AutoFHE: **Auto**mated Adaption of CNNs for Efficient Evaluation over **FHE**

WEI AO

Vishnu Boddeti

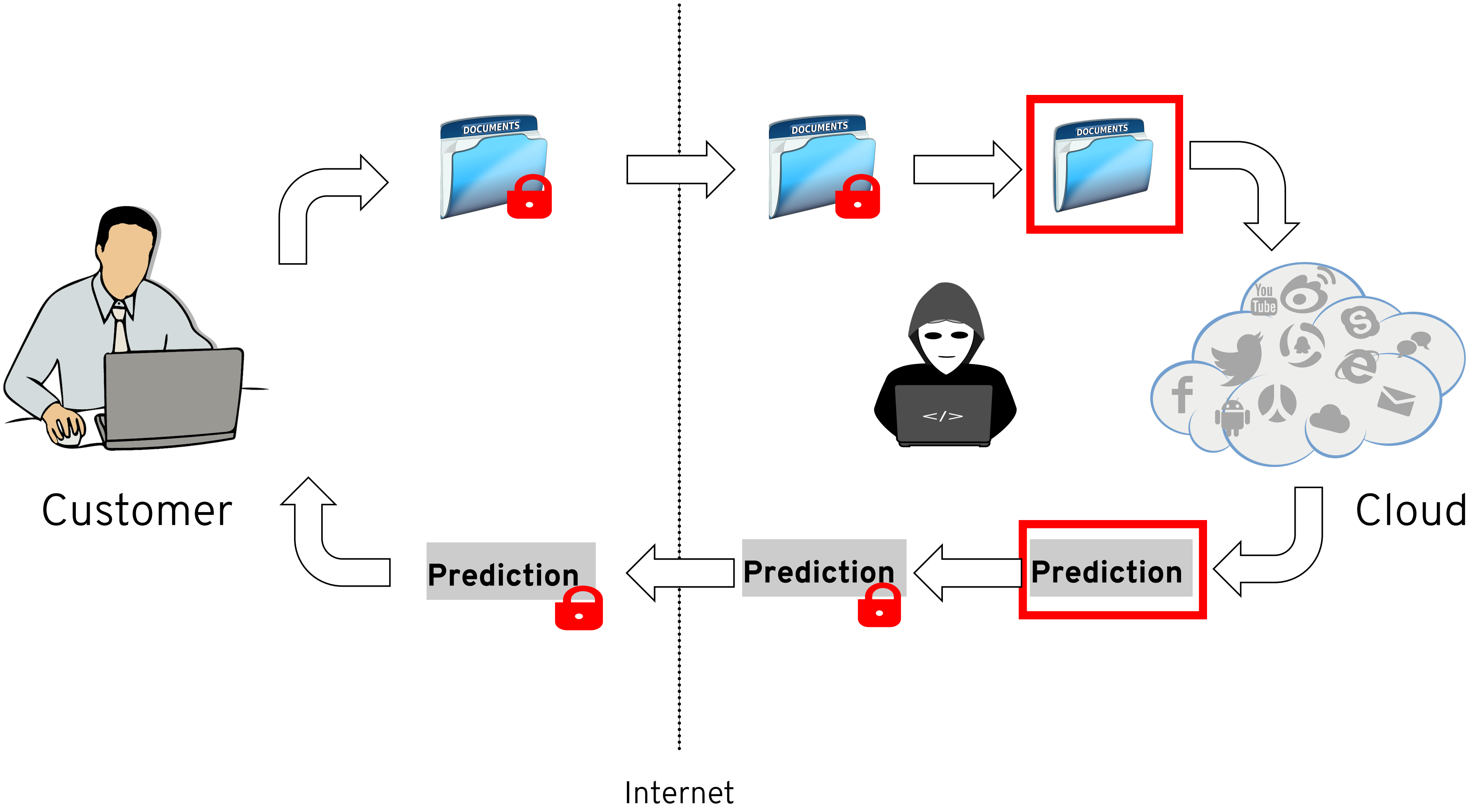
wei-ao.github.io

Michigan State University

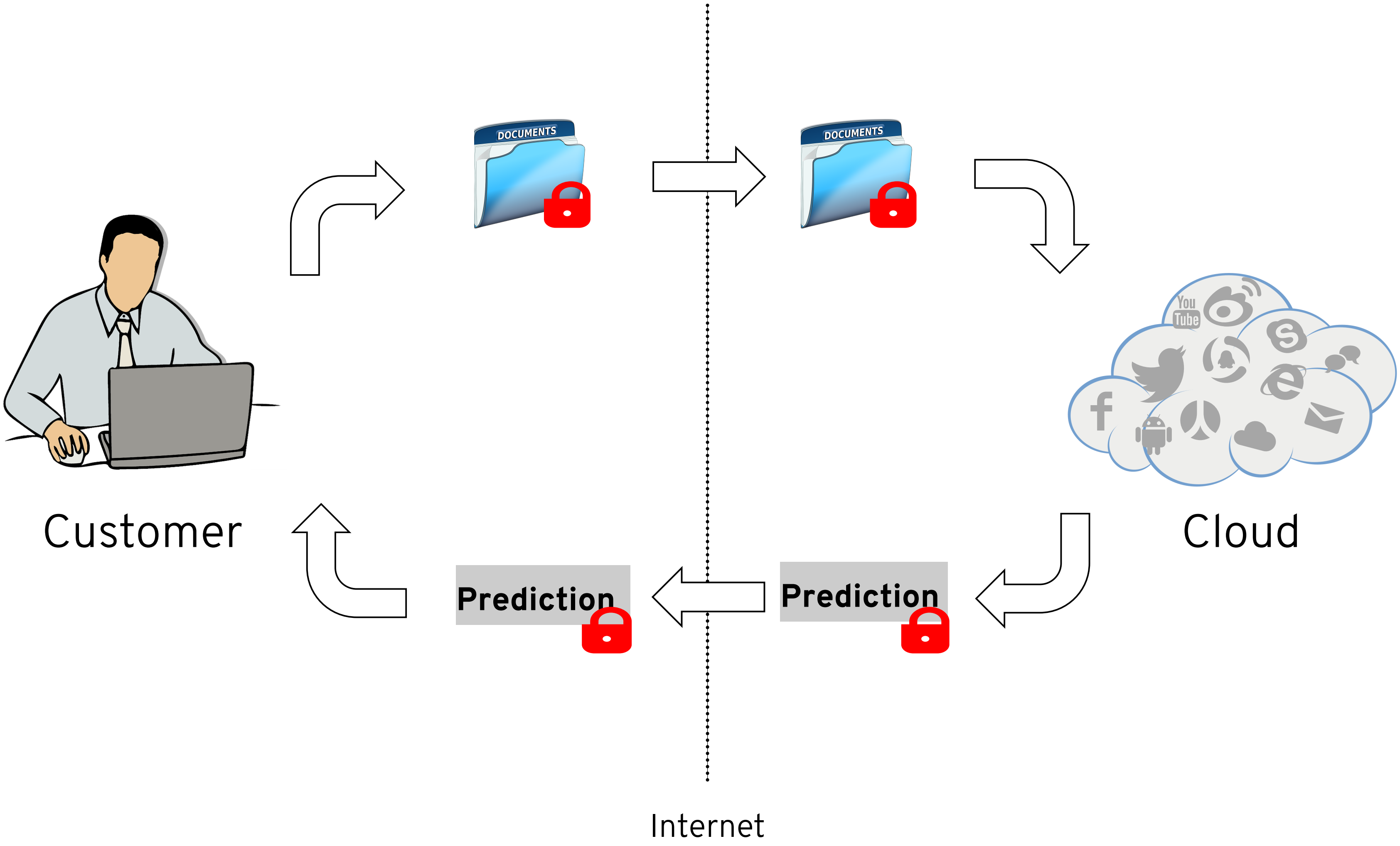
Philadelphia, PA, USA 2024

Secure deep learning under fully homomorprhic encryption

Deep Learning as a **Service (DLaaS)**



Secure DLaaS under **Fully Homomorphic Encryption (FHE)**



From **Secure Computation** to **Secure Deep Learning**

1. **Initiative:** privacy homomorphisms, 1978

2. **Gentry's Blueprint** uses Ideal lattices, 2009

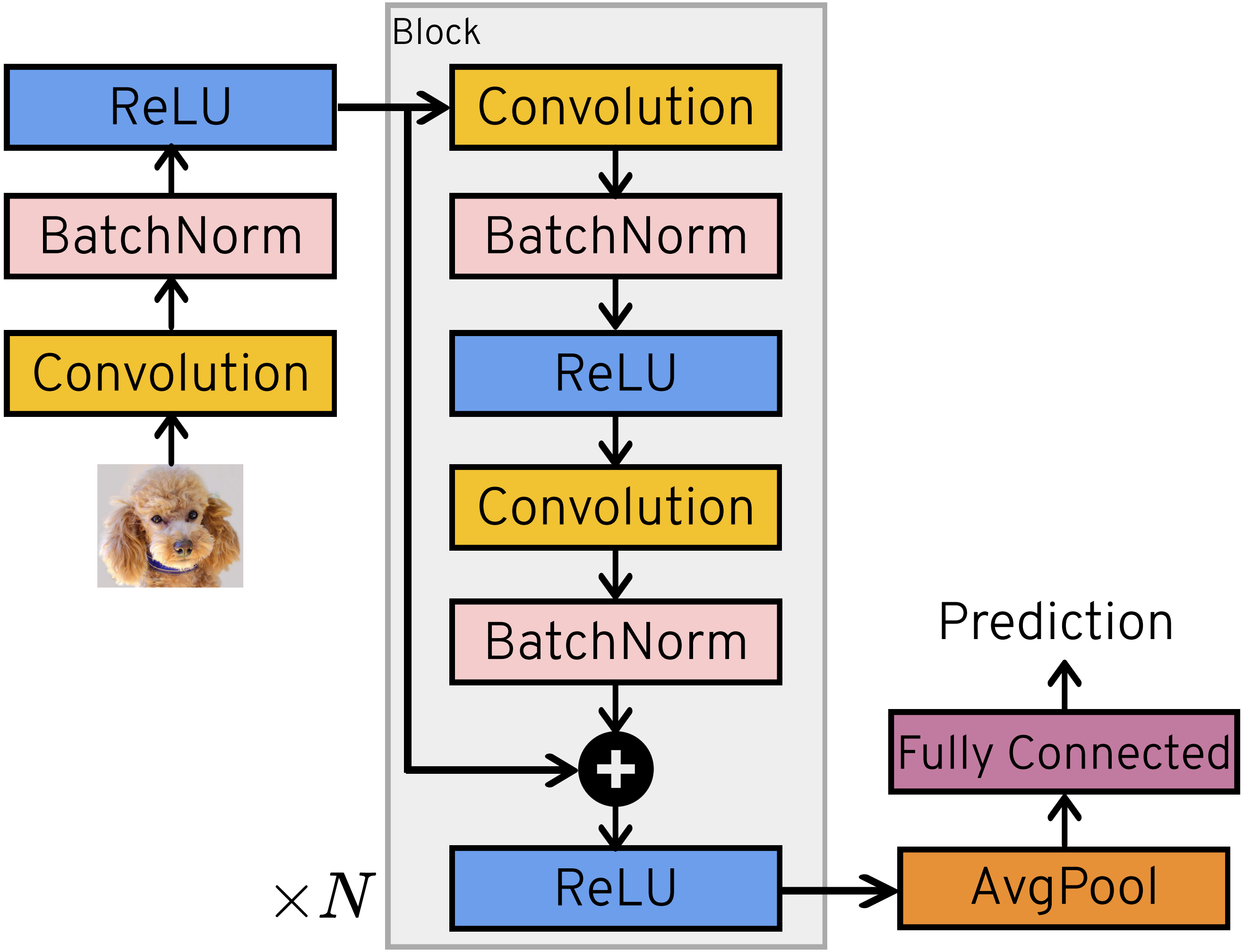
3. **Approximate Arithmetic** under leveled homomorphic encryption: CKKS (HEAAN), ASIACRYPT 2017

