Packet Processing Algorithm Identification using Program Embeddings

S. VenkataKeerthy, **Yashas Andaluri, Sayan Dey**,
Rinku Shah, Praveen Tammana, Ramakrishna Upadrasta

2nd July, 2022
6th Asia-Pacific Workshop on Networking (APNET)

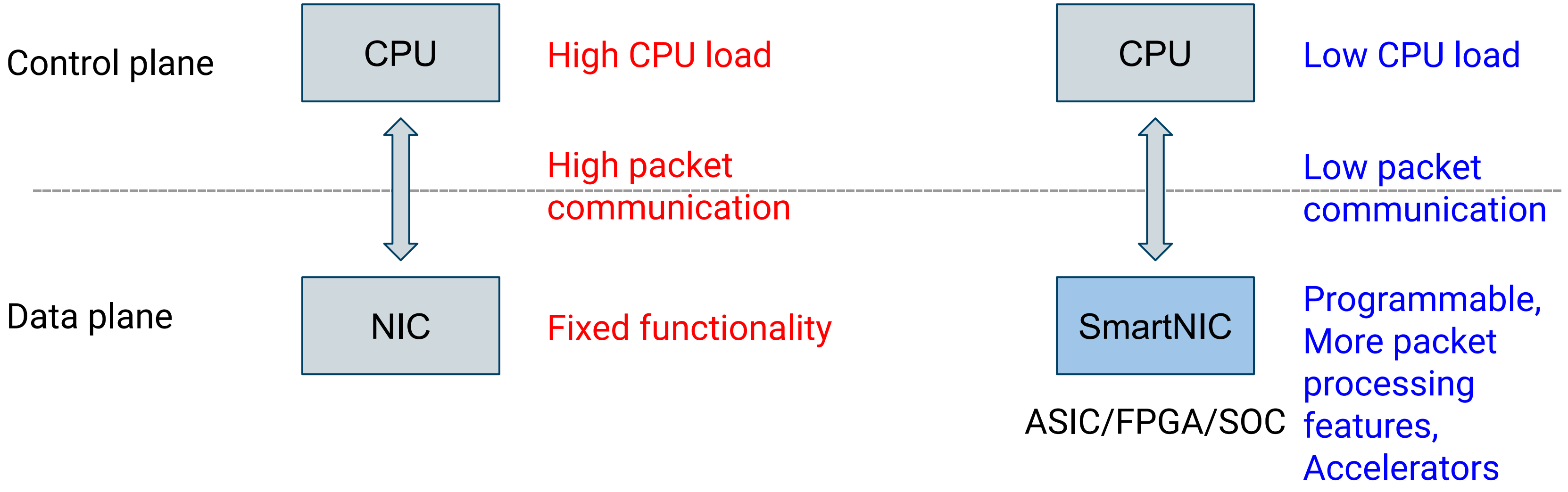
భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Introduction: Overall theme

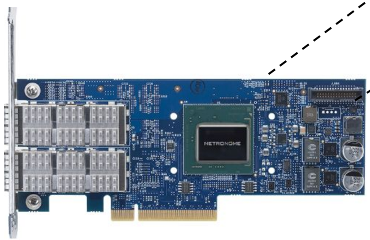
Fact: CPU packet-processing speeds ≪ network speeds

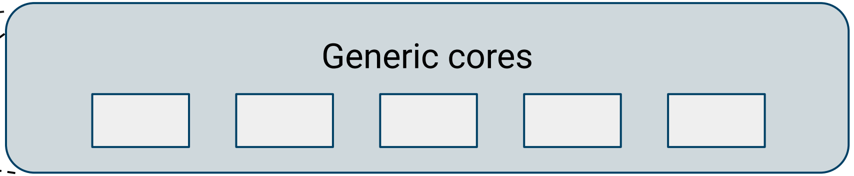
SmartNICs: High speed programmable hardware for packet processing



