Model Compression
Towards Efficient Deep Learning Inference

Xuefei Ning
Department of Electronic Engineering, Tsinghua University
2023.03
foxdoraame@gmail.com

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

目录 Contents

目 录 Contents

Goal of Artificial Intelligence

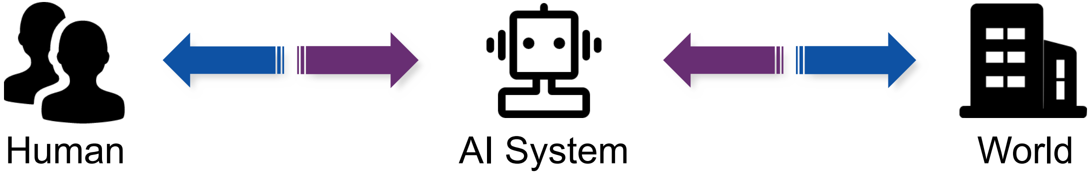
Goal: To serve humanity and improve human life

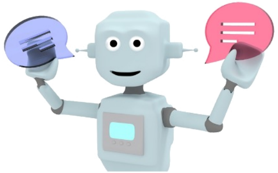
Fei-Fei Li
Co-Director of HAI

——"ImageNet Project"

"Artificial Intelligence has the potential to help us realize our shared dream of a better future **for all of humanity**. …, our vision is led by our commitment to studying, guiding and developing **human-centered AI technologies and applications**."

Interaction with Human | **Interaction with World**





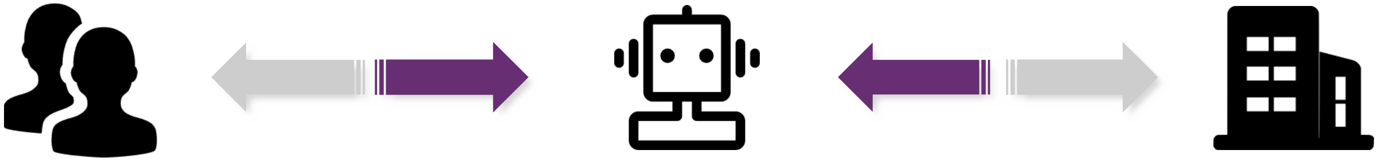
- Understand **human's instructions**
- Improve their **sensory experiences**

- Understand the **physical world**
- Make decisions to **influence the world**

Pathway to Improve the Interaction

Interaction with Human

Interaction with World

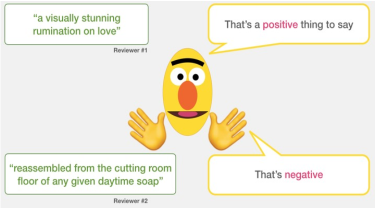


Human

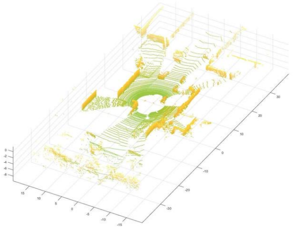
AI System

World

Large scale of data



Text



Point Cloud

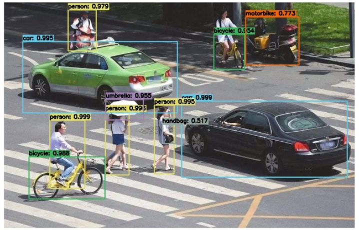
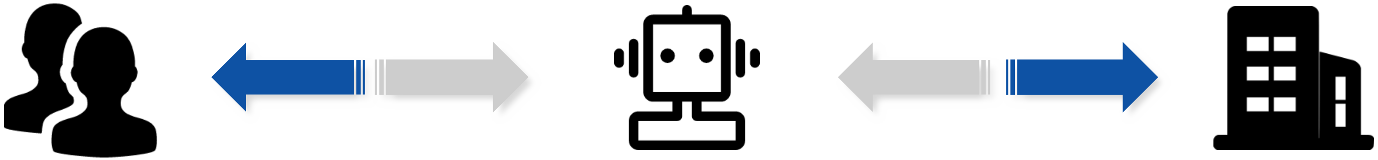


