

A Quick User Manual of
Peking University CoMN Platform
Version: 1.0

CoMN: Algorithm-Hardware Co-design Platform for
Non-volatile Memory Based Convolution
Neural Network Accelerators

Developers:

Lixia Han¹, Renjie Pan², Peng Huang^{1,*}, Xiaoyan Liu¹ and Jinfeng Kang¹.

Affiliation:

1. School of Integrated Circuits, Peking University, Beijing 100871, China.
2. School of Software and Microelectronics, Peking University, Beijing 102600, China.

April 10, 2023

Terms of Use

Peking University and the authors provide CoMN platform to you subject to the Terms of Use, which may be updated by us from time to time without notice to you.

By using the Peking University CoMN platform (“CoMN”), you acknowledge that you have read the most up-to-date Terms of Use and agree to abide by the Terms of Use (“Terms”).

These Terms include, but are not limited to, the following:

License Agreement

Peking University grants you a non-transferable license to use CoMN on a single computer for a single user (You). This license may not be sub-leased, sub-licensed, sold or otherwise transferred to another individual, company, or third party. Peking University reserves the right to revoke this license at any time, at which point you must stop using the Model and delete all Model files.

Acceptable Usage

CoMN shall be used solely for non-commercial academic and industrial research by the individual to whom this Model, and its license, is granted. CoMN may not be used, in part or in whole, by another individual, institution, or a third party other than the original individual without prior written approval from Peking University. CoMN may not be copied, redistributed, or otherwise transferred, in part or in whole, to a third party without prior written approval from Peking University.

You agree not to disclose the ideas and inventions inherent in CoMN to other individuals, institutions, or third parties. You further agree not to decompile or otherwise reverse-engineer CoMN, in part or in whole, not to decompose CoMN and its files, and not to misrepresent CoMN through modifications and add-ons.

Disclaimer and Limitation of Liability

CoMN is provided to you “As Is,” without warranty of any kind, either expressed or implied. By using CoMN, you agree that you and your representing institution or company will not hold Peking University, the CoMN inventors, the CoMN authors, as well as all other contributing members to the CoMN and its official distribution, liable for damage of any kind resulting from the download or use of CoMN and documents.

Legal Notice

CoMN, including the files, documents, and inherent ideas, are protected by Chinese Copyright Law and Chinese Patent Law. Peking University and authors reserve all rights. Unauthorized reproduction of the files and/or the documents included in the CoMN package is unlawful.

Contents

1. Introduction	4
2. Installation (Linux)	5
2.1. System requirements	5
2.2. Installation and Usage	5
3. nvCIM chip architecture	6
4. Algorithm-level evaluation and optimization	7
4.1. Function description	7
4.2. Parameters description	7
5. Algorithm-to-hardware mapping and evaluation	9
5.1. Function description	9
5.2. Parameters description	9
6. Hardware-level design exploration	11
6.1. Function description	11
6.2. Parameters description	11
Reference	12

1. Introduction

CoMN is an algorithm–hardware co-design platform with the graphical user interface to evaluate and explore accuracy and performance for convolution neural network accelerators based on nonvolatile computing in memory (nvCIM) architecture. In evaluation mode, CoMN platform develops mapper, accuracy evaluator and performance evaluator to jointly estimate accuracy, energy, latency and area in account of coupling effect of algorithm and hardware. In exploration mode, CoMN platform builds algorithm optimization engine and hardware exploration engine to preliminarily guide algorithm and hardware design in pursuit of higher energy efficiency and throughput. CoMN supports various convolutional neural networks (VGG, ResNet, MobileNet, etc.) and various nonvolatile memories NVM (RRAM, PCM, MRAM). Meanwhile, hardware aware training, mixed precision quantization, architecture configurations exploration and circuit performance guidance all are supported by CoMN.

Including but not limited to the following researchers or engineers can use CoMN to preliminarily verify and further optimize their designments:

- **Device or circuit designers:** user-defined device and circuit as basic components of nvCIM accelerators, CoMN can evaluate CNN inference accuracy as well as architecture-level energy consumption, latency, and area overheads.
- **Chip designers:** for the objective energy consumption, latency and area, CoMN can provide preliminary guidance of architecture configuration and circuit performance to narrow down the scope of subsequent detailed optimization.
- **System designers:** For target CNN models and nvCIM hardware, CoMN can support weights retraining and mapping optimization to achieve higher inference accuracy with smaller hardware overheads (energy, latency and area).