SmartNICs

- Generic cores for packet processing specified by Network Function (NF) program





Significant performance improvements

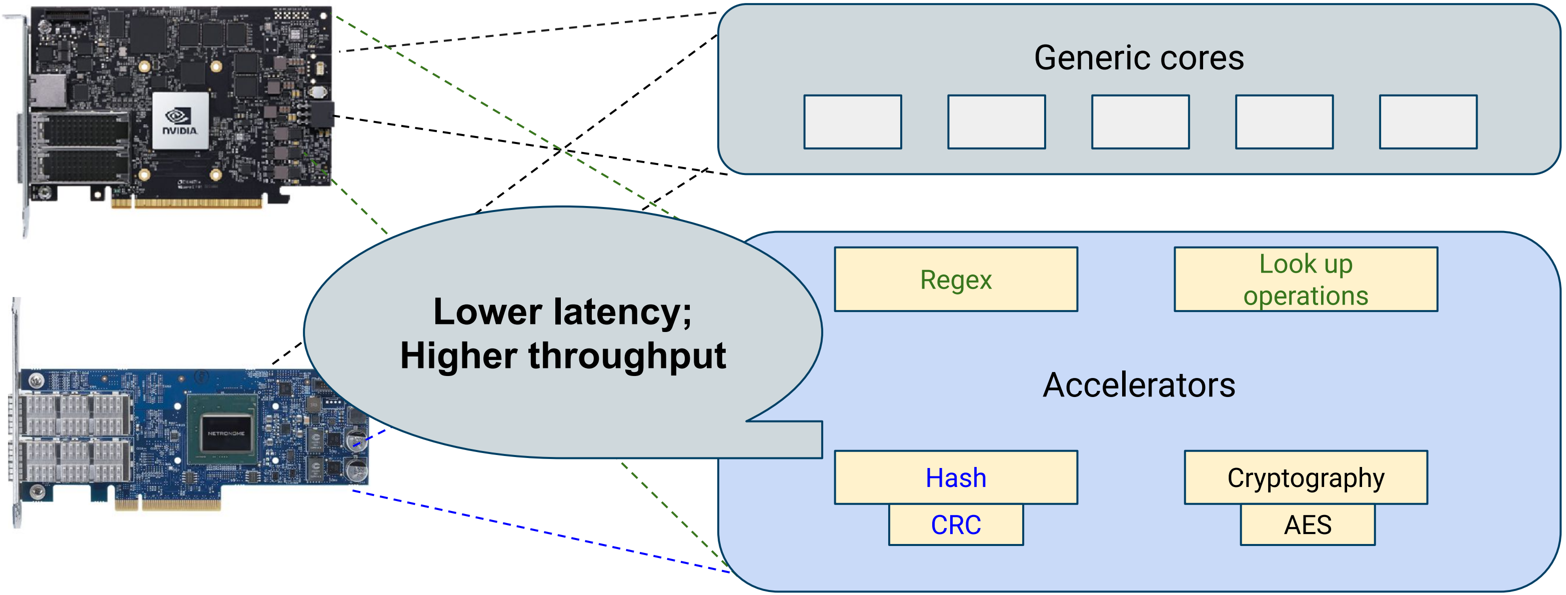
- Latency, throughput and power

IPipe[1]: IPSec program achieved 22.9 Gbps on 25GbE SmartNIC

IPIPE[1]: Offloading Distributed Applications onto SmartNICs Using IPipe: Ming Liu, Tianyi Cui, Henry Schuh, Arvind Krishnamurthy, Simon Peter, and Karan Gupta (ACM Special Interest Group on Data Communication 2019)