Image / Video

Pathway to Improve the Interaction

Interaction with Human

Interaction with World

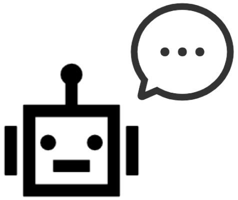


Human

AI System

World

Generative tasks



Dialogue Generation

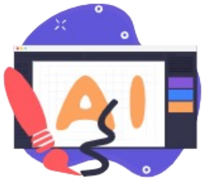


Image Generation

Code Generation

3D Assets Generation

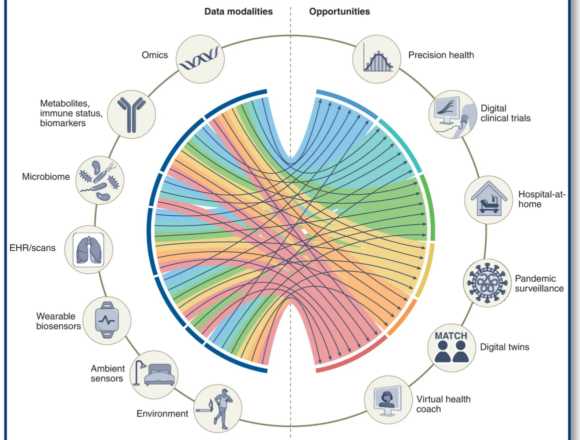
DL Trend: Data and Task

Big data, multimodal, generative tasks

Big Data

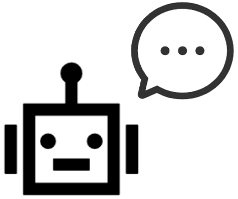
90% data in last 2 years [1]

Multimodal

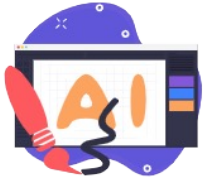


multimodal data in healthcare [2]

Generative Tasks



Text/Code Symbol Generation



Image/3D assets generation

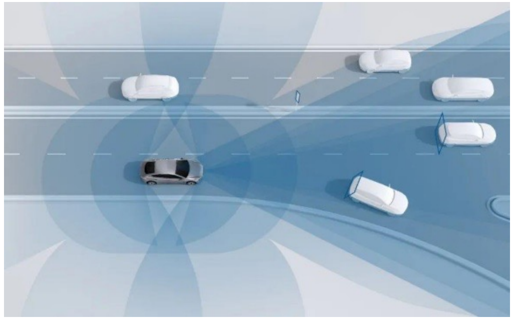
generative tasks are attracting attentions

[1] MIT 6.S191 Course: https://www.youtube.com/watch?v=QZxcTZj0L-M
[2] Acosta, J.N., Falcone, G.J., Rajpurkar, P. *et al.* Multimodal biomedical AI. *Nat Med* **28**, 1773–1784 (2022). https://doi.org/10.1038/s41591-022-01981-2

