南开大学智能计算系统研究室

高性能智能合约体系架构

——高性能链上DNN计算（推理）

SmartVM: A Smart Contract Virtual Machine for Fast on-Chain DNN Computations

Tao Li, **Yaozheng Fang**, Ye Lu* (luye@nankai.edu.cn), Jinni Yang, Zhaolong Jian, Zhiguo Wan, Yusen Li

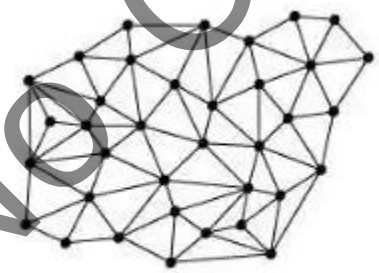
IEEE Transactions on Parallel and Distributed Systems, **DOI:** *10.1109/TPDS.2022.3177405*

房耀政 南开大学　　2023年1月

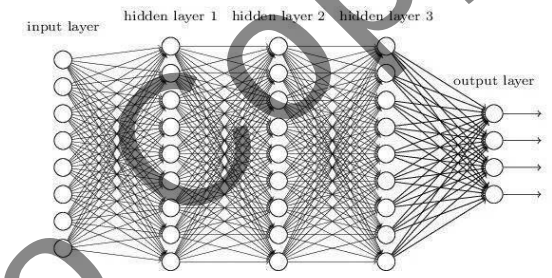
目录

目录

▶ Blockchain & Artificial Intelligence

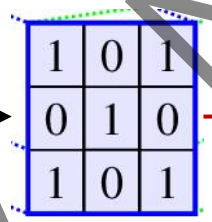


- 分布式数据库+状态复制机
- 存储（数据）防篡改、操作（计算）可共识
- 公开透明



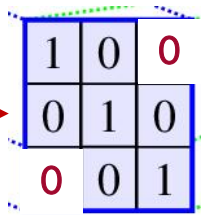
深度神经网络

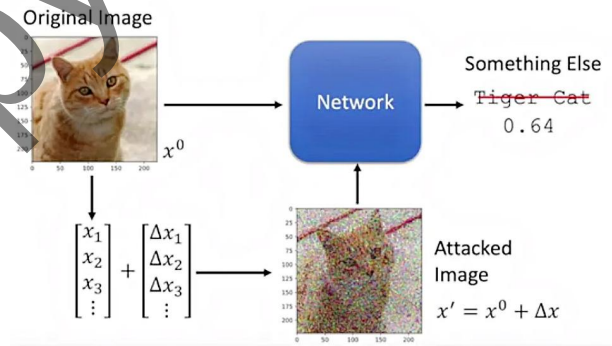
迭代训练



网络权重

权重篡改等安全问题





▶ Blockchain-based Artificial Intelligence (BC-AI)

区块链可以和分布式AI天然结合，构建安全、可信的分布式AI系统 (Blockchain-based AI Systems)

▶ Problems of off-chain DNN inference

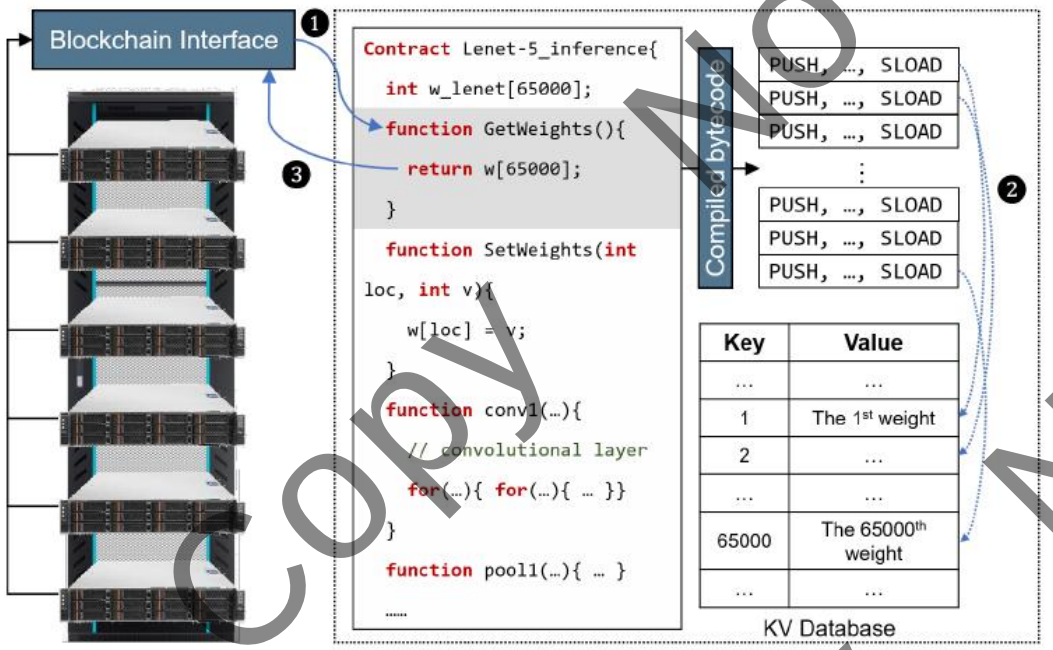


Fig. 2: CNN Computing Process in Typical BC-AI.

① 合约方法调用，**请求**获取权重

② 字节码执行，**按顺序**从数据库中**取权重**

③ 权重数据**返回**传输

问题：

① 合约方法执行时字节码数量庞大，达千万条

② 权重分散存储，数万条记录，且CNN权重越来越多

③ 数据转移代价

EVM+Solidity组合：

LeNet网络数分钟，AlexNet网络近1小时；数据搬运代价高

目录

■ 智能组件的思考——构建分布式人工智能决策和协作平台

智能组件是一个新概念，最早由西安交通大学郑南宁院士于2018年科技部“科技创新2030-新一代人工智能重大专项”项目组的一次研讨会中提出。

[1] 袁勇, 欧阳丽炜, 王晓, 王飞跃, “基于区块链的智能组件：一种分布式人工智能研究新范式”, 数据与计算发展前沿, 2021, Vol. 3, No. 1, pp. 1-14.

Running Deep Learning on EVM

EVM

kladkogex Jan '18

At my company we are starting an experimental project to extend EVM with basic deep learning capabilities.

This is not to train a neural network, but to use a pre-trained neural network inside a smart contract. Computation-wise using a pre-trained neural network is actually not so much more expensive than doing, say, RSA.

I understand that this may be a bit too heavy for the official Ethereum blockchain, so in our case we will run the EVM on a separate permissioned cluster with BFT-like consensus.

The current plan is that:

1. A pre-trained network is saved on the blockchain. We can use some of existing neural network serialization standards such as the ones used in Keras framework 36
2. The EVM will need to pull the neural network from the blockchain.
3. In the simplest case there we add a single **predict** instruction similar to predict from Keras framework 22. This instruction will take a fully qualified name of the neural network and an input data array, run the neural network and produce output data.

As an example input data could be an English-language string, and output will be a German translation of this string.

One problem that we will need to solve in the process is introducing deterministic floating point numbers such as IEEE 754-2008 into the EVM in some way.

If there are other people interested to run AI on EVM, we would be willing to cooperate on this to establish a standard that everyone uses …

- **社区内的真实需求**
- Distributed Uber
- Trusted analysis

Well - I think many people have expressed different ideas about running a neural network as a trusted application so that all parties agree to the outcome.

As a toy example, you can consider an example of a smart contract that is an decentralized Uber which pays to drivers based on their behavior. The smart contract needs to differentiate bad drivers from good drivers by running a neural network on driver historical behavior. Good drivers are get paid and bad drivers do not get paid.
What you do, you feed driver's trajectory into a neural network, and then the driver gets either paid in-full or penalized based on her behavior …

Another example is when a smart contract buys apples from a supplier and it needs to find out whether an apple is a good apple or a bad apple based on chemical analysis data. Essentially you feed into the neural network 100 data points of chemical analysis and the network tells you whether the apple is good or bad.

In general, I think to answer your question, anytime a smart contract runs on data, there is no need to use neural networks. On the other hand if a smart contract runs on Big Data, then, arguably, you need a neural network to extract important info from the data, as the data itself is too complex.

In the examples above the neural network does not need to be confidential or encrypted in anyway, it could be a pre-trained network which is trusted by all participants.

