Evolutionary Multi-objective Neural Architecture Search of Deep Neural Network

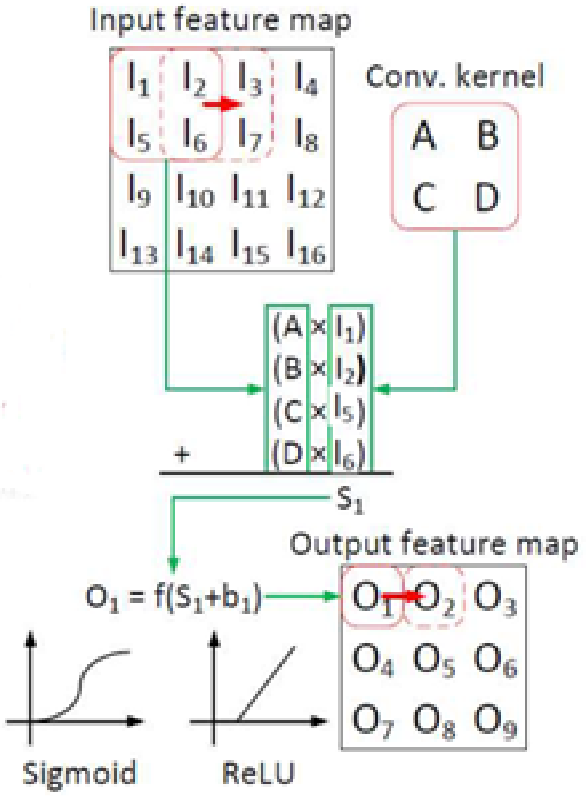
Presenter: Jiayi Feng
Supervisor: Vahid Geraeinejad
Examiner: Masoumeh Ebrahimi

Agenda

- **Introduction**
  - ➢Deep Neural Network (DNN), Convolutional Neural Network (CNN), Embedded Systems
  - ➢DNN Implementation in Embedded system
- **Background**
  - ➢Neural Architecture Search (NAS)
    - ❑Random Search
    - ❑Evolutionary Search
  - ➢Multi-objective Optimization
    - ❑ Pareto Front
- **Method**
  - ➢Binary One Optimization (BOO) Definition
  - ➢BOO Availability Analysis
- **Result Analysis**
  - ➢Searched Architectures
  - ➢Analysis of BOO, FLOPs, and Parameter size
- **Conclusion**

Introduction: DNN and CNN

- DNN, Multiple Layer, Fully connected
  - DNN Layer example:
    - Convolutional Layer (CNN),
    - Recurrent Layer (RNN),
    - Embedding Layer (Natural Language Processing (NLP)),
    - Pooling Layer,
    - Fully-Connected/Dense Layer.
  - Tasks autonomous driving, healthcare, and finance
- CNN, specialized DNN with convolutional layer
  - CNN image recognition and processing



Picture: Hardware Acceleration for Deep Learning IL2230 L4_IL2230_HT20_CNN.pdf Page 19

Embedded Systems

- Definition
  - Computer-based systems for specific functions but NOT perceived as computers
  - Safety-critical applications: automotive and medical systems
  - Design complexities and safety responsibilities
- Characteristics of Embedded Systems
  - Designed for specific tasks
  - Mass-produced, and cost-sensitive design
  - Real-time environmental interaction.
  - Power efficiency and market delivery are crucial
  - Design demands cost-efficient, rapid development, and accurate functionality

Reference: Embedded System IL2206 lecture_notes.pdf and intro-embedded-systems-1x2.pdf