4. **Bootstrapping** for CKKS: fully homomorphic encryption, EUROCRYPT 2018

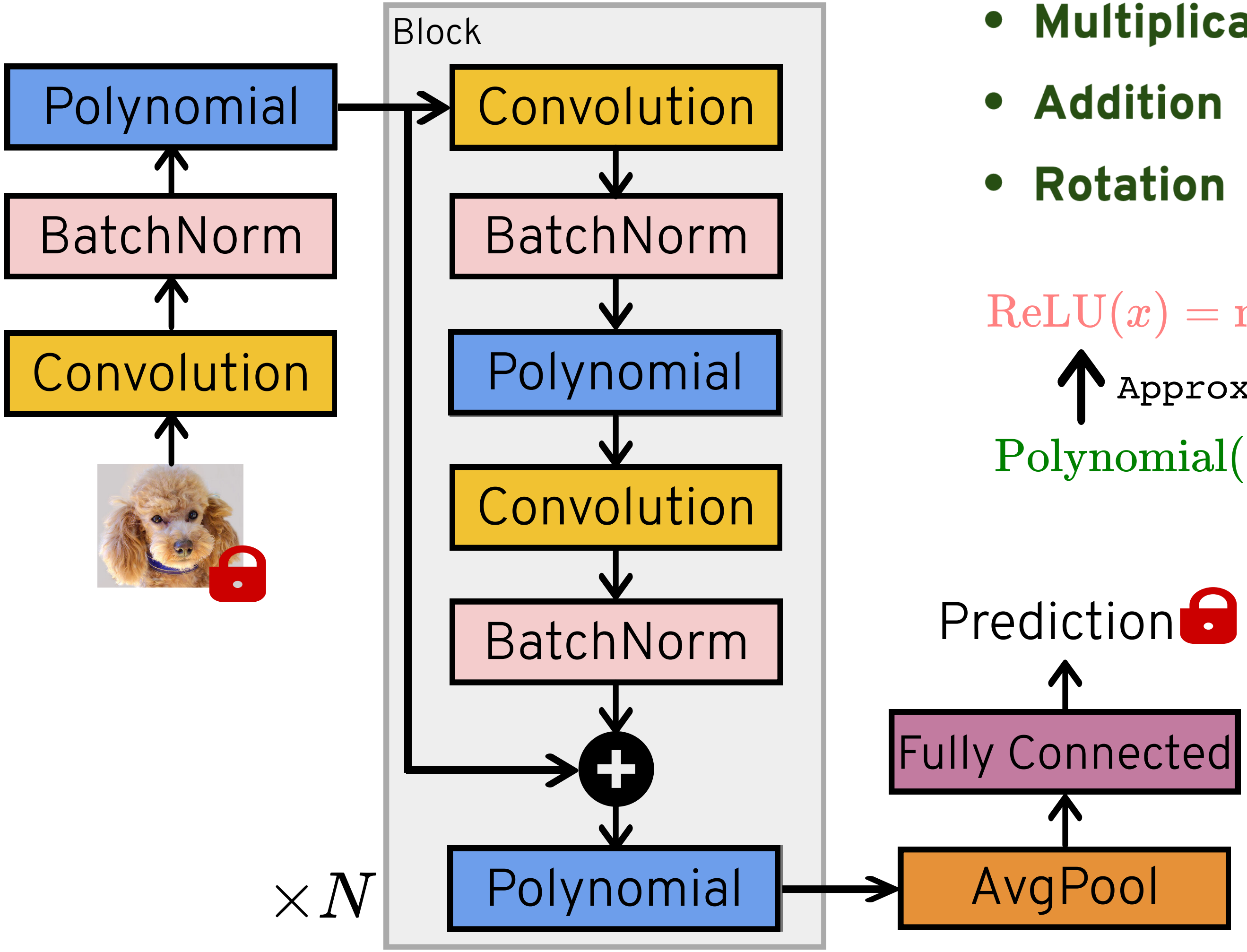
5. **MPCNN:** Deep CNNs on RNS-CKKS FHE scheme, ICML 2022

6. **AutoFHE: Auto**mated Adaption of CNNs for Efficient Evaluation over **FHE**, USENIX 2024

Convolutional Neural Networks (CNNs)



CNNs under Homomorphic Encryption (HE)



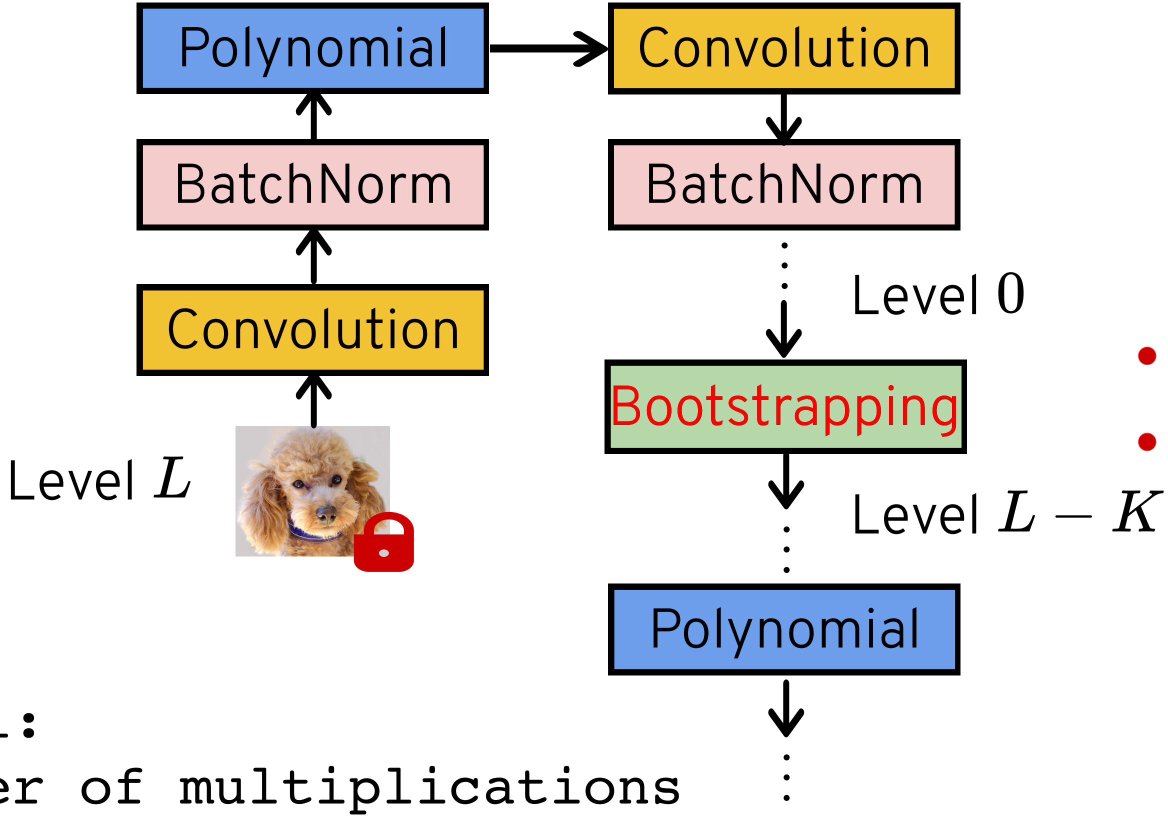
- **Multiplication**
- **Addition**
- **Rotation**

$$\mathrm{ReLU}(x) = \max(x, 0)$$

Approximate

$$\mathrm{Polynomial}(x) = \sum_i a_i x^i$$

Deep CNNs under **Fully Homomorphic Encryption (FHE)**



- **High Latency**
- **High Memory Footprint**

```
Level:
number of multiplications
allowed to evaluate
```

Deep CNNs under **Fully Homomorphic Encryption (FHE)**

- Security Requirement
- Inference Latency
- Prediction Accuracy

Deep CNNs under **Fully Homomorphic Encryption (FHE)**

- Security Requirement
- Inference Latency
- Prediction Accuracy

Cryptographic Parameters

Cyclotomic polynomial degree	N
Level	L
Modulus	$Q_\ell = \prod_{i=0}^{\ell} q_\ell, 0 \leq \ell \leq L$
Bootstrapping depth	K
Hamming weight	h

Deep CNNs under **Fully Homomorphic Encryption (FHE)**

- Security Requirement
- Inference Latency
- Prediction Accuracy

Cryptographic Parameters

Cyclotomic polynomial degree	N
Level	L
Modulus	$Q_\ell = \prod_{i=0}^{\ell} q_\ell, 0 \leq \ell \leq L$
Bootstrapping depth	K
Hamming weight	h

Polynomial CNNS

Conv, BN, pooling, FC layers: packing

Polynomials: degree -> depth

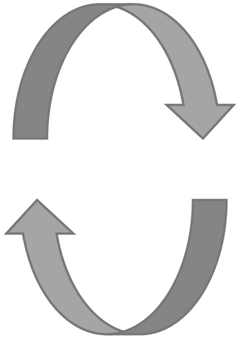
Number of layers: ResNet20, ResNet32

Input image resolution

Channels/kernels

Deep CNNs under **Fully Homomorphic Encryption (FHE)**

- Security Requirement
- Inference Latency
- Prediction Accuracy



Cryptographic Parameters

Cyclotomic polynomial degree	N
Level	L
Modulus	$Q_\ell = \prod_{i=0}^{\ell} q_\ell, 0 \leq \ell \leq L$
Bootstrapping depth	K
Hamming weight	h

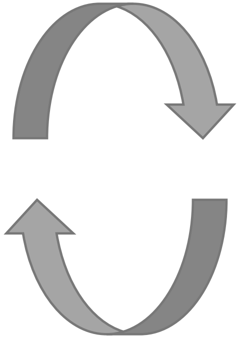
Polynomial CNNS

- Conv, BN, pooling, FC layers: packing
- Polynomials: degree -> depth
- Number of layers: ResNet20, ResNet32
- Input image resolution
- Channels/kernels

Hand-crafted Design of **Polynomial** for **CNNs** under **FHE**

Cryptographic Parameters

$$N, L, Q_\ell = \prod_{i=0}^{\ell} q_\ell (0 \le \ell \le L), K, h$$



Polynomial CNNS

`Polynomials: degree -> depth`

MPCNN [1]:

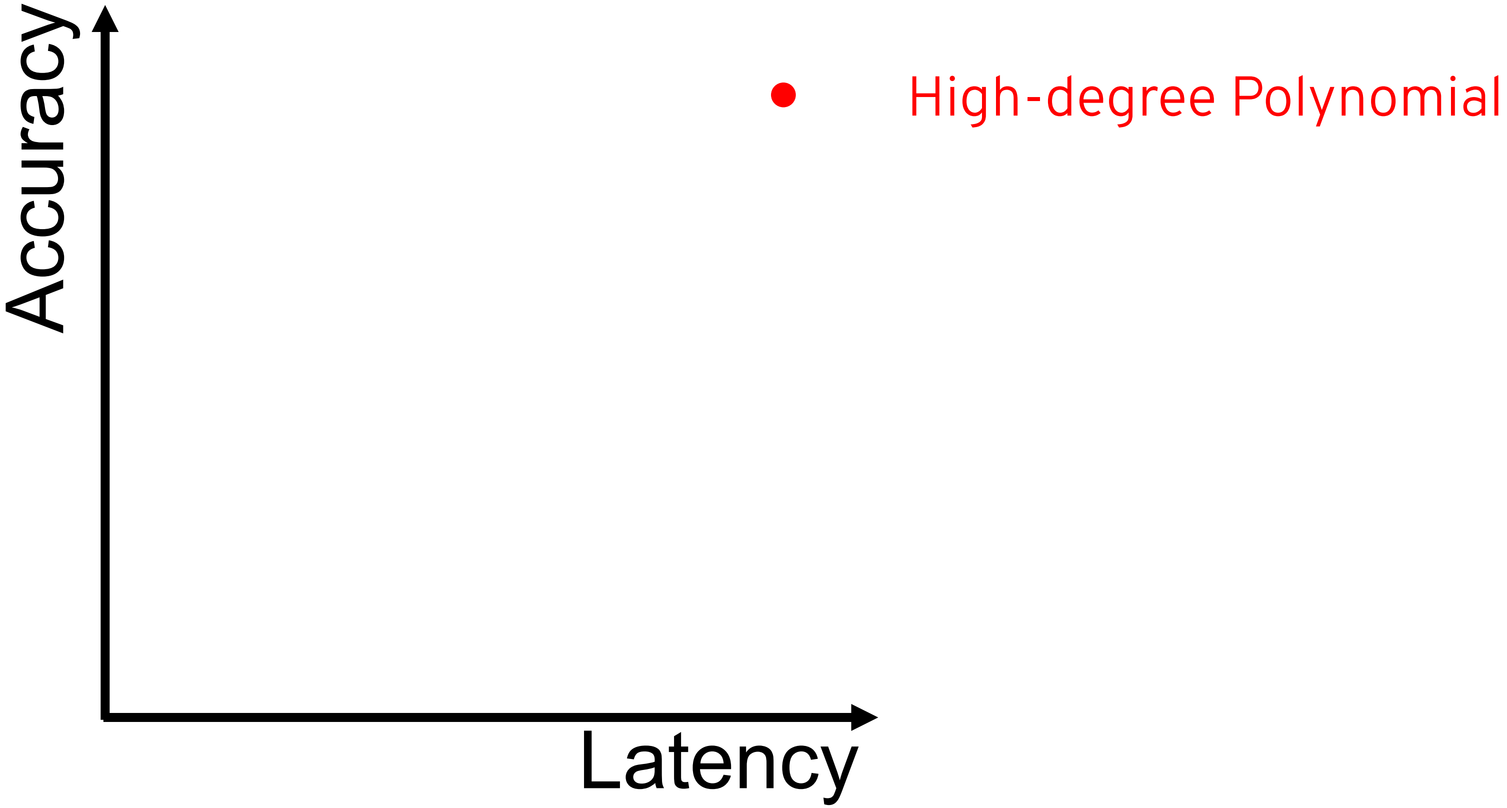
→ Convolution → BatchNorm → Bootstrapping → High-degree Polynomial →