I am not saying EVM is a perfect place to run neural networks, on the other hand making it some kind of an simple extension to EVM/Solifity would draw many developers. Another possibility is to run a totally different thing and then feed the results into Ethereum somehow …

- **Moving computations rather than moving data!**
- **Convolutional Neural Network**

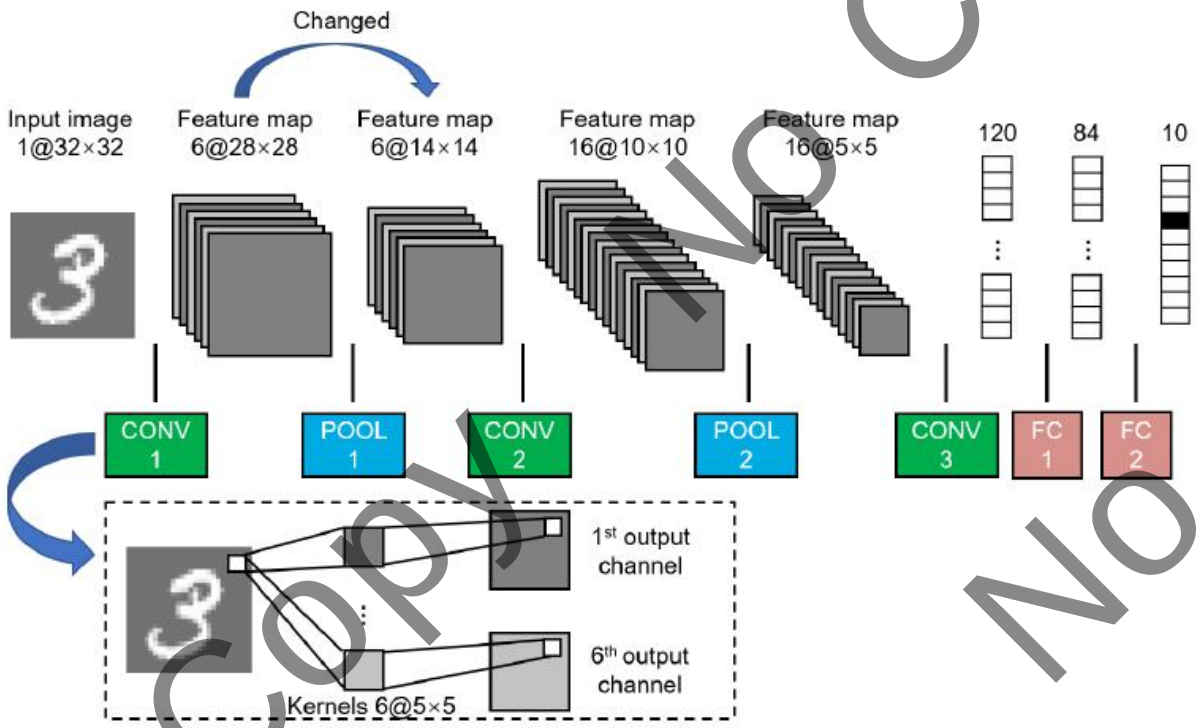
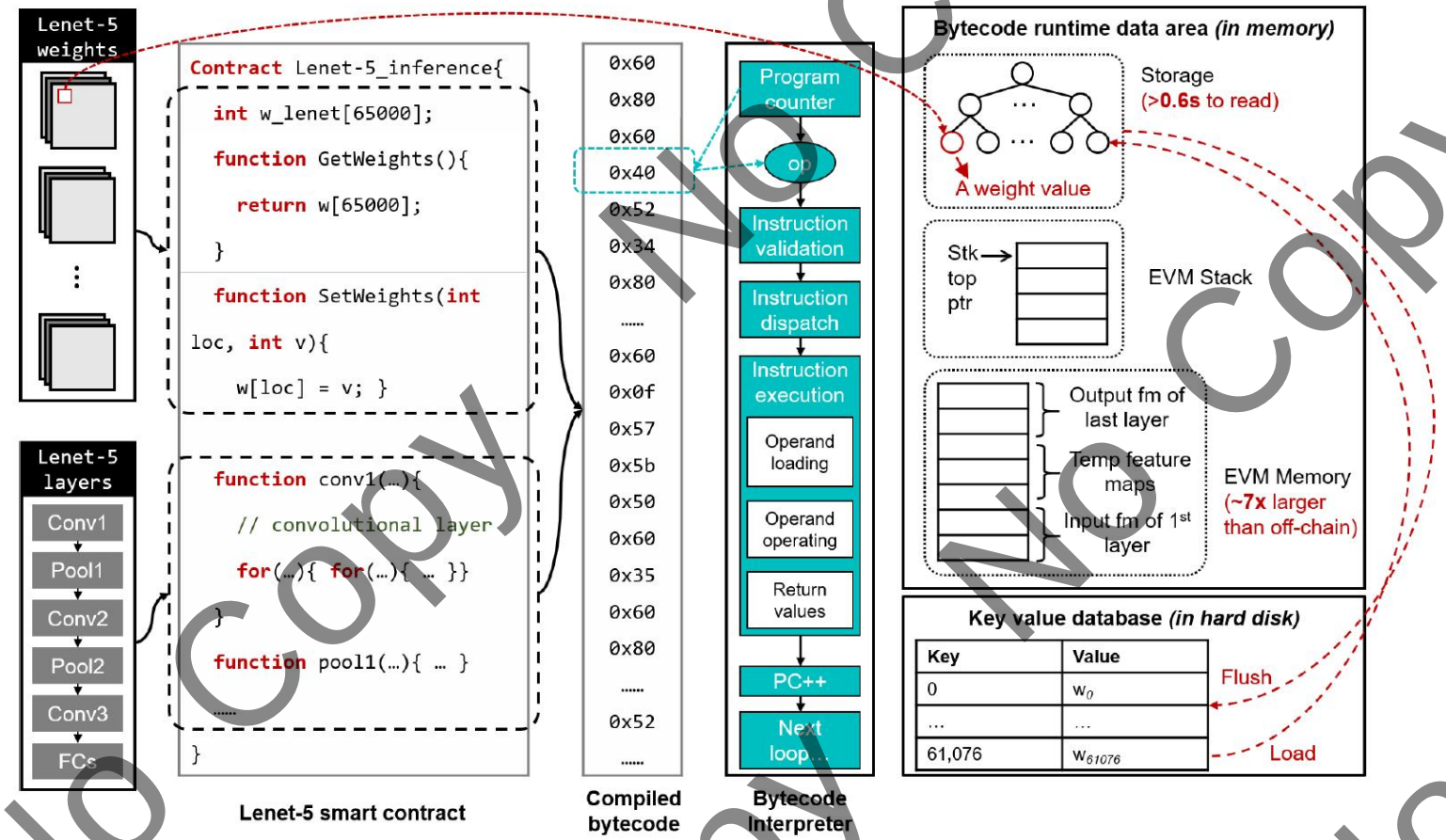


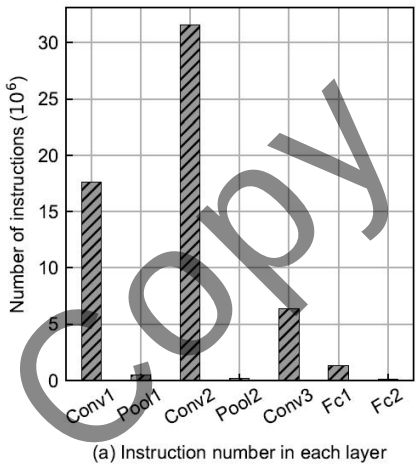
Fig. 1: LeNet-5 architecture of inference.

- **经过某层计算后，计算前后的特征图大小一般会不同**
  - 例如，卷积层增大，池化层减小
- **各同类型层计算逻辑基本相同**
  - 例如，Conv1和Conv2
- **权重获取和计算有先后顺序，且各通道相互独立**
  - 例如，第1通道和第6通道

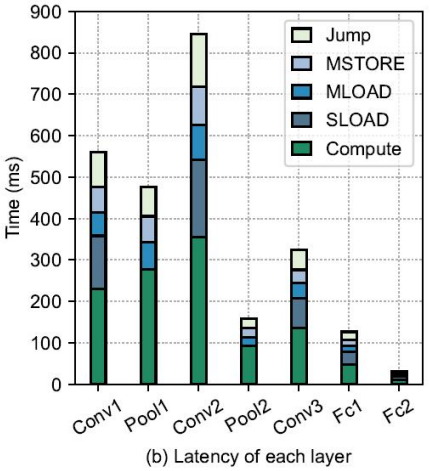
- **Moving computations rather than moving data!**
- **Problems of on-chain CNN inference**



