Neural Architecture Search and Architecture Encoding

Xuefei Ning Email: foxdoraame@gmail.com

NICS-EFC Lab, EE, Tsinghua University

Lab Leader: Yu Wang yu-wang@tsinghua.edu.cn

2022/11/24

Menu

1. Basics
2. Field Summary: Questions and Development
   a) What to Search
   b) How to Search
   c) What Can Search Tell Us
3. Our Work: Utilizing Architecture Encoding to Answer “How to Search”
4. Summary

Menu

Neural Architecture Matters

- Neural architecture matters for task performance, hardware efficiency, and so on.
- Extensive expert efforts have been devoted to developing new architectures, pushing forward the wide application of NN.

Task Performances

- Classification Accuracy
- Detection mAP
- Segmentation IoU
- …

AlexNet (2014)	**VGG** (2015)	**ResNet** (2016)	**DenseNet** (2017)
#Param: 61M	#Param: 143M	#Param: 25M	#Param: 8M
#FLOPs: 7.1E8	#FLOPs: 7.6E9	#FLOPs: 4.0E9	#FLOPs: 2.9E9
56.5%	71.5%	76.1%	74.7%

Hardware Efficiency

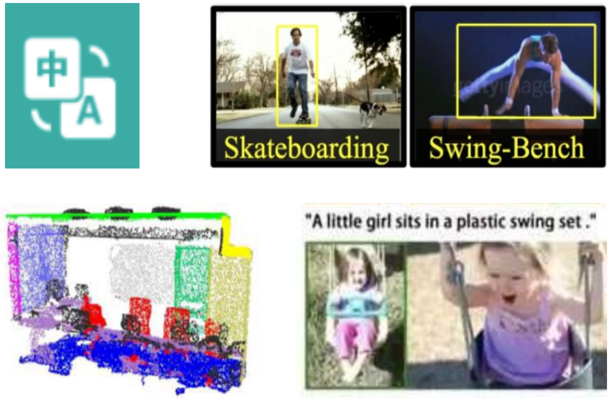
- #Parameter / #FLOPs
- Latency
- Energy
- …

SqueezeNet (2016)	**MobileNet** (2016)	**ShuffleNetV2** (2018)	**GhostNet** (2020)
#Param: 1M	#Param: 4.2M	#Param: 2M	#Param: 7.3M
#FLOPs: 8.2E8	#FLOPs: 2.8E8	#FLOPs: 1.5E8	#FLOPs: 2.3E8
58.2%	70.6%	69.4%	75.7%

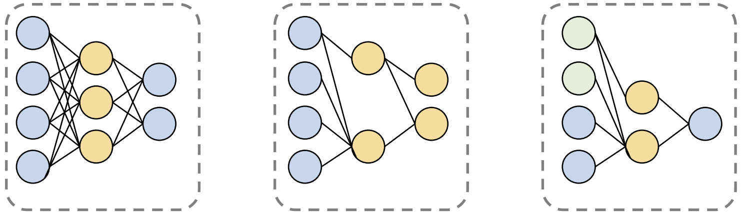
From Manual Design to Automated Design

- Varying tasks and hardware may need different architectures
- Need to consider multiple objectives or constraints in the meantime

Varying Tasks



Neural Architecture



Varying Hardware

A100 GPU | Versal ACAP | TPU v3

NVIDIA Xavier GPU | Xilinx ZU11 FPGA | TI TDA4 ASIC

Expensive/Suboptimal

Purely Manual Design

We need automated architecture design!

Neural Architecture Search (NAS)

- Basic Problem Definition
  - Architecture a; Search space S; Network parameters w
  - Valid task perf R_{val}; Train task loss L_{train}; Complexity function F; Budget B

$$\min_{a \in S} \; R_{val}(w^*(a), a)$$

$$\text{s.t.} \;\; w^*(a) = \text{argmin}_w \, L_{train}(w, a)$$

$$\boldsymbol{F}(a) \leq \boldsymbol{B}$$

Inner optimization: Embeds the optimization for w

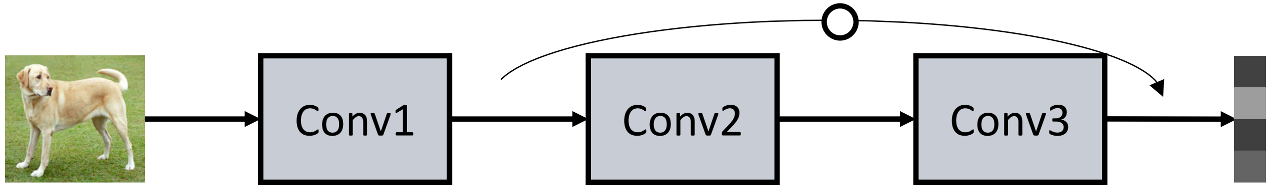
Constraints: #Param, #FLOPs, latency, energy, …

- A black-box optimization problem => Using "search" to solve
- Three components in search:
  - Search space: Set of all possible architectures
  - Search strategy: Explore the search space and sample candidate architectures
  - Evaluation strategy: Evaluate the performances of candidate architectures

Neural Architecture Search (NAS)

- A minimal search space example
  - Search space: Set of architectures; Cartesian product of decisions
  - Architecture
  - Decision: Set of choices
  - Choice

A topological decision: whether this connection exists: $d_0 = \{0, 1\}$ [2 choices]



The i-th Conv has two operation parameter **decisions**:
#channel: $d_{i1} = \{32, 64, 128\}$ [3 **choices**]
Kernel size: $d_{i2} = \{1, 3, 5\}$ [3 choices]

The **search space** can be represented as the Cartesian product of all 7 decisions:

$$S = d_0 \times \prod d_{i1} \times d_{i2} \ ; |S|=1458$$

Each element in S describes an **architecture**. Deciding the choice for all decisions yields an architecture.

Menu

1. Basics
2. **Field Summary: Questions and Development**
   a) What to Search
   b) How to Search
   c) What Can Search Tell Us
3. Our Work: Utilizing Architecture Encoding to Answer "How to Search"
4. Summary

Menu

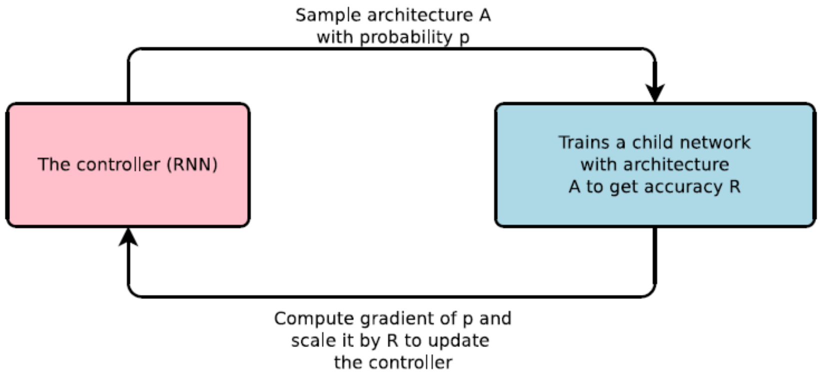
What to Search

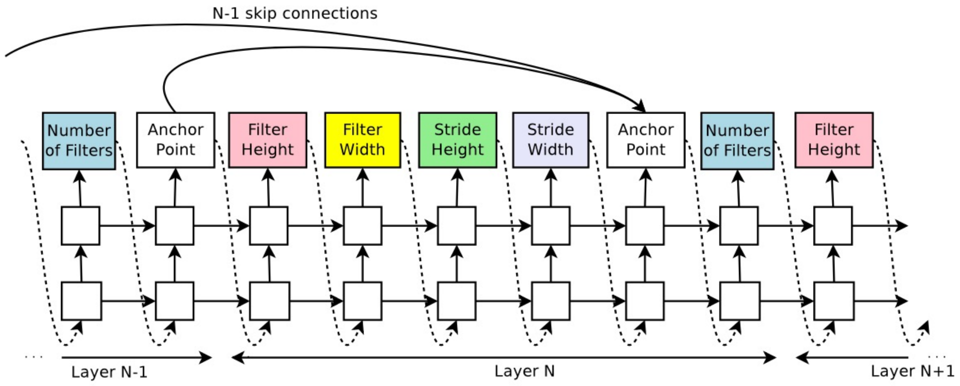
- Which decisions' choice matters for performance/objectives (including task performance & hardware efficiency & ...)?
  - Topology: Connection pattern, depth
  - Feature size: #Channels (width), #Spatial size (resolution)
  - Operation type
  - Operation parameters: conv kernel size, conv group number, …
- Types of search spaces
  - Topological: Macro, Micro, Hierarchical
  - Hardware-friendly (Non-topological, Non-micro)
  - Extension:
    - Outside of architecture: Other components in the DL pipeline
    - Outside of DL

Topological Search Spaces: Macro

Google Brain: Neural Architecture Search with Reinforcement Learning[Zoph et al., 2017]

- Macro search space
  - Skip connections
  - Filter width/height/number, stride height/width
- Results: error rate of 3.65% on CIFAR-10
- **Inefficient**: Search space size: 3.9×10^{73}; Search cost ~22.4k GPU days



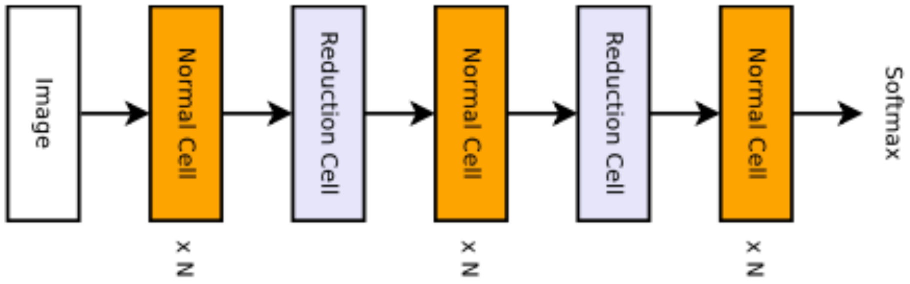


Zoph et al., “Neural architecture search with reinforcement learning”, ICLR’17.

Topological Search Spaces: Micro

Google Brain: Learning Transferable Architectures for Scalable Image Recognition[Zoph et al., 2018]

- Cell-based (micro) search space (**NASNet** search space) to reduce the space complexity while keeping the space containing well-performing architectures
  - Repeat: stacking together more copies of this cell, each with its own parameters
- Results: 2.4% error rate on CIFAR-10; 82.7% top-1 accuracy on ImageNet
- **More Efficient than NAS-RL**: Search space size: 5.2×10^{33}; Search time ~2k GPU days



Stack of Normal cell & Reduction cell

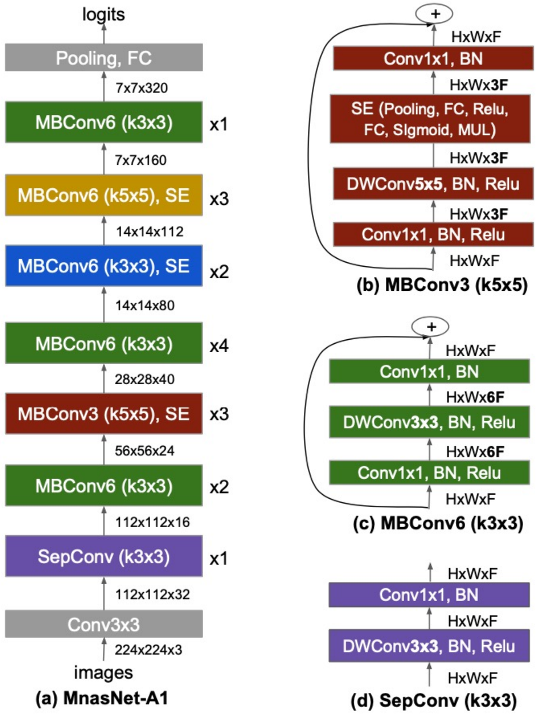
Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR'18.

Hardware-friendly Search Spaces

- More friendly to efficiency objectives (better task performance - efficiency trade-off

- MNasNet[Tan et al., 2019]

- **Stage-wise space** with **non-topological** decisions
  - **More layer diversity** than cell-based NASNet space, important for efficiency
    - Convolutional ops $ConvOp$: regular conv (conv), depthwise conv (dconv), and mobile inverted bottleneck conv [29].
    - Convolutional kernel size $KernelSize$: 3x3, 5x5.
    - Squeeze-and-excitation [13] ratio $SERatio$: 0, 0.25.
    - Skip ops $SkipOp$: pooling, identity residual, or no skip.
    - Output filter size F_i.
    - Number of layers per block N_i.



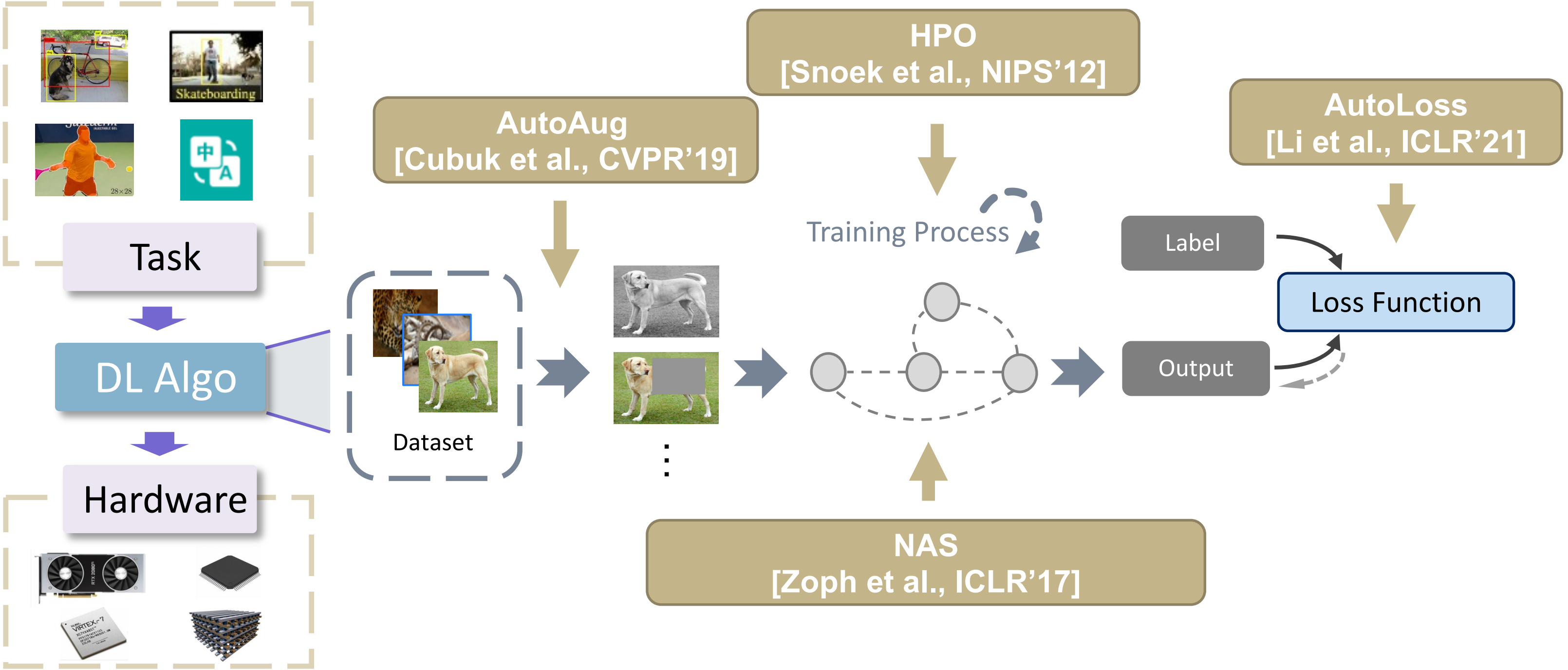
Reward	Search Space	Latency	Top-1 Acc.
NASNet	NASNet	183ms	74.0%
Multi-obj	NASNet	100ms	72.0%
Multi-obj	**MnasNet**	**78ms**	**75.2%**

Better efficiency-task performance trade-offs than cell-based spaces (NASNet)

Tan et al., “MnasNet: Platform-Aware Neural Architecture Search for Mobile”, CVPR’19.