Level 2 Level 0 Level 0 Level 16 Level 2

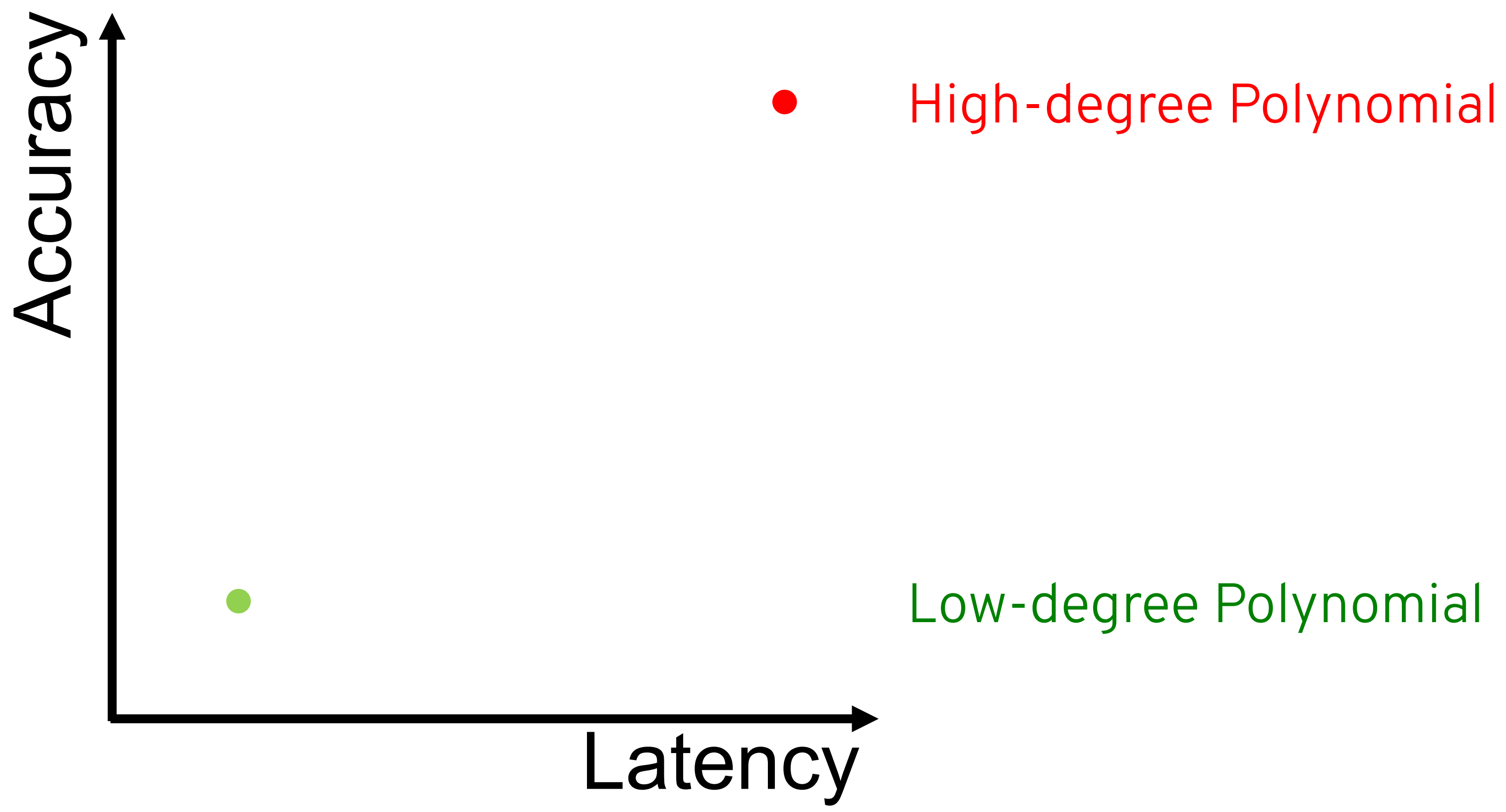
- **Fixed Architecture**
- **Periodic Bootstrapping**

[1] Lee, Eunsang, et al. "Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel convolutions." *International Conference on Machine Learning*. PMLR, 2022.

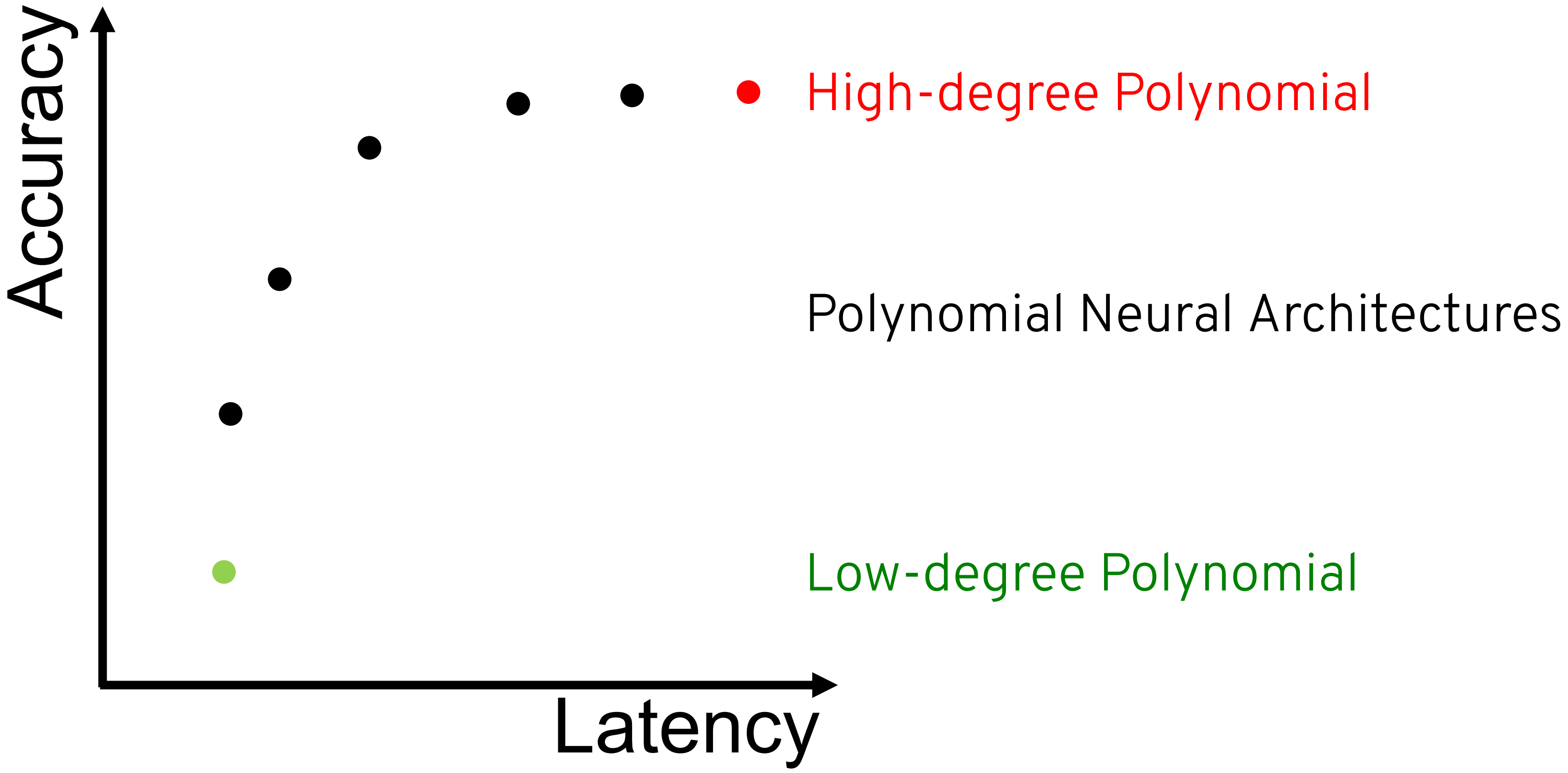
Hand-crafted Design of **Polynomial** for **CNNs** under **FHE**



Hand-crafted Design of **Polynomial** for **CNNs** under **FHE**



Hand-crafted Design of **Polynomial** for **CNNs** under **FHE**



How to obtain all possible polynomial neural architectures?