2. Installation (Linux)

2.1. System requirements

The platform is expected to be installed under Linux, and CoMN is workable in the ubuntu 16.04, gcc v5.4.0, glibc v2.23 and python 3.8 environments. The following libraries are still required:

```
numpy==1.24.2  
Pillow==6.2.1  
pyinstaller==5.8.0  
pyinstaller-hooks-contrib==2023.0  
PyQt5==5.12  
PyQt5-sip==4.19.19  
scikit-learn==1.0.1  
scipy==1.10.1  
torch==1.4.0  
torchvision==0.2.2.post3
```

2.2. Installation and Usage

- 1) Get the tool from Github:

```
>> git clone #####
```

- 2) Open terminal, then print:

```
>> export LD_LIBRARY_PATH="./CoMN/torch/lib":${LD_LIBRARY_PATH}  
>> ./gui
```

3. nvCIM chip architecture

The typical architecture of CNN accelerators based on nvCIM is shown in Fig. 1. In nvCIM chips, nvCIM tiles, instruction units, control units, and L2 buffer are connected with Mesh-based network on chip. Macros, L1 buffer, special function units (SFU), adders, and shift registers are typically contained in each CIM tile. Several macros are connected in hierarchical routers and shared by four surrounding macros to enable the adding, shifting, and storage of partial results, which enables to efficiently distribute input data and gather output data. NVM devices, DACs, ADCs, multiplexers, registers, and adders are all components of a CIM macro.

VMM computation is in-situ performed with high parallelism according to Ohm's law and Kirchhoff's law in CIM macro. The matrix is stored in the non-volatile memory crossbar in advance. The vector is converted into voltage by DACs and applied to NVM crossbar. Output current is converted to digital signal by ADCs. However, CIM macros suffer from various non-idealities. The limited ADC/DAC resolution and limited NVM states limit the VMM computation precision. Interconnect wire resistance cause that the actual applied voltage of NVM devices deviates from the target voltage. Conductance stack-at-fault and conductance variation make actual conductance deviate from target conductance. In addition, nonlinear I-V response of NVM device also introduce VMM computation errors.

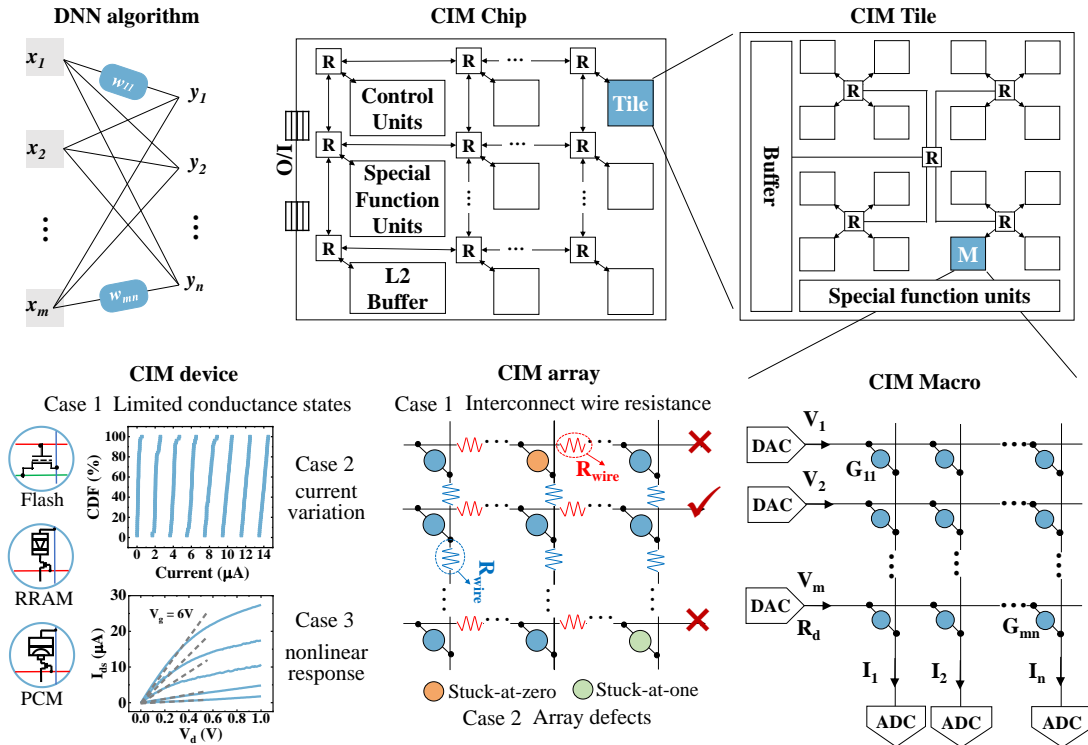


Fig. 1. The diagram of NVM-based CIM hierarchy architecture and NVM device, array nonidealities

4. Algorithm-level evaluation and optimization

4.1. Function description

The non-idealities from nvCIM chip would degrade accuracy. CoMN platform provides two operating modes, namely evaluation mode and optimization mode. In the evaluation mode, CoMN platform can evaluate the actual inference accuracy when mapping CNN model to nvCIM chip with above non-ideal characteristics. In the optimization mode, CoMN platform can recover the inference accuracy as much as possible through hardware aware training and weight mapping correction.