Challenges in Implementing DNNs in Embedded Systems

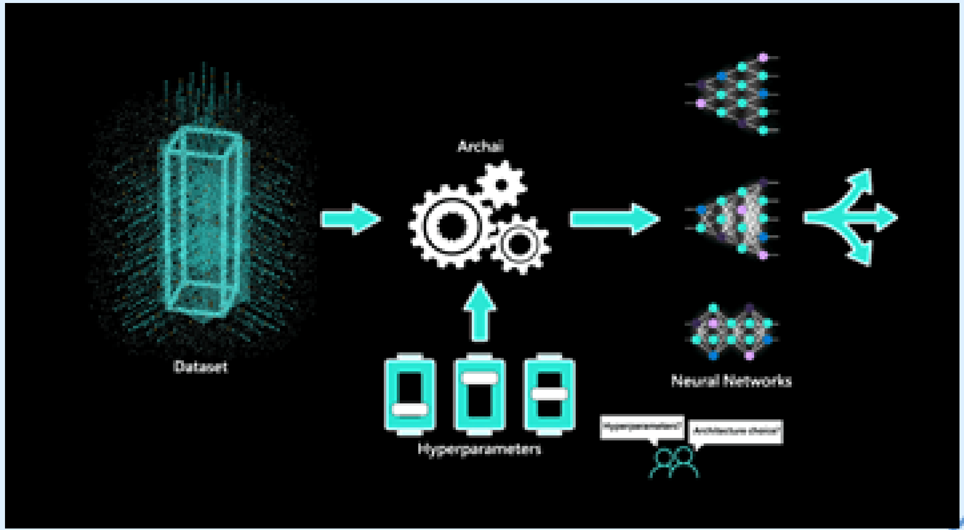
- Limited available embedded system resources versus increasing resource-demanding DNNs
  - ➢ Embedded system: Computational Power, Memory and Storage, Energy Efficiency, and Heat Dissipation.
  - ➢ DNNs: Complex and Resource-demanding

- Challlenges to find appropriate DNNs suitable to be deployed in embedded system
  - ➢ Any numerical merit that illustrate the difficulty in the implementation
  - ➢ Optimization trade-off between different or even confliction objectives
  - ➢ Automation to the DNN search process, manually VGG and ResNet
  - ➢ Efficient DNN search process

Background: Neural Architecture Search

- Automated Machine Learning (AutoML)
  - Automated Data Preprocessing
  - Automated Feature Engineering
  - Automated Model Selection

- NAS is a subset of AutoML
  - Automates the design of neural network models
  - Stochastic Gradient Descent (SGD)
  - Reinforcement Learning (RL)
  - Evolutionary/Genetic Algorithms



Picture: Archai can design your neural network with state-of-the-art neural architecture search (NAS), https://www.microsoft.com/en-us/research/blog/archai-can-design-your-neural-network-with-state-of-the-art-neural-architecture-search-nas/

Parameter (Weights) and Hyperparameter

- Parameters in NAS
  - Weights and biases within the network
  - Learned directly from training data
  - Example: Weights in convolutional layers

- Hyperparameters in NAS
  - Settings that govern the architecture search
  - Not learned from data, but set prior to training
  - Examples: Number of layers, type of layers, and learning rate

Search Space

- Search space is essentially a set of hyperparameters that encompasses potential neural network architecture
- Size of the search space is measured as the total number of possible neural network architectures

Hyper-parameter Types	Candidates
Number of filters	{16, 36, 64, 100}
Size of filters	{3×3, 5×5, 7×7}
Stride	{1, 2}

Search Space

$$\begin{gathered} Search\ Space: \\ \{1,2\}^N \times \\ \{3\times 3, 5\times 5, 7\times 7\}^N \times \\ \{16, 36, 64, 100\}^N \end{gathered}$$

$$\begin{aligned} &Size\ of\ Search\ Space: \\ &(\{candidates\ of\ number\ of\ filters\} \\ &\times\ \{candidates\ of\ size\ of\ filters\} \\ &\times\ \{candidates\ of\ stride\})\text{^}N \\ &=\ \{4\ \times\ 3\ \times\ 2\}\text{^}N \end{aligned}$$