SmartNICs

- Hardwired logic for frequently used operations and algorithms



Goal: Offloading NF programs from CPUs to SmartNIC accelerators

Accelerators available on SmartNICs

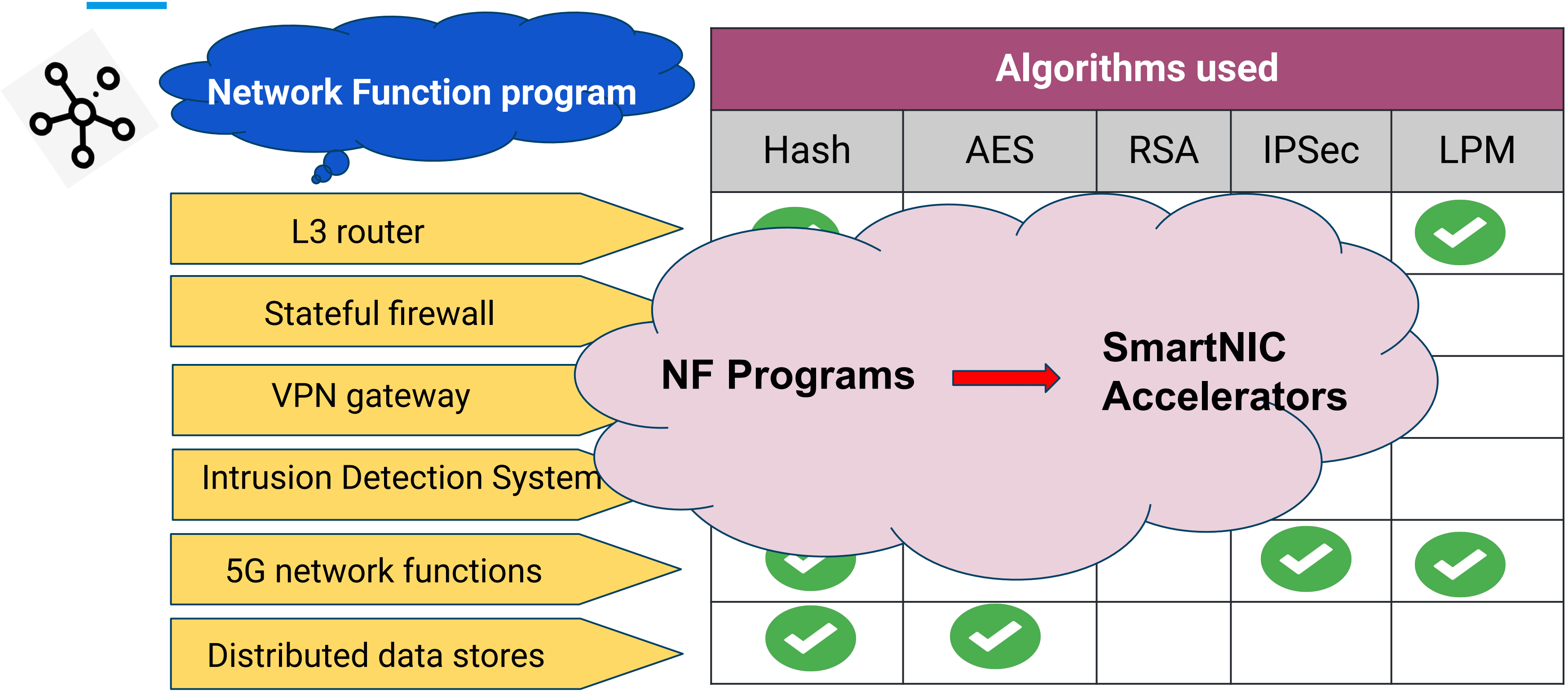
SmartNIC model	Hash	AES	RSA	IPSec	LPM
Netronome Agilio CX	✓	✓	✓	✓	✓
Nvidia Bluefield 3 DPU		✓	✓	✓	
Cavium LiquidIO				✓	
Marvell OCTEON 10		✓		✓	
Pensando DSC-100	✓	✓		✓	✓
⋮					

Algorithms with Accelerator Support

Network Functions and Associated Algorithms

Network Function program	Algorithms used				
	Hash	AES	RSA	IPSec	LPM
L3 router	✓				✓
Stateful firewall	✓				
VPN gateway	✓	✓	✓	✓	
Intrusion Detection System					
5G network functions	✓			✓	✓
Distributed data stores	✓	✓			