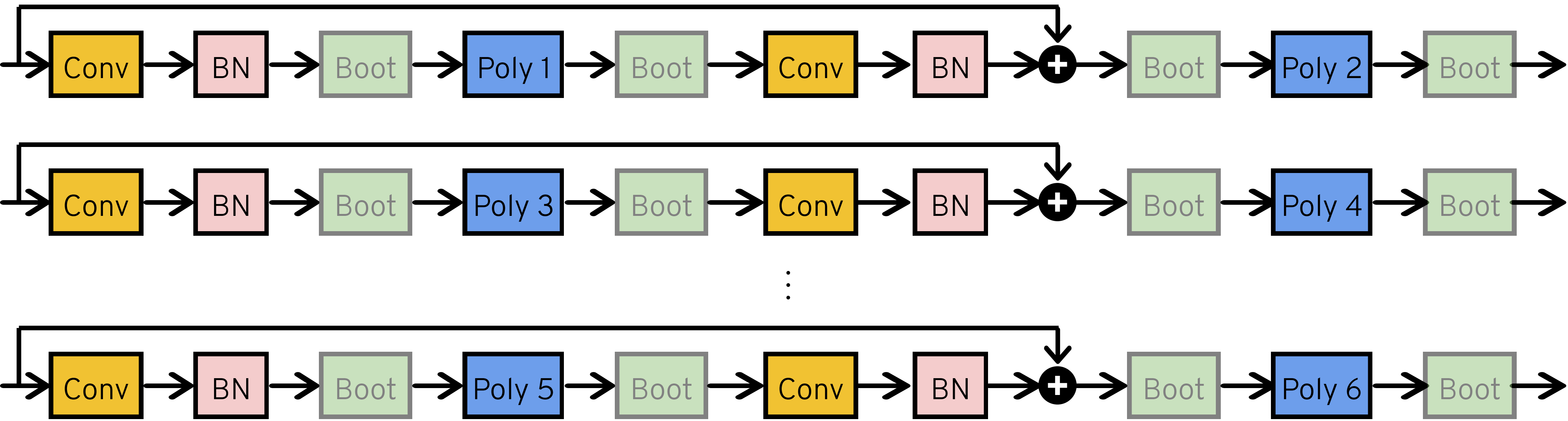
Key Insight

Optimize the

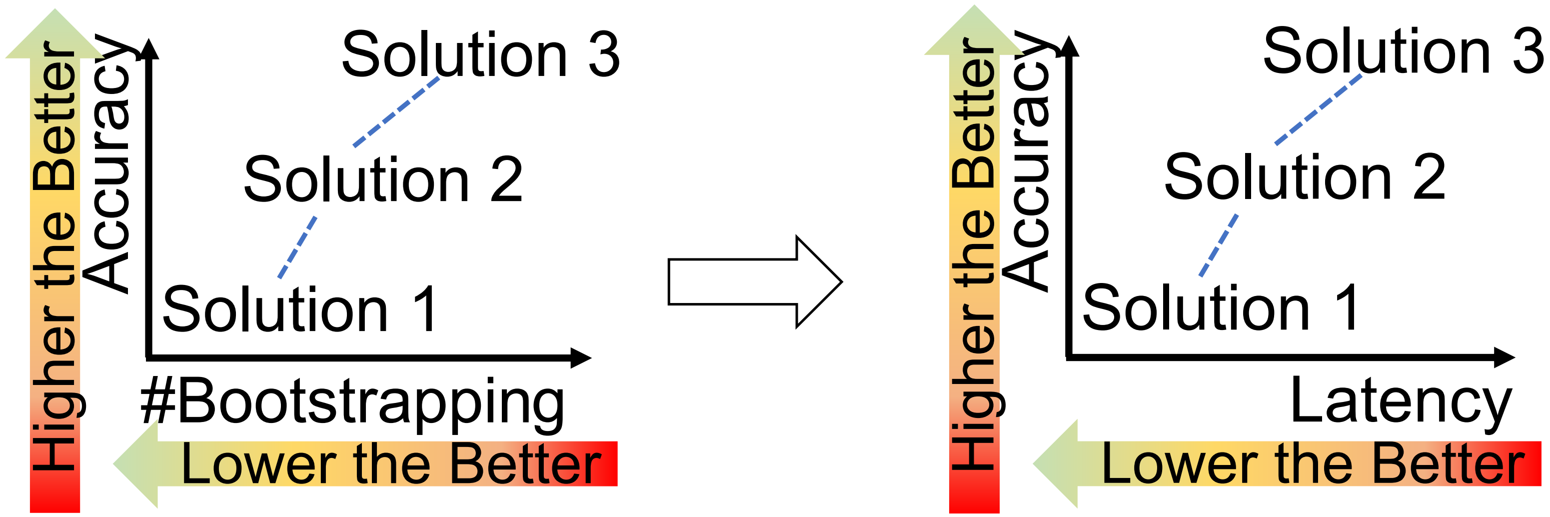
end-to-end polynomial neural architecture

instead of the polynomial function

Optimization of End-to-End Polynomial Neural Architecture



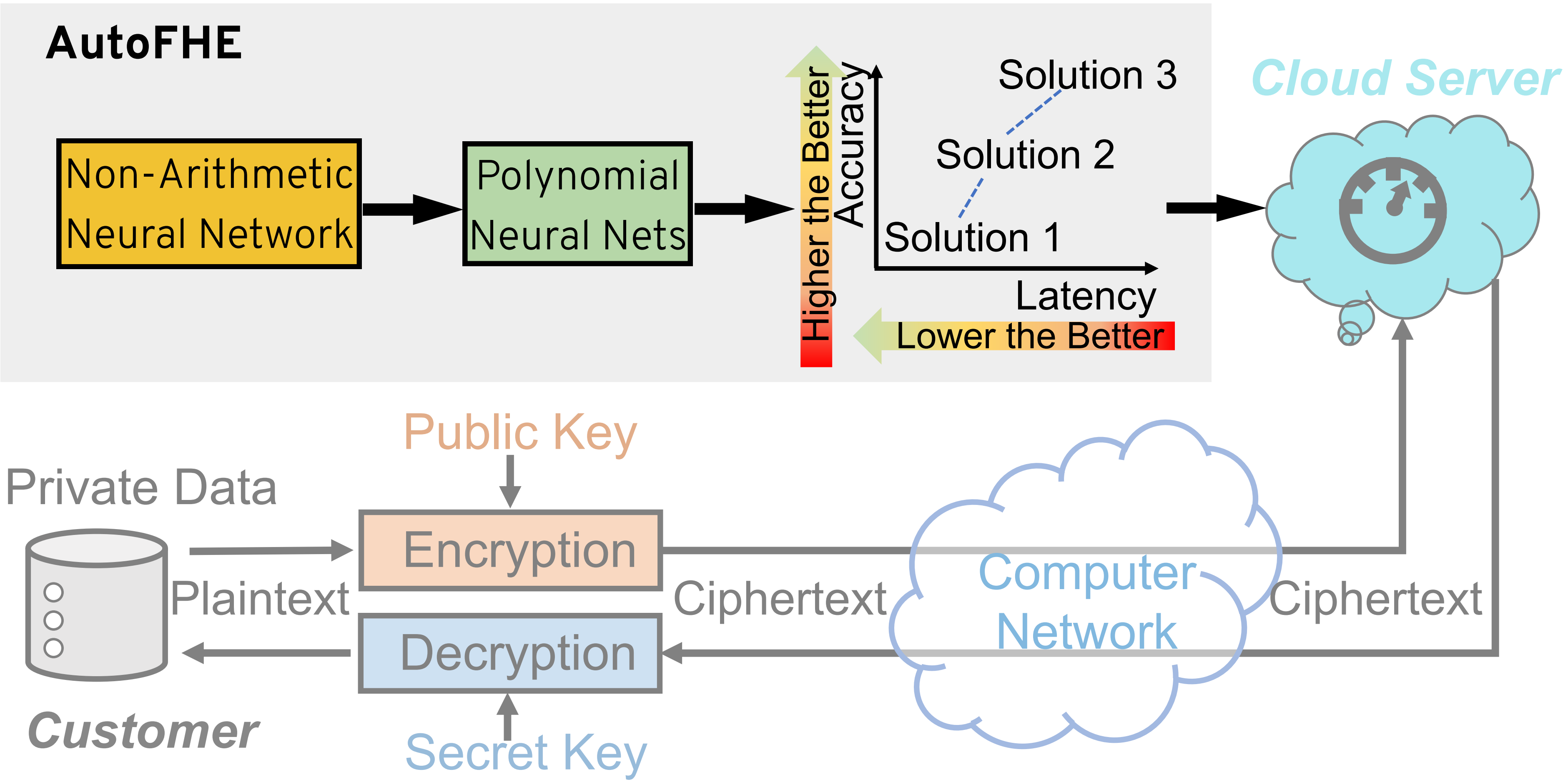
Optimization of **End-to-End Polynomial Neural Architecture**



To meet different requirements in real world

- I want a faster response
- I can wait for an accurate result

AutoFHE: Automated Adaption of CNNs under FHE



EvoReLU: Evolutionary Mixed-Degree Polynomial Approximation of ReLU

Forward Propagation

$$\mathrm{EvoReLU}(x) = \begin{cases} x, & d = 1 \\ \alpha_2 x^2 + \alpha_1 x + \alpha_0, & d = 2 \\ x \cdot (\mathcal{F}(x) + 0.5)\,, & d > 2 \end{cases}$$

High-degree composite polynomial [2]:

$$\mathcal{F}(x) = (f_K^{d_K} \circ \cdots \circ f_k^{d_k} \circ \cdots \circ f_1^{d_1})(x), 1 \leq k \leq K$$

⬅ Differentiable Evolution

- **Pruning**: DeepReDuce, SAFENet, Delphi
- **Quadratic**: LoLa, CryptoNets, HEMET
- **High-degree approximation**: MPCNN

[2] Lee, Eunsang, Joon-Woo Lee, Jong-Seon No, and Young-Sik Kim. "Minimax approximation of sign function by composite polynomial for homomorphic comparison." *IEEE Transactions on Dependable and Secure Computing* 19, no. 6 (2021): 3711-3727.

EvoReLU: Evolutionary Mixed-Degree Polynomial Approximation of ReLU

Backward Propagation

$$\frac{\partial \mathrm{EvoReLU}(x)}{\partial x} = \begin{cases} 1, & d = 1 \\ 2\alpha_2 x + \alpha_1, & d = 2 \\ \partial \mathrm{ReLU}(x)/\partial x, & d > 2 \end{cases}$$

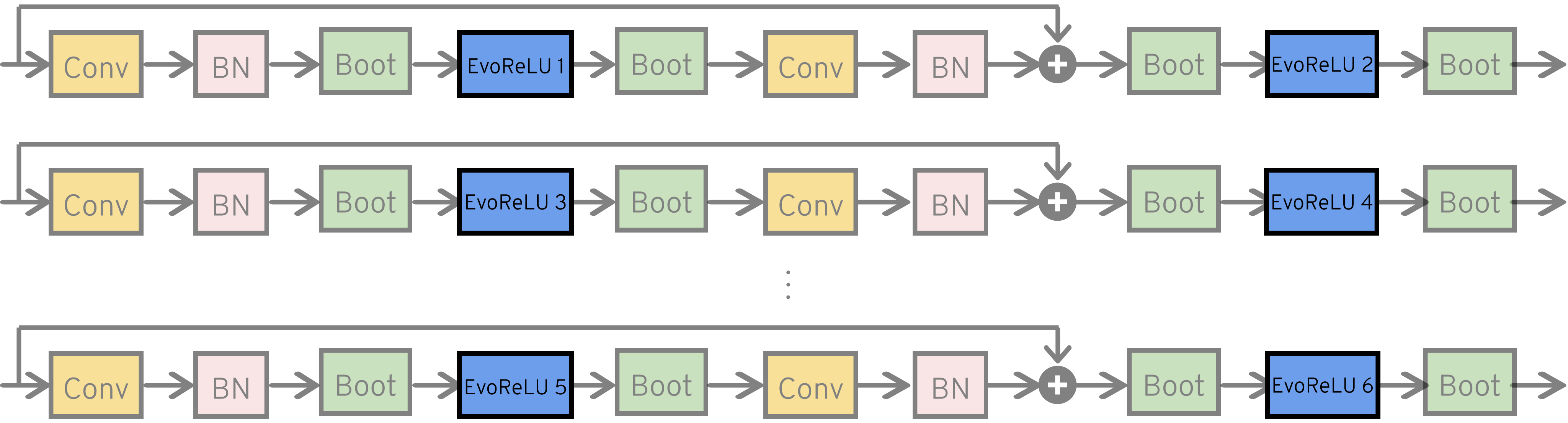
- Gradient
- Gradient
- Straight-through estimated gradient

- **Make training more stable**

How to **optimize** end-to-end polynomial neural architecture?