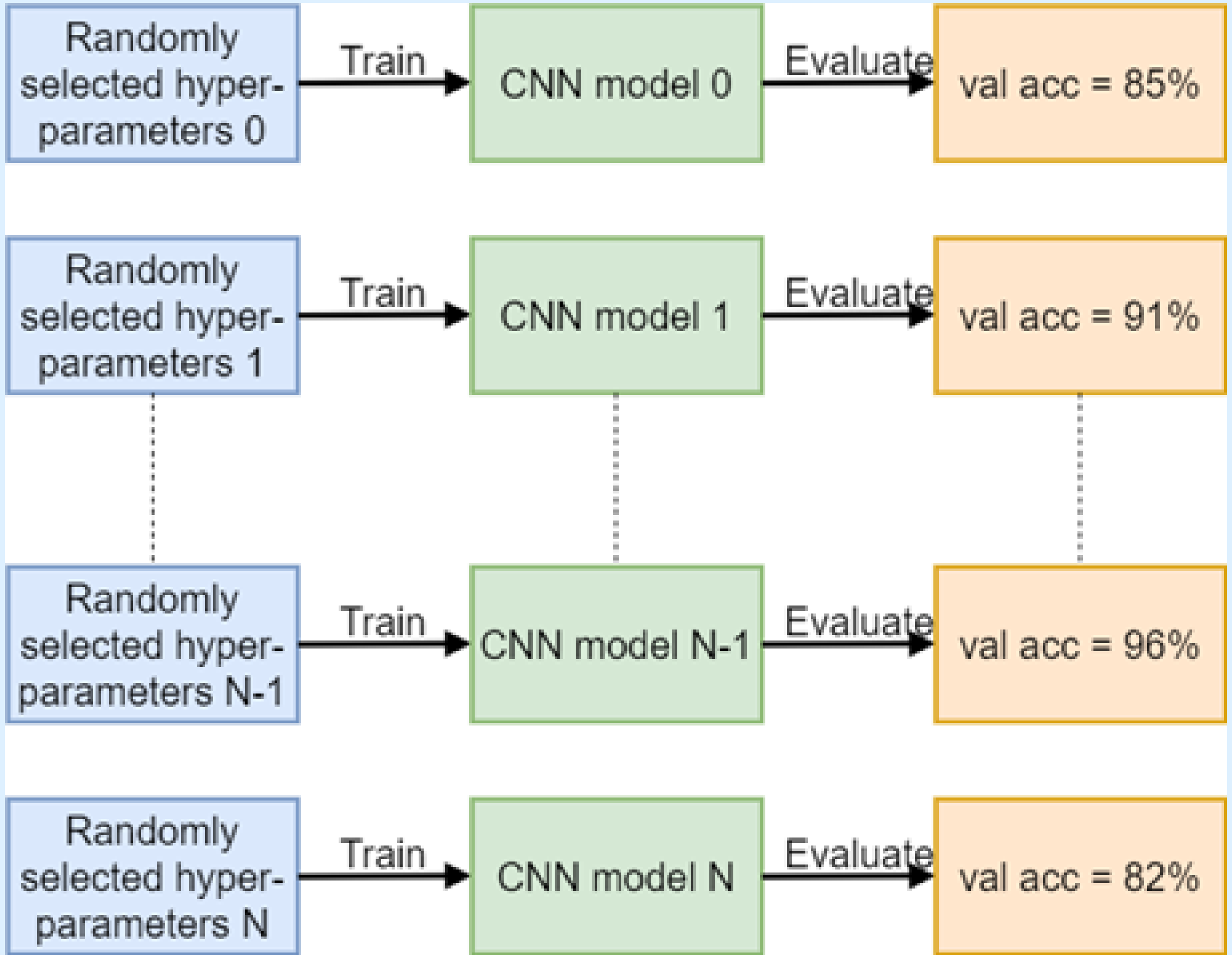
	Layer 1	Layer 2	…	Layer (N-1)	Layer N
Number of filters	16	36	…	100	64
Size of filters	3×3	3×3	…	7×7	5×5
Stride	1	2	…	1	1