挑战一：通用指令集及特定指令缺失，基于栈的虚拟机需要频繁处理栈数据，导致CNN推理需执行上千万条指令

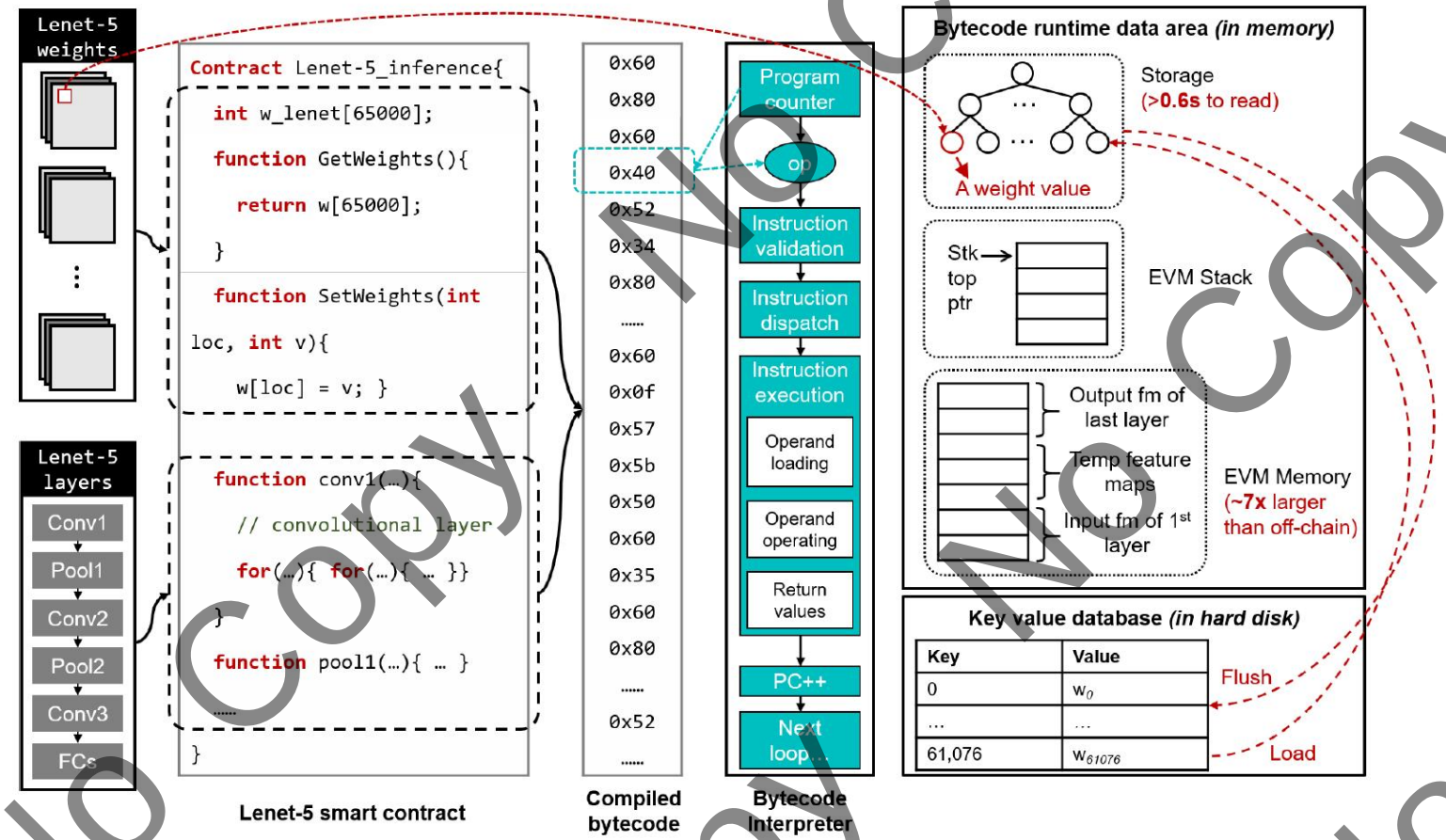


(a) Instruction number in each layer

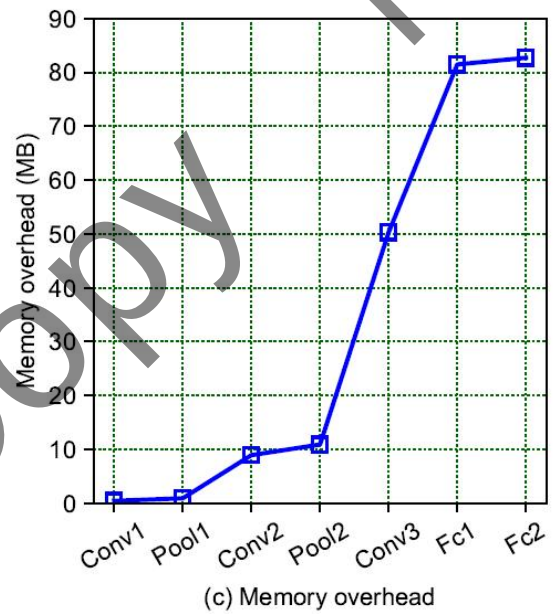


(b) Latency of each layer

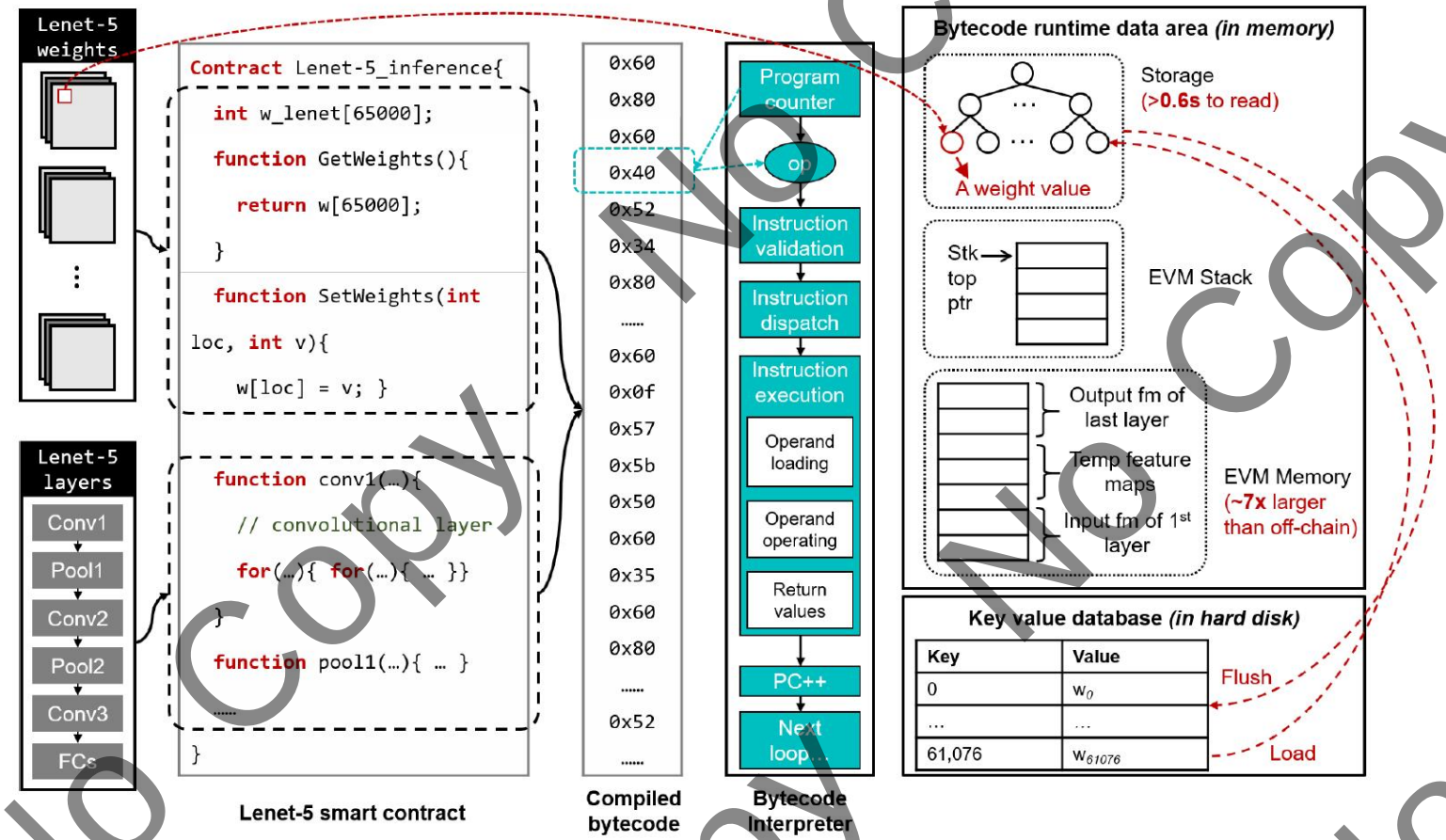
- **Moving computations rather than moving data!**
- **Problems of on-chain CNN inference**



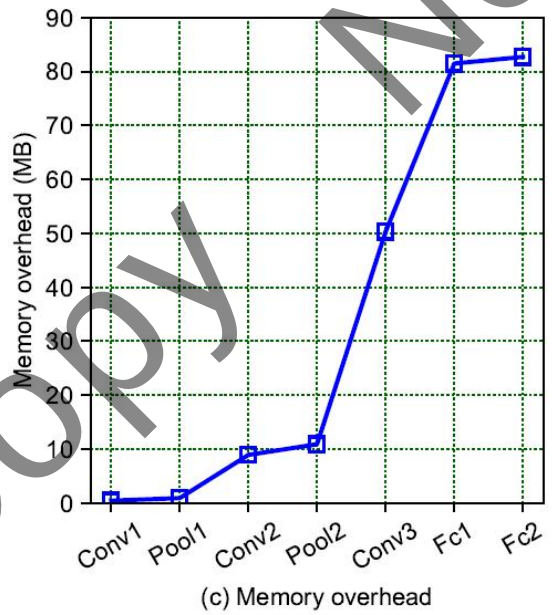
挑战二：内存管理欠缺，仅包括一个空闲指针管理(0x40)，导致CNN推理运行时大数据无法被有效管理



- **Moving computations rather than moving data!**
- **Problems of on-chain CNN inference**