DL Trend: Data and Task

Big data, multimodal, generative tasks

Big Data

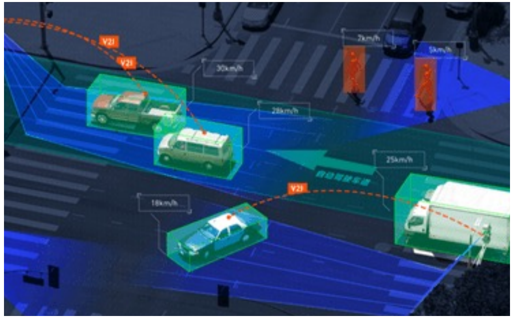


L4

Autonomous Driving

GB/Day/Car → TB/Day/Car

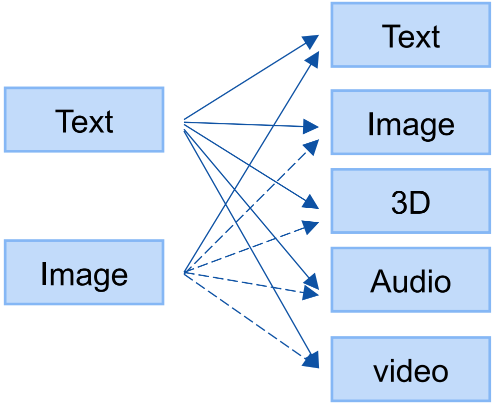
Multimodal



Intelligent perception of vehicle road collaboration

5-6 types, 30-40 sensors

Generative Tasks



Enormous generative applications

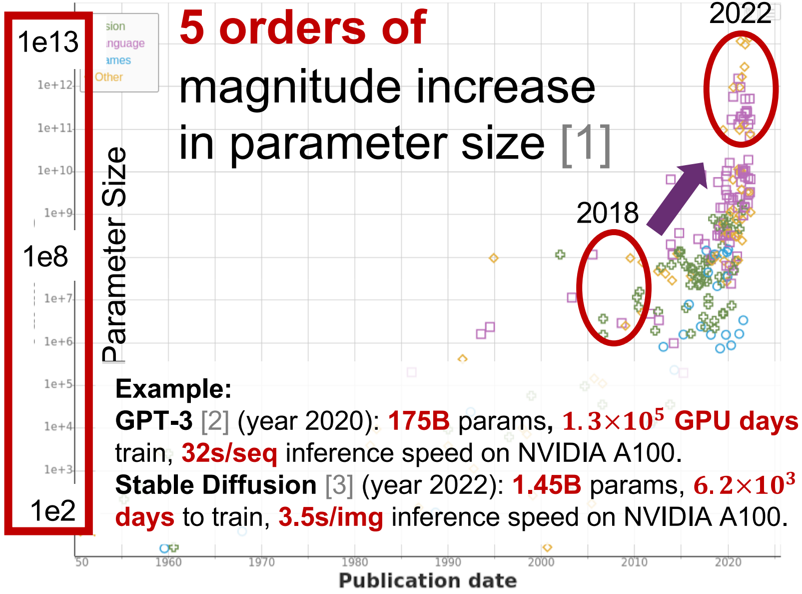
Source: EqualOcean Intelligence 2021, Computing Power Driven Vehicles - 2021 Research Report on the Development of Computing Power of China's Intelligent Vehicles

DL Trend: Model and Algorithm

The model size has being rapidly increased

2018 - 2022

5 orders of magnitude increase in parameter size [1]



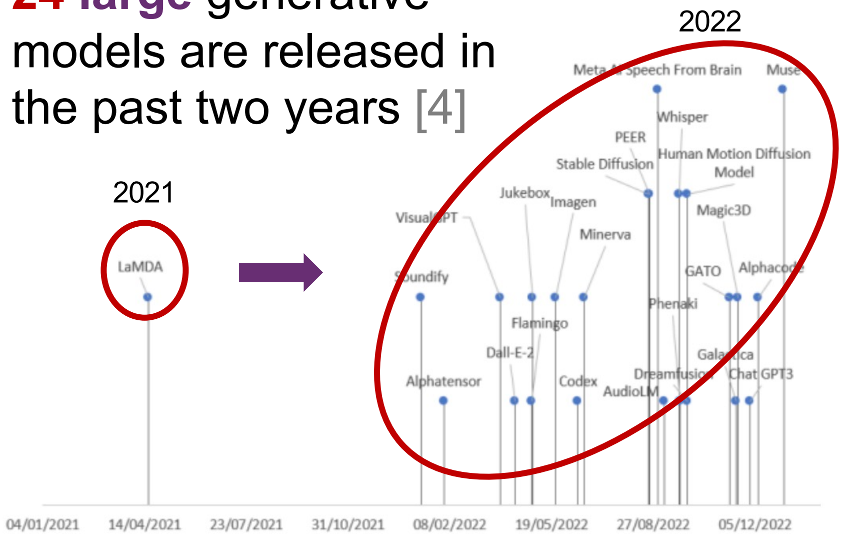
Example:

GPT-3 [2] (year 2020): **175B** params, $\mathbf{1.3\times10^5}$ **GPU days** to train, **32s/seq** inference speed on NVIDIA A100.

Stable Diffusion [3] (year 2022): **1.45B** params, $\mathbf{6.2\times10^3}$ **GPU days** to train, **3.5s/img** inference speed on NVIDIA A100.