Search Space Encoding

- Encoding in Neural Network Architecture
  - Translates neural architectures into binary and integer strings for computation
- Macro Encoding
  - Represents block connections within CNN architectures
  - Simplified representation focusing on connectivity
- Micro Encoding
  - Details the selection and quantity of computational blocks
  - More computationally intensive due to finer granularity
- Automation in CNN Design
  - Facilitates automated search for optimized architectures
  - Reduces dependency on manual design expertise

Baseline Random Search

- Random Search is efficient in small and simple search space, and can be perfect baseline to the proposed NAS method



Evolutionary Search

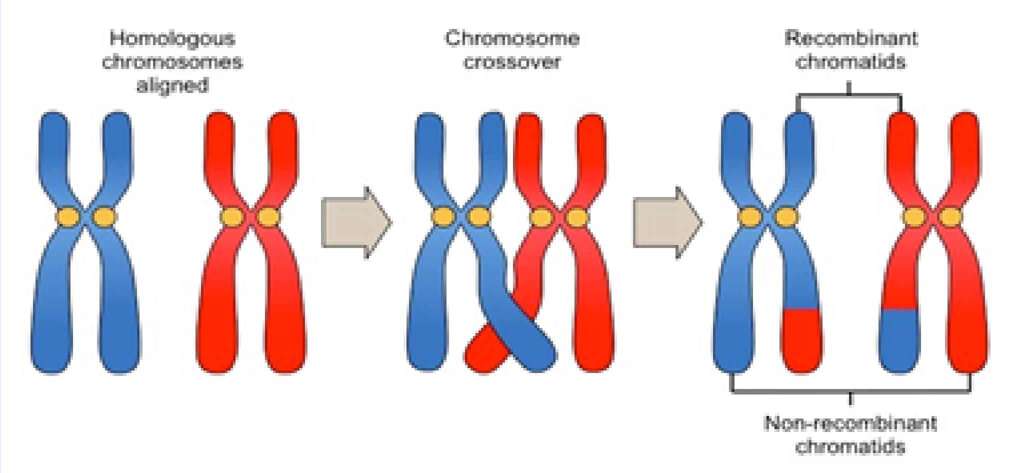
- Evolutionary Search Overview
  - Population-based iterative approach
  - Aims to improve hyperparameter solutions
  - Inspired by Darwin's theory of evolution
  - Mimics survival of the fittest
  - Leads to better solutions by natural selection
- Evolutionary Search Phases in NAS
  - Initialization: Generate initial population
  - Selection: Choose fittest individuals
  - Mutation: Introduce random changes
  - Crossover: Combine individuals to create offspring
  - Termination: End process upon solution or limit

Evolutionary Search: Mutation

- Mutation
  - *Wiki: In biology, a mutation is an alteration in the nucleic acid sequence of the genome of an organism, virus, or extrachromosomal DNA Viral genomes contain either DNA or RNA.*
  - In NAS, a mutation is an alternation of tensor elements that represents the neural network
  - Introduces genetic diversity
  - Prevents population pool premature convergence
- Bit-Flipping Mutation Example
  - Original Genotype: [0, 1, 0]
  - Post-Mutation Genotype: [0, 0, 0]
- Polynomial Mutation Example
  - Original Genotype: [3, 6, 5]
  - Post-Mutation Genotype: [3, 5, 5]

Evolutionary Search: Crossover

- Overview of Crossover
  - ➢ Exchange and combine genetic information of both parents
  - ➢ Introduces diversity, aids in finding global optimum
- Multi-point Crossover
  - ➢ Multiple points along the length of parents
  - ➢ Crossover mask determines gene exchange
- Crossover Numerical Example
  - ➢ Parents: [[A, B, C, D, E], [1, 2, 3, 4, 5]]
  - ➢ Mask: [F, F, T, T, T]
  - ➢ Offspring: [[A, B, 1, 2, 3], [1, 2, C, D, E]]