Multi-Objective evolutionary optimization

Joint Search for **Layerwise EvoReLU** and **Bootstrapping Operations**



Joint search problem → Multi-objective optimization →

- **Flexible Architecture**
- **On-demand Bootstrapping**

Multi-Objective Optimization

Single Objective

- Accuracy
- Latency

Scalarization of Multiple Objectives

$$\alpha \cdot \text{Accuracy} + \beta \cdot \text{Latency}$$

- Only generate a single solution
- Hard to tune balancing weights
- Not Pareto optimal

Multi-Objective Optimization

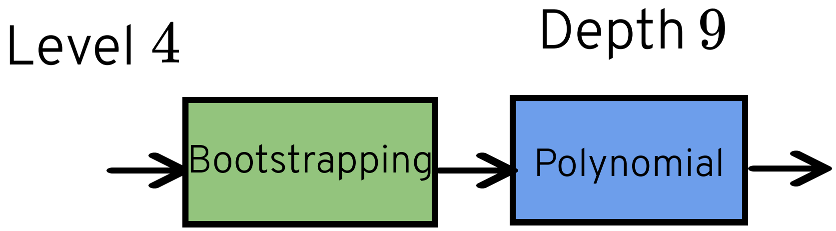
$$\min \{1 - \text{Accuracy}, \#\text{Bootstrapping}\}$$

- Multiple solutions on the Pareto front
- No need to tune weights
- Pareto optimal

Multi-Objective Optimization

Multi-Objective Optimization

$$\min \{1 - \text{Accuracy}, \text{Depth of polys}\}$$



Drop 4 Levels

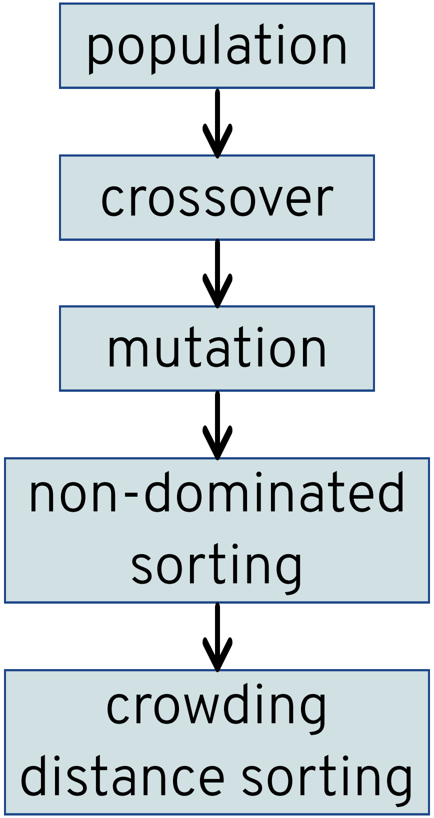
- Not necessarily reduce bootstrapping operations

Multi-Objective Optimization

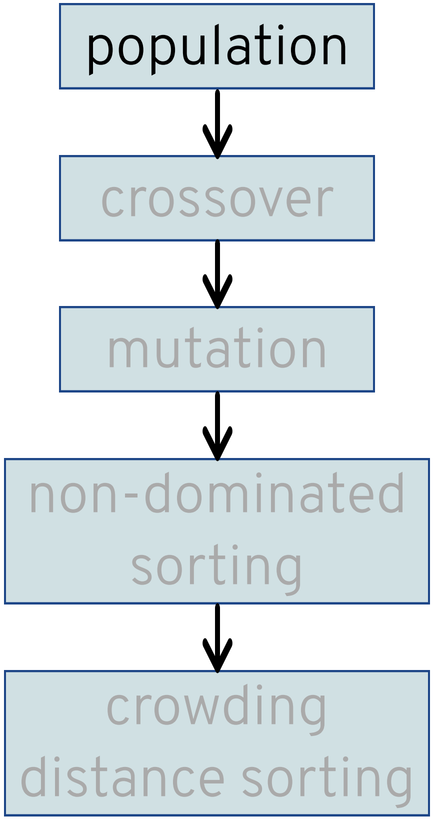
$$\min \{1 - \text{Accuracy}, \#\text{Bootstrapping}\}$$

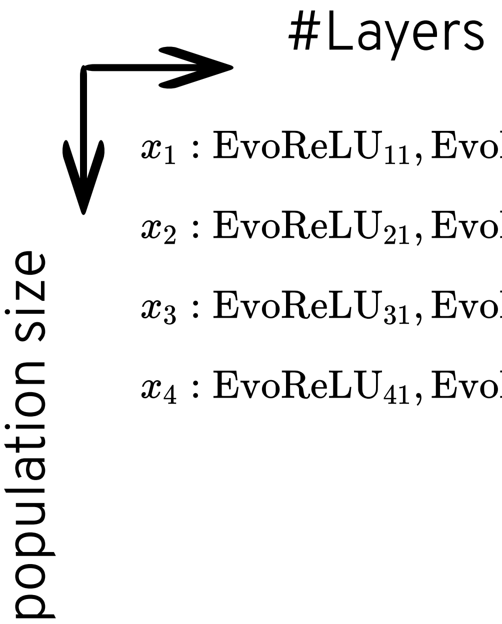
- Directly reduce bootstrapping operations

Evolutionary Multi-Objective Optimization



Evolutionary Multi-Objective Optimization





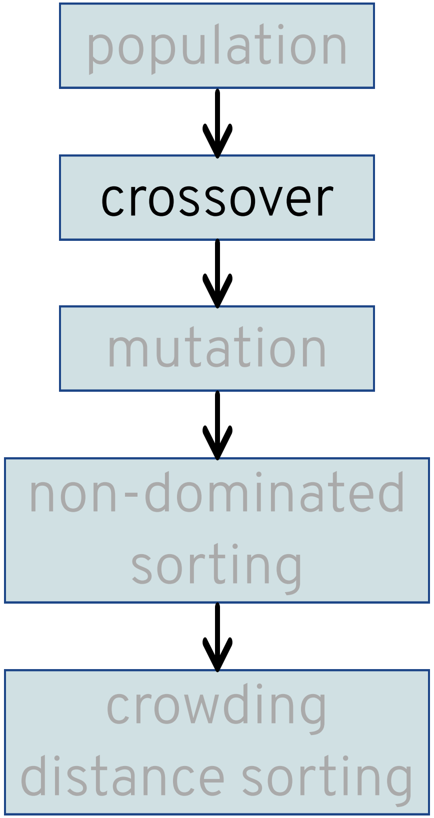
$$x_1 : \mathrm{EvoReLU}_{11}, \mathrm{EvoReLU}_{12}, \mathrm{EvoReLU}_{13}, \mathrm{EvoReLU}_{14}, \cdots$$

$$x_2 : \mathrm{EvoReLU}_{21}, \mathrm{EvoReLU}_{22}, \mathrm{EvoReLU}_{23}, \mathrm{EvoReLU}_{24}, \cdots$$

$$x_3 : \mathrm{EvoReLU}_{31}, \mathrm{EvoReLU}_{32}, \mathrm{EvoReLU}_{33}, \mathrm{EvoReLU}_{34}, \cdots$$

$$x_4 : \mathrm{EvoReLU}_{41}, \mathrm{EvoReLU}_{42}, \mathrm{EvoReLU}_{43}, \mathrm{EvoReLU}_{44}, \cdots$$

Evolutionary Multi-Objective Optimization



$$x_1 : \boxed{\begin{matrix}\mathrm{EvoReLU}_{11}, \\ \mathrm{EvoReLU}_{21},\end{matrix}} \begin{matrix}\mathrm{EvoReLU}_{12}, \\ \mathrm{EvoReLU}_{22},\end{matrix} \boxed{\begin{matrix}\mathrm{EvoReLU}_{13}, \\ \mathrm{EvoReLU}_{23},\end{matrix}} \begin{matrix}\mathrm{EvoReLU}_{14}, \cdots \\ \mathrm{EvoReLU}_{24}, \cdots\end{matrix} : x_2$$

$$x_1 : \mathrm{EvoReLU}_{11}, \mathrm{EvoReLU}_{12}, \mathrm{EvoReLU}_{13}, \mathrm{EvoReLU}_{14}, \cdots$$

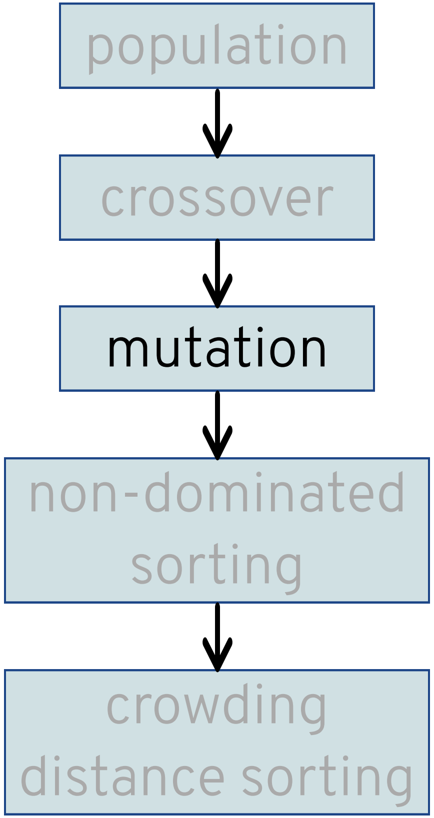
$$x_2 : \mathrm{EvoReLU}_{21}, \mathrm{EvoReLU}_{22}, \mathrm{EvoReLU}_{23}, \mathrm{EvoReLU}_{24}, \cdots$$

$$\Downarrow$$

$$x_1' : \mathrm{EvoReLU}_{21}, \mathrm{EvoReLU}_{12}, \mathrm{EvoReLU}_{23}, \mathrm{EvoReLU}_{14}, \cdots$$

$$x_2' : \mathrm{EvoReLU}_{11}, \mathrm{EvoReLU}_{22}, \mathrm{EvoReLU}_{13}, \mathrm{EvoReLU}_{24}, \cdots$$

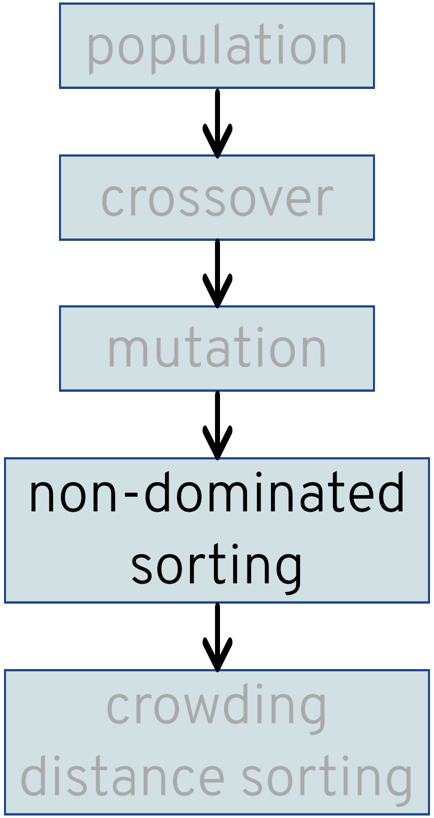
Evolutionary Multi-Objective Optimization

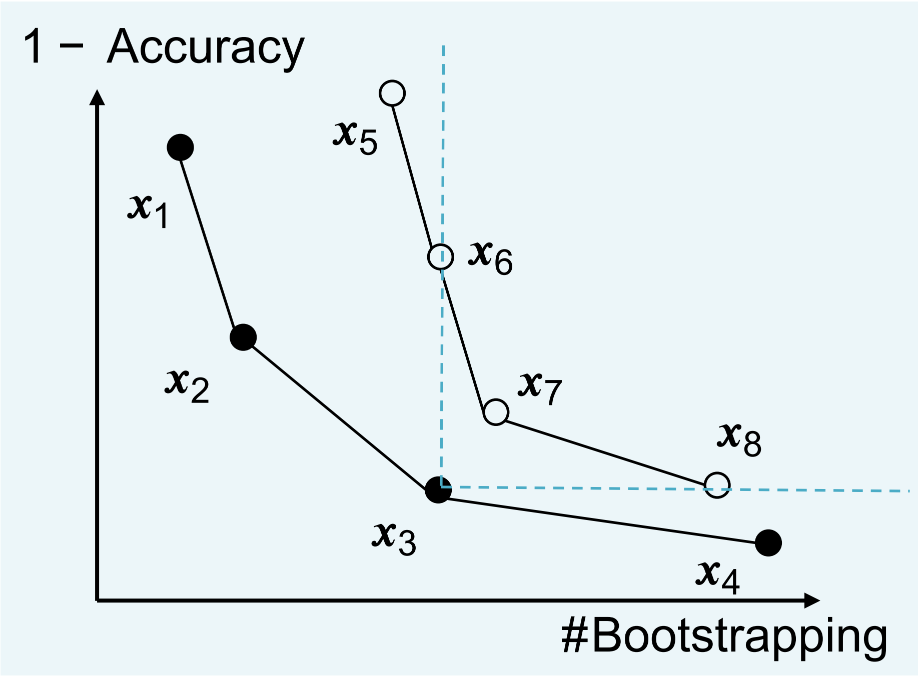


$$
\begin{aligned}
x_1 &: \mathrm{EvoReLU}_{11}, \mathrm{EvoReLU}_{12}, \boxed{\mathrm{EvoReLU}_{13},} \mathrm{EvoReLU}_{14}, \cdots \\
x_2 &: \mathrm{EvoReLU}_{21}, \mathrm{EvoReLU}_{22}, \boxed{\mathrm{EvoReLU}_{23},} \mathrm{EvoReLU}_{24}, \cdots \\
x_3 &: \boxed{\mathrm{EvoReLU}_{31},} \mathrm{EvoReLU}_{32}, \mathrm{EvoReLU}_{33}, \mathrm{EvoReLU}_{34}, \cdots \\
x_4 &: \mathrm{EvoReLU}_{41}, \boxed{\mathrm{EvoReLU}_{42},} \mathrm{EvoReLU}_{43}, \mathrm{EvoReLU}_{44}, \cdots
\end{aligned}
$$

$$
\begin{aligned}
x'_1 &: \mathrm{EvoReLU}_{11}, \mathrm{EvoReLU}_{12}, \mathrm{EvoReLU}'_{13}, \mathrm{EvoReLU}_{14}, \cdots \\
x'_2 &: \mathrm{EvoReLU}_{21}, \mathrm{EvoReLU}_{22}, \mathrm{EvoReLU}'_{23}, \mathrm{EvoReLU}_{24}, \cdots \\
x'_3 &: \mathrm{EvoReLU}'_{31}, \mathrm{EvoReLU}_{32}, \mathrm{EvoReLU}_{33}, \mathrm{EvoReLU}_{34}, \cdots \\
x'_4 &: \mathrm{EvoReLU}_{41}, \mathrm{EvoReLU}'_{42}, \mathrm{EvoReLU}_{43}, \mathrm{EvoReLU}_{44}, \cdots
\end{aligned}
$$