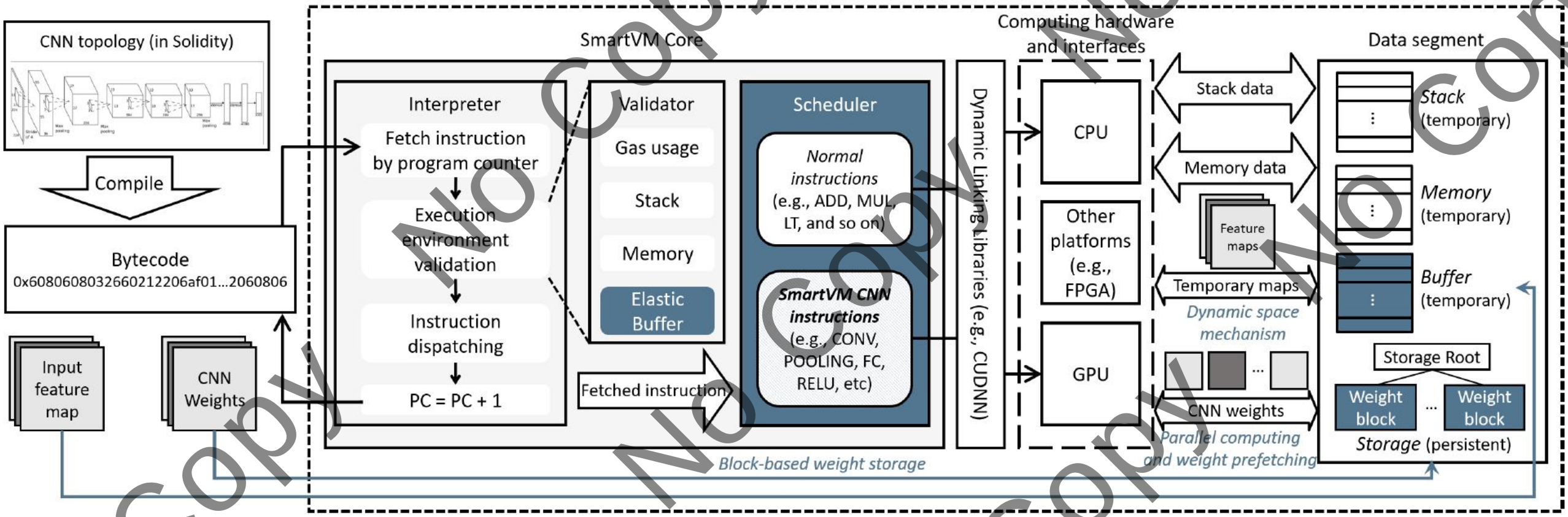
挑战三：串行(for)单平台计算(CPU)无法匹配CNN并行度高的特点和需求



(c) Memory overhead

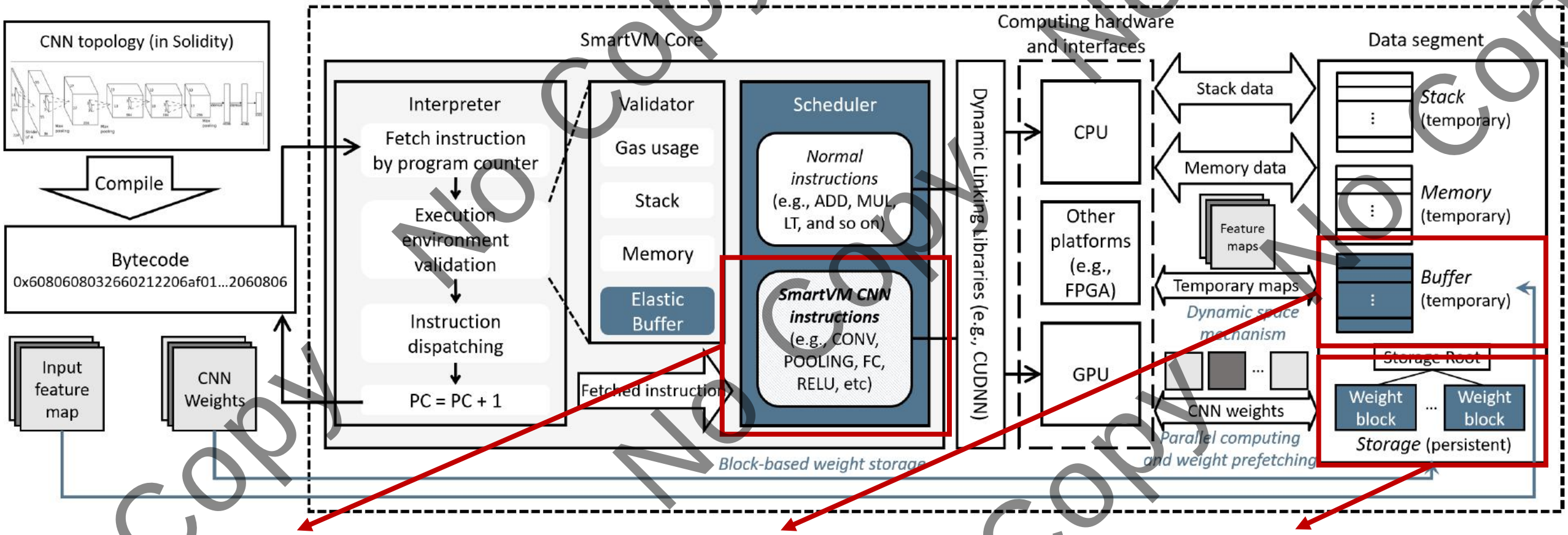
目录

SmartVM Overview



- CNN-oriented Instructions
- Dynamic Memory Management Method
- CNN Weight Prefetching and Parallel Computation

▶ SmartVM Overview



◆ CNN-oriented Instructions

◆ Dynamic Memory Management Method

◆ CNN Weight Prefetching and Parallel Computation

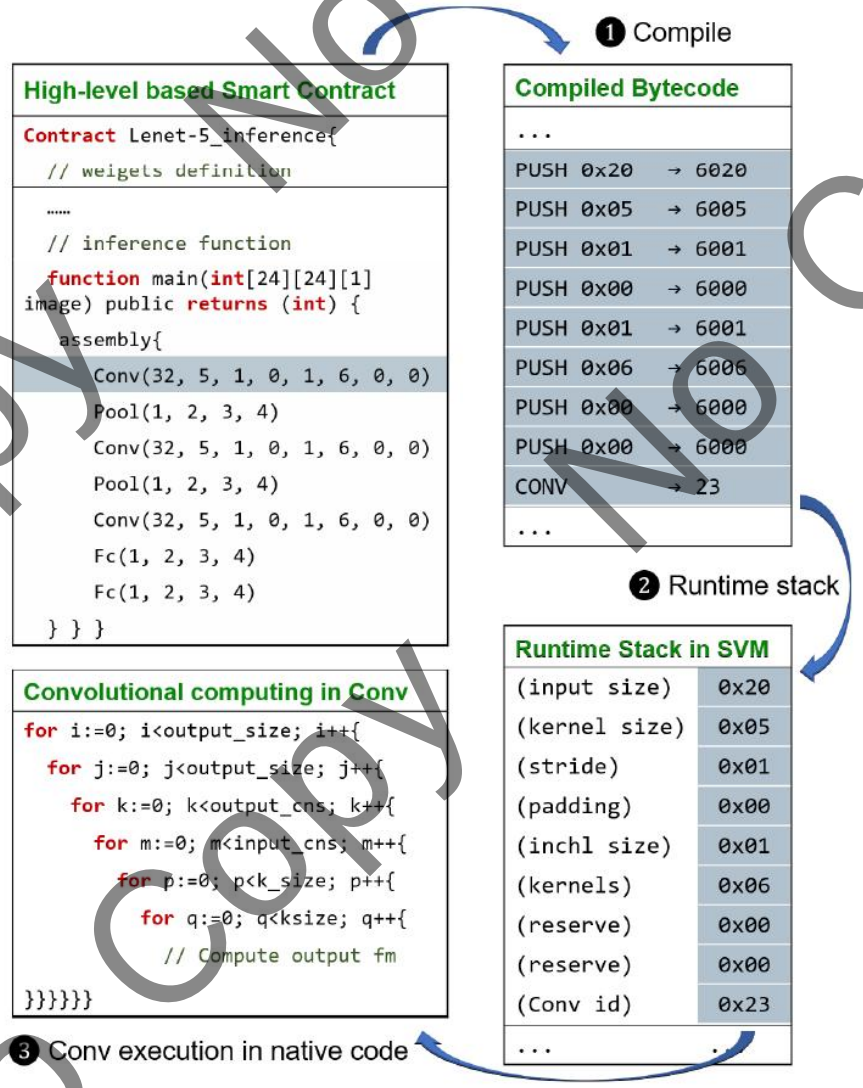
▶ CNN-oriented Instructions

Type	Name	Opcode	Description	Stack required (Key arguments)
Computation (Convolution)	CONV_SING	0x21	Implement single channel convolution	8 (Kernel, Output channel, Stride)
	CONV_MUL	0x22	Implement multi-channel convolution	8 (Kernel, Output channel, Stride)
	CONV_3D	0x23	Implement 3D convolution	8 (Kernel, Output channel, Stride)
	CONV_TPD	0x24	Implement transposed convolution	8 (Kernel, Output channel, Stride)
(Pooling)	POOL_MAX	0x25	Implement max pooling	5 (Stride, Input channel)
	POOL_AVG	0x26	Implement average pooling	5 (Stride, Input channel)
	POOL_OL	0x27	Implement overlapping pooling	5 (Stride, Input channel)
(Full connected)	FULL_CON	0x28	Implement full connected layer	5 (Input channel, Output channel)
	MAT_MUL	0x29	Implement matmul	2 (Addresses of two matrix)
(Active)	ACT_SM0	0x2a	Implement softmax function	1 (Value)
	ACT_SM1	0x2b	Implement Sigmoid function	1 (Value)
	ACT_RL	0x2c	Implement ReLU function	1 (Value)
	ACT_TANH	0x2d	Implement Tanh function	1 (Value)
(Buffer)	BUF_SCL0	0x2e	Increase Buffer's data with specific times	1 (Specific times)
	BUF_SCL1	0x2f	Reduce Buffer's data with specific times	1 (Specific times)
	BUF_BIAS	0x30	Add Buffer's data and bias	1 (Base address of bias)
Data transfer	MTOB	0x31	Transfer data from Memory to Buffer	2 (Data offset)
	BTOM	0x32	Transfer data from Buffer to Memory	2 (Data offset)
	BTOS0	0x33	Transfer data from Buffer to Stack	2 (Data offset, Size)
	BTOS1	0x34	Transfer data from Buffer to Storage	2 (Data offset, Size)
(Buffer set)	BUF_CLS	0x35	Clean Buffer's data	1 (Clean number)
	BUF_FIL	0x36	Fill Buffer's data with specific data	1 (Specific filled data)
	BUF_INIT	0x37	Initial Buffer with specific size	1 (Specific size)
	BUF_ALLO	0x38	Allocate specific size to Buffer	1 (Specific size)
	BUF_FREE	0x39	Free specific size from Buffer	1 (Specific size)
	BUF_COPY	0x3a	Copy a same Buffer	2 (Start and end pointers)