Network Functions and Associated Algorithms



Difficulties in Mapping

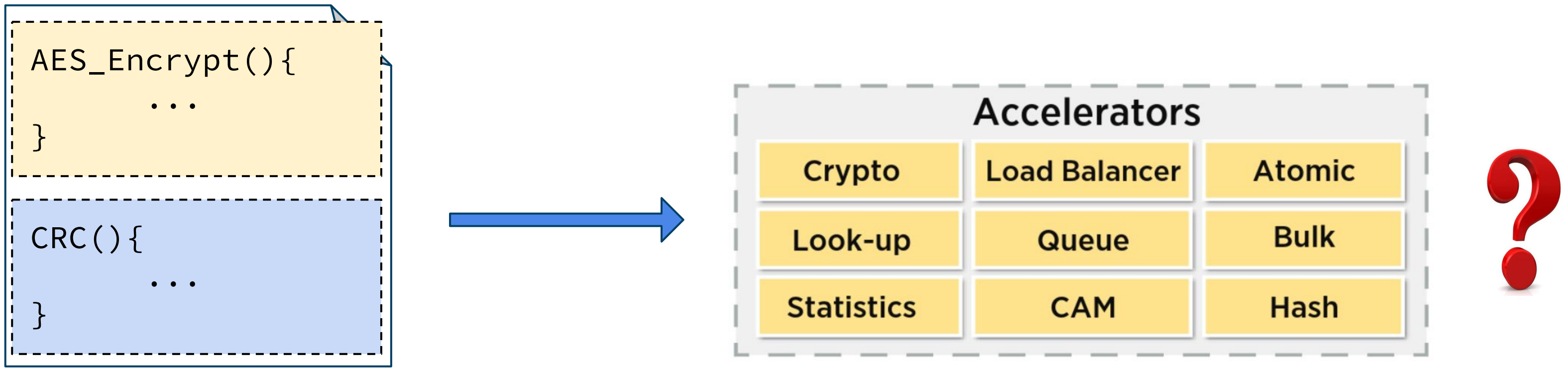
- Identification of regions of code suitable for accelerators
  - Same algorithm can be implemented in multiple ways
- Porting to SmartNICs needs analysis and multiple rounds of manual tuning
  - Tune program by utilizing SmartNIC accelerators

Identifying + Mapping NF to SmartNIC is a tedious and laborious process

Can this process be simplified?

Problem Statement

- Need a workflow to simplify the cross-platform porting process
- Automatic identification of regions in Network Functions



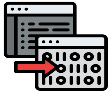
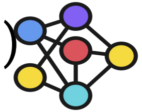
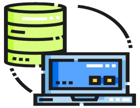
Network Function

Accelerators Figure Source: Netronome NFP-4000 Flow Processor Product Brief

Approach

<u>Our view:</u> This is a ML classification problem.

<u>Our approach:</u>

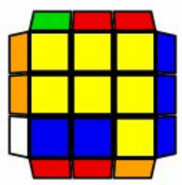
- **Use Compilers** to aid developers to map NF program to SmartNIC 
- **Use ML** to identify code regions performing a specific task (algorithm) 
- **Create realistic dataset** of packet processing algorithms 

Why ML?

- **Undecidability**
  - It is hard to identify algorithms in a program

- **Laborious**
  - Manually assigning accelerators for functions in a large NF program is tedious

- **Scale of variation**
  - Diverse algorithms and SmartNIC architectures



Challenges

1. **Represent** algorithms and programs as input to ML model

2. **Create dataset** of packet processing algorithms
    - Realistic
    - Diverse
    - Wide range of applicability

Challenge #1: Representations of Programs

Background: LLVM IR

LLVM IR: LLVM IR is the Intermediate Representation (IR) of the LLVM compiler toolchain.