Evolutionary Multi-Objective Optimization



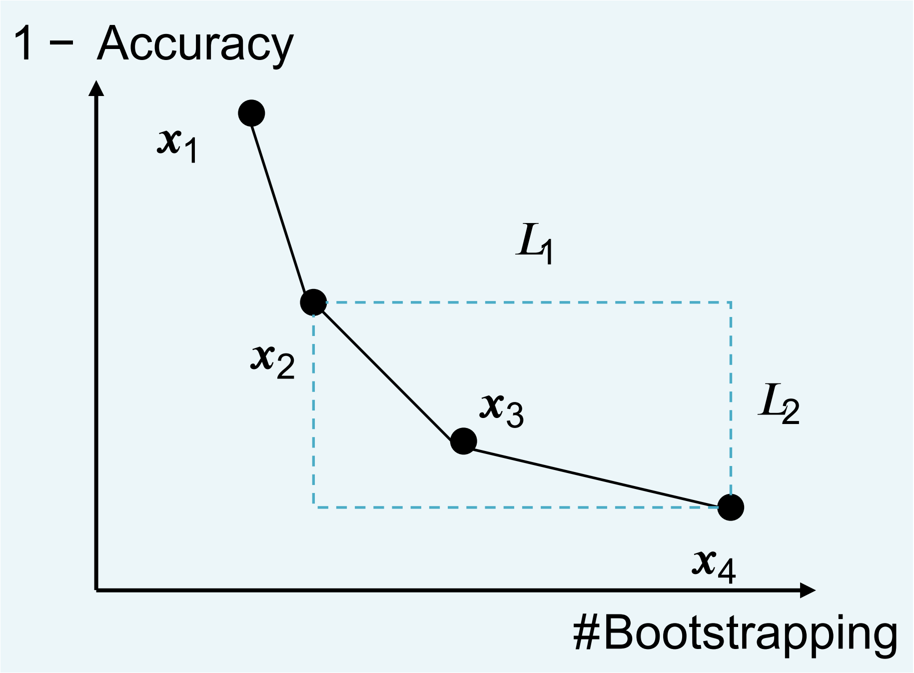


x_3 dominates $x_6, x_7,$ and x_8

i.e. x_3 is better than $x_6, x_7,$ and x_8

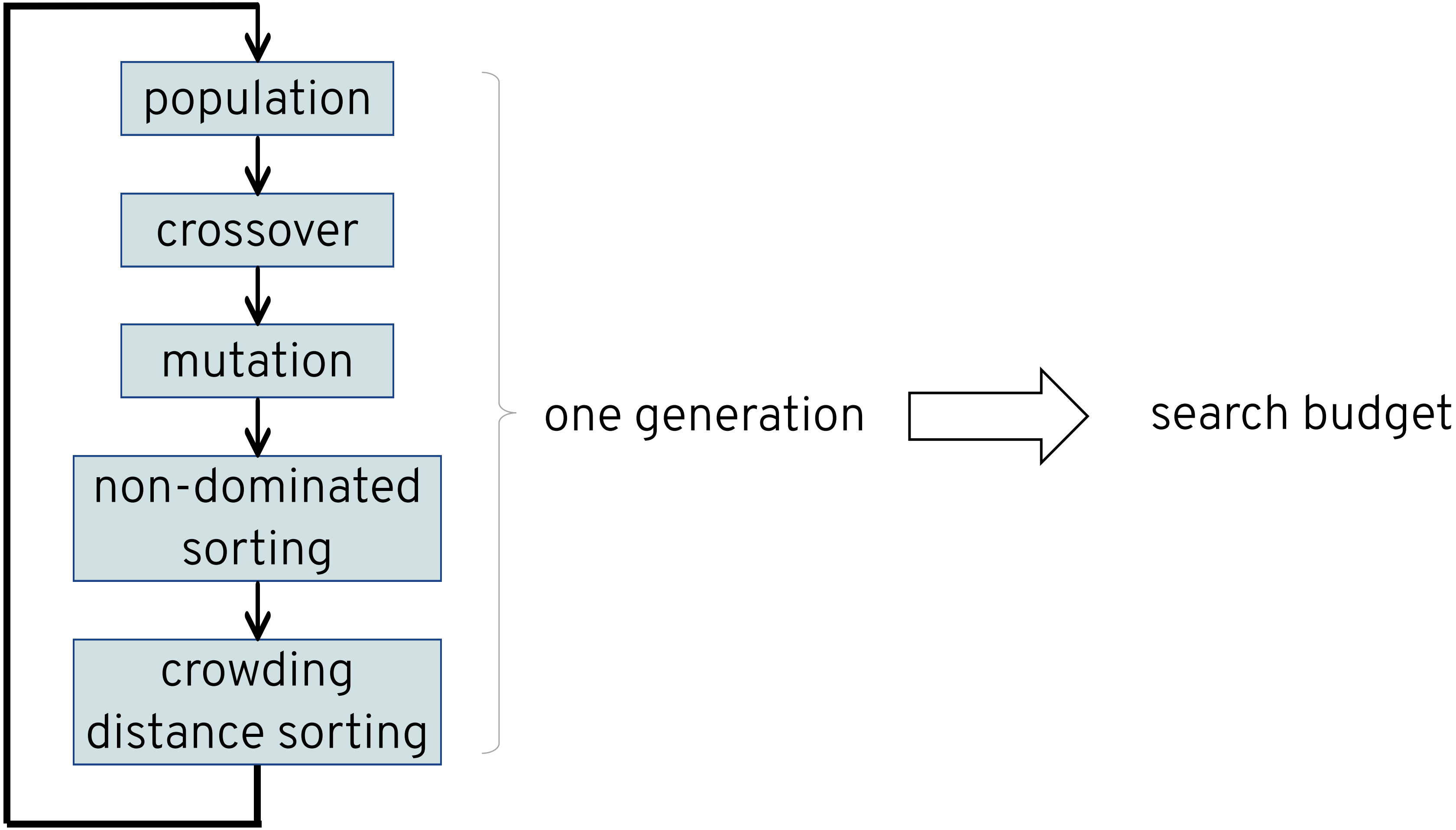
Evolutionary Multi-Objective Optimization

population
↓
crossover
↓
mutation
↓
non-dominated sorting
↓
crowding distance sorting



crowding distance of x_3 is $\frac{L_1 + L_2}{2}$

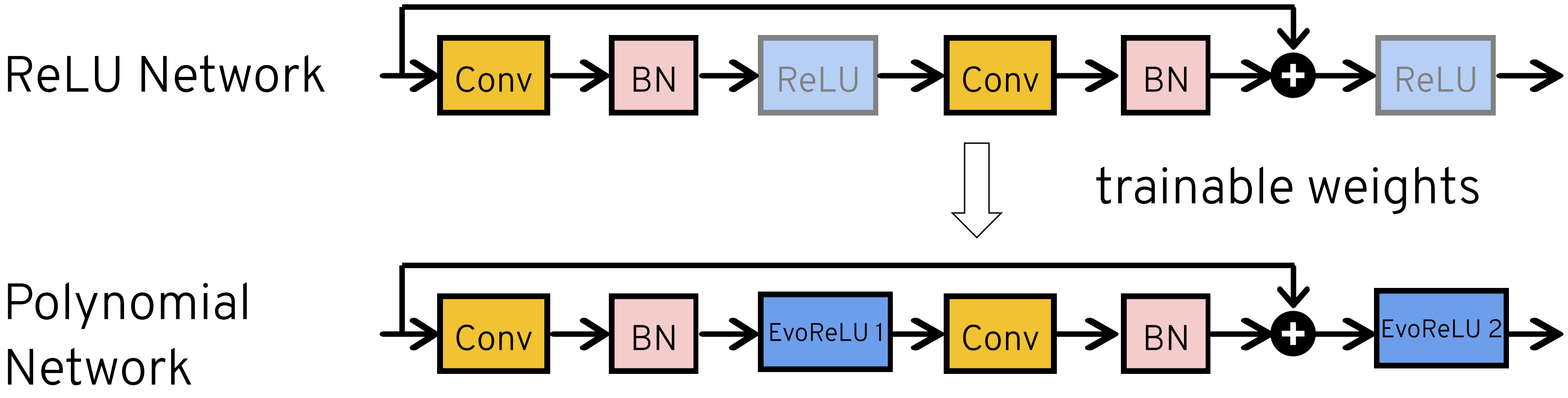
Evolutionary Multi-Objective Optimization



How to **fine-tune** polynomial CNNs?