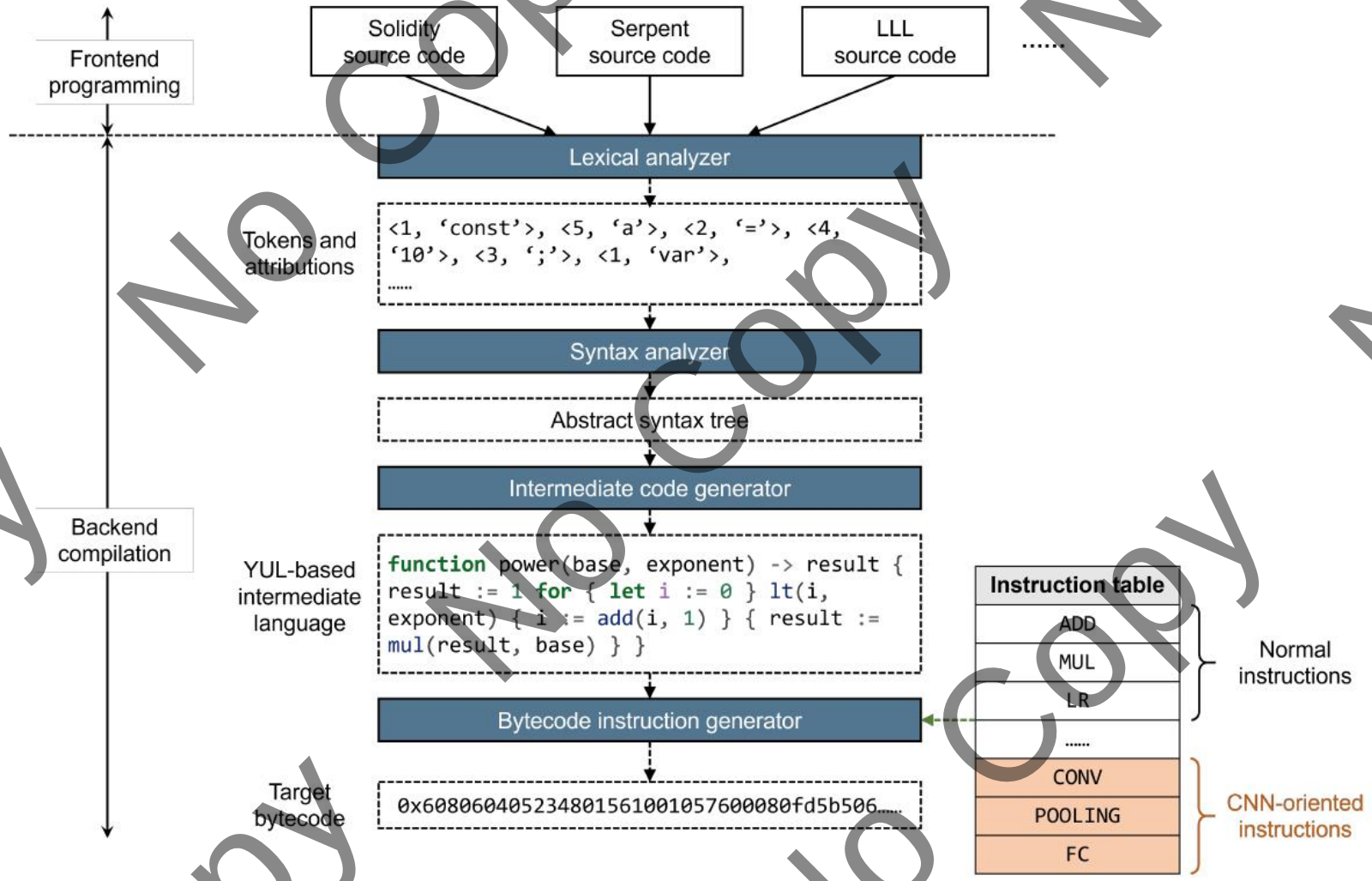
- 计算指令
  - Conv
  - Pool
  - FC
  - ...
- 数据转移指令
  - Stg.
  - Stk.
  - Mem.
  - Buf.

▶ CNN-oriented Instructions Workflow

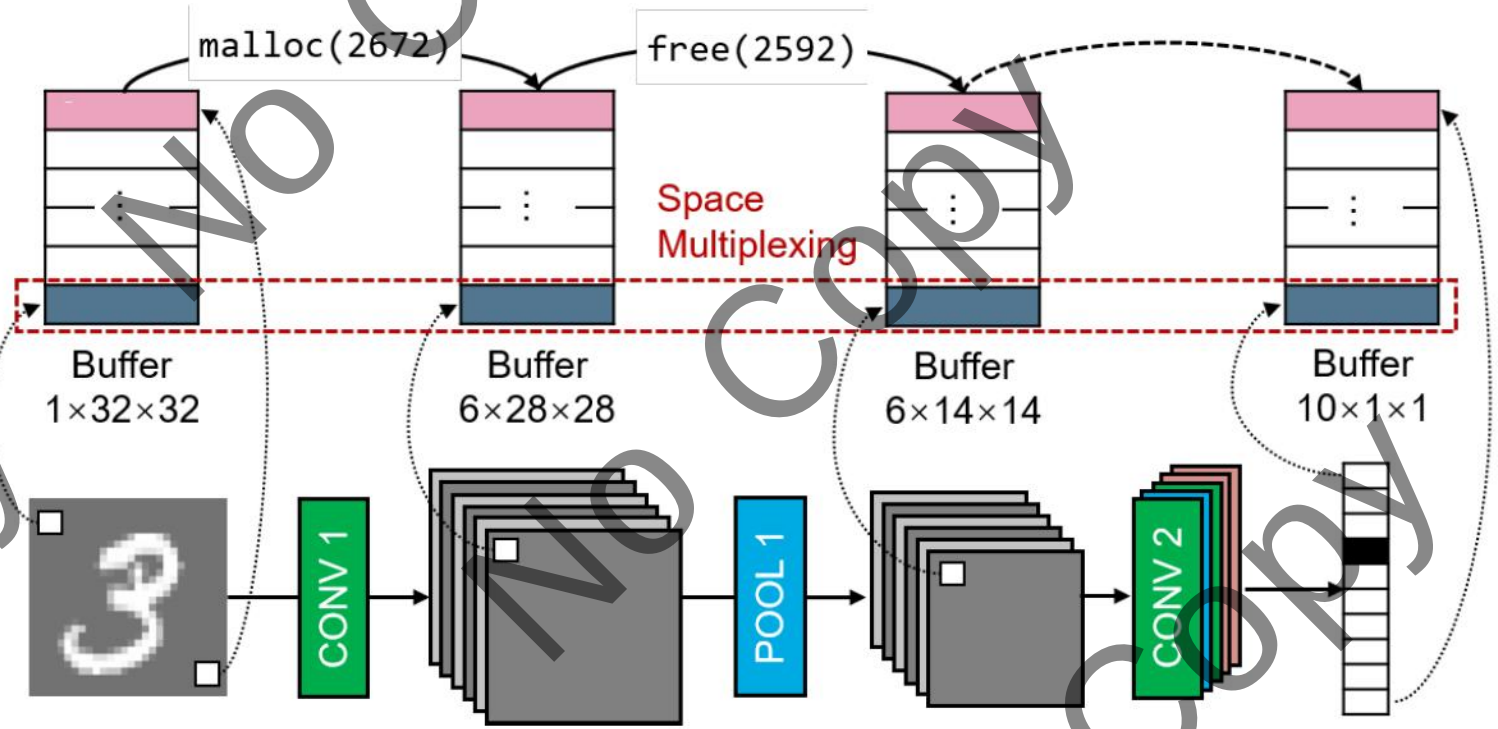
- 编译Implementation
  - 采取行内汇编的形式
  - 参数记录在栈中
  - N*Push + 1*CNNI
- 运行时参数入栈
- Native Code执行循环



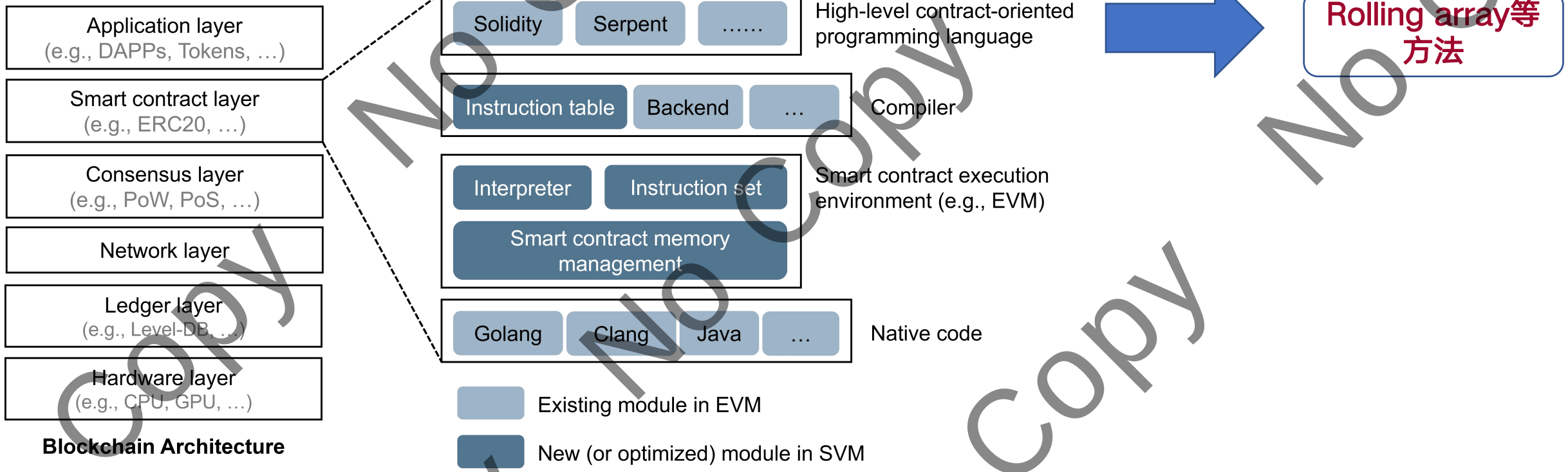
▶ Compiler for CNN-oriented Instructions（编译层次）