C → Front-end → **LLVM IR**

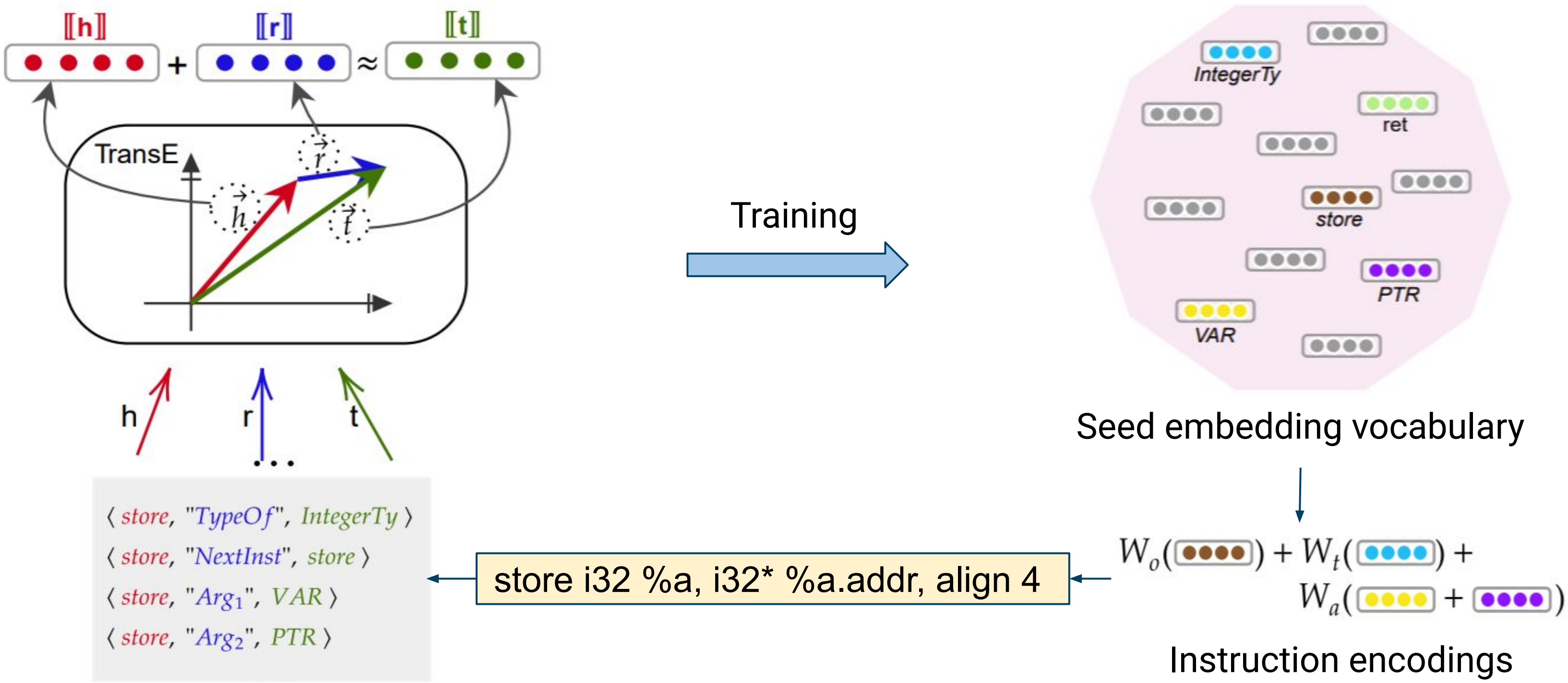
```c
int sum(int a, int b)
{
    return (a+b);
}
```

```llvm
define i32 @sum(i32 %a, i32 %b) #0 {
entry:
 %a.addr = alloca i32, align 4
 %b.addr = alloca i32, align 4
 store i32 %a, i32* %a.addr, align 4
 store i32 %b, i32* %b.addr, align 4
 %0 = load i32, i32* %a.addr, align 4
 %1 = load i32, i32* %b.addr, align 4
 %add = add nsw i32 %0, %1
 ret i32 %add
}
```

Background: Program Representations

Various techniques in use	Information captured
Collecting **features** using domain expertise	Specific task (Domain expertise)
Programs as **tokens** of natural languages	Syntactic
Abstract Syntax Tree representations	Syntactic + Limited Semantic
IR-based representations	Syntactic + Semantic + Generalized

Background: IR2Vec: IR based Program Embeddings

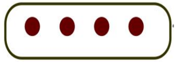


IR2Vec: LLVM IR based Scalable Program Embeddings: S. VenkataKeerthy, Rohit Aggarwal, Shalini Jain, Maunendra Sankar Desarkar, Ramakrishna Upadrasta, Y. N. Srikant (ACM Transactions on Architecture and Code Optimization 2020)

Proposed Methodology

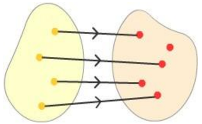
- Identify appropriate accelerators for the program
  - Use ML based techniques