AutoDL other than NAS or Outside of DL

- Other components in the DL pipeline besides architectures



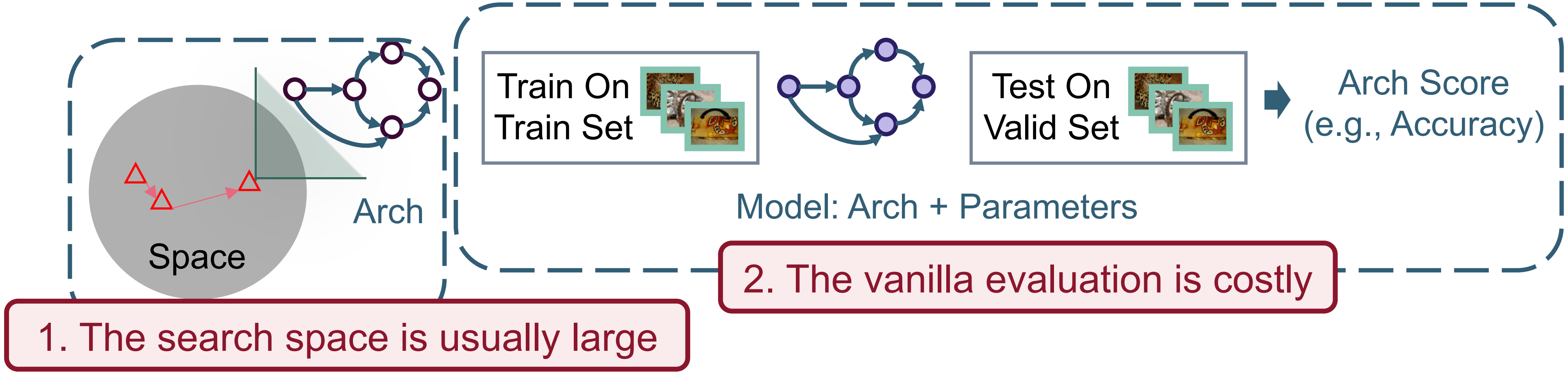
- More ambitious[Real et al., 2020]: Jump out of the DL paradigm to discover novel things. E.g., can we discover things like "back-propagation" or "SGD"

Cubuk et al., "Autoaugment: Learning augmentation strategies from data", CVPR'19.
Snoek et al., "Practical bayesian optimization of machine learning algorithms", NeurIPS'12.
Zoph et al., "Neural architecture search with reinforcement learning", ICLR'17.
Li et al., "Auto Seg-Loss: Searching Metric Surrogates for Semantic Segmentation", ICLR'21.
Real et al., "Automl-zero: Evolving machine learning algorithms from scratch", ICML'20.

Menu

Target: Efficient Search

- Challenges
  - Large search space: Need to evaluate many architectures to explore sufficiently
  - Costly vanilla evaluation: The evaluation of each architecture is costly



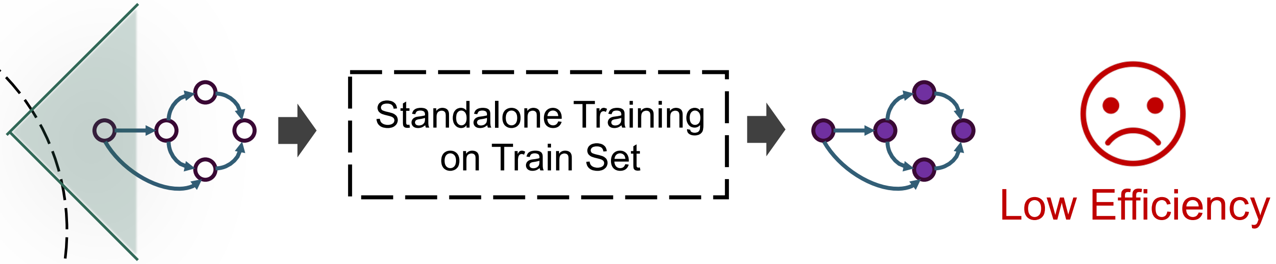
Work	Search space size	#Evaluated architectures	#Train epochs	#GPU day
NASRL [Zoph et al., 2017]	3.9×10^{73}	12.8k	50	800x28d=22.4k
NASNet [Zoph et al. 2018]	5.2×10^{33}	20k	20	500x4d=2k
AmoebaNet-A [Real et al. 2019]	3.1×10^{28}	20k	25	450x7d=3.2k

Zoph et al., "Neural architecture search with reinforcement learning", ICLR'17.
Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR'18.
Real et al.., "Regularized Evolution for Image Classifier Architecture Search", AAAI'19.

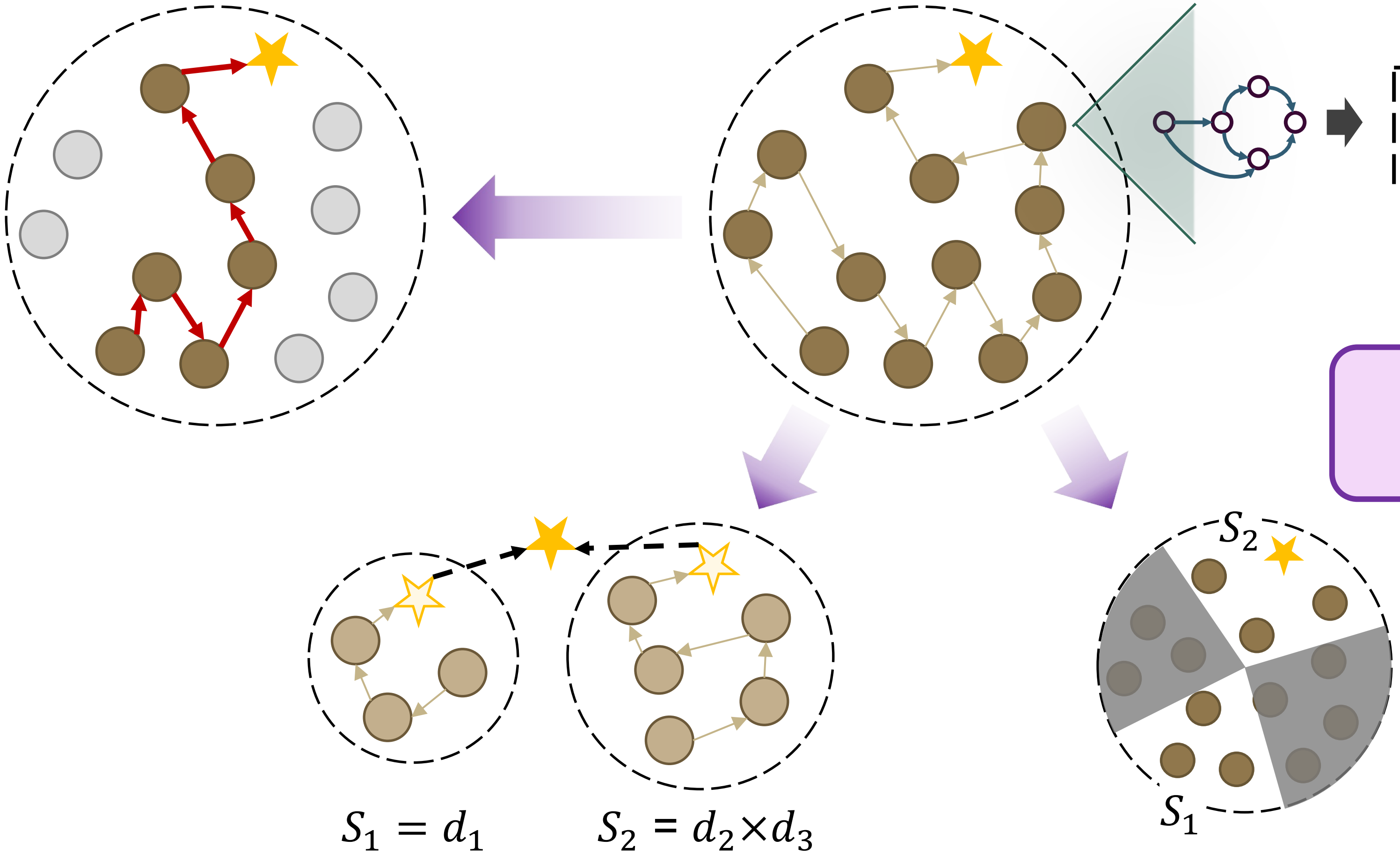
Directions for Efficient Search

1. Improve the sample efficiency of the search strategy

3. Accelerate the evaluation strategy



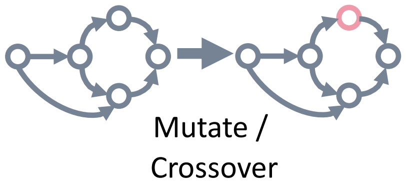
2. Factorize or partition the search space to reduce space complexity.



Direction 1: Improve the sample efficiency of search strategy

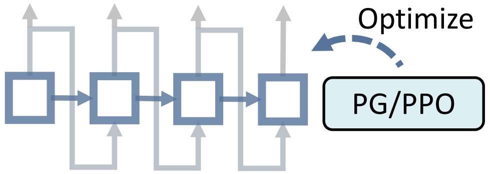
- How to sample new architectures given evaluated architectures and their rewards? Decrease #architectures that need to be evaluated before finding a well-performing one.
  - Search strategy types

Local Search[Real et al., 2019]



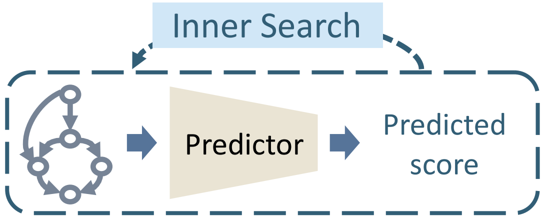
Sample new candidates by applying **local** mutations / cross-overs on past candidates

Reinforcement Learning[Zoph et al., 2017]



- **RL** to learn the sampler from past experiences
- Use the sampler to sample new candidates

Predictor-based[Luo et al., 2018]



- Learn a **predictor** from past experiences
- Predict performances of unseen architectures and sample new candidates

  - The search strategy design should consider **two aspects that influence the sample efficiency**
    - **Exploitation:** Exploiting the already evaluated information (experiences) to sample promising architectures, i.e., avoid sampling expectedly poor-performing ones
    - **Exploration**: Avoid getting stuck in local optima; Sample uncertain architectures (whose evaluations are informative)

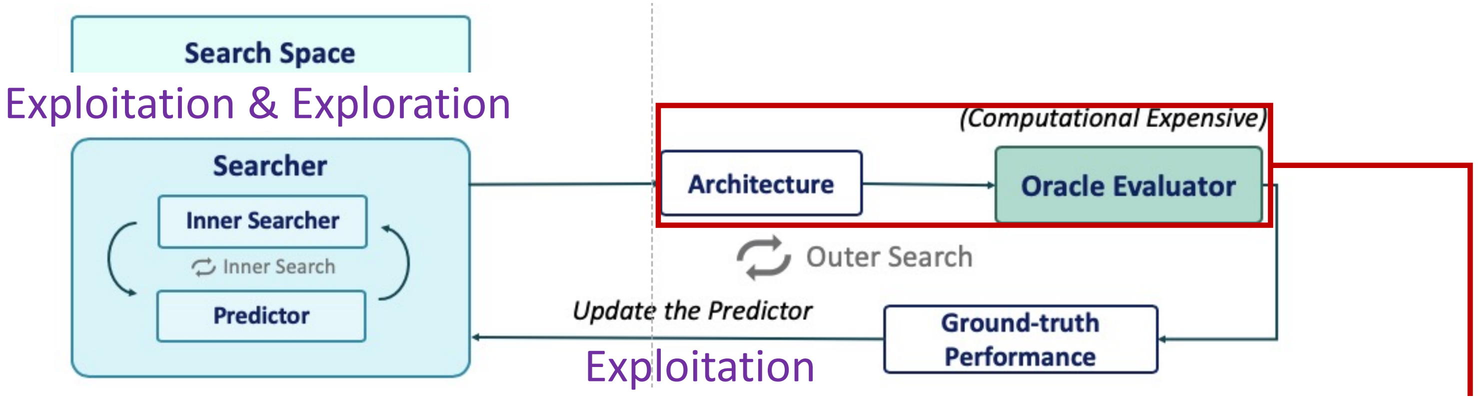
Zoph et al., Neural Architecture Search with Reinforcement Learning, ICLR’17.
Real et al., Regularized evolution for image classifier architecture search, AAAI’19.
Luo et al., Neural architecture optimization, NeurIPS’18.

Direction 1: Improve the sample efficiency of search strategy

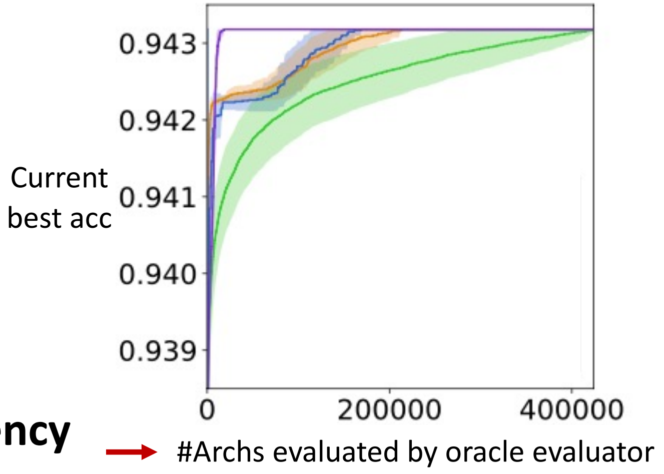
- How to sample new architectures given evaluated architectures and their rewards?
  - **Exploitation of past experiences**: Exploiting the already evaluated information (experiences) to sample promising architectures, i.e., avoid sampling expectedly poor-performing ones
    - Local Search method: Mutate from well-performing architectures in past experiences
    - Reinforcement Learning method: Learn the sampler from past experiences
    - Predictor-based / Bayesian Optimization method: Learn the predictor from past experiences
  - **Exploration of unknown areas**: Avoid getting stuck in bad local optima; Explore uncertain architectures (whose evaluations are informative)
    - Local Search method: Population sample & update design
    - Reinforcement Learning method: ϵ-greedy exploration
    - Predictor-based / Bayesian Optimization method: Acquisition function design

Example of enhancing exploitation: Predictor-based NAS

- Predictor-based NAS

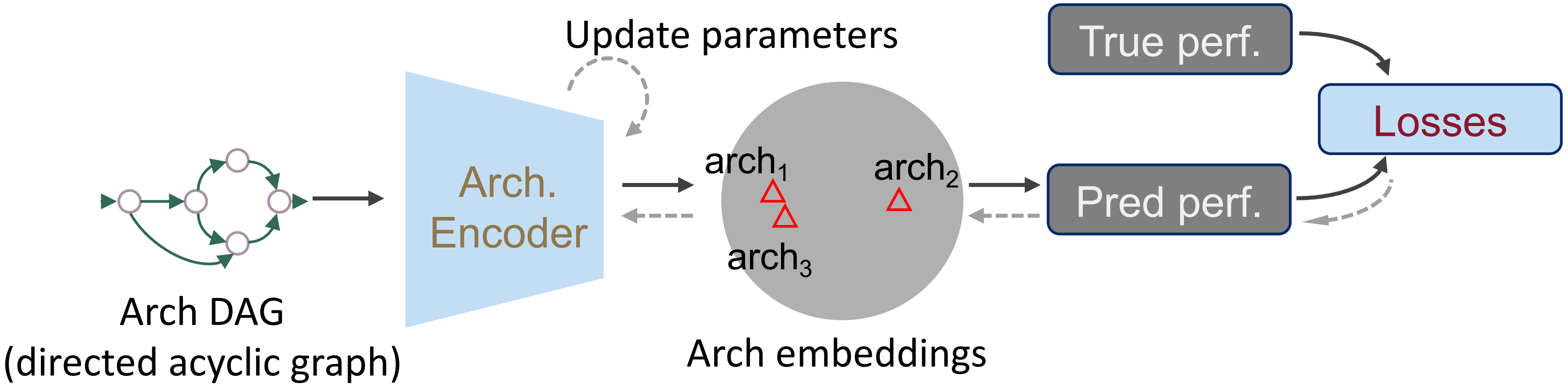


Example plot of sample efficiency



The predictor's fitness is vital to the predictor-based searcher's sample efficiency

- Typical parametric predictor construction



Enhancing Exploitation

"Trained with the same set of true perf. data, how can we get accurate predictions for unseen architectures?"