▶ Dynamic Memory Management Method



Dynamic Memory Management Method



▶ CNN Weight Prefetching and Parallel Computation

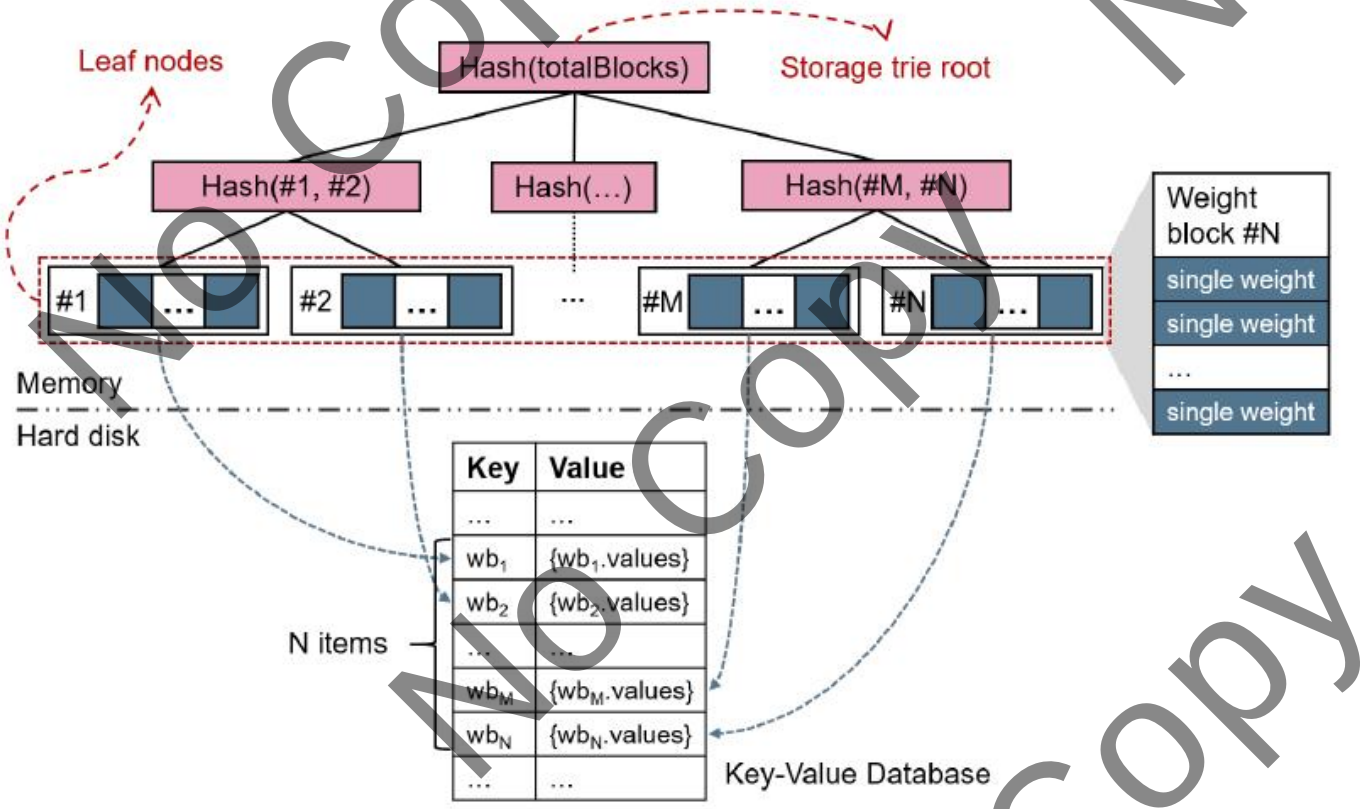
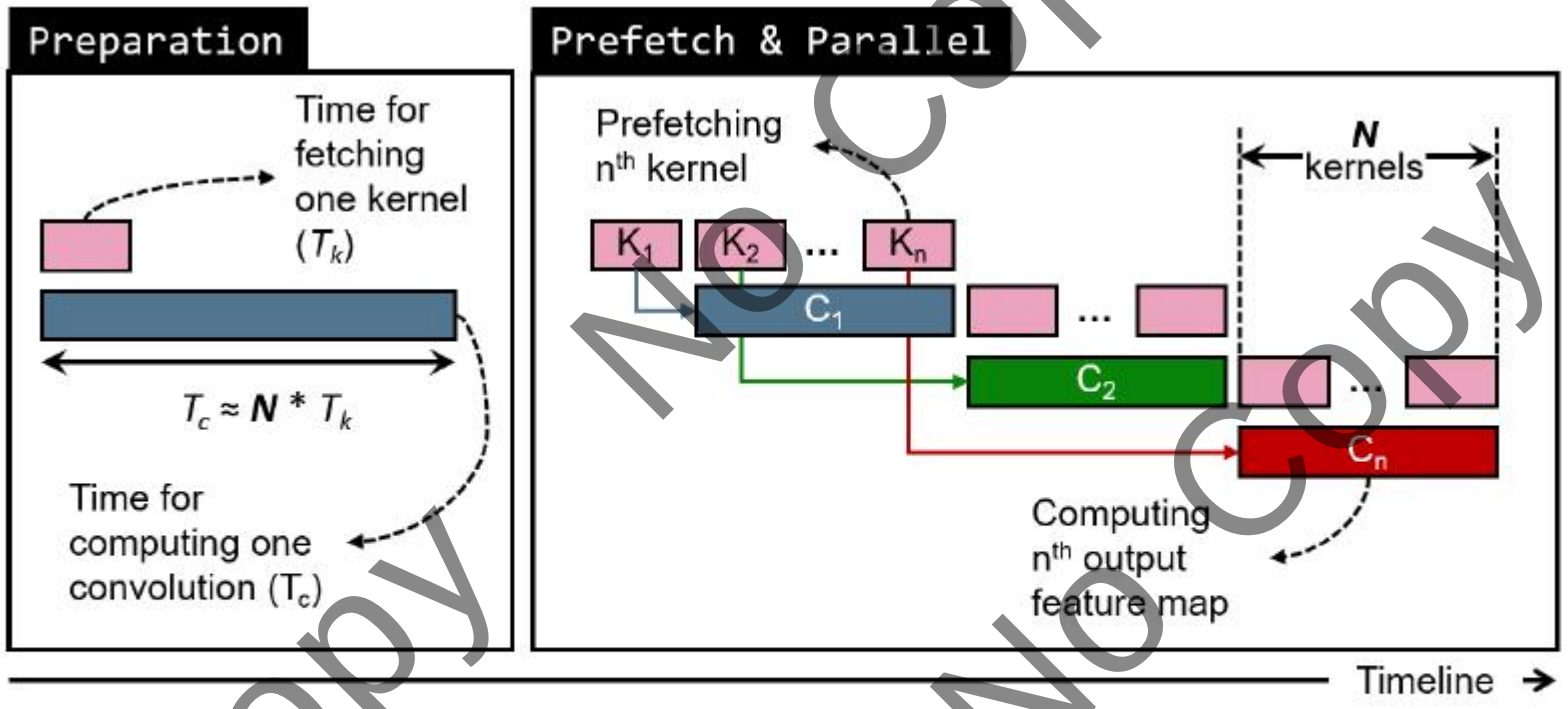


Fig. 9: The block-based weight storage.

▶ CNN Weight Prefetching and Parallel Computation



Inner-instruction parallel

Fig. 10: The weights prefetching and parallel computation model in Conv instruction.

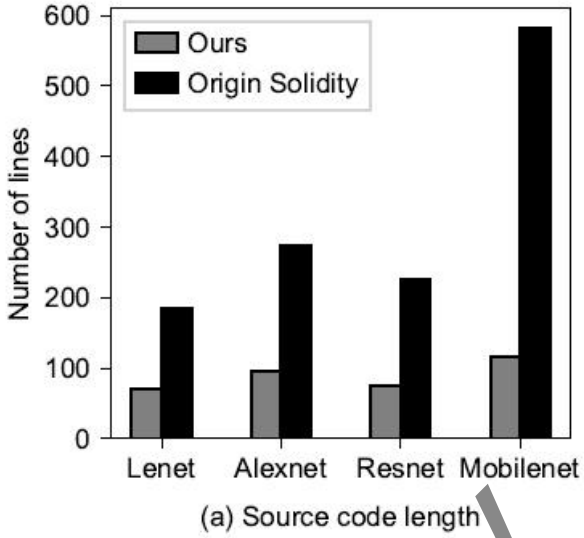
目录

▶ Setup

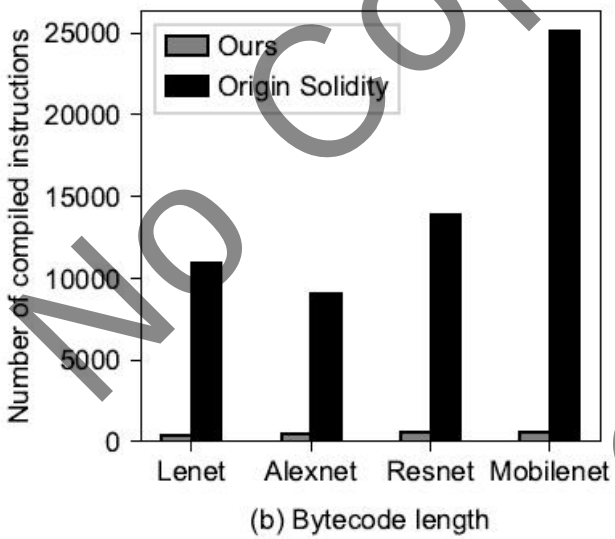
Code Length	Source code	Compiled bytecode	
Latency	End-to-end	Weights fetching latency	Computing latency
Memory footprint	EVM Memory vs. Buffer	SmartVM vs. Native code	