- Utilize IR2Vec embeddings

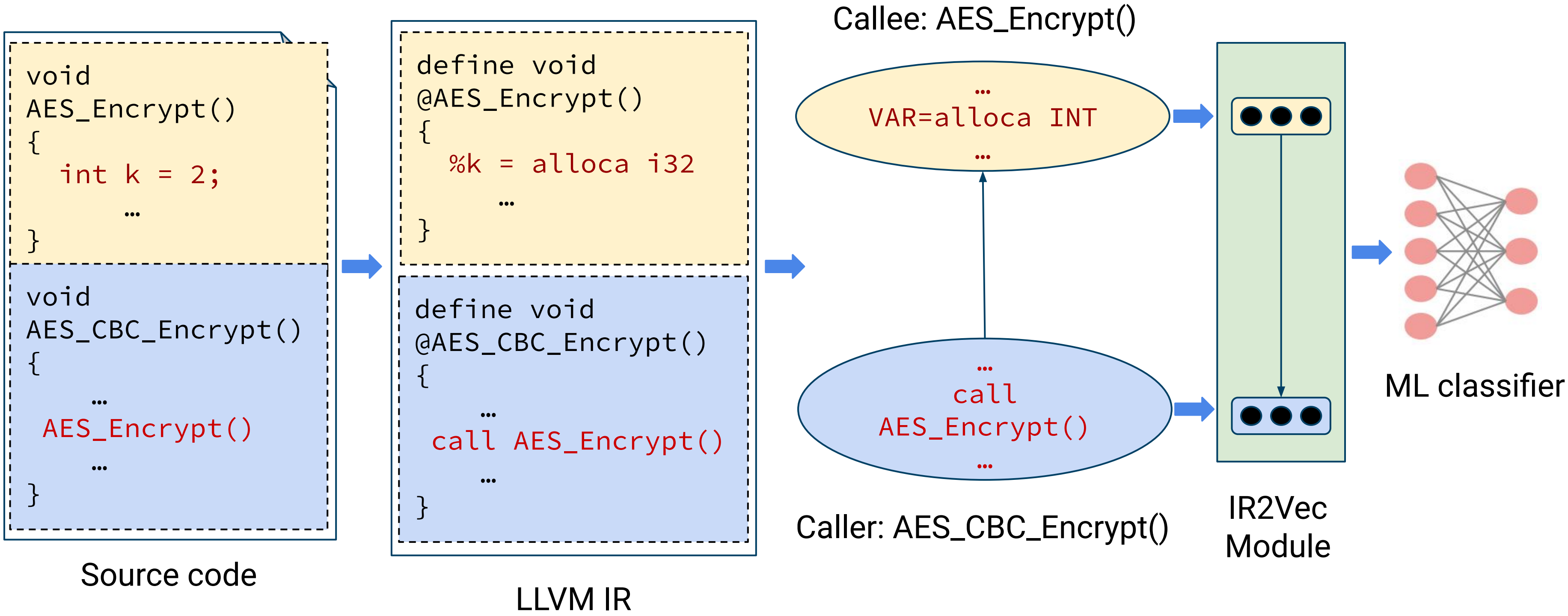


  - Encodes syntactic and semantic information of the program

- Predict accelerator label for each function



Proposed Methodology (contd.)



Challenge #2: Generation of Dataset

Dataset Creation

- ML needs more data
  - ImageNet - 14 million images
  - COCO - 330K images

- Lack of availability of sufficient real world NF programs
  - Earlier datasets [Clara]: only around 7.5k programs

- Need to create a custom dataset
  - using programs from NF domain

Clara: Automated SmartNIC Offloading Insights for Network Functions: Yiming Qiu, Jiarong Xing, Kuo-Feng Hsu, Qiao Kang, Ming Liu, Srinivas Narayana, Ang Chen (ACM Symposium on Operating Systems Principles 2021)

Initial Steps: Seed Dataset Collection

Seed dataset: Collected functions for algorithms used in cryptography libraries

Library / Algorithm	OpenSSL (v1.1)	OpenSSL (v3)	CryptoPP (v8.6)	Botan (v2.19)	Nettle (v3.7)	WolfCrypt (v5.1)	MbedTLS (v3.1)	**Total**
AES	8	7	8	6	2	6	5	**42**
DES	13	13	8	4	4	2	4	**48**
RSA	5	7	4	5	0	7	3	**31**
Total	**26**	**27**	**20**	**15**	**6**	**15**	**12**	**121**