Should mind 2 aspects
- How to encode
- How to train

We'll cover it later!

Ning et al., "A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS", ECCV'20.

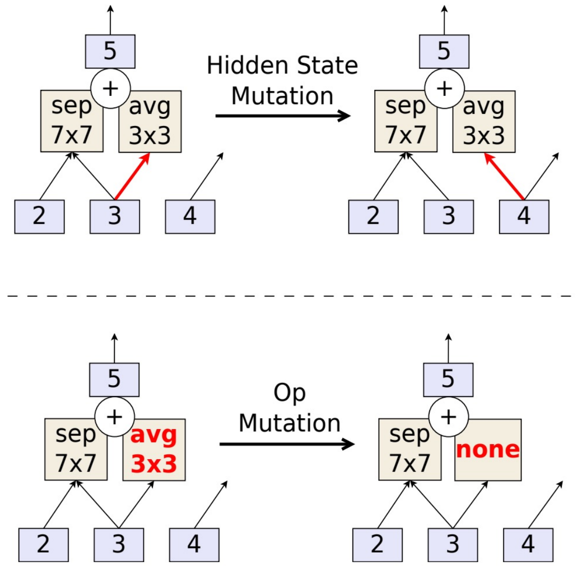
Example of enhancing exploration: Aging evolutionary

- Modify the **population update** strategy in the evolutionary (local search) method

1. **Choose** parent
   1. Randomly select a parent pool with size S from population — Exploration
   2. Choose the highest-accuracy model in parent pool as parent — Exploitation
2. **Mutate** parent to get arch
3. **Evaluate** arch
4. **Population update**
   1. **Add** arch into population
   2. **Eliminate** the **oldest** arch in population

⇩

Exploration: **Aging evolution** encourages exploration by avoiding “zooming in on good models too early”. **Forced to pay attention to architectures rather than models: architecture should retrain well.**

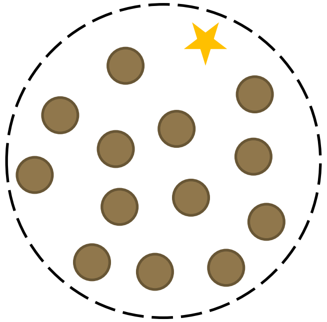


Mutation in NASNet search space

Real et al.., “Regularized Evolution for Image Classifier Architecture Search”, AAAI’19.

Direction 2: Reduce the search space complexity

- **Factorize** (exponential complexity decrease) or **partition** (linear complexity decrease)?



Original space $S = d_1 \times d_2 \times d_3$

S_1 **Partition** S_2

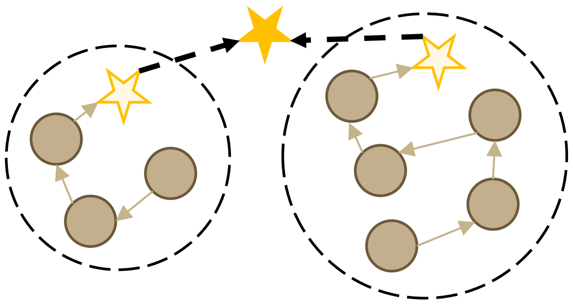
d_1: $c_1^1\ c_1^2, c_1^3, c_1^4$

d_2: $c_2^1\ c_2^2, c_2^3, c_2^4$

d_3: $c_3^1\ c_3^2, c_3^3, c_3^4$

S_1 **Factorize** S_2

Factorization[Liu et al., 2018]



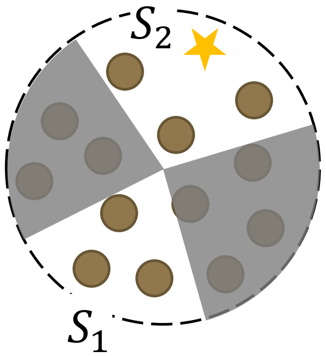
$S_1 = d_1$ $\qquad$ $S_2 = d_2 \times d_3$

Search in S.

$S = S_1 \times S_2$

- Search in S_1
- Search in S_2

Partition[Ci et al., 2021][Guo et al., 2020]



Search in S. $S = d_{12} \times d_3 = (d_{12}^1 + d_{12}^2) \times (d_3^1 + d_3^2) \approx d_{12}^1 \times d_3^1 + d_{12}^2 \times d_3^2$

- Search in $S_1 = d_{12}^1 \times d_3^1$
- Search in $S_2 = d_{12}^2 \times d_3^2$

- If factorize, how to evaluate partial architecture?
- How to orchestrate the search in different space factors & partitions?

Liu et al., "Progressive Neural Architecture Search", ECCV'18. Ci et al., "Evolving Search Space for Neural Architecture Search", ICCV'21.
Guo et al.., "Breaking the Curse of Space Explosion: Towards Efficient NAS with Curriculum Search", ICML'20.

Direction 3: Accelerate the evaluation strategy

- How to get the parameters of an architecture?
  - **Standalone**: Standalone train from scratch[Zoph et al., 2017]
  - **Morphism**: Inherit from parents & finetune, (used in conjunction with local search method, e.g., evolutionary)[Cai et al., 2018]
  - **One-shot**: Amortize the parameter training costs of multiple architectures
    - **SuperNet-based**: Share parameters in one SuperNet with other architectures[Pham et al., 2018]
    - **HyperNet-based**: Generate parameters using one HyperNet for all architectures[Brock et al., 2018]
  - **Zero-shot**: Randomly initialize parameters[Abdelfattah et al., 2021]

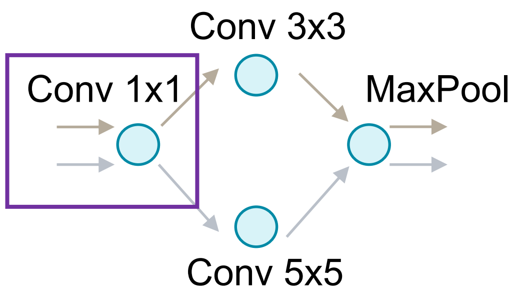
A widely recognized **trade-off between the "accuracy" and "efficiency"** of evaluation strategy: Overly strong proxy of getting parameters (excessive parameter sharing, zero-shot) results in inaccurate evaluation, and thus suboptimal search results.

Zoph et al., "Neural architecture search with reinforcement learning. " ICLR'17.
Pham et al., "Efficient Neural Architecture Search via Parameters Sharing." ICML'18.
Cai et al., "Efficient architecture search by network transformation." AAAI'18.
Brock et al., "Smash: one-shot model architecture search through hypernetworks." ICLR'18.
Abdelfattah et al., Zero-cost proxies for lightweight NAS, ICLR'21.

One-shot Evaluation (Sharing parameters)

- **Parameter sharing technique**[Pham et al., 2018][Bender et al., 2018]
  - Construct the SuperNet: Construct an over-parametrized network called SuperNet, containing the parameters needed by all the architectures in the search space

    Parameters are shared between many architectures
    E.g., Conv 1x1 shared between architectures
    - 1x1 -> 3x3 -> MaxPool
    - 1x1 -> 5x5 -> MaxPool



  - Train the SuperNet: Randomly sample architectures, and train the corresponding parameter subset in the SuperNet
  - Evaluating Candidates: Evaluate architectures using the corresponding subset of SuperNet parameters, without separate training the parameters for each candidate

Pham et al., "Efficient Neural Architecture Search via Parameters Sharing." ICML'18.
Bender et al. "Understanding and Simplifying One-Shot Architecture Search." ICML'18.

Zero-shot Evaluation (Randomly initialized parameters)

- **One-Shot** Evaluators avoid separately training each architecture. Instead, the training costs of all architectures are amortized into the cost of training **ONE** SuperNet

Can we further reduce the training cost to **ZERO**?
Zero-Shot Evaluators

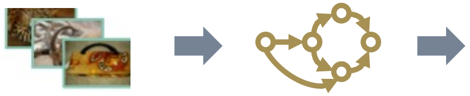
Parameter-level ZSEs[Abdelfattah et al., 2021] measure the architecture's score by adding up parameter-wise sensitivity measures

e.g. $S(\alpha) = \sum \frac{\partial L}{\partial \theta} \theta$ add up parameter-wise sensitivities of all parameters

Architecture-level ZSEs[Mellor et al., 2020, 2021] measure the architecture's score (discriminability) by inference differences between input images.

[Mellor, et al., arXiv 2020]
input gradients difference of different inputs

$$jacob\_cov(\alpha) = f\left(\frac{\partial L}{\partial x_1}, \dots, \frac{\partial L}{\partial x_n}\right)$$



Loss L

$x_1, \dots x_n$

randomly initialized θ

Abdelfattah et al., "Zero-cost proxies for lightweight NAS", ICLR'21.
Mellor et al., "Neural architecture search without training", arXiv:2006.04647v1, 2020.
Mellor et al., "Neural architecture search without training", ICML 2021.

Quality of These Proxy Efficient Evaluators?

One-shot and Zero-shot Evaluators (OSEs and ZSEs) are efficient.
Are current OSEs and ZSEs powerful enough for evaluating architectures on various search spaces?

How are the OSE and ZSE scores correlated with the architectures' true ranking?

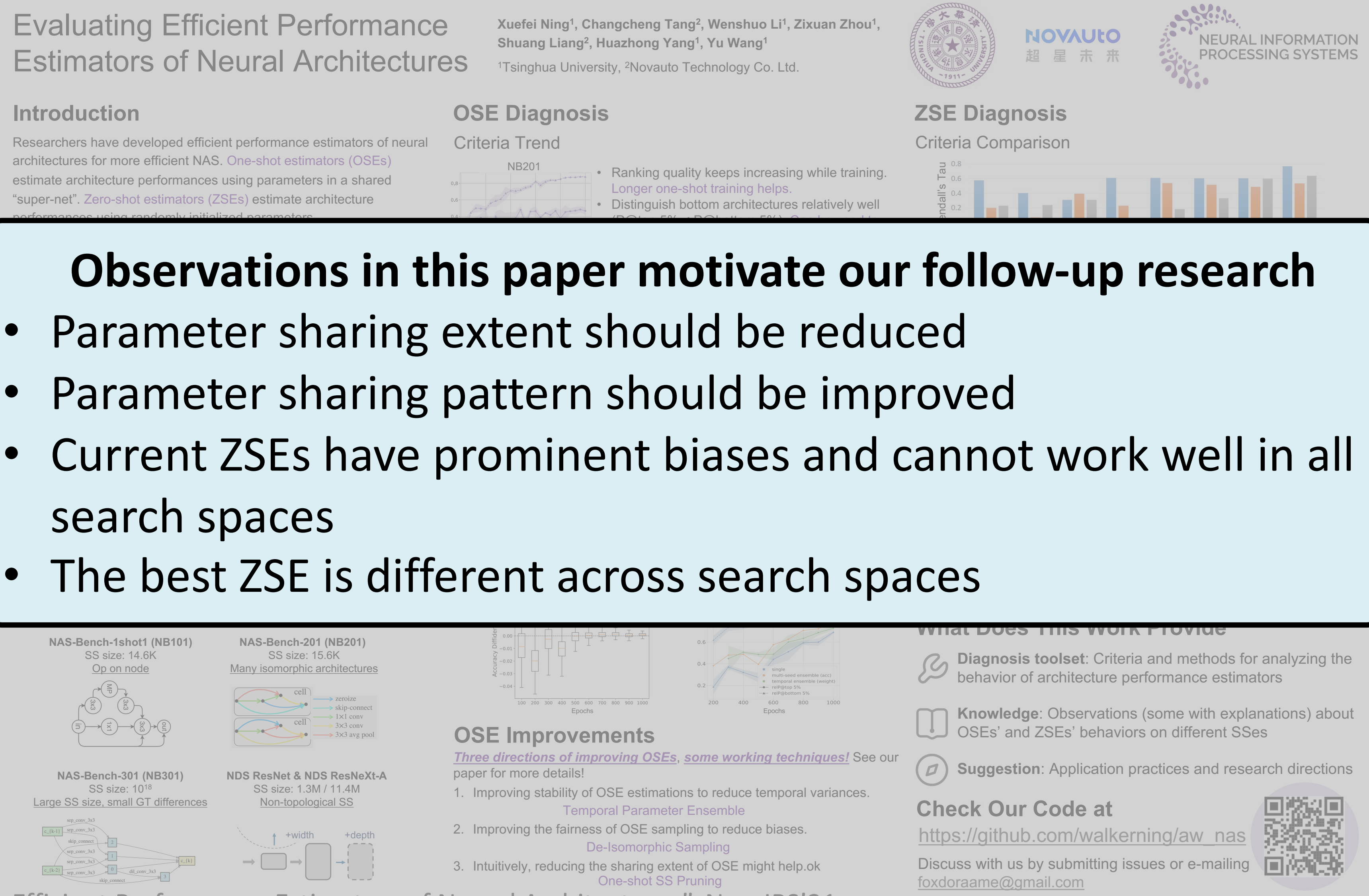
What architectures do they overestimate or underestimate?

Is there a general ZSE that is powerful on different types of search spaces.

What should we do to further improve OSEs and ZSEs?

- **Targets**: OSEs and **eight** types of ZSEs.
- **Aspects**:
  - Kendall's Tau & SpearmanR: Overall ranking correlation.
  - P@top / bottom K & Best / WorstRanking@K: Distinguishing ability of top or bottom architectures.
  - Bias: Which architectures are over- or under-estimated?
  - Variance: How rankings vary w.r.t. random factors?

Our Work at NeurIPS'21



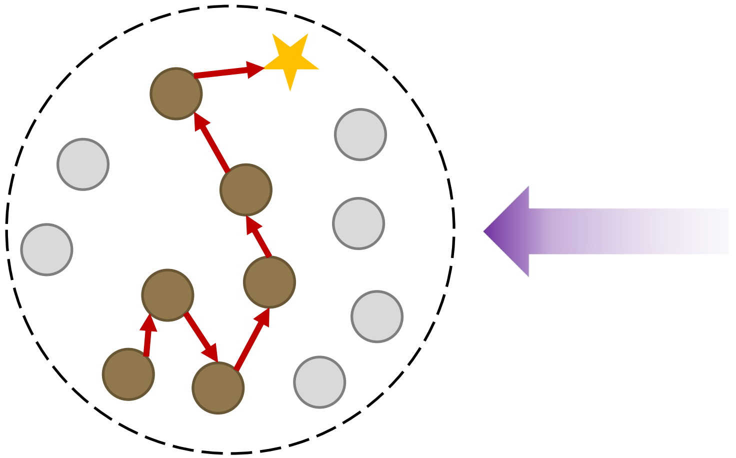
Observations in this paper motivate our follow-up research

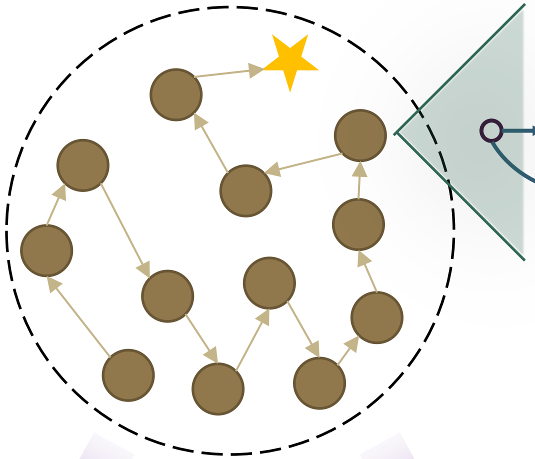
- Parameter sharing extent should be reduced
- Parameter sharing pattern should be improved
- Current ZSEs have prominent biases and cannot work well in all search spaces
- The best ZSE is different across search spaces

Ning et al., "Evaluating Efficient Performance Estimators of Neural Architectures", NeurIPS'21.