Neural network adaption

Trainable Weight Adaption and **Knowledge Transferring**



Fine-tuning objective

$$\mathcal{L}_{train} = (1-\tau)\mathcal{L}_{CE} + \tau\mathcal{L}_{KL}$$

- Inherit representation learning ability
- Adapt trainable weights to EvoReLU

Experiments on encrypted CIFAR10 dataset under FHE

Experimental Setup

Dataset: CIFAR10

50,000 training images

10,000 test images

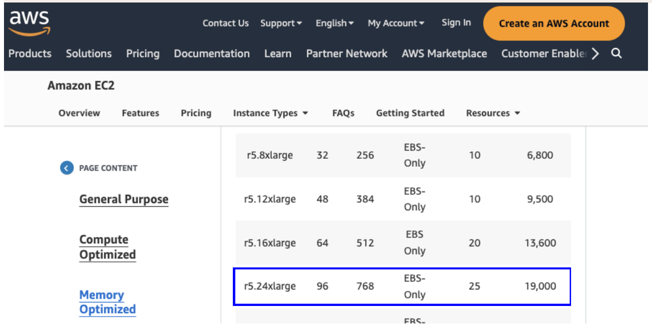
32x32 resolution, 10 classes

Hardware & Software

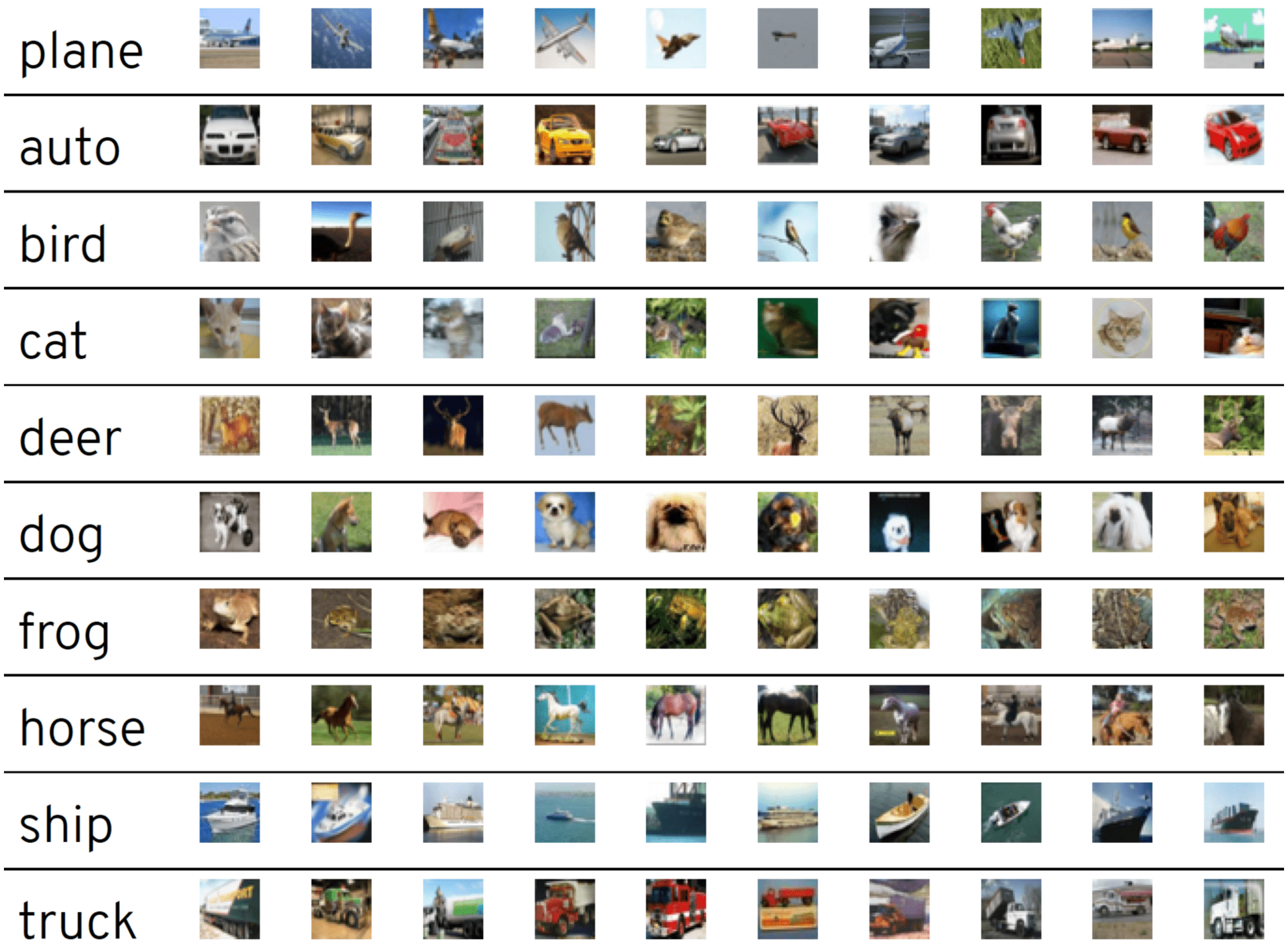
Amazon AWS, r5.24xlarge

96 CPUs, 768 GB RAM

Microsoft SEAL, 3.6



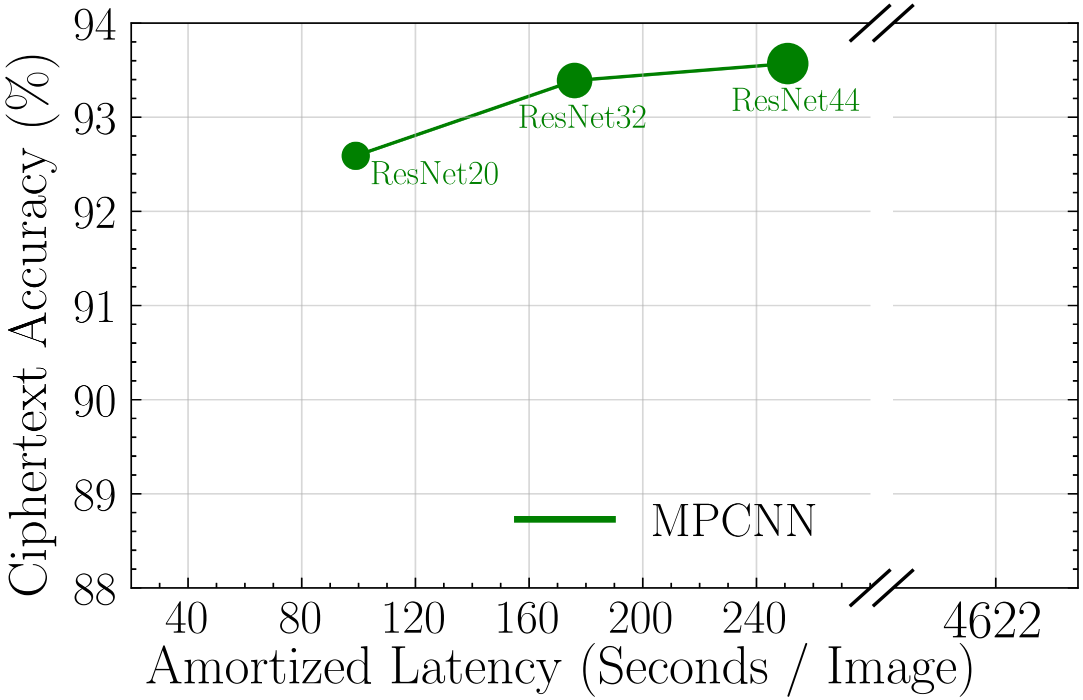
r5.8xlarge	32	256	EBS-Only	10	6,800
r5.12xlarge	48	384	EBS-Only	10	9,500
r5.16xlarge	64	512	EBS Only	20	13,600
r5.24xlarge	96	768	EBS-Only	25	19,000



[3]Alex Krizhevsky. CIFAR example images (online).

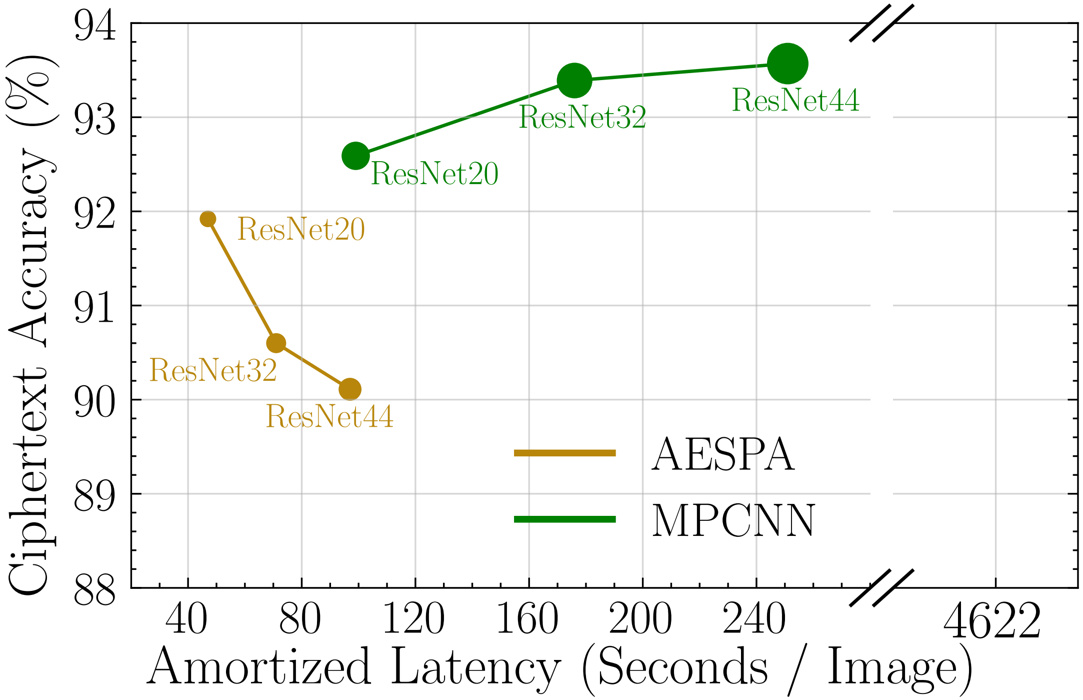
Latency and **Accuracy Trade-offs** under **FHE**

Approach	**MPCNN**
Venue	ICML22
Scheme	CKKS
Polynomial	high-degree
Layerwise	no
Strategy	approx
Arch	manual



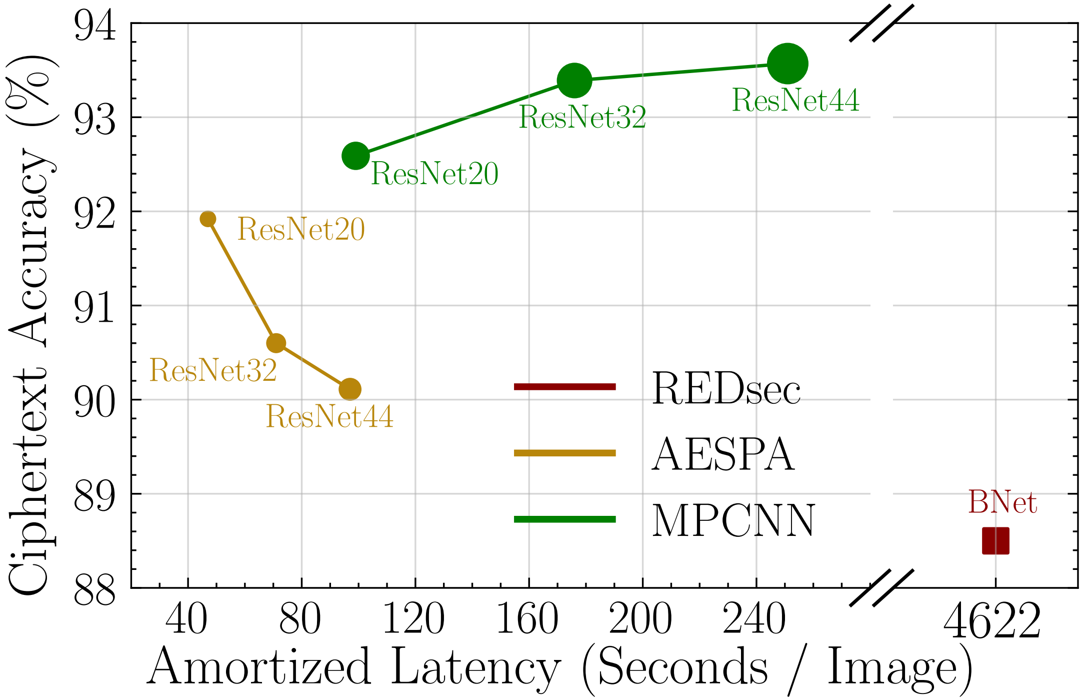
Latency and **Accuracy Trade-offs** under **FHE**

Approach	**MPCNN**	**AESPA**
Venue	ICML22	arXiv22
Scheme	CKKS	CKKS
Polynomial	high-degree	low-degree
Layerwise	no	no
Strategy	approx	train
Arch	manual	manual