4.2. Parameters description

The Algorithm-level evaluation and optimization interface is as follows, including the definition of convolutional neural network and nvCIM chip nonidealities. In addition, users can set epochs and learning rate of CNN training, and whether to adopt hardware aware training and weight mapping correction to retrain and correct CNN weights.

- **Convolution neural network**

- 1) CoMN platform provides ten default models, including VGG11, VGG13, VGG16, VGG19, ResNet18, ResNet34, ResNet50, ResNet101, ResNet152, MobileNet v1.
- 2) CoMN platform also allows users to define their own models, which are composed of the following parts: the convolution layer, the pooling layer, the activation layer and the full connection layer.

Defined CNN model	description
Conv	Convolution layer
ConvRelu	Relu layer after convolution layer All sub-parameters are immutable
ConvSigmod	Sigmoid layer after convolution layer All sub-parameters are immutable
Maxpool	Maxpool layer
Avgpool	Avgpool layer
Linear	Fully-connected layer, sub-parameters are immutable except inchannels and outchannels
LinearRelu	Relu layer after fully-connected layer All sub-parameters are immutable

- **nvCIM chip nonidealities**

nvCIM chip nonidealities include Quantization, Variation, Nonlinearity and IR drop, detailed parameters are shown as follows.

Nonidealities	parameters	description
Quantization	Weight precision	Weight precision, optional 1~8bit
	ADC resolution	Partial sums precision, optional 1~8bit
	Input precision	Input coding precision, optional 1~ 8bit
Variation	Conductance STD %	Ratio of conductance variance to conductance window
	NVM states	Number of states per device (bit)
Nonlinearity	a1, a2, a3	Polynomial fitting the nonlinear response of the device ($Y=a_1 + a_2*x + a_3*x^2$)
IR drop	Row resistance	Row-oriented interconnect resistance
	Column resistance	Column-oriented interconnect resistance
	MaxConductance	The maximum value of programmed device
	MinConductance	The minimum value of programmed device
	Subarray size	NVM subarray size

The train result of CNN optimization is shown as follows:

```

#####Training neural network models to adapt hardware nonidealities#####
Epoch: [0][0/391] Time 0.573 (0.573) Data 0.162 (0.162) Loss 2.3142 (2.3142) Prec@1 10.938 (10.938)
Epoch: [0][50/391] Time 0.242 (0.250) Data 0.000 (0.004) Loss 1.8771 (2.0445) Prec@1 29.688 (21.630)
Epoch: [0][100/391] Time 0.236 (0.246) Data 0.001 (0.002) Loss 1.5763 (1.8865) Prec@1 42.969 (27.498)
Epoch: [0][150/391] Time 0.235 (0.243) Data 0.000 (0.002) Loss 1.7512 (1.7970) Prec@1 30.469 (31.328)
Epoch: [0][200/391] Time 0.250 (0.243) Data 0.000 (0.001) Loss 1.4457 (1.7201) Prec@1 50.781 (34.725)
Epoch: [0][250/391] Time 0.243 (0.243) Data 0.000 (0.001) Loss 1.2755 (1.6642) Prec@1 53.906 (37.347)
Epoch: [0][300/391] Time 0.242 (0.243) Data 0.000 (0.001) Loss 1.4432 (1.6118) Prec@1 41.406 (39.696)
Epoch: [0][350/391] Time 0.242 (0.243) Data 0.000 (0.001) Loss 1.5303 (1.5677) Prec@1 46.094 (41.622)
Test: [0/79] Time 0.170 (0.170) Loss 1.5407 (1.5407) Prec@1 43.750 (43.750)
Test: [50/79] Time 0.038 (0.042) Loss 1.7900 (1.5405) Prec@1 46.875 (46.752)
* Prec@1 46.490
Epoch: [1][0/391] Time 0.404 (0.404) Data 0.145 (0.145) Loss 1.3353 (1.3353) Prec@1 50.781 (50.781)
Epoch: [1][50/391] Time 0.243 (0.246) Data 0.000 (0.003) Loss 1.3078 (1.1960) Prec@1 50.781 (57.659)
Epoch: [1][100/391] Time 0.241 (0.244) Data 0.000 (0.002) Loss 0.9444 (1.1839) Prec@1 67.188 (58.253)
Epoch: [1][150/391] Time 0.243 (0.241) Data 0.000 (0.001) Loss 1.3966 (1.1674) Prec@1 54.688 (58.940)
  