Dataset Expansion using Compiler Transformations

Method: Apply compiler transformations on original (seed) NF programs

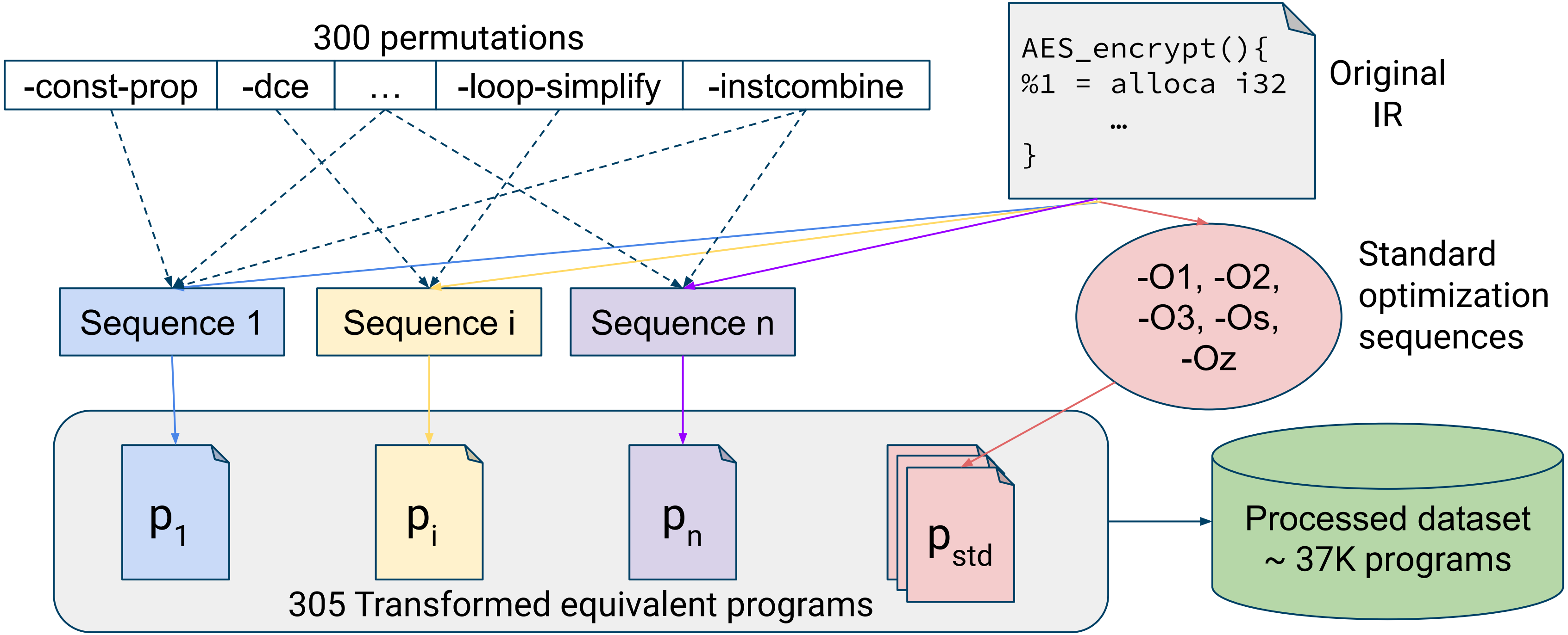
- Adds diversity to dataset
    - Code size of the program
    - Latency
    - Throughput
    - Power usage

- Semantics of original codes are preserved

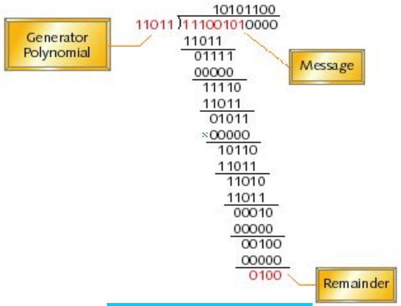
Result: Produces sufficient data for training a ML model

Dataset Expansion using Compiler Transformations

- Apply random permutations of LLVM transformations to programs

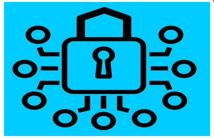


Experimentation & Implementation details



Experimentation

- **Detection of CRC algorithm**
  - Classifying CRC and non-CRC programs
- **Detection of cryptography algorithms**
  - CRC, AES, DES/3DES, RSA, non-NF programs



Implementation

- Manually labelled functions for classification
- Used IR2Vec embeddings of programs compiled to LLVM IR (v12.0)
- Compared results from our approach with Clara