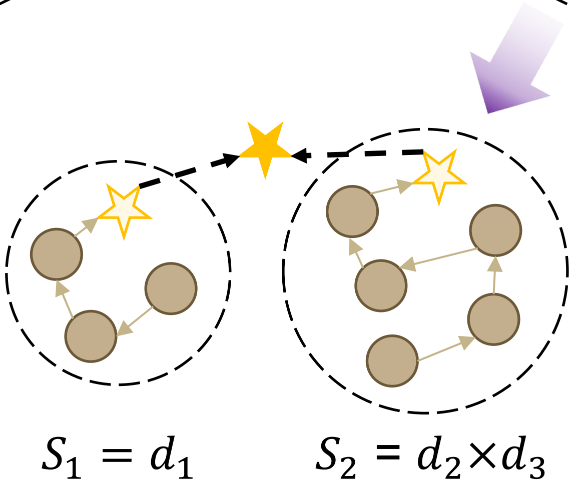
(Review) Directions for Efficient Search

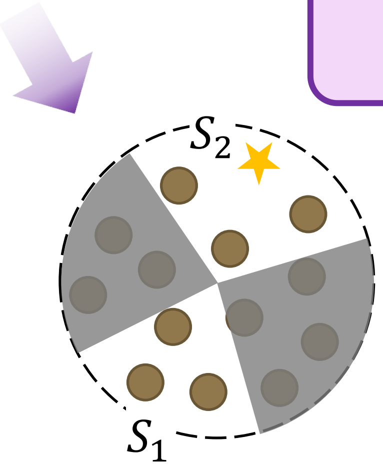
1. Improve the sample efficiency of the search strategy





3. Accelerate the evaluation strategy





2. Factorize or partition the search space to reduce space complexity.

Menu

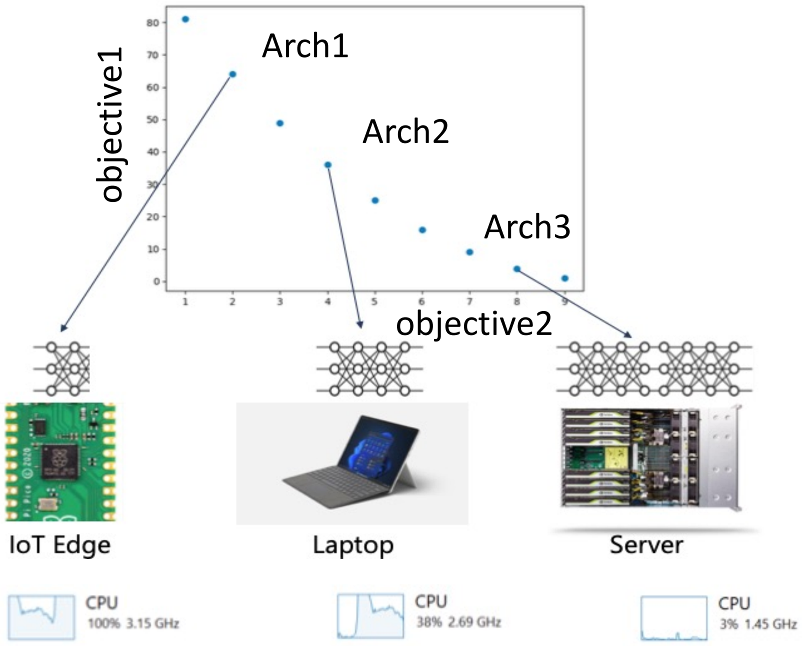
1. Basics
2. Field Summary: Questions and Development
    a) What to Search
    b) How to Search
    c) **What Can Search Tell Us**
3. Our Work: Utilizing Architecture Encoding to Answer “How to Search”
4. Summary

What does NAS Return: Architecture, Pareto, Choice Rules

- Instead of returning an architecture, it can be useful for NAS to return:

Pareto Curve: Multiple <u>architectures</u> to achieve Pareto Perf[Elsken et al., 2018][Cai et al., 20][Dey et al., 2022]

Easy deployment for varying budgets / hardware



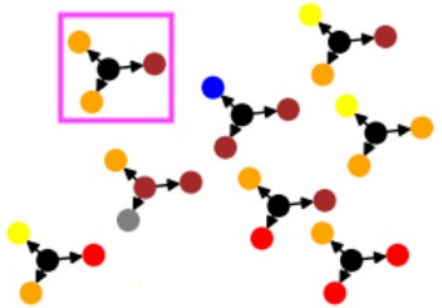
Different hardware
Different budget

Dynamic budget

Choice Rules: Rules of decision choices to achieve good perf

Easy interpretability for human:
<u>Choice</u> rules are <u>low-dimensional constraints</u> that can "efficiently" describe a <u>large sub-space</u> of <u>architectures</u>

NASBOWL [Ru et al., 2021][extracts topology structure



RegNet[Dey et al., 2022] extracts choice relationship between decision

$+\, b_{i+1} = b_i$
$+\, g_{i+1} = g_i$
$+\, w_{i+1} \geq w_i$
$+\, d_{i+1} \geq d_i$
quantized linear

Elsken et al., "Efficient Multi-Objective Neural Architecture Search via Lamarckian Evolution", ICLR'19.
Cai et al., "Once for All: Train One Network and Specialize it for Efficient Deployment", ICLR'20.
Dey et al., Neural Architecture Search Tutorial Part2, AutoML'22.
Radosavovic et al., "Designing Network Design Spaces", CVPR'20.
Ru et al., "Interpretable Neural Architecture Search via Bayesian Optimization with WL Kernels", ICLR'21.

History

- NAS Efficiency Improvement
- Hardware-Aware, Co-Exploration
- Search Space Analysis & Auto Design
- Benchmark

2017

Zoph@Google, ICLR'17
Neural Architecture Search with Reinforcement Learning

Zoph@Google, CVPR'18
NASNet
More compact cell-based micro search space design

2018

Pham@Google, ICML'18
ENAS
One-shot evaluation

Luo@MSRA, NIPS'18
NAO
Predictor-based NAS controller

Liu@DeepMind & CMU, ICLR'19
DARTS Differentiable NAS controller

Real@Google, AAAI'19
AmoebaNet Regularized evolution controller

Tan@Google, CVPR'19
MNasNet
Hardware-aware search with tailored search space design

2019

Ying@Google&UFreiburg, ICML'19
NAS-Bench-101
The first NAS benchmark

Radosavovic@FAIR, ICCV'19
NDS evaluating search spaces

Xie@FAIR, ICCV'19
Randomly Wired NN analyzing search spaces

Cai@MIT ICLR'20
Once-for-all
One-shot NAS for multiple hardware

Radosavovic@FAIR, CVPR'20
RegNet Auto design of non-topological search space

Yang@UND, DAC'20
Co-exploration ASIC and NAS

2020

Ning@Tsinghua&Tencent, ECCV'20
GATES Improving predictor-based controller

Abdelfattah@Samsung, ICLR'21
Zero-cost proxies
Zero-shot evaluation w.o. training

Zhao@WPI&UBrown&FAIR, ICML'21
Few-shot NAS
Improving one-shot evaluation

2021

Ning@Tsinghua&Novauto, NIPS'21
Evaluating Efficient Estimation Strategies
Analyze one-shot / zero-shot evaluation

Mehta@UFreiburg&Abacus&BCAI, ICLR'22
NAS-Bench-Suite
A collection of NAS Benchmarks

2022

Zhou@Tsinghua, ECCV'22
CLOSE
Improving one-shot evaluation

Menu

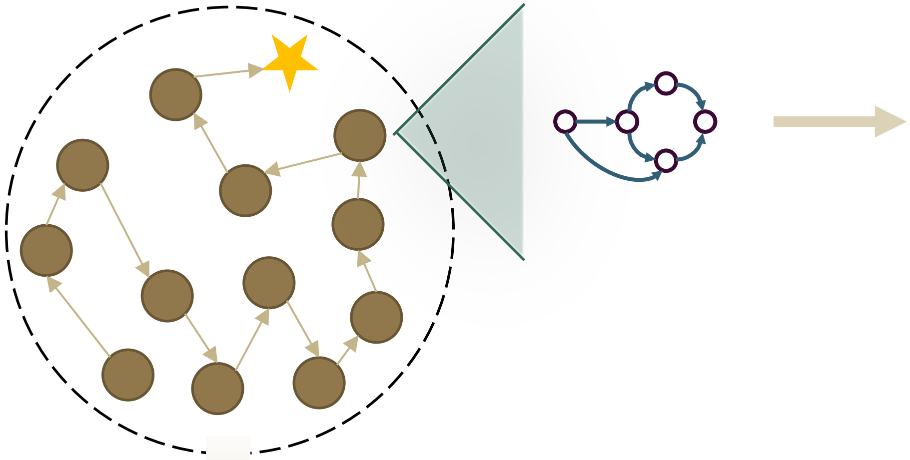
Menu

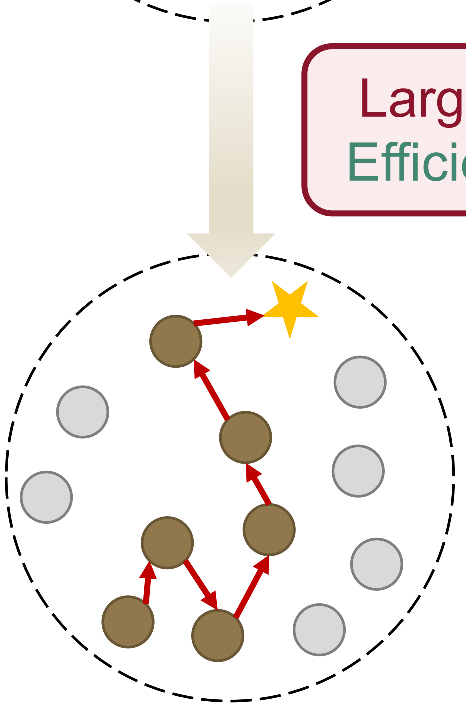
- Utilizing Architecture Encoding for Efficient and Accurate Search
- How to Encode: GATES@ECCV’20, TA-GATES@NeurIPS’22
- How to Learn the Encoder: GATES@ECCV’20, CLOSE@ECCV’22, DELE@AAAI’23

Menu

- Utilizing Architecture Encoding for Efficient and Accurate Search
- How to Encode: GATES@ECCV'20, TA-GATES@NeurIPS'22
- How to Learn the Encoder: GATES@ECCV'20, CLOSE@ECCV'22, DELE@AAAI'23

Review: Two Basic Challenges in “How to Search”



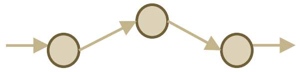


Large search space
Efficient exploration?

Evaluation: How to get the parameters

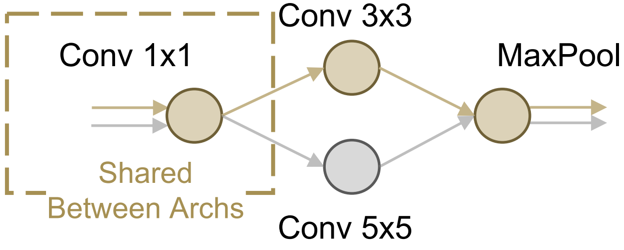
Standalone Training

Training on Train Set



Low Efficiency

Parameter-Sharing Training

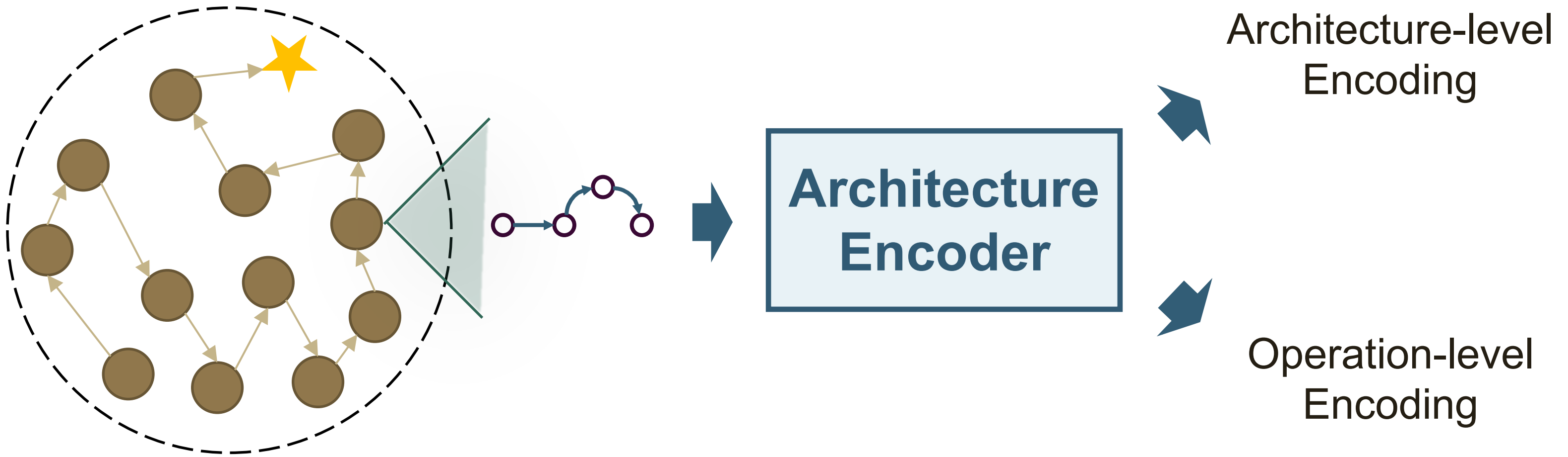


Low Quality

Cannot guarantee efficiency and quality in the meantime
How to improve the quality of parameter-sharing evaluation?

Our Major Idea

Utilize the learnable encoding of architecture to accelerate exploration and improve the quality of evaluation



Accelerate Exploration
Which <u>architectures</u> are promising and need explore

Improve Evaluation
Which <u>operations</u> to share parameters

Then, how do we get a good architecture encoding?

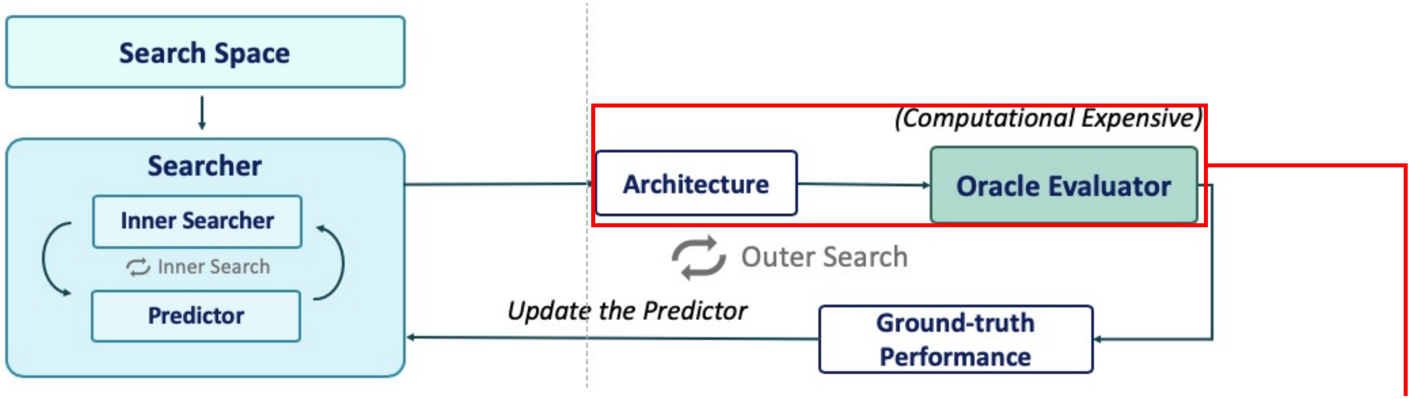
- How to encode?
- How to learn the encoder?