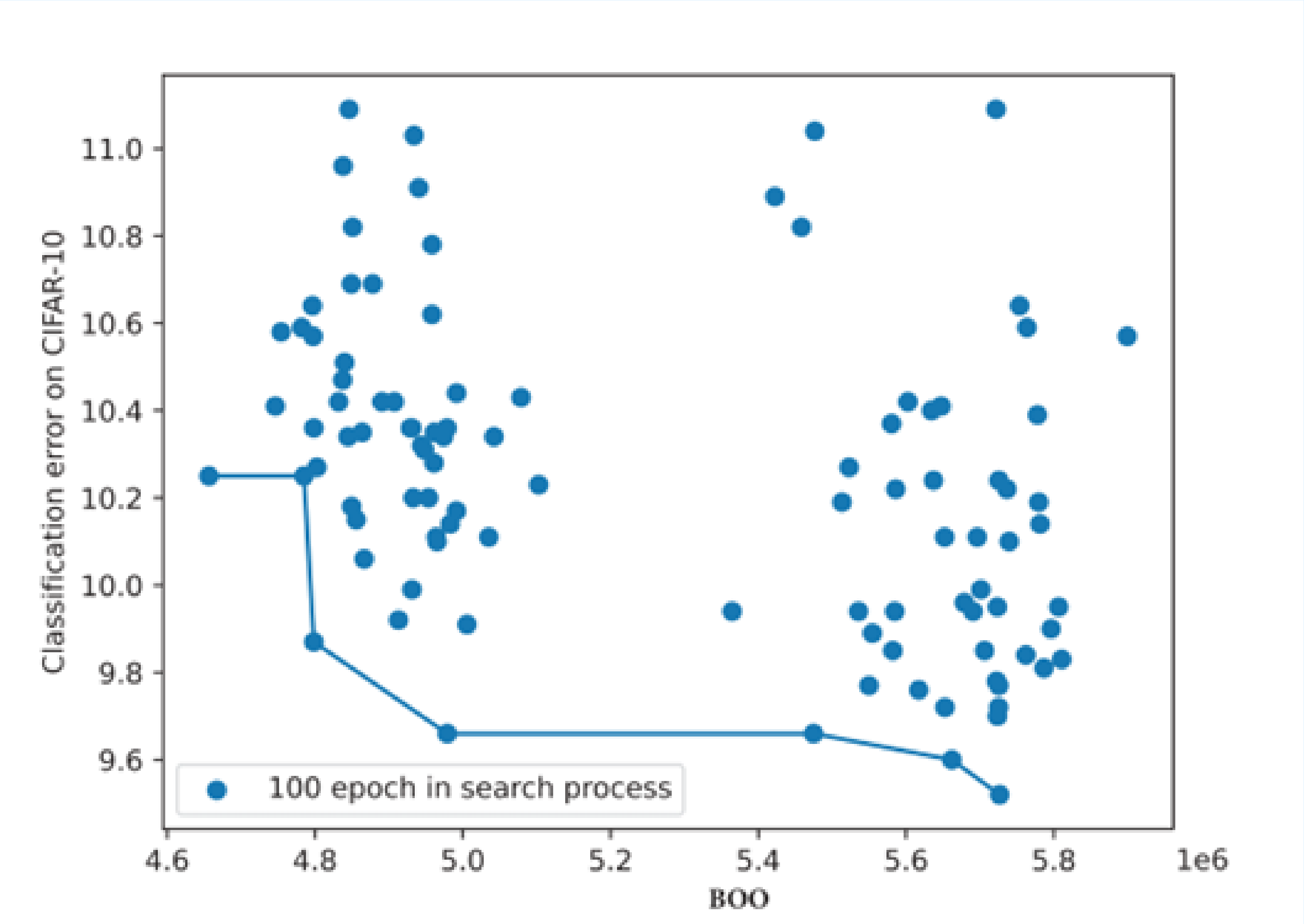


Picture: BioNinja at https://ib.bioninja.com.au/standard-level/topic-3-genetics/33-meiosis/crossing-over.html