Clara: Automated SmartNIC Offloading Insights for Network Functions: Yiming Qiu, Jiarong Xing, Kuo-Feng Hsu, Qiao Kang, Ming Liu, Srinivas Narayana, Ang Chen (ACM Symposium on Operating Systems Principles 2021)

Results: Precision

Model	CRC		CRC + Cryptography	
	Clara	IR2Vec	Clara	IR2Vec
Gradient Boosted Decision Tree	0.992	0.974	0.664	0.949
Decision Tree	0.983	0.994	0.661	0.969
Multi-layer Perceptron	0.983	0.997	0.666	0.959
Support Vector Machine	0.992	0.999	0.646	0.898
k-Nearest Neighbour	0.980	0.999	0.596	0.976
AutoML	0.980	0.999	0.661	0.979

Results: Recall

Model	CRC		CRC + Cryptography	
	Clara	IR2Vec	Clara	IR2Vec
Gradient Boosted Decision Tree	0.435	0.997	0.594	0.947
Decision Tree	0.488	0.995	0.622	0.968
Multi-layer Perceptron	0.437	0.999	0.590	0.958
Support Vector Machine	0.484	0.995	0.604	0.894
k-Nearest Neighbour	0.486	0.999	0.630	0.974
AutoML	0.487	0.999	0.621	0.978

Results: F1 Score

Model	CRC		CRC + Cryptography	
	Clara	IR2Vec	Clara	IR2Vec
Gradient Boosted Decision Tree	0.605	0.985	0.627	0.948
Decision Tree	0.652	0.994	0.641	0.968
Multi-layer Perceptron	0.605	0.998	0.626	0.958
Support Vector Machine	0.651	0.997	0.624	0.896
k-Nearest Neighbour	0.650	0.999	0.613	0.975
AutoML	0.651	0.999	0.640	0.978

IR2Vec can capture semantics of the algorithm

Summary & Future Work

Contributions

- Using embedding techniques (IR2Vec) to represent programs from network domain
- Modeling algorithm identification problem with a scalable ML approach
- Realistic dataset collection and generation of semantically equivalent programs

Future Work

- Applying to real-world network functions
- Identifying other algorithms

Authors

S. VenkataKeerthy
PhD student, IITH

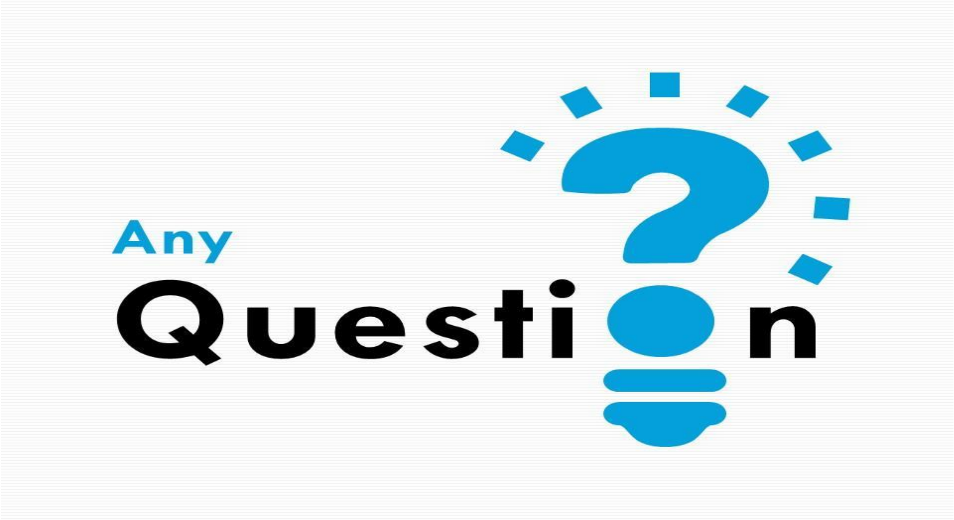
Yashas Andaluri
MTech student, IITH

Sayan Dey
MTech student, IITH

Rinku Shah
Asst. Professor, IIITD

Praveen Tammana
Asst. Professor, IITH

Ramakrishna Upadrasta
Asst. Professor, IITH


Any
Question