Menu

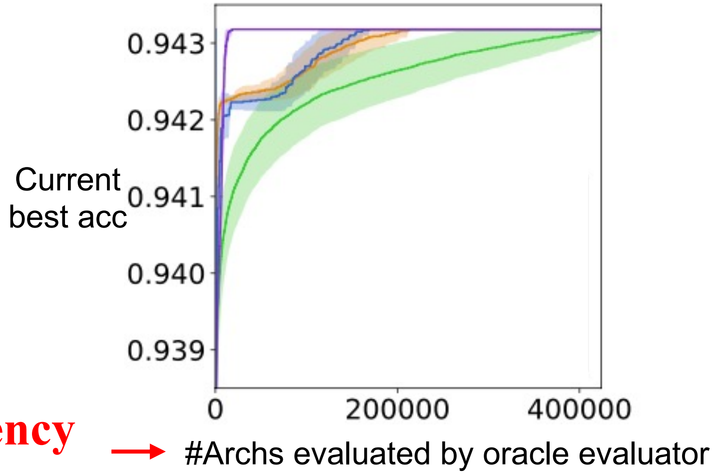
- Utilizing Architecture Encoding for Efficient and Accurate Search
- How to Encode: GATES@ECCV’20, TA-GATES@NeurIPS’22
- How to Learn the Encoder: GATES@ECCV’20, CLOSE@ECCV’22, DELE@AAAI’23

GATES: Background

- Predictor-based NAS

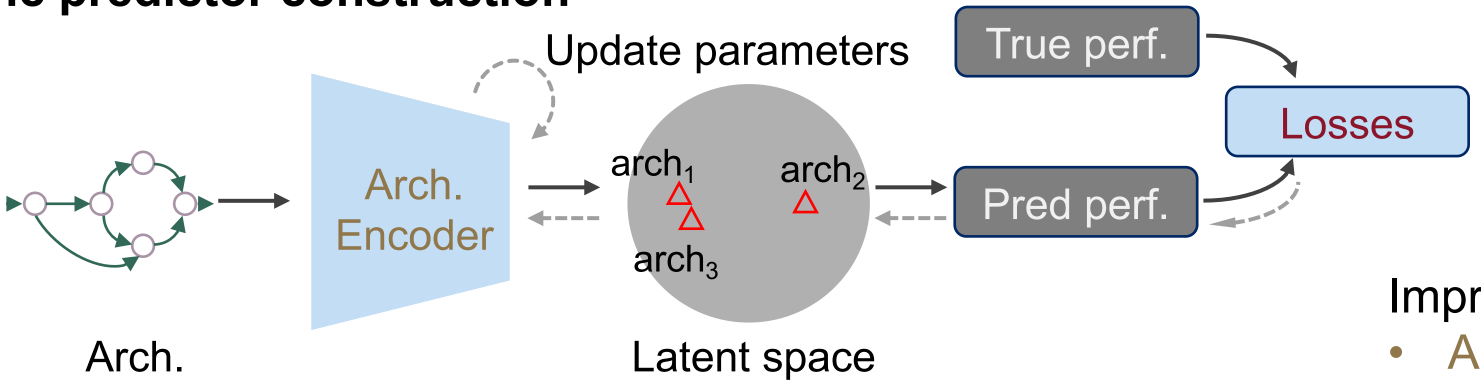


Example plot of sample efficiency



The predictor's fitness is vital to the predictor-based searcher's sample efficiency

Typical parametric predictor construction

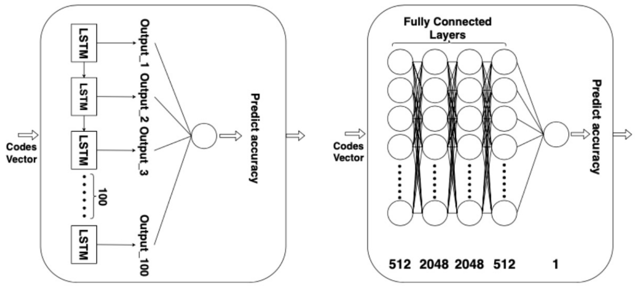


"Trained with the same set of true perf. data, we want to get more accurate predictions for unseen architectures."

Improvements from 2 aspects
- Arch. encoder
- Training loss

GATES: Existing Methods

- **Arch. Encoder**

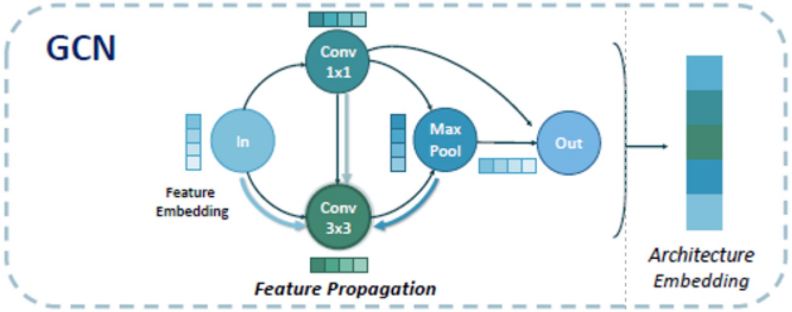


RNN MLP

Sequence-based encoder [Luo et al. NIPS 2018]

Not suitable for handling DAGs

An architecture and its isomorphic counterparts
can have multiple different encodings



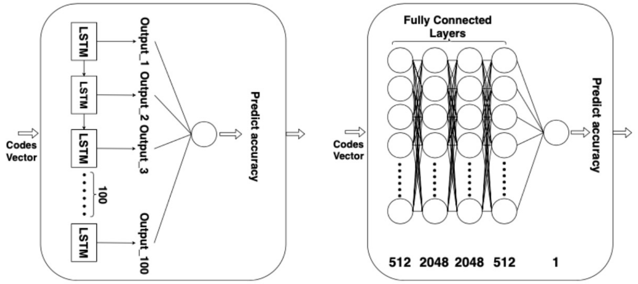
GCN-based encoder [Guo et al. NIPS 2019, Shi et al. 2019]

Not suitable for handling **data-processing DAG** (NN architecture)

Existing GCN encoder models the operation (Conv, Pooling) as the propagated information on the graph. The aggregation method does not intuitively match the node/edge semantics of data-processing DAG

GATES: Existing Methods

- **Arch. Encoder**

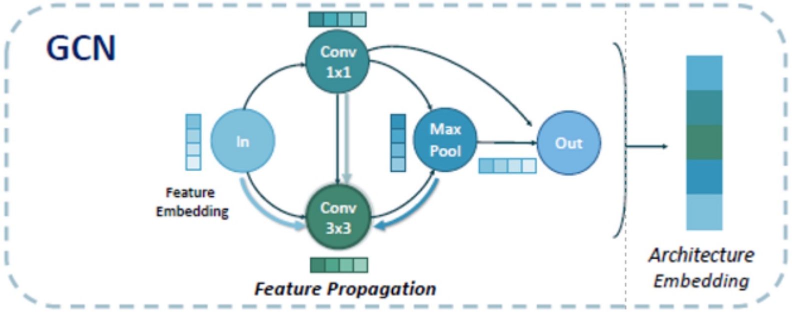


RNN MLP

Sequence-based encoder [Luo et al. NIPS 2018]

Not suitable for handling DAGs

An architecture and its isomorphic counterparts can have multiple different encodings



GCN-based encoder [Guo et al. NIPS 2019, Shi et al. 2019]

Not suitable for handling **data-processing DAG** (NN architecture)

Existing GCN encoder models the operation (Conv, Pooling) as the propagated information on the graph. The aggregation method does not intuitively match the node/edge semantics of data-processing DAG

- **Training loss**

What is important in NAS is the relative ranking order of architectures, not the absolute score

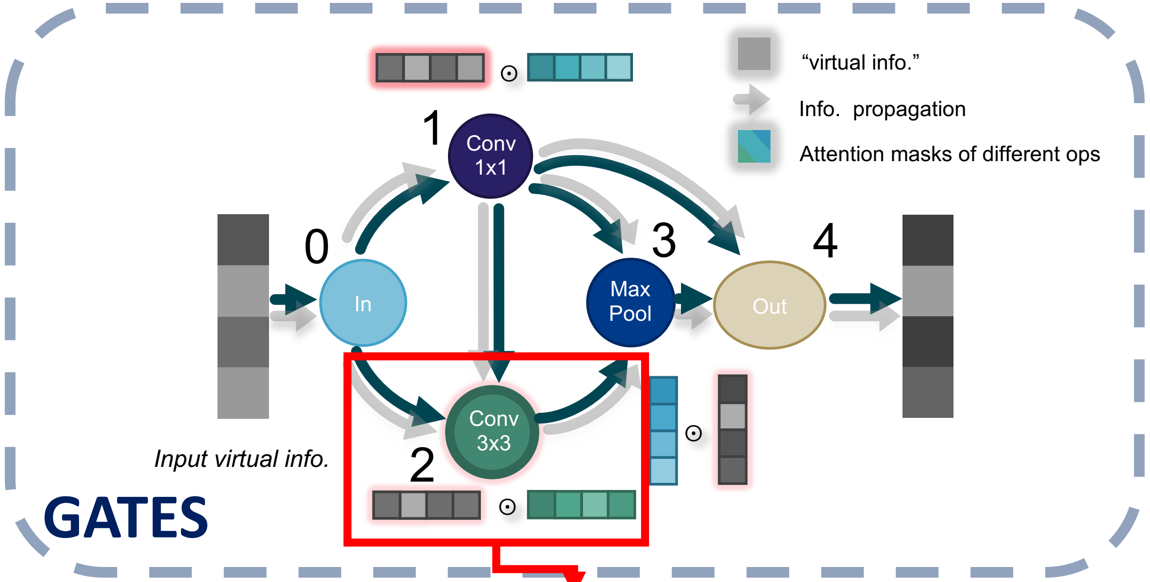
- Regression loss: make predicted score $\mathrm{P}(a_j)$ close to true performance y_j

$$L(\{a_j, y_j\}_{j=1,\cdots,N}) = \sum_{j=1}^{N} (\mathrm{P}(a_j) - y_j)^2$$

L is not a good surrogate of the ranking measures

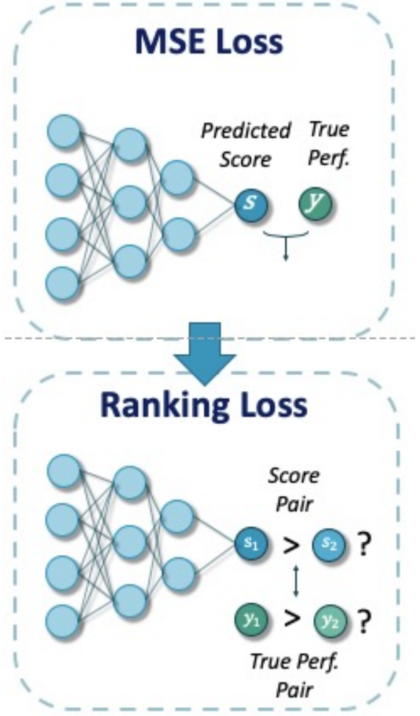
GATES

- Improve **Encoder** and **Training losses**
  - A more generic **G**raph-based neural **A**rchi**T**ecture **E**ncoding **S**cheme (**GATES**)
    - Mimic the information propagation in the architecture to encode it
  - Learning to Rank (LtR) losses (Relative order matters rather than absolute perf.)
    - Ranking Losses are better surrogate of ranking measures than regression losses



- feature map computation: F2=Conv3x3(F0 + F1)
- "Virtual info transformation" during architecture encoding: N2=$m_2 \odot$ (N0+N1)

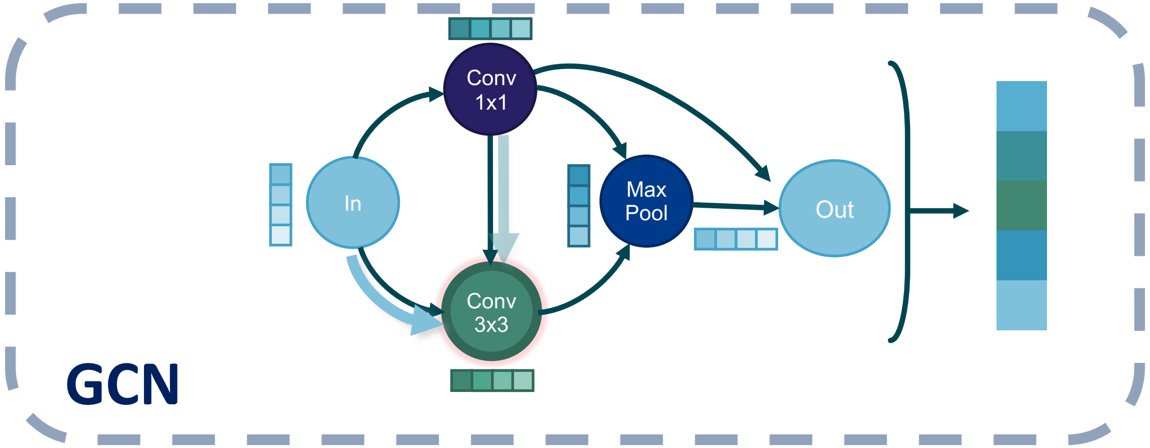
$m_2 = \sigma(\mathrm{EMB}_{Conv3x3}\mathrm{W_o})$is the **attention mask** of Conv3x3



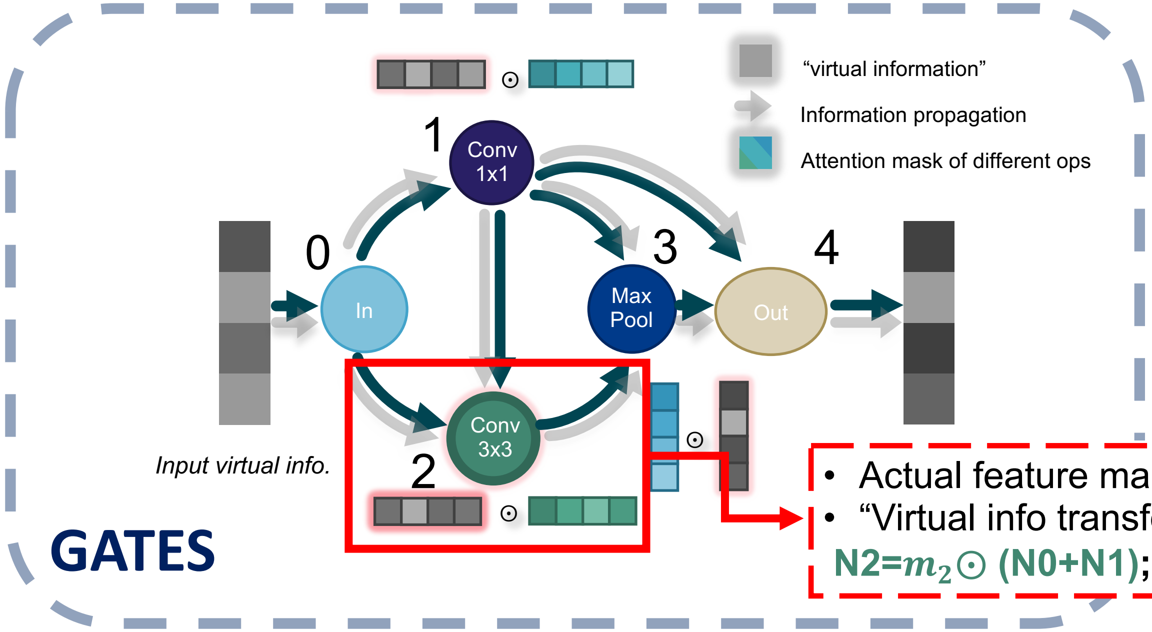
$$\sum_{j=1}^{N}(\mathrm{P}(a_j)-y_j)^2$$

$$\sum_{j=1}^{N}\sum_{i,y_i>y_j}\max[0, m-(P(a_i)-P(a_j))]$$

GATES



- **Operation** modeled as the information to be propagated on the graph
- **Architecture encoding**: After the information is propagated for several steps, the rep. of all nodes are read out (aggregated) as the architecture representation



Mimic the information propagation of NN computation

- **Operation** modeled as the transformation/processing of the propagating information (attention mask)
- **Architecture encoding** : output "information" is used as the architecture representation