2021 - 2022

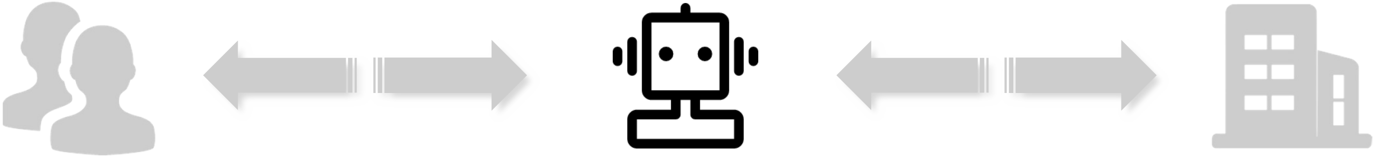
24 large generative models are released in the past two years [4]



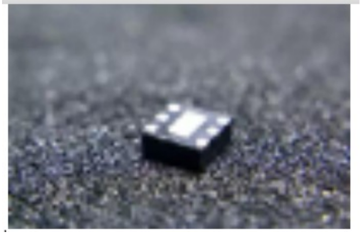
[1] Villalobos et al. "Machine Learning Model Sizes and the Parameter Gap." arXiv preprint arXiv:2207.02852 (2022).
[2] Brown et al. "Language Models are Few-Shot Learners." arXiv, 2020, https://doi.org/10.48550/arXiv.2005.14165.
[3] Rombatch et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR'22.
[4] Roberto Gozalo-Brizuela, et al. "ChatGPT is not all you need. A State of the Art Review of large Generative AI models." arXiv, 2023.