LeNet, AlexNet, ResNet18, MobileNet, and LSTM

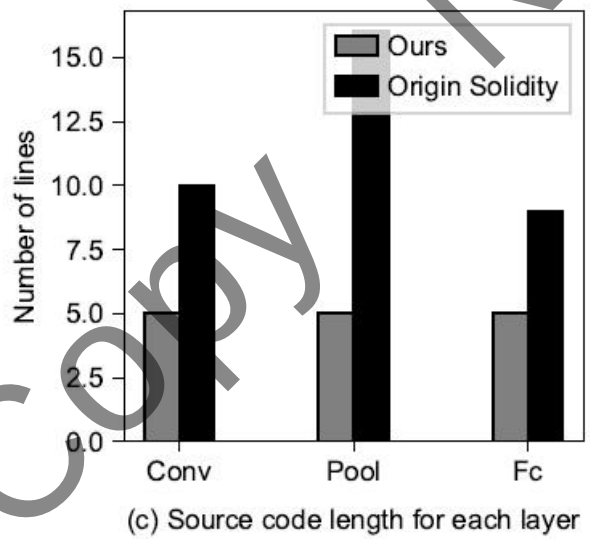
▶ Code length



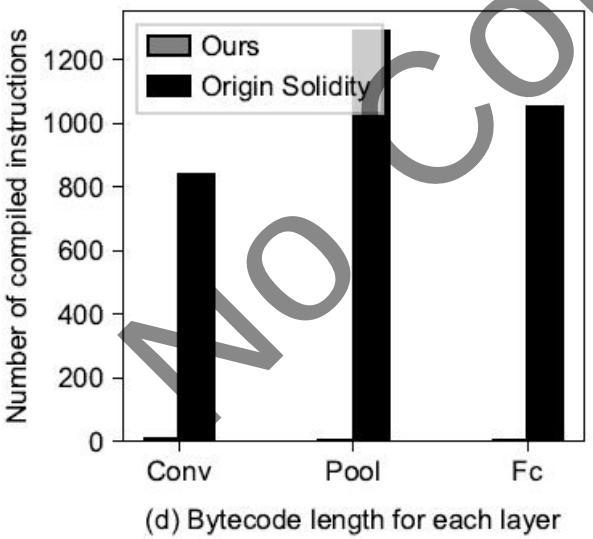
(a) Source code length



(b) Bytecode length



(c) Source code length for each layer



(d) Bytecode length for each layer

110 lines
vs.
600 lines

↓ 95%

► Code length

Native Solidity-based Lenet-5 inference contract

```
pragma solidity ^0.5.1;
contract lenet {
 ……
  function convolution1(uint input_size, uint kernel_size,
uint stride, uint padding, uint input_channel_number,
uint output_channel_number) public view
returns(int[24][24][6] memory output_fm){
  uint output_size = (input_size - kernel_size + 2 *
padding) / stride + 1;
  for(uint i=0; i<output_size; i++){
   for(uint j=0; j<output_size; j++){
    for(uint k=0; k<output_channel_number; k++){
     for(uint m=0; m<input_channel_number; m++){
      for(uint p=0; p<kernel_size; p++){
       for(uint q=0; q<kernel_size; q++){
          output_fm[k][i][j] = output_fm[k][i][j] +
conv1_kernel[k][m][p][q] * input_fm_conv1[m][stride * i +
p][stride * j + q];
  }}}}}}
  return output_fm;
 }
 ……
}
```

Compile

0x6080604052348015610010576000080fd5b50612cce806100206000396000f3fe6080604052600436106101645760003557c0100000000000000000000000000000000000000000000000000000009004806310a0991a1461016957806311260048036036020811015610181018c57600080fd5b810190808035906020019092919050505061113cb565b60405180828152602001915050604051f35b34801561……

Compiled bytecode of convolution1(…)
(~17,580,000 instructions)

Compiled results are the same

Compiled results are different

Solidity-based Lenet-5 inference smart contract (with function abstraction)

```
pragma solidity ^0.5.1;
import ‘nnlib.sol’
contract lenet {
 ……
  function convolution1() public view
returns(int[24][24][6] memory output_fm){
  network{
   Conv(…, …)
   ……
  }
  return output_fm;
 }
 ……
}
```

Compile

91a146101695780631126004803603602081101561018c57600080fd5b8101908080359060200190929190505050611 13cb565b60405180828152602001915050604051f35b34801561……

Compiled bytecode of convolution1(…)
(~17,580,000 instructions)

Solidity-based Lenet-5 inference smart contract (with CNN-oriented instructions)

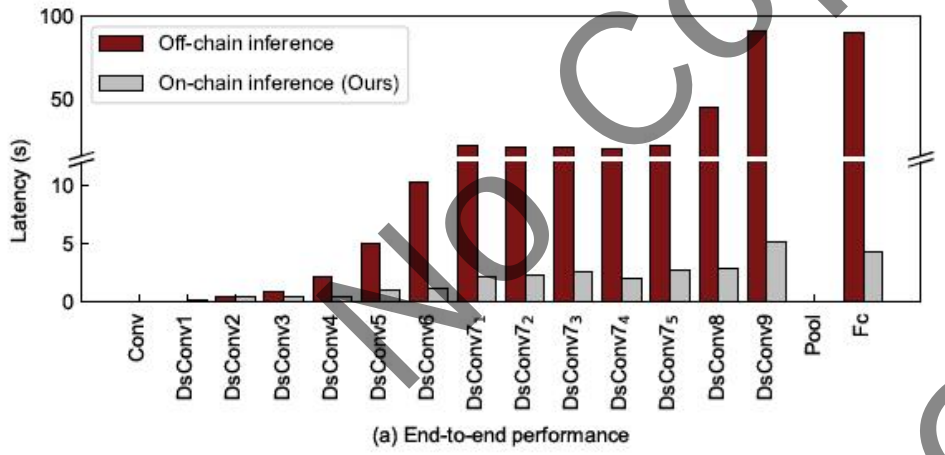
```
pragma solidity ^0.5.1;
contract lenet {
 ……
  function convolution1() public view
returns(int[24][24][6] memory output_fm){
  assembly{
   Conv(…, …)
   ……
  }
  return output_fm;
 }
 ……
}
```

Compile

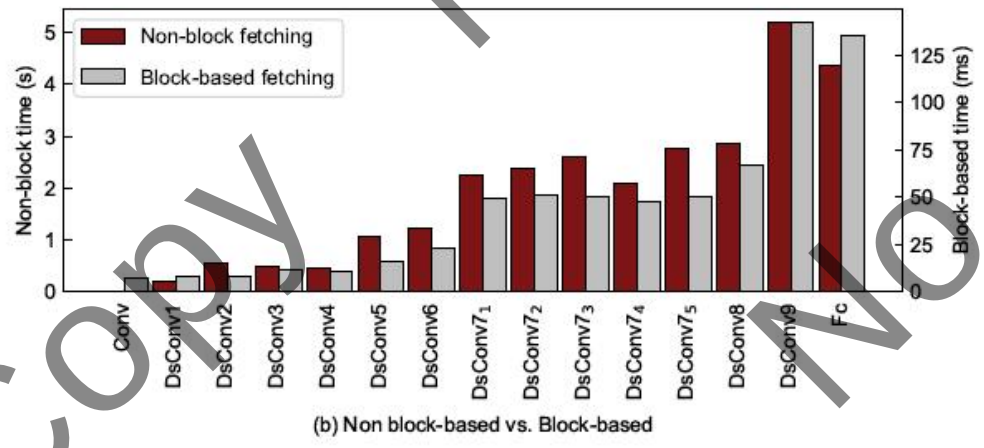
0x6080……6020600560016006600060023……

Compiled bytecode of convolution1(…)
(~30 instructions)

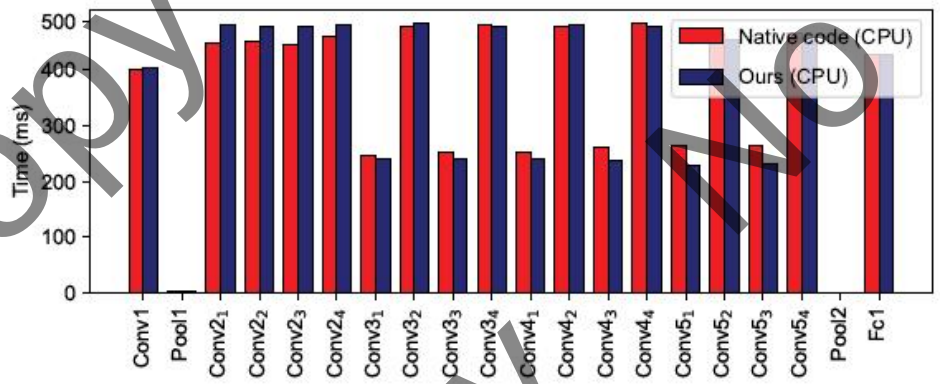
Latency



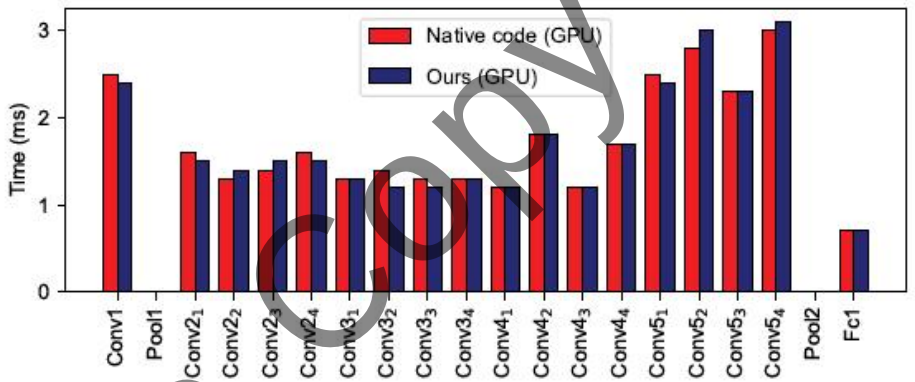
(a) End-to-end performance



(b) Non block-based vs. Block-based



(c) Inference time on CPU



(d) Inference time on GPU

▶ Latency

- CNN-oriented Instructions
  - 降低97.3%计算时延
- **基于块存储**
  - 降低98.7%权重获取时延
- Pipeline method
  - 降低13.1%
- Overall
  - 93.6%

TABLE II
PIPELINE LATENCY.