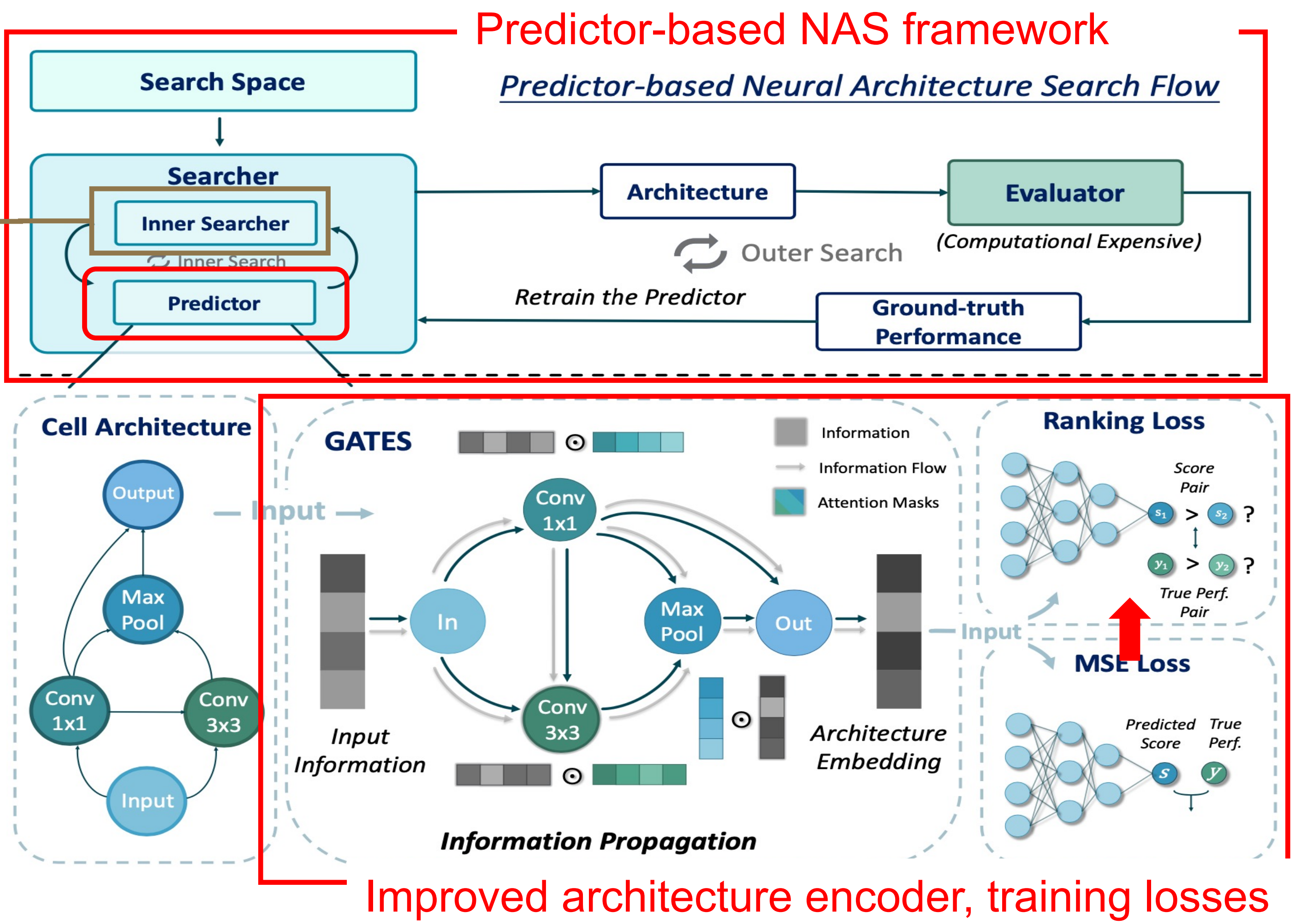
- Actual feature map computation: **F2=Conv3x3(F0 + F1)**
- "Virtual info transformation" in the arch. encoding process:

N2=$m_2\odot$ (N0+N1); $m_2 = \sigma(\mathrm{EMB}_{Conv3x3} \mathrm{W}_o)$ is the **attention mask** of Conv3x3

Overall framework

- The overall framework of predictor-based NAS with GATES and LtR

- Evolutionary Algorithm (EA)
- Random Search (RS)



Results on NAS-Bench-101

- Ranking correlation (Kendall's Tau) of the predictors
  - Encoder comparison

Encoder	Proportions of 381262 training samples							
	0.05%	0.1%	0.5%	1%	5%	10%	50%	100%
MLP [21]	0.3971	0.5272	0.6463	0.7312	0.8592	0.8718	0.8893	0.8955
LSTM [21]	0.5509	0.5993	0.7112	0.7747	0.8440	0.8576	0.8859	0.8931
GCN (w.o. global node)	0.3992	0.4628	0.6963	0.8243	0.8626	0.8721	0.8910	0.8952
GCN (global node) [20]	0.5343	0.5790	0.7915	0.8277	0.8641	0.8747	0.8918	0.8950
GATES	**0.7634**	**0.7789**	**0.8434**	**0.8594**	**0.8841**	**0.8922**	**0.9001**	**0.9030**

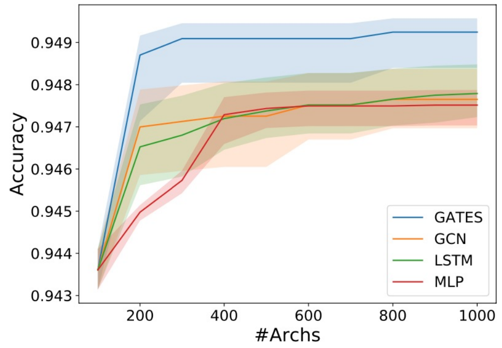
GATES outperform other encoders consistently, especially when there are few training samples

  - Loss function comparison

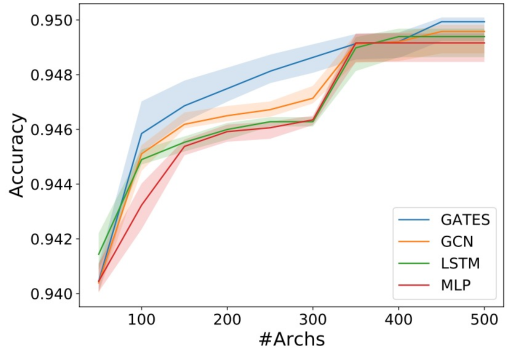
Loss	Proportions of 381262 training samples							
	0.05%	0.1%	0.5%	1%	5%	10%	50%	100%
Regression (MSE) + GCN†	0.4536	0.5058	0.5587	0.5699	0.5846	0.5871	0.5901	0.5941
Regression (MSE) + GATES†	0.4935	0.5425	0.5739	0.6323	0.7439	0.7849	0.8247	0.8352
Pairwise (BCE)	0.7460	0.7696	0.8352	0.8550	0.8828	0.8913	0.9006	0.9042
Pairwise (Comparator)	0.7250	0.7622	0.8367	0.8540	0.8793	0.8891	0.8987	0.9011
Pairwise (Hinge)	0.7634	0.7789	0.8434	0.8594	0.8841	0.8922	0.9001	0.9030
Listwise (ListMLE)	0.7359	0.7604	0.8312	0.8558	0.8852	0.8897	0.9003	0.9009

Ranking losses are better surrogate to ranking measures than regression losses

- Sample efficiency
  - Encoder comparison

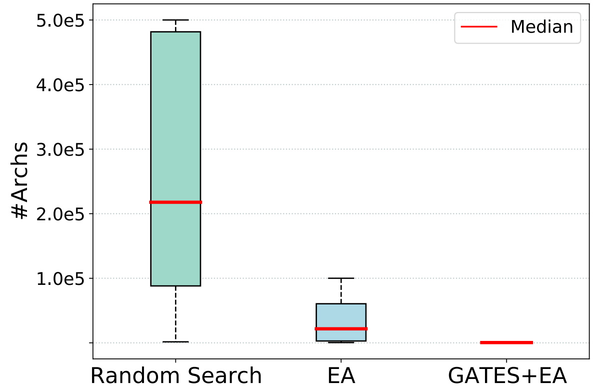


(a) RS inner search method ($r = 500$)



(b) EA inner search method ($r = 100$)

  - Comparison with baseline search strategies

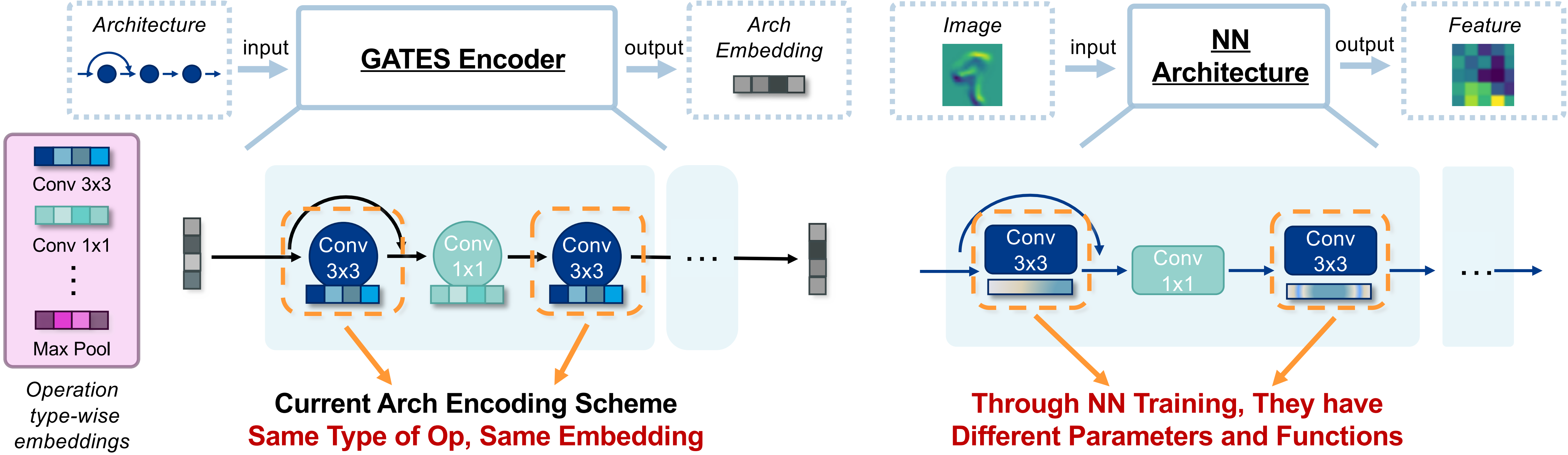


Median: 220k 24k 0.4k

551.0× and 59.25× more efficient than RS/EA

TA-GATES: More Discriminative Encoding of Operation / Architectures

GATES views an architecture as a **DAG with operations**.
Drawback: Neglect the "operations are trainable" property of NN architecture.

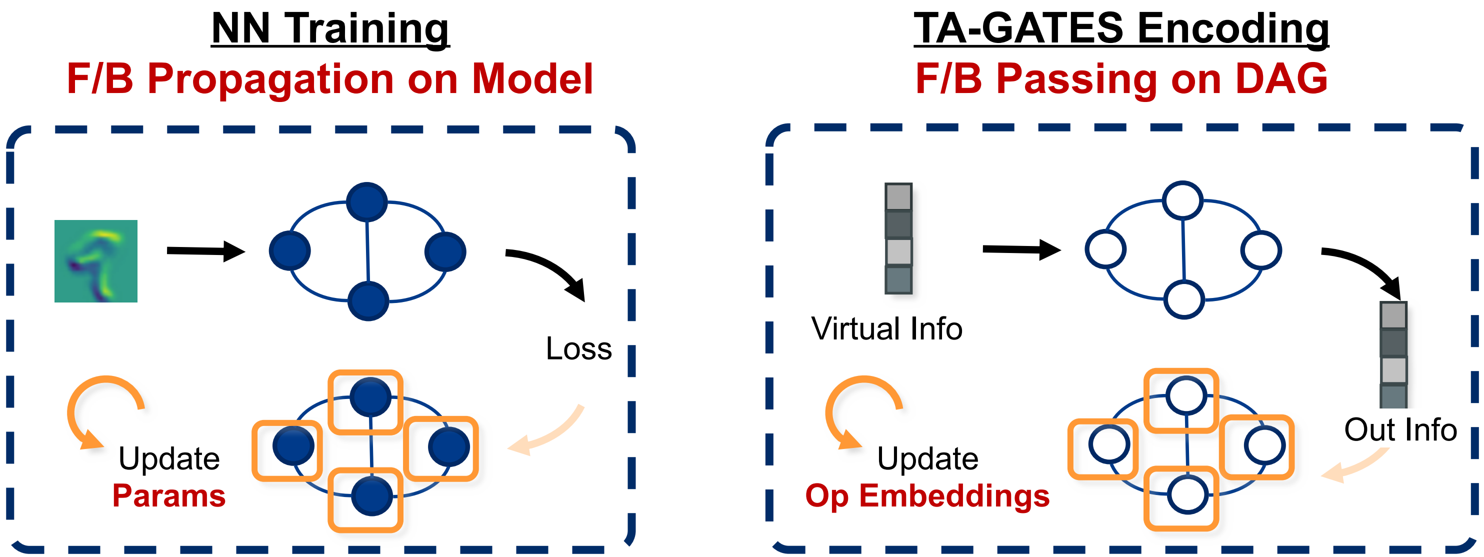


To improve the discriminative modeling of operation and architecture,
should **give contextualized embeddings for operations according to the architectural context**.

TA-GATES: A Training-Analogous Encoding Scheme

An NN architecture **determines the NN training dynamics**. And it is precisely through the training process that an operation interacts with other operations (the architectural context) and gets its parameters and functionalities.

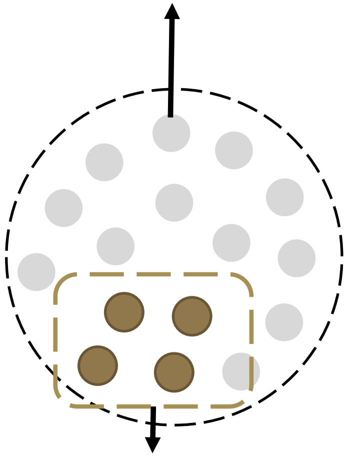
TA-GATES is designed considering this intrinsic property of NN architectures. It encodes architectures in an “**encoding by training-mimicking**” manner, **naturally providing discriminative operation and architecture embeddings**.



Results: Comparison with Baseline Encoders

Spaces: NB101, NB201, NB301, NDS ENAS
Measures: Kendall's Tau, Precision@K, Mean Square Error (MSE), Pearson coefficient of LC

2. Predict the performances of unseen architectures, measure how close the predictions are to the GT performances (ranking and regression quality)



1. Train the encoder using the GT performances of some architectures

Kendall's Tau Comparison Example

	Encoder	Proportions of 7290 training samples				
		1%	5%	10%	50%	100%
NB101	MLP [42]	0.3937$_{(0.0302)}$	0.5318$_{(0.0185)}$	0.5703$_{(0.0167)}$	0.6225$_{(0.0078)}$	0.6307$_{(0.0069)}$
	LSTM [42]	0.5476$_{(0.0341)}$	0.5876$_{(0.0245)}$	0.6040$_{(0.0154)}$	0.6196$_{(0.0142)}$	0.6131$_{(0.0185)}$
	GCN (global node) [35]	0.3668$_{(0.0563)}$	0.5973$_{(0.0233)}$	0.6927$_{(0.0108)}$	0.7520$_{(0.0075)}$	0.7689$_{(0.0083)}$
	NASBOWL [33]	0.5850$_{(0.0232)}$	0.6416$_{(0.0241)}$	0.6536$_{(0.0193)}$	0.6833$_{(0.0022)}$	0.6872$_{(0.0000)}$
	SemiNAS [22]	0.5273$_{(0.0589)}$	0.6055$_{(0.0294)}$	0.5953$_{(0.0279)}$	0.6040$_{(0.0284)}$	0.6043$_{(0.0179)}$
	XGBoost [44]	0.4517$_{(0.0470)}$	0.5987$_{(0.0365)}$	0.5680$_{(0.0125)}$	0.5677$_{(0.0077)}$	0.6175$_{(0.0000)}$
	GATES [27]	0.6321$_{(0.0251)}$	0.7493$_{(0.0166)}$	0.7690$_{(0.0077)}$	0.7999$_{(0.0071)}$	0.8119$_{(0.0071)}$
	TA-GATES	**0.6686**$_{(0.0338)}$	**0.7744**$_{(0.0211)}$	**0.7839**$_{(0.0063)}$	**0.8133**$_{(0.0053)}$	**0.8217**$_{(0.0057)}$
	Encoder	Proportions of 7813 training samples				
		0.1%	0.5%	1%	5%	10%
NB201	MLP [42]	0.0162$_{(0.0859)}$	0.0863$_{(0.0556)}$	0.1756$_{(0.0332)}$	0.3885$_{(0.0237)}$	0.5492$_{(0.0092)}$
	LSTM [42]	0.1935$_{(0.1806)}$	0.5079$_{(0.0715)}$	0.5691$_{(0.0110)}$	0.6690$_{(0.0189)}$	0.7395$_{0.0061)}$
	Line-Graph GCN [35]	0.2461$_{0.1549)}$	0.3113$_{0.0626)}$	0.4080$_{0.0369)}$	0.5461$_{0.0138)}$	0.6095$_{0.0164)}$
	NASBOWL [33]	0.4980$_{(0.0408)}$	0.6674$_{(0.0077)}$	0.5912$_{(0.0874)}$	0.7259$_{((0.0098)}$	0.7625$_{(0.0083)}$
	XGBoost [44]	0.0706$_{(0.1238)}$	0.3719$_{(0.0560)}$	0.4178$_{(0.0288)}$	0.6412$_{(0.0053)}$	0.7084$_{(0.0123)}$
	GATES [27]	0.4309$_{0.1062)}$	0.6702$_{0.0254)}$	0.7571$_{0.0169)}$	0.8583$_{0.0019)}$	0.8823$_{0.0024)}$
	TA-GATES	**0.5382**$_{(0.0478)}$	**0.6707**$_{(0.0256)}$	**0.7731**$_{(0.0249)}$	**0.8660**$_{(0.0060)}$	**0.8890**$_{(0.0049)}$

Trained with the same architecture-performance pairs, TA-GATES consistently outperforms other encoders

Summary: Methodology of Encoding - Mimic the Information Flow

Need to consider special properties of the input space!
Mimic the propagation and processing of actual data by architecture a to encode a

GATES[1]

a describes how the data are propagated and processed.