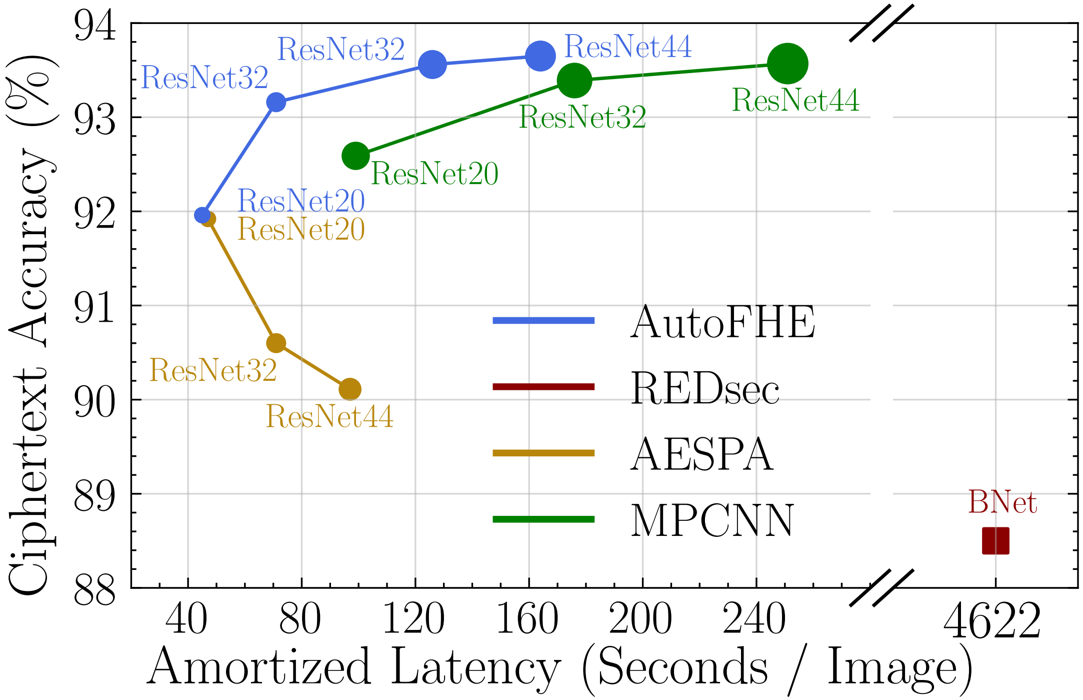
Latency and **Accuracy Trade-offs** under **FHE**

Approach	**MPCNN**	**AESPA**	**REDsec**
Venue	ICML22	arXiv22	NDSS23
Scheme	CKKS	CKKS	TFHE
Polynomial	high-degree	low-degree	n/a
Layerwise	no	no	n/a
Strategy	approx	train	train
Arch	manual	manual	manual

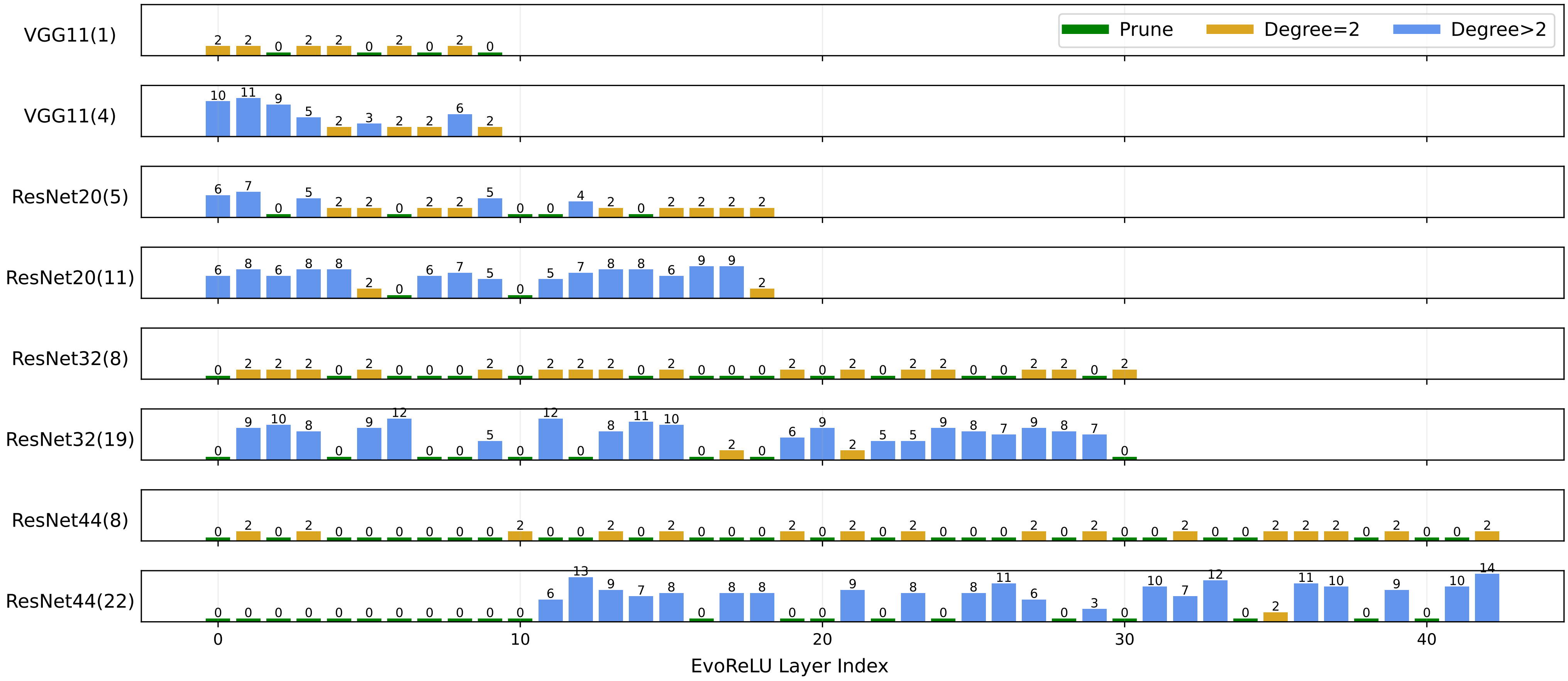


Latency and **Accuracy Trade-offs** under **FHE**

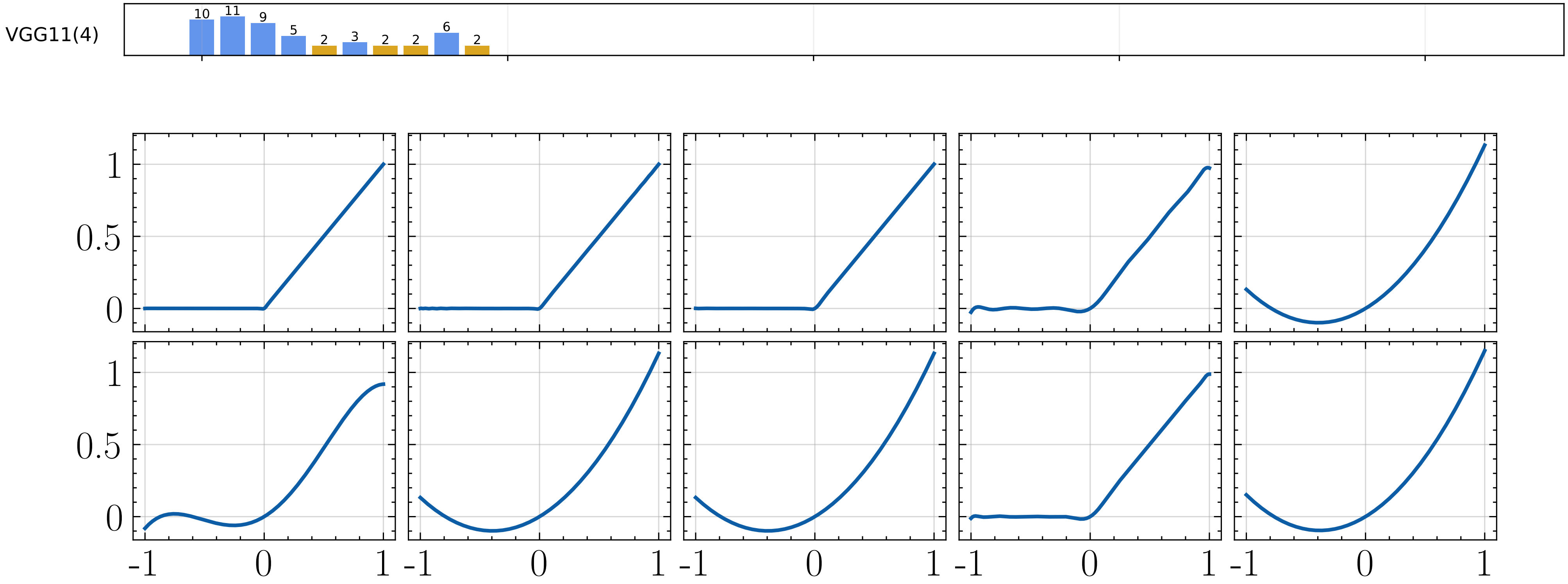
Approach	**MPCNN**	**AESPA**	**REDsec**	**AutoFHE**
Venue	ICML22	arXiv22	NDSS23	USENIX24
Scheme	CKKS	CKKS	TFHE	CKKS
Polynomial	high-degree	low-degree	n/a	mixed
Layerwise	no	no	n/a	**yes**
Strategy	approx	train	train	**adapt**
Arch	manual	manual	manual	**search**



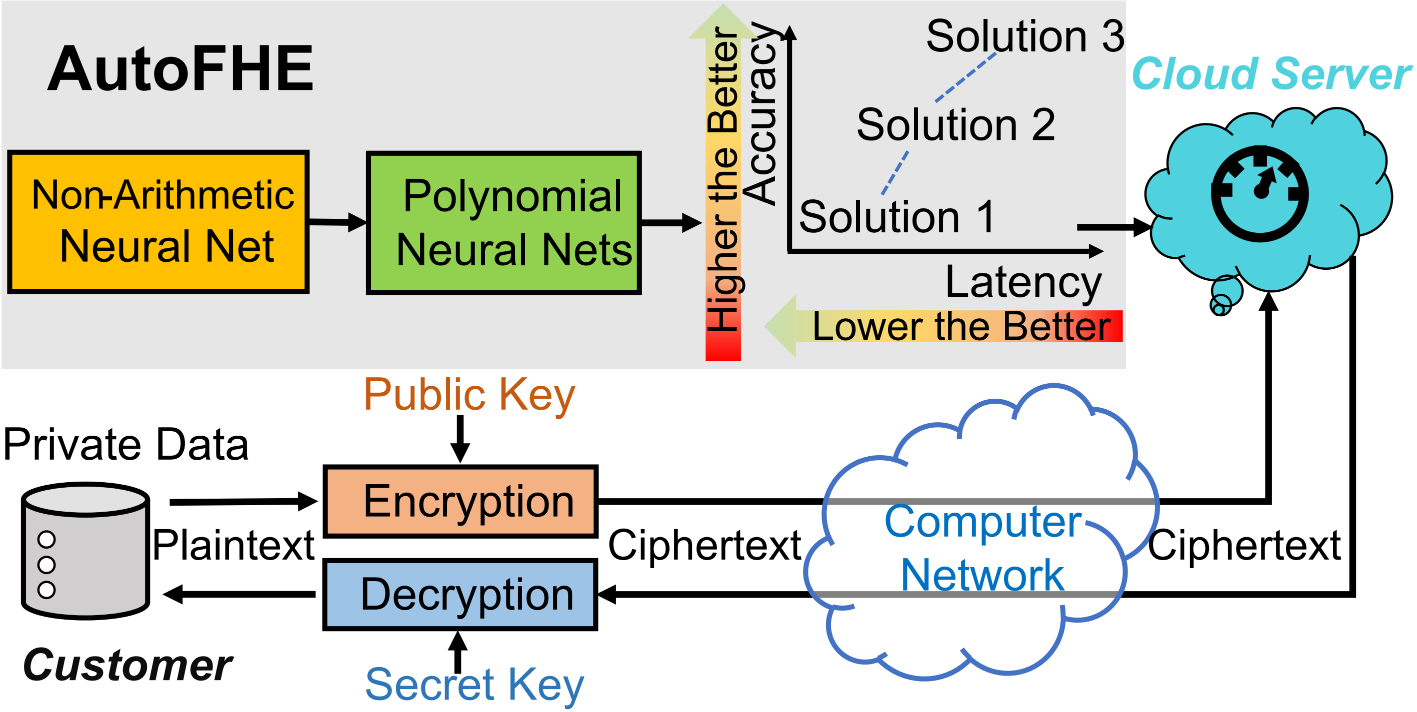
Multiplicative Depth of Layerwise EvoReLU



Layerwise EvoReLU



Conclusion



Paper, Code & Online Slides

AutoFHE optimizes end-to-end polynomial neural architecture

- Multi-objective optimization generates Pareto-effective solutions to meet different requirements
- Joint optimization of layerwise EvoReLU and bootstrapping results in optimal polynomial neural architectures
- Adapted neural networks can inherit representation learning ability from ReLU networks