Network	Latency (ms)			
	No pipeline			Pipelined
	Weights fetching	Computing	Total	Total
LeNet	2	10	12	9.87
Alexnet	1228	4370.98	5598.98	4832.68
ResNet18	328	7037	7365	6871
MobileNet	677.7	2826.47	3504.17	3005.52

▶ Memory/RAM footprint

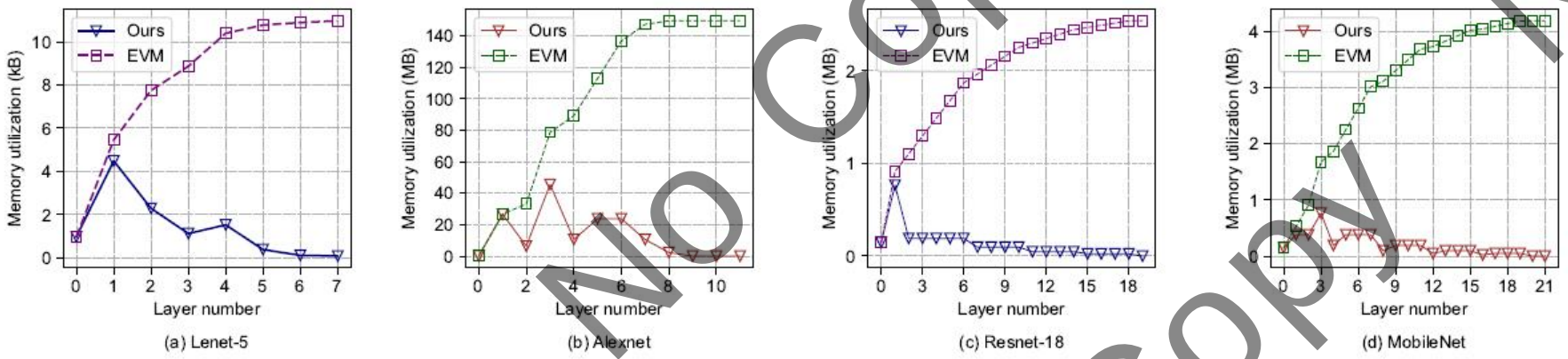
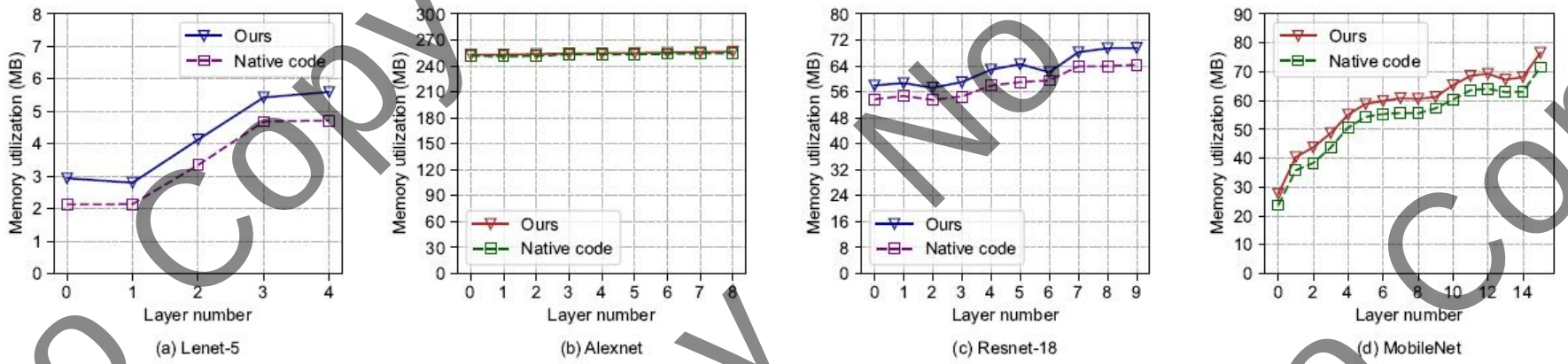


Fig. 16: RAM footprint comparison between SmartVM Buffer and EVM Memory.



- 内存分别降低84%, 90.8%, 94.3%, 及93.7%
- 内存占用近似（由于外部库调用，略高）

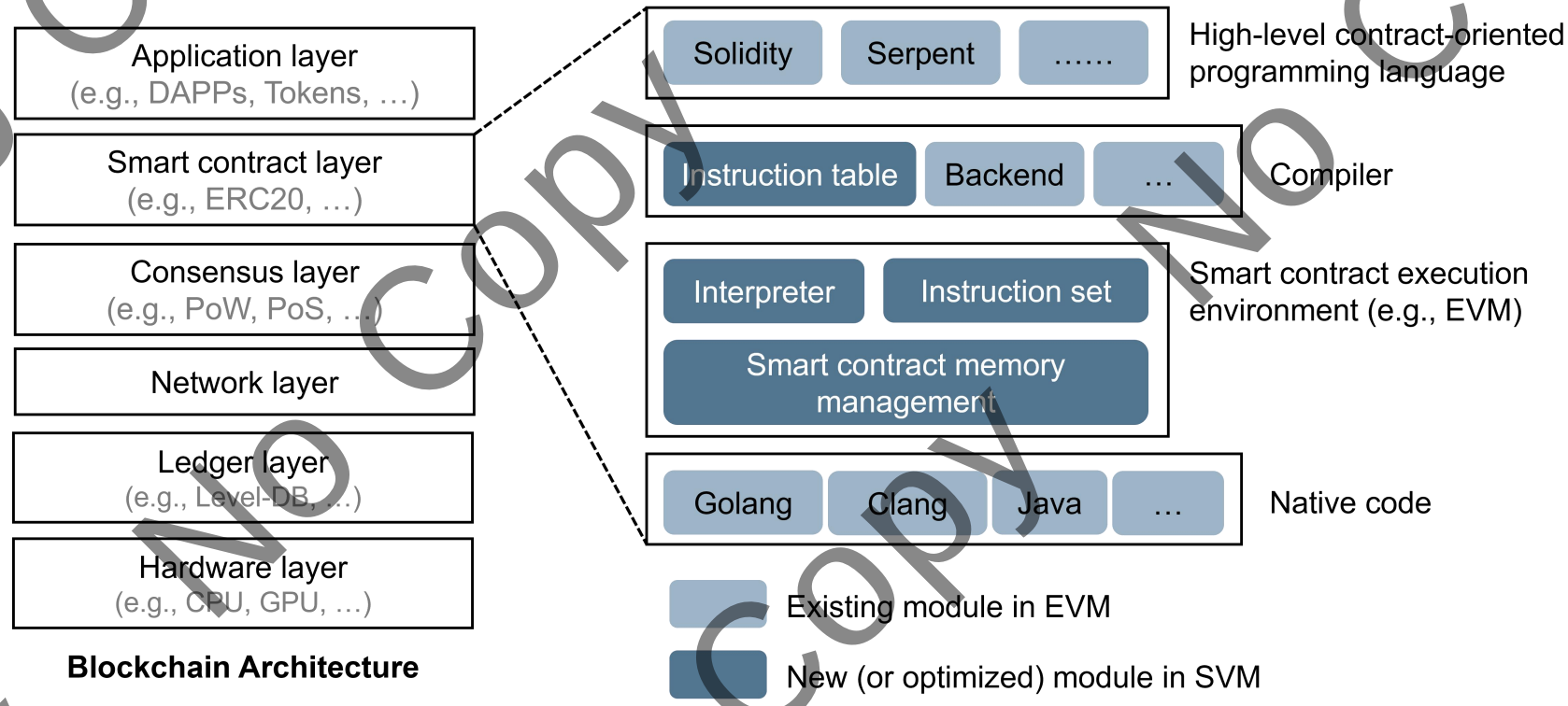
目录

Conclusion

- 从底层支持并加速了链上CNN推理
- 为EVM等合约执行环境提供了面向CNN推理的内存管理方法

未来方向：

- CNN compression for on-chain execution
- Data compression and direct data processing
- Parallel architecture



Blockchain Architecture

南開大學 Nankai University

报告内容到此结束，谢谢！

Q & A

- 欢迎访问：https://ics.nankai.edu.cn
- 研究室邮箱：aics@nankai.edu.cn
- 邮箱：fyz@mail.nankai.edu.cn
- 官方仓库：https://github.com/nkicsl

This work is partially supported by the CCF-AFSG Research Fund (CCF-AFSG RF20210031), the Special Funding for Excellent Enterprise Technology Correspondent of Tianjin (21YDTPJC00380), the National Natural Science Foundation (62002175), the Open Project Fund of State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences (CARCHB202016), the Key Research Project of Zhejiang Lab (No. 2021KF0AB04), the Natural Science Foundation of Tianjin (20JCZDJC00610), the Open Project Foundation of Information Security Evaluation Center of Civil Aviation, Civil Aviation University of China (NO. ISECCA-202102), the People's Republic of China ministry of education science and technology development center (2019J02019), and the Tianjin Graduate Scientific Research Innovation Project (2021YJSB014).