Multi-objective Optimization

- Deals with multiple, often conflicting, objectives
- Examples of Conflicting Objectives
  - In computer manufacturing: computational power vs. weight vs. cost
  - In automotive industry: carrying capacity vs. speed
- Goal: Finding Non-Dominated Solutions
- Identify solutions that balance divergent objectives
- Pareto Optimal Solutions
  - Improvement in one objective leads to deterioration in another
  - Cannot improve one objective without worsening another
  - Pareto Front: Set of all Pareto optimal solutions
- Application in Evolutionary Multi-objective Neural Architecture Search (EMONAS)
  - Optimize both image classification errors and Binary Ones (BOO) performance

Multi-objective Optimization Visualization



Method: BOO Definition

- BOO in DNNs for Embedded Systems
  - Converts DNN parameters to binary format
  - Minimizes the number of binary ones ('1')
- BOO's Importance in Embedded Systems
  - Optimizes DNN performance on hardware
  - Criteria: Lower number of binary ones indicates better performance
- BOO Calculation
  - Convert floating-point weights to binary
  - Sum the total number of binary ones
- Comparison with Traditional Metrics
  - Complements Parameter Size and FLOPs
  - Offers a different perspective on DNN efficiency

BOO Availability Analysis

- Benefits of BOO in DNNs
  - Arithmetic Efficiency
  - Hardware Acceleration
  - Power Dissipation
- Arithmetic Efficiency Example
  - Simplification of binary operations
  - Less 1's lead to reduced arithmetic operations
- Hardware Acceleration with BOO
  - Sparsity in weight matrix
  - Optimized for sparse matrix operations
- Power Dissipation
  - Switching frequency