Mimic the data propagation & processing to encode a.

1. **Graph-based**: Map architectures with isomorphic graphs to the same embedding.
2. Special **aggregation function**: Reasonable aggregation of other nodes' passed information and local node's feature. Since passed information indicates "data", local feature indicates "data processing" (edge represents "flow direction" instead of "affinity"). These two piece of information should be treated differently.
3. Special **aggregation considering operation semantics**: Skip connections should be treated specially. Map architectures with isomorphic computations to the same embedding.
4. Special **readout function**: Corresponding to the arch output, guarantee discriminative ability related to actual computation.

TA-GATES[2]

a determines the learning dynamics

Mimic the training process to encode a

1. **Iteratively refining** OP embedding: Discriminate same-type OPs.
Based on the root cause that operations get different parameters through training, we use forward-backward loops for refining.
2. **Symmetry-breaking** OP embedding: Discriminate same-type OPs at symmetry position.
Based on the root cause that their parameters are initialized differently, we use zero-shot with randomly initialized parameters when encoding begins for symmetry-breaking.

[1] Ning et al., "A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS", ECCV'20.
[2] Ning, Zhou et al., "TA-GATES: An Encoding Scheme for Neural Network Architectures", NeurIPS'22.

Summary: What Spaces Do Our Encoder Support Now

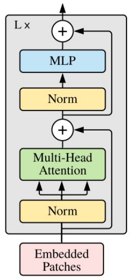
Encoder design is clearly related to the space. What spaces have we supported?

Levels of Spaces

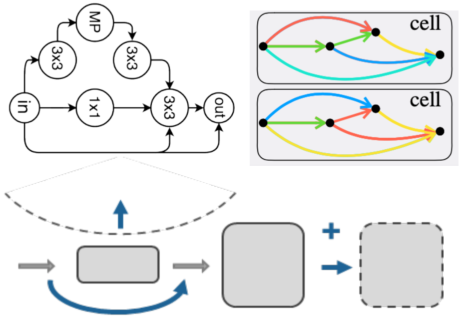
Task / Dataset

Architecture

ViT



CNN



Hardware

A100 GPU Versal ACAP TPU v3 Xilinx ZU11

Arch + Task[4]

Efficiently compress ViT to adapt to downstream tasks
(Utilize knowledge on some tasks to generalize to other tasks)

Arch: ViT Space[4]

Arch: Cell-based CNN Space[1,2]

Efficient exploration and accurate evaluation
(Utilize knowledge on some archs to generalize to other archs)

Arch + Hardware[3]

Efficient exploration

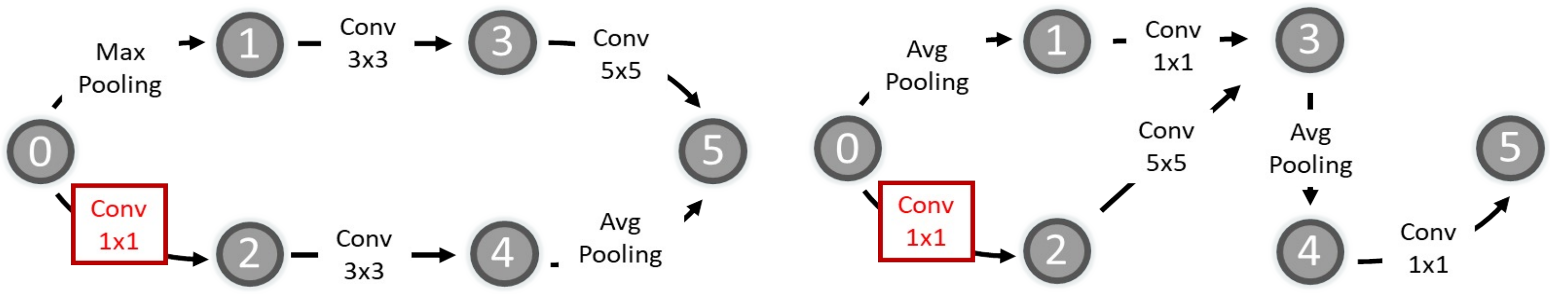
[1] Ning et al., "A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS", ECCV'20.
[2] Ning, Zhou et al., "TA-GATES: An Encoding Scheme for Neural Network Architectures", NeurIPS'22.
[3] Sun, Wang et al., "Gibbon: An Efficient Co-Exploration Framework of NN model and Processing-In-Memory Architecture", DATE'22.
[4] Zhou, Ning et al., "GiTE: A Generic Vision Transformer Encoding Scheme for Efficient ViT Architecture Search", under review for CVPR'23.

Menu

- Utilizing Architecture Encoding for Efficient and Accurate Search
- How to Encode: GATES@ECCV’20, TA-GATES@NeurIPS’22
- How to Learn the Encoder: GATES@ECCV’20, CLOSE@ECCV’22, DELE@AAAI’23

CLOSE: Improving Parameter-Sharing Evaluation

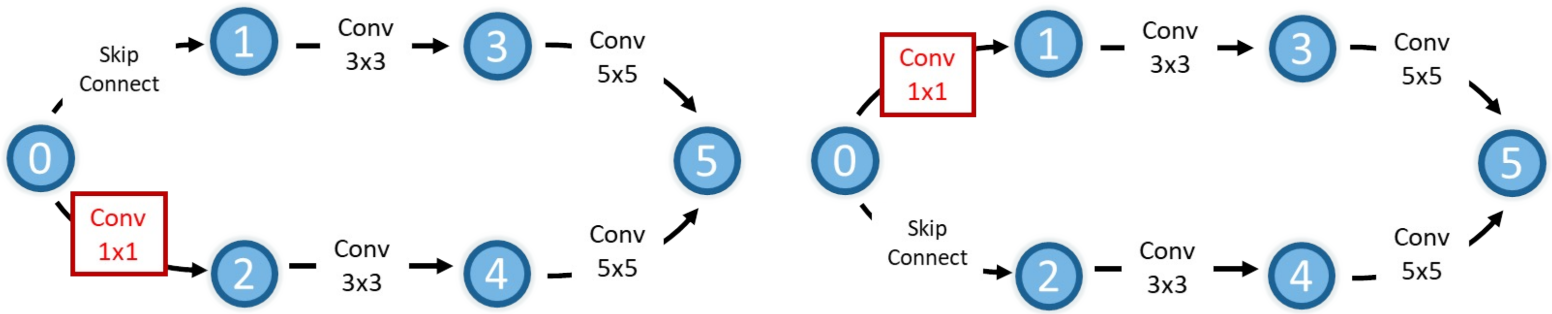
- Motivation: Deciding the sharing scheme based on **operation positions** is **inappropriate**.



Should have the flexibility to **assign different Params** to Ops with vastly **different** contexts (thus different optimal Params)

More appropriate sharing scheme:

Decide the sharing scheme based on the **operations' architectural context**.

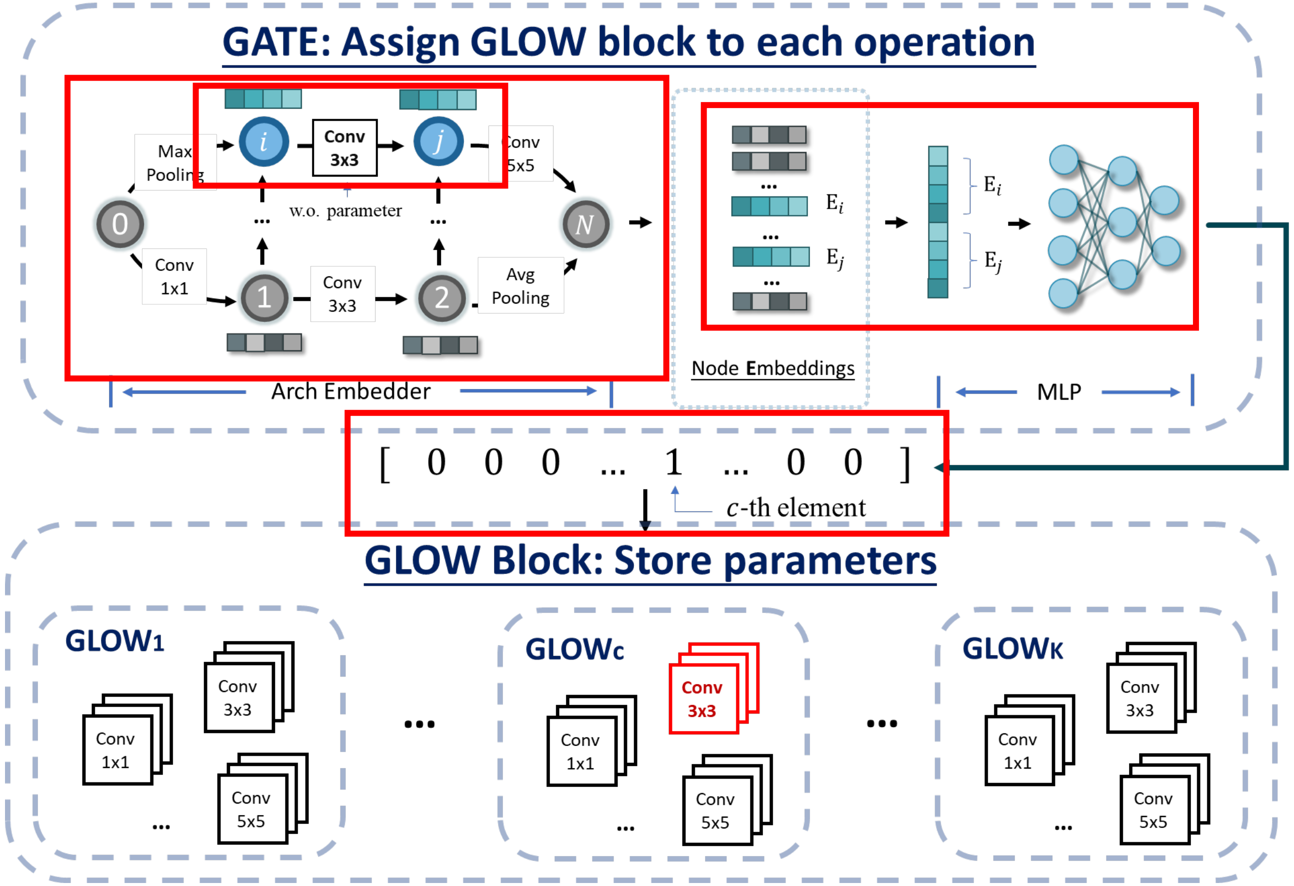


Should **share Params** for Ops with **similar or equivalent** contexts

[1] Zhou, Ning et al., "CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS", ECCV'22.

CLOSE: Improving Parameter-Sharing Evaluation

- Method: A new way to construct & learn the SuperNet.
  - **Decouple** the operations and parameters.
  - Properly **deciding the sharing scheme based on the operation encodings**.



GATE Module: Decide the parameters assignment to the operations based on the operation encodings.

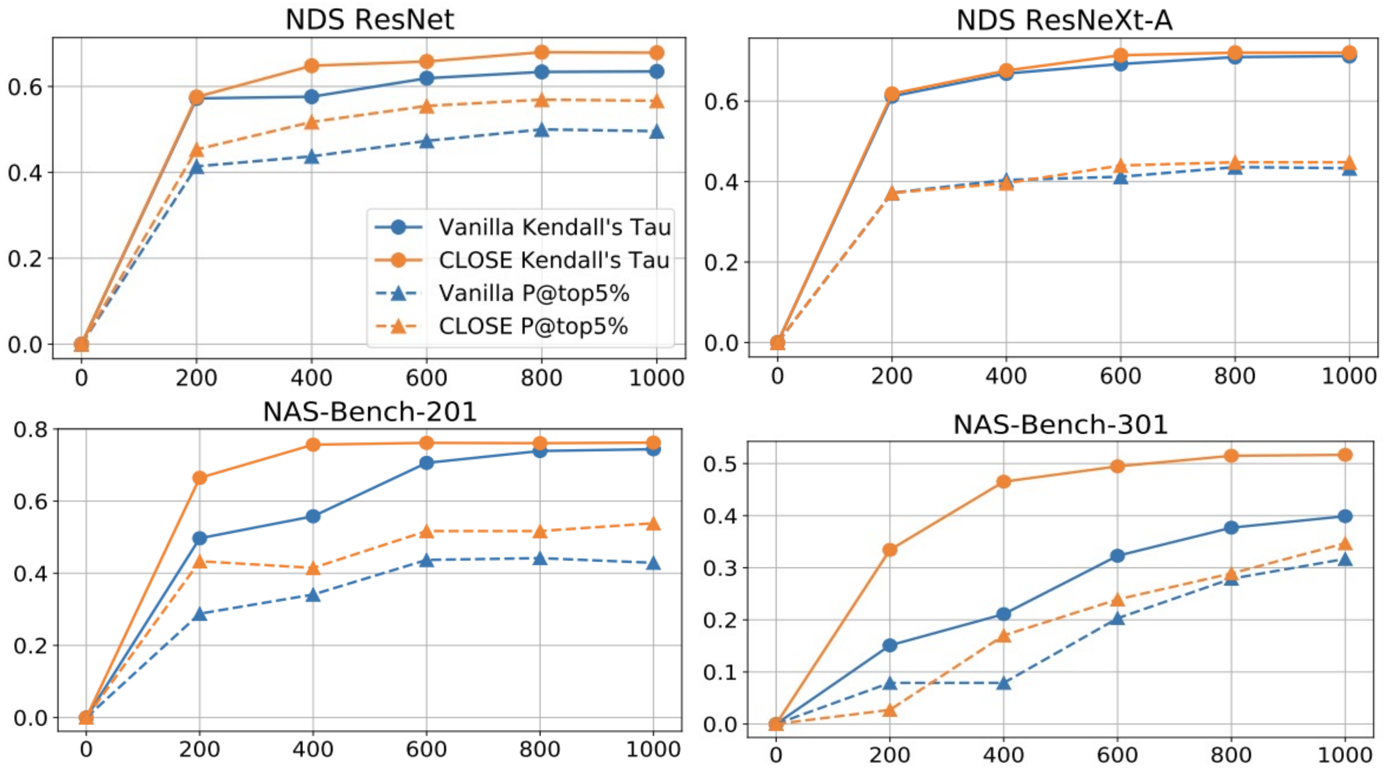
Jointly trained using the training loss

GLobal **O**peration **W**eight (GLOW) block: Store the operations' parameters

[1] Zhou, Ning et al., "CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS", ECCV'22.

