator
  - Build hardware evaluator with neural network
  - Get optimal hardware design with evaluator
  - Propagate HW costs to NAS module via gradient
- Reduce co-exploration costs
- Applicable to the various tasks
  - e.g. video processing, NLP