```


5. Algorithm-to-hardware mapping and evaluation

5.1. Function description

The algorithm-to-hardware mapping scheme significantly determines whether workload can benefit from nvCIM accelerators. An elaborate mapping scheme is desired to optimize computational parallelism, data reuse, etc., CoMN platform can automatically map various CNN model to nvCIM chips according to CNN structures and hardware resource constraints. CoMN platform develops mapper to optimize pipeline design, weight transformation, partition, and placement according to user-defined power-prior, delay-prior, or area-prior principles. The performance of macros, buffers and NoC routers are evaluated by calling relevant simulation tools (Macro: NeuroSIM[1], Buffer: CACTI[2], Router: Orion[3]).

5.2. Parameters description

CoMN platform supports the modification of the following chip characteristic parameters.

	Parameters	
Technology node	Optional 180nm, 130nm, 90nm, 65nm, 40nm, 28nm, 14nm	
Macro performance (Optional)	Macro power (J)	
	Macro delay (s)	
	Macro area (mm ²)	
Device/Circuit performance (Optional)	NVM states	Optional 1~8
	NVM MaxConductance (S)	
	NVM MinConductance (S)	
	read Voltage (V)	
	read Pulse (s)	
	ADC power (mW)	
	ADC sample rate (Hz)	
	ADC active area (mm ²)	
nvCIM architecture key configurations	Macro size	Optional [64,64], [128,128], [256,256], [512,512], [1024,1024]
	ADC nums per Macro	Optional 1~Macro columns
	Macro nums per Tile	Optional [2,2], [2,4], [4,4], [4,8], [8,8], [8,16], [16,16]
	Buffer size per Tile (KB)	Optional 4 ~ 128
	Buffer bandwidth (bit)	Optional 16 ~ 128
	Htree router flit bandwidth (bit)	Optional 16 ~ 128
	Mesh router flit bandwidth (bit)	Optional 16 ~ 128

The output of mapper includes the mapping relationship of CNN weight to Tiles and the estimation of power consumption, delay and area. The example result of algorithm-to-hardware mapping is shown as follows:

```
#####
#####Mapping CNN models to nvCIM chip and evaluate energy, latency, area#####
#####
```

Layer: the layer of the CNN model

Type: the computation type of this layer

Tiles: The corresponding virtual tiles after weight transformation and partition

Energy: energy consumption of this layer

Latency: delay of this layer

Area: area overheads mapped by this layer

```
#####
#####Mapping relationship between virtual tiles and physical tiles#####
#####
```

tile: virtual tiles after weight transformation and partition

locations: locations of physical tiles

6. Hardware-level design exploration

6.1. Function description

Hardware architecture configuration and circuit optimization are heavily dependent on the experience of engineers, and a rapid preliminary hardware design space exploration tool can effectively narrow the feasible optimization space. CoMN platform can automatically search optimal range of key architecture configurations and guide neuron circuit optimization from the point of architecture view.