CLOSE: Improving Parameter-Sharing Evaluation

- Results
  - **Kendall's Tau (KD)**: The relative difference of the number of concordant pairs and discordant pairs
  - **P@top5%**: The proportion of true top-5% architectures in the top-5% architectures according to the one-shot estimations



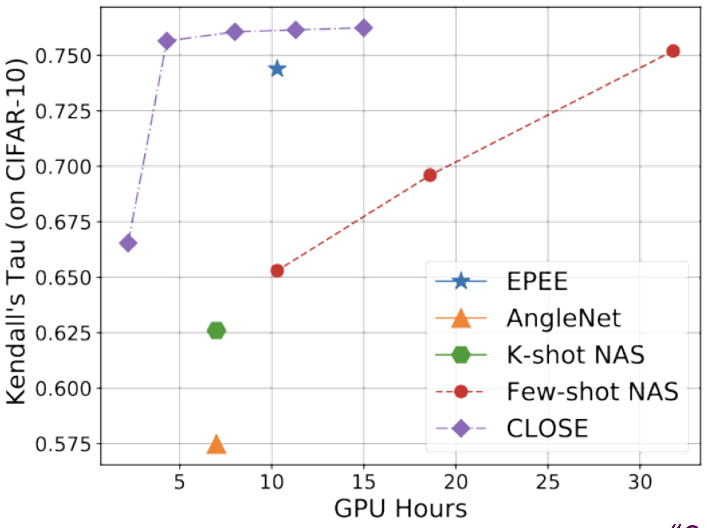
Consistently better ranking quality on multiple NAS benchmarks

- Higher **ranking correlation**
- Higher **distinguishing** ability on **top-performed** architectures

[1] Zhou, Ning et al., "CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS", ECCV'22.

CLOSE: Improving Parameter-Sharing Evaluation

- Results
  - **Kendall's Tau (KD)**: The relative difference of the number of concordant pairs and discordant pairs
  - **P@top5%**: The proportion of true top-5% architectures in the top-5% architectures according to the one-shot estimations



Method	Kendall's Tau	
	C-100	IN-16
EPEE	0.5600	0.5400
AngleNet	0.6040	0.5445
K-shot NAS	0.6122	0.5633
CLOSE	**0.6693**	**0.6632**

"C-100" and "IN-16" denotes the CIFAR-100 and ImageNet-16 datasets.

Consistently better ranking quality on multiple datasets

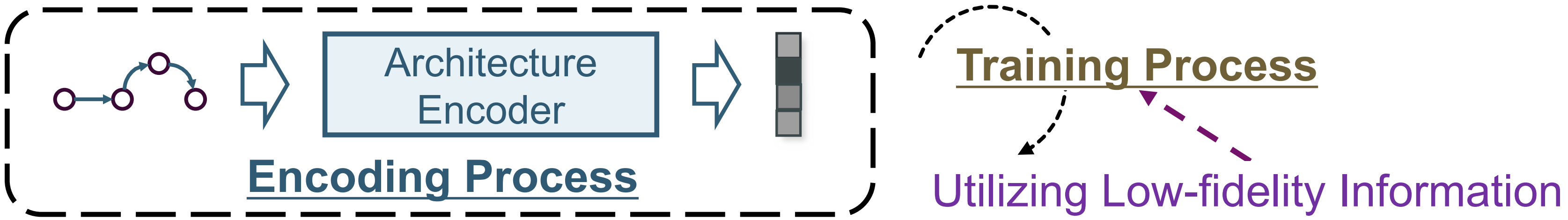
- Higher **ranking correlation**
- Less **training cost**

[1] Zhou, Ning et al., "CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS", ECCV'22.

DELE

- Predictor-based NAS suffers from the "Cold-Start" problem
  - Limited training data for predictor (limited number architecture-performance pairs) => Low ranking quality for unseen architectures (Poor exploitation) => Low sample efficiency
- **How to get better ranking?** Utilize other proxy evaluations to help rank directly?
  - Challenge: One-shot / Zero-shot evaluations alone cannot work well[2]
    - Have prominent biases, cannot work well in all search spaces / for comparing all architectures
    - The best ZSE is different across search spaces

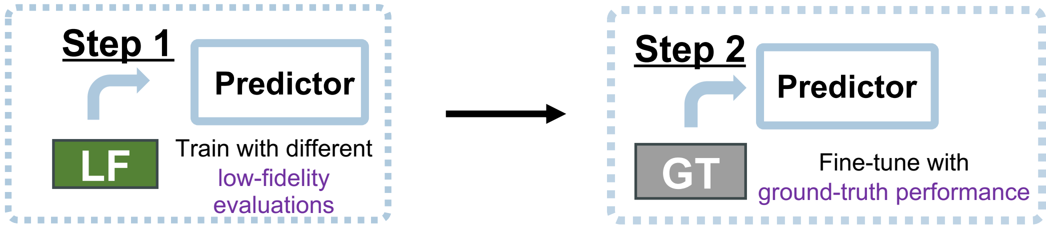
Utilize these proxy evaluations as the auxiliary information in the training process of performance predictor.



[1] Zhao, Ning et al., "Dynamic Ensemble of Low-fidelity Experts: Mitigating NAS Cold-Start", AAAI'23.
[2] Ning et al., "Evaluating Efficient Performance Estimators of Neural Architectures", NeurIPS'21.

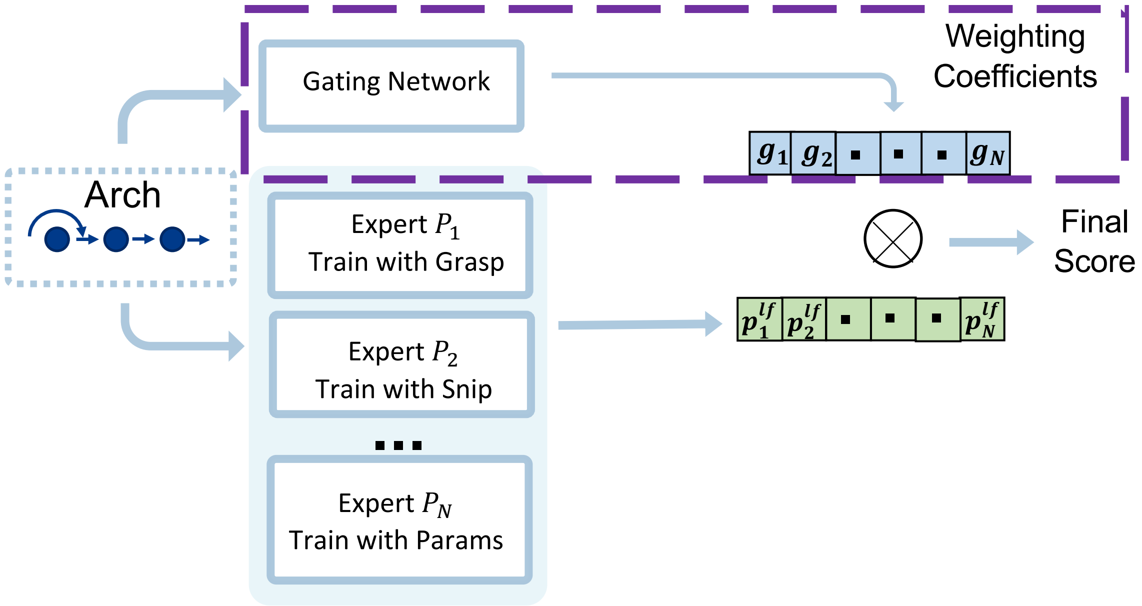
DELE

- Dynamic Ensemble Predictor Framework[1]
  - Training: **Two-step** training framework
    - Step 1: Pretrain multiple experts independently using low-fidelity evaluations
    - Step 2: Finetune ensemble network using ground-truth evaluations



Learn to fuse knowledge from different low-fidelity experts

  - Construction: **Learnable gating network** to fuse knowledge from different low-fidelity experts according to the architecture encoding



[1] Zhao, Ning et al., "Dynamic Ensemble of Low-fidelity Experts: Mitigating NAS Cold-Start", AAAI'23.

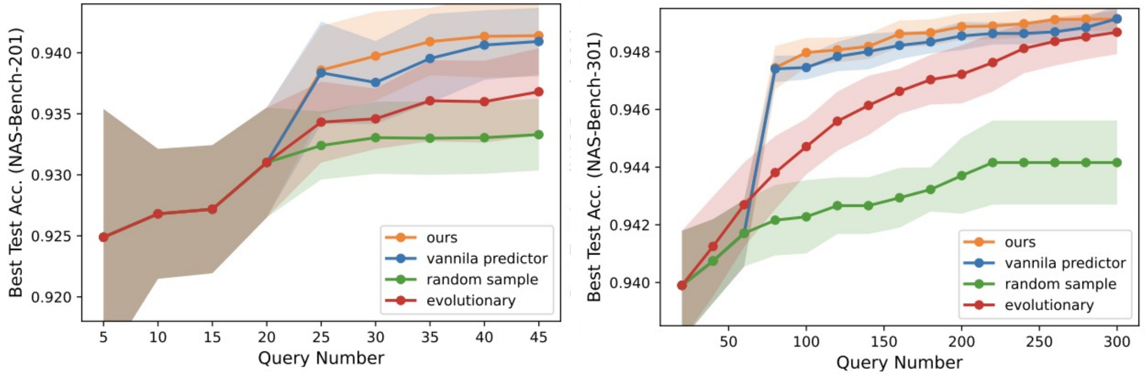
DELE

- Ranking Quality

Search Space	Encoder	Manner	Proportions of training samples				
			1%	**5%**	**10%**	**50%**	**100%**
NAS-Bench-201	GATES	Vanilla	0.7332(0.0110)	0.8582(0.0059)	0.8865(0.0045)	0.9180(0.0029)	0.9249(0.0019)
		Ours	**0.8244**(0.0081)	**0.8948**(0.0021)	**0.9075**(0.0015)	**0.9216**(0.0019)	**0.9250**(0.0020)
	LSTM	Vanilla	0.5692(0.0087)	0.6410(0.0018)	0.7258(0.0053)	0.8765(0.0010)	0.9000(0.0008)
		Ours	**0.7835**(0.0062)	**0.8538**(0.0029)	**0.8683**(0.0015)	**0.8992**(0.0010)	**0.9084**(0.0010)
NAS-Bench-301	GATES	Vanilla	0.4160(0.0450)	0.6752(0.0088)	0.7354(0.0044)	0.7693(0.0041)	**0.7883**(0.0011)
		Ours	**0.5529**(0.0135)	**0.6830**(0.0038)	**0.7433**(0.0018)	**0.7752**(0.0026)	0.7842(0.0022)
	LSTM	Vanilla	0.4757(0.0150)	0.6116(0.0099)	0.6923(0.0044)	0.7516(0.0017)	0.7667(0.0007)
		Ours	**0.4805**(0.0083)	**0.6405**(0.0035)	**0.7075**(0.0022)	**0.7544**(0.0028)	**0.7751**(0.0011)

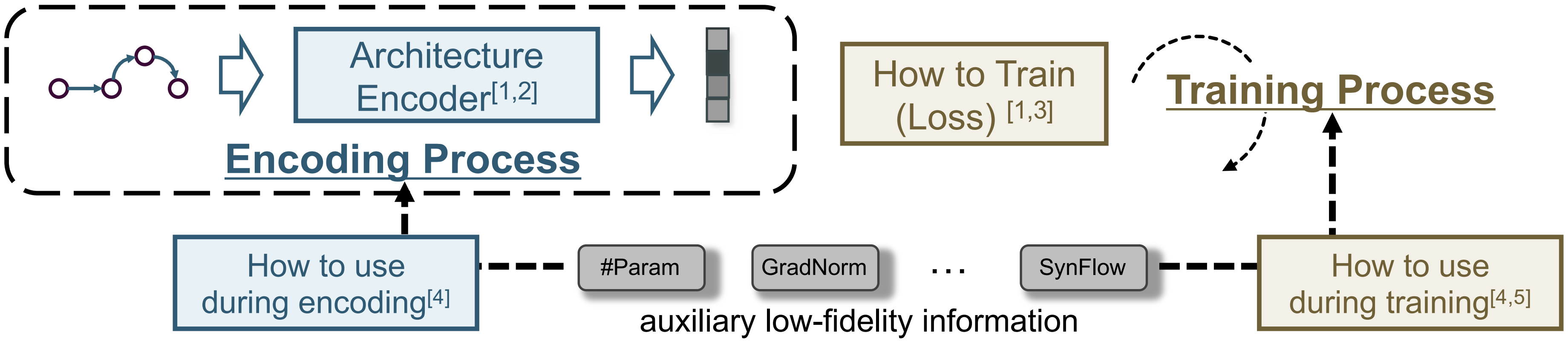
Achieve better ranking quality across different search spaces, encoders and training proportions.

- Sample Efficiency



Discover better architectures with less query number.

Summary: Methodology of Encoder Training



How to train the encoder?
Depends on how we use the encoder!

How to utilize more information (what information)?
Incorporate the knowledge into encoding or through training.

GATES[1]

- Use encoder to rank architecture performance
- Maximize ranking ability i.e., Minimize ranking loss

CLOSE[3]

- Use encoder to choose parameters for operations
- Minimize parameter-sharing loss, train jointly with supernet

DELE/GATES++[4,5]

- Low-fidelity information can be beneficial for modeling
- **Utilize low-fidelity information** in encoding and training

[1] Ning et al., "A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS", ECCV'20.
[2] Ning, Zhou et al., "TA-GATES: An Encoding Scheme for Neural Network Architectures", NeurIPS'22.
[3] Zhou, Ning et al., "CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS", ECCV'22.
[4] Ning et al., "A Generic Graph-based Neural Architecture Encoding Scheme with Multifaceted Information", under review for TPAMI.
[5] Zhao, Ning et al., "Dynamic Ensemble of Low-fidelity Experts: Mitigating NAS Cold-Start", AAAI'23.

Menu

1. Basics
2. Field Summary: Questions and Development
   a) What to Search
   b) How to Search
   c) What Can Search Tell Us
3. Our Work: Utilizing Architecture Encoding to Answer “How to Search”
4. **Summary**

Neural Architecture Search

- Problem Definition
- Three components of NAS: Search space; Search strategy; Evaluation strategy
  - Search space; Architecture; Decision; Choice
- Three Research Questions
  - **What to search**: Which decisions' choice matters for objectives
    - Topological space: Macro -> Micro; Hardware-friendly space; Outside NAS or outside DL
  - **How to search**: How to efficiently search for promising architectures
    - Improve the sample efficiency of <u>search strategy</u>: Exploration V.S. Exploitation
    - Reduce the complexity of <u>search space</u>: Factorization / Partition
    - Accelerate the <u>evaluation strategy</u>: Standalone / Morphism / One-shot / Zero-shot
  - **What can search tell us**: What does NAS process return to the user
    - Architecture; Pareto curve of architectures; Rules of decision choices (sub-space of architectures)

Architecture Encoding Can Help Answer These Questions

- How to encode: Mimic the information flow to encode the operation and architecture
- How to use: A good encoding of architectures can be used for…

1. Exploration Acceleration[1,2,3,4]

Improve the sample efficiency of NAS or Co-Exploration

2. Better Evaluation[5]

Improve parameter-sharing evaluation

How to Search

3. Interpretation[Ru et al., 2021]

Interpret which architectural
pattern is beneficial efficiently (though biased)

What Can Search Tell Us

- How to train: The encoder should be trained according to its usage (training loss, utilize other information)

[1] Ning et al., “A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS”, ECCV’20.
[2] Ning, Zhou et al., “TA-GATES: An Encoding Scheme for Neural Network Architectures”, NeurIPS’22.
[3] Sun, Wang et al., “Gibbon: An Efficient Co-Exploration Framework of NN Model and Processing-In-Memory Architecture”, DATE’22.
[4] Zhao, Ning et al., “Dynamic Ensemble of Low-fidelity Experts: Mitigating NAS Cold-Start”, AAAI’23.
[5] Zhou, Ning et al., “CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS”, ECCV’22.
Ru et al., “Interpretable Neural Architecture Search via Bayesian Optimization with WL Kernels”, ICLR’21

What's Next

- Mind the search space (what to search)
- Open the black box (what can search tell us)
- Handle the dynamicity (how to search)
- Concrete applications (how to search)

Thanks for Listening!

- Check our website introducing NAS and summarizing our work at https://sites.google.com/view/nas-nicsefc
- Discuss with me by emailing foxdoraame@gmail.com