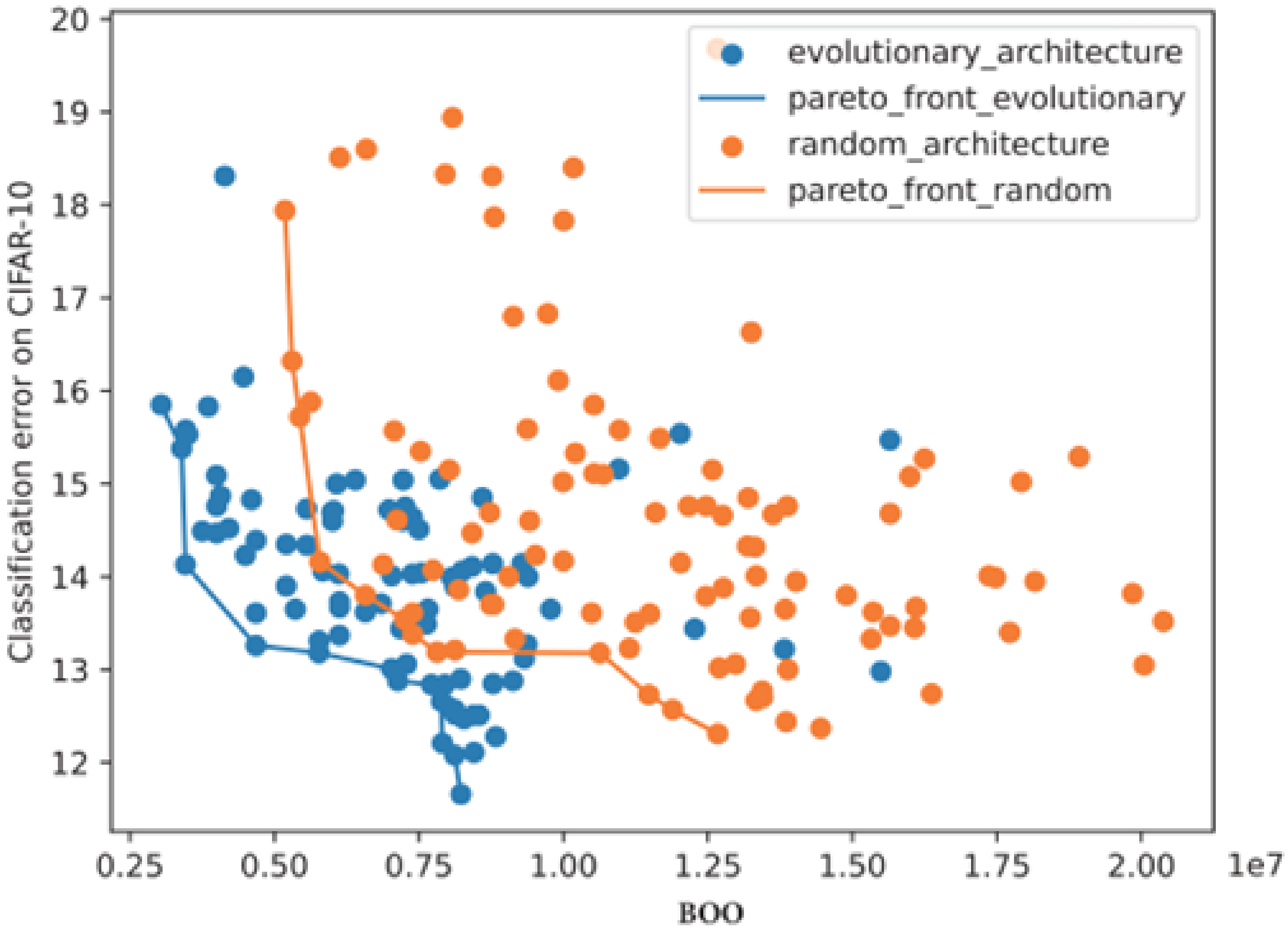
Result Analysis: Simulation Setup

Description	Hyperparameter Name	Default Value
Epoch of training	epoch	20
Filters for the first cell	init_channels	16
Layer	layers	11
Block in a cell	n_block	5
Cell	n_cell	2
Generation value	n_gen	10
Node per phase	n_node	6
Offspring per generation	n_offspring	10
Operations considered	n_ops	9
Population size	pop_size	10
Micro or macro encoding	search_space	'macro'
Random seed	seed	0

Searched Architecture Configuration

Hyperparameter	EMONAS-BOO	Random
epoch	20	20
init_channels	16	16
layers	11	11
n_block	5	5
n_cell	2	2
n_gen	10	0
n_node	6	6
n_offspring	10	0
n_ops	9	9
pop_size	10	100
search_space	'micro'	'micro'
seed	0	0