6.2. Parameters description

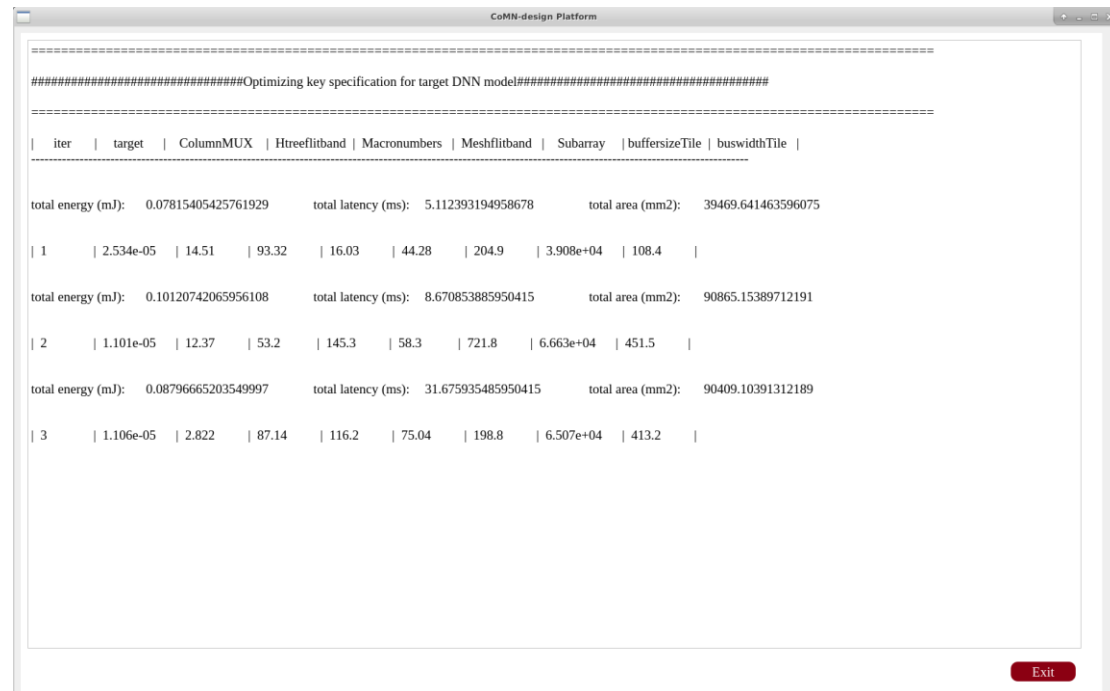
The key configurations of nvCIM architecture includes tile size, macro size, buffer size per tile, ADC numbers per macro, network on chip (NoC) bandwidth, and so on. We adopt Bayesian optimization method[4] to explore key configurations of architecture in purist of lower energy, latency and area. CoMN platform provides default parameters.

	Default Parameters	Down bound	Up bound
Architecture configurations	Macro size	64	1024
	ADC nums per Macro	1	Macro columns
	Macro nums per Tile	16	256
	Buffer size per Tile (KB)	4	1024
	Buffer bandwidth (bit)	4	128
Target you expect to optimize	Optional energy, latency and area (one of them)		

ADC power consumption and active area dominate the energy consumption and area overhead of nvCIM based CNN accelerators. CoMN platform can explore the feasible performance range of ADC circuits and NVM devices to **meet the user-defined architecture-level performance PPA_{target}** . CoMN platform provides default parameters.

	Default Parameters	Initial	Down bound	Up bound
ADC circuit	ADC power (W)	0.001	1e-6	1e-3
	ADC sampling rate (Hz)	1e8	1e6	1e10
	ADC active area (mm ²)	0.001	1e-5	0.1
User-defined performance	Objective energy (J)	e.g. 1e-2		
	Objective latency (s)	e.g. 1e-3		
	Objective area (mm ²)	e.g. 1000		

The example result of nvCIM architecture exploration is shown as follows:



CoMN-design Platform

#####Optimizing key specification for target DNN model#####

iter	target	ColumnMUX	Htreeflitband	Macronumbers	Meshflitband	Subarray	buffersizeTile	buswidthTile
total energy (mJ): 0.07815405425761929 total latency (ms): 5.112393194958678 total area (mm2): 39469.641463596075								
1	2.534e-05	14.51	93.32	16.03	44.28	204.9	3.908e+04	108.4
total energy (mJ): 0.10120742065956108 total latency (ms): 8.670853885950415 total area (mm2): 90865.15389712191								
2	1.101e-05	12.37	53.2	145.3	58.3	721.8	6.663e+04	451.5
total energy (mJ): 0.08796665203549997 total latency (ms): 31.675935485950415 total area (mm2): 90409.10391312189								
3	1.106e-05	2.822	87.14	116.2	75.04	198.8	6.507e+04	413.2

Exit

Reference

- [1] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 12, pp. 3067-3080, 2018.
- [2] N. Muralimanohar, R. Balasubramonian, and N. P. Jouppi, "CACTI 6.0: A tool to model large caches," *HP laboratories*, vol. 27, p. 28, 2009.
- [3] H.-S. Wang, X. Zhu, L.-S. Peh, and S. Malik, "Orion: A power-performance simulator for interconnection networks," in *35th Annual IEEE/ACM International Symposium on Microarchitecture, 2002.(MICRO-35). Proceedings.*, 2002: IEEE, pp. 294-305.
- [4] F. Nogueira, "Bayesian Optimization: Open source constrained global optimization tool for Python," URL <https://github.com/fmfn/BayesianOptimization>, 2014.