Various Application Scenario

Interaction with Human | **Interaction with World**



Human | AI System | World



Sensor
Wearable Device

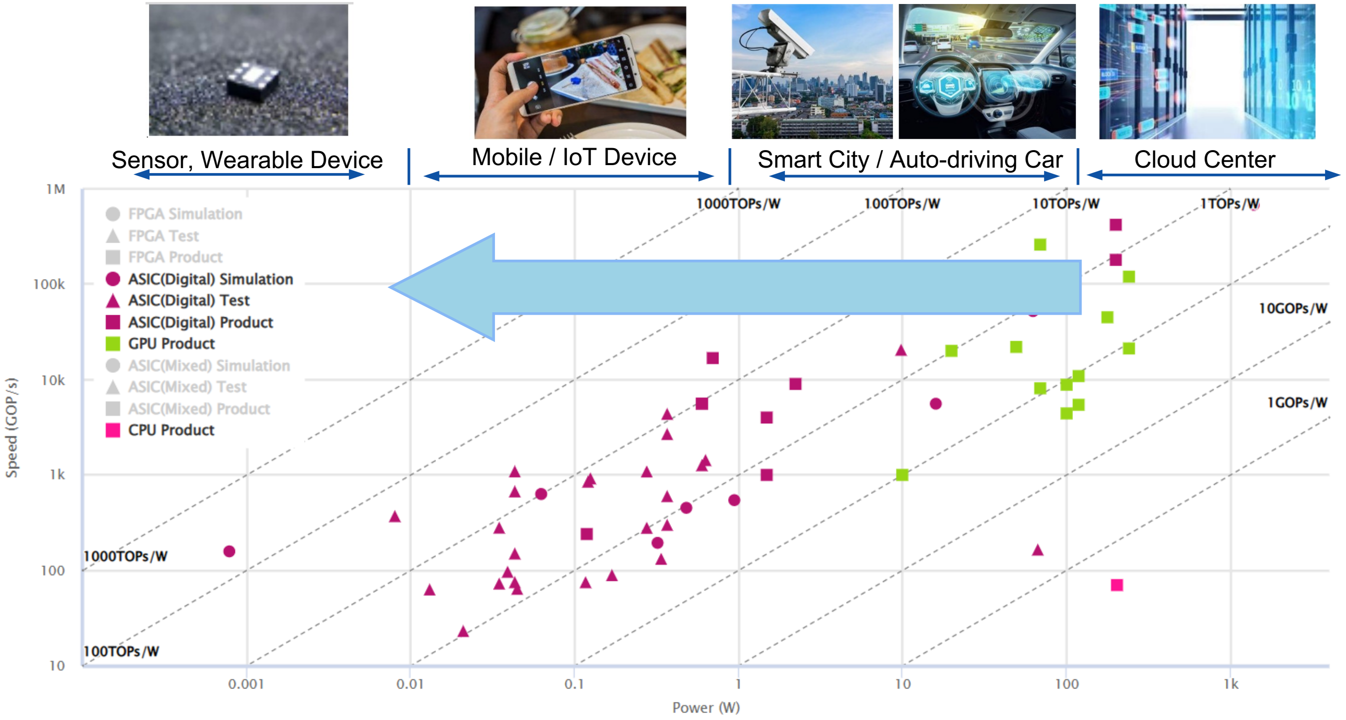
Mobile Phone
IoT Device

Smart City
Auto-driving Car

Smart City
Auto-driving Car

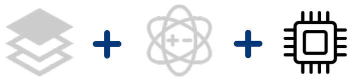
DL Trend: Computing Power

From cloud center to tiny edge device

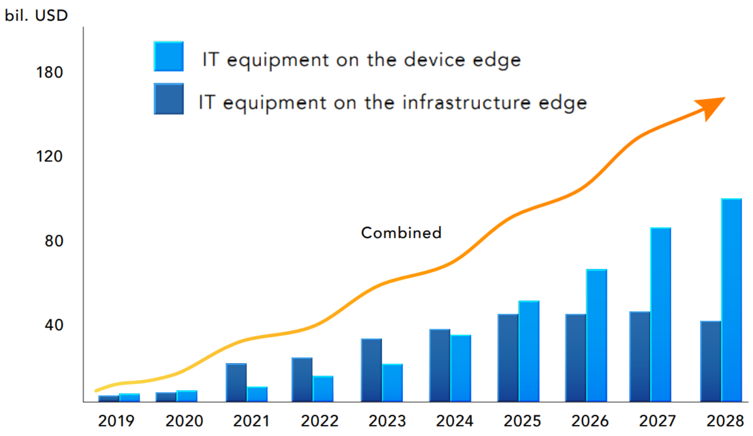


DL Trend: Computing Power

From cloud center to tiny edge device

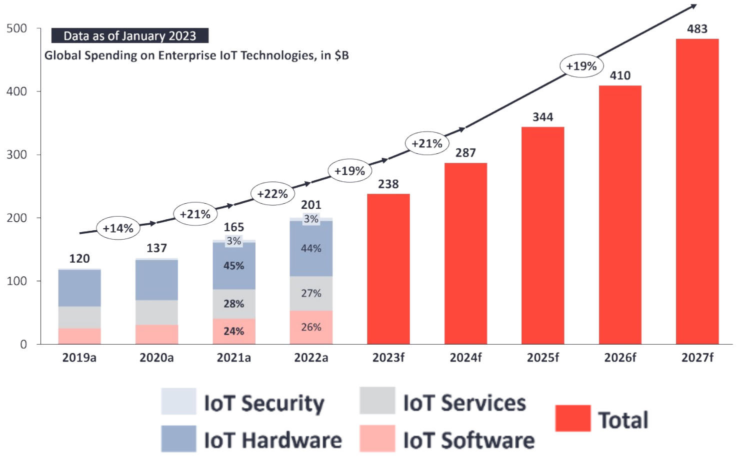


Global Annual CAPEX Spend on Edge [1]



Edge: 88% *v.s.* Cloud: 12%

Enterprise IoT market 2019-2027 [2]



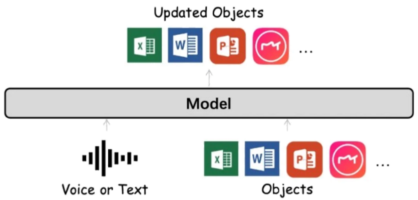
IoT market grew 21.5% in 2022

[1] State of the Edge 2021: A Market and Ecosystem Report for Edge Computing
[2] https://iot-analytics.com/iot-market-size/

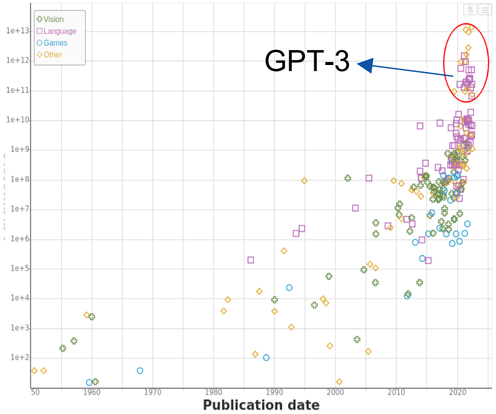
DL Trend: Summary

Data & Task + Model & Algorithm + Computing Power

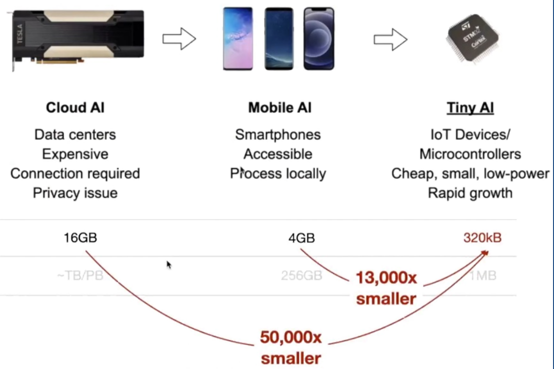

Big data **Multimodal**



Generative tasks



Large model



Tiny edge device

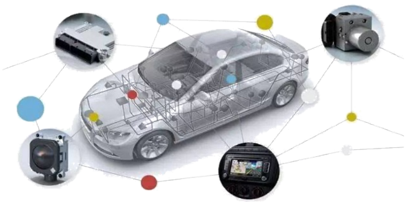
Application Challenges

- **Gap** caused by the large model and the tiny edge device

Real-time Requirement
on Constrained Platform

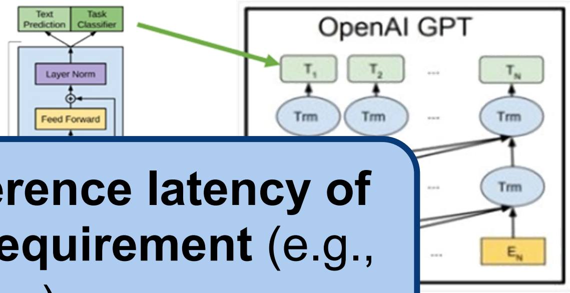
Video Conference
~30FPS

Outdoor Detection
10~25Hz



Auto-driving Car
(10~100 TOPS)

High Latency
caused by Large Model Size



Our research goal is to **reduce the inference latency of the model to satisfy the application requirement** (e.g., latency on certain platform)

<1FPS

NVIDIA A100
(19.5 TFLOPS)

E.g., stable diffusion: 1.45B, 5.7TOPs, **2s/img**
GPT-3: 175B, 664TOPs, **32s/seq**

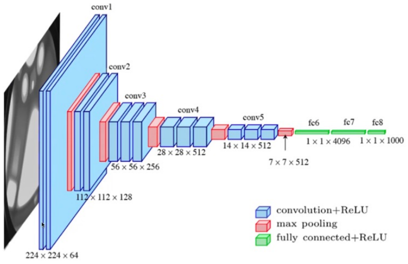
Direction to Mitigate the Gap

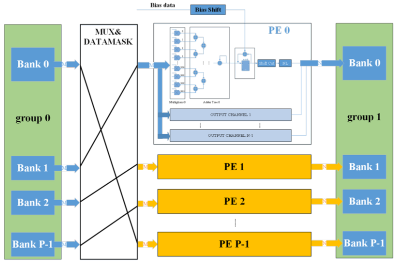
$$Latency = \frac{Computing\ Demand}{Computing\ Performace * Hardware\ Uilization}$$

VGG-16
~30 GOPs

~191ms

Xilinx DPU
(400 GOPS)
~40% Utilization





Direction to Mitigate the Gap

$$\downarrow \mathit{Latency} = \frac{\mathit{Computing\ Demand} \downarrow}{\mathit{Computing\ Performace} * \mathit{Hardware\ Uilization} \uparrow}$$