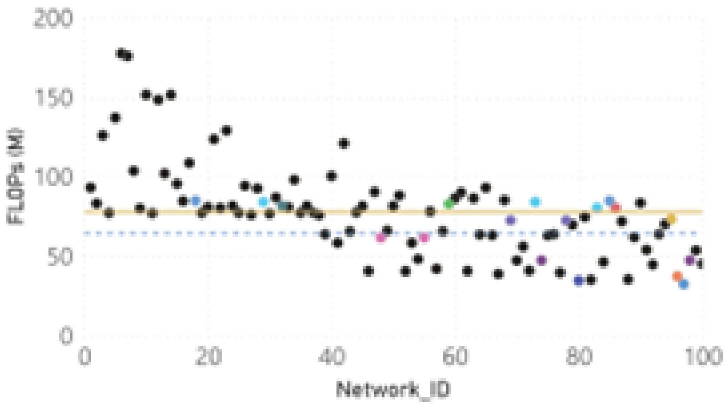
Searched Architecture Display

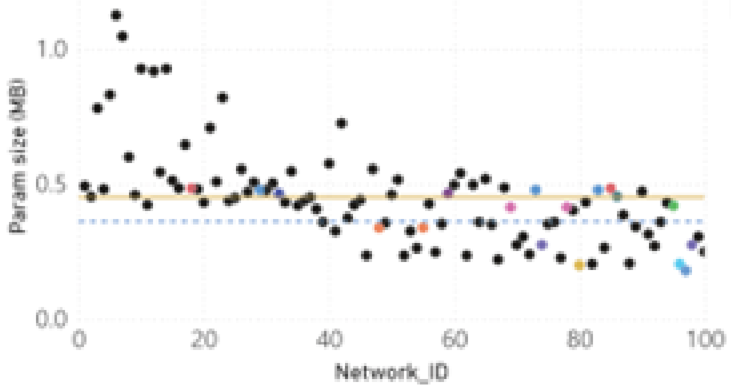


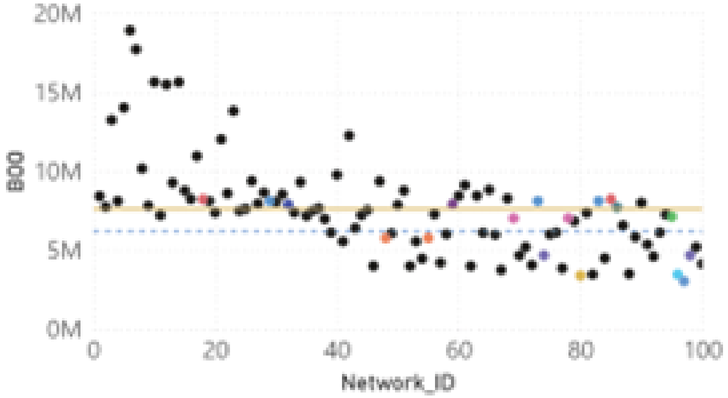
Searched Architecture Display Analysis

- Variable Elements in Search
  - ➢ Comparison of evolutionary and random searches via hyperparameters
  - ➢ Hyperparameters *n_gen, n_offspring, and pop_size*
- Consistency in Experimentation
  - ➢ Unified framework with micro search, 20 epochs, and specific blocks/channels
- Performance Observations
  - ➢ Higher precision in evolutionary search
  - ➢ Dual optimization of image classification error and BOO in evolutionary search
- Pareto Front Analysis
  - ➢ Evolutionary search Pareto front surpasses random search
  - ➢ Better DNNs' models solutions

BOO, FLOPs, and Parameter Size







Network_ID	Acc	Param_Size (MB)	BOO	FLOPs (M)
97	84.15	0.18	3040227	32.52
80	84.62	0.20	3394436	34.87
96	85.87	0.20	3461520	37.43
74	86.74	0.27	4680097	47.52
48	86.82	0.34	5769896	61.71
69	86.99	0.41	7024962	72.66
95	87.12	0.42	7134078	73.42
86	87.16	0.45	7715063	80.48
32	87.34	0.46	7889428	81.64
59	87.79	0.46	7907194	83.00
29	87.92	0.48	8119060	84.33
85	88.34	0.48	8227340	85.00

BOO, FLOPs, and Parameter Size Analysis

- Assessing Performance Metrics
  - Comparison of BOO with classical evaluators: FLOPs and parameter sizes.
  - Evolutionary micro search
- Pareto Front Neural Networks
  - 12 out of 100 networks optimized for accuracy and BOO.
- Key Observations
  - Correlation between BOO, FLOPs, and parameter size indicates optimization effectiveness.
  - Pareto front networks display evolutionary search's potential in multi-objective optimization.

Conclusion

- DNNs and Embedded Systems Challenge
  - High demand for DNN deployment in embedded systems
  - Conflicting needs: DNN complexity vs. embedded system simplicity
- Binary One Optimization (BOO)
  - Innovative performance assessment tool
  - Balances DNN efficiency with embedded system constraints
- EMONAS Framework
  - Automates DNN architecture optimization
  - Employs evolutionary search algorithm
  - Achieves accuracy and BOO efficiency in DNNs by multi-objective optimization
- Effective DNN Search Result
  - 12 out of 100 neural networks optimized for embedded systems
  - Maintains image classification accuracy while optimizing BOO merits

Thank you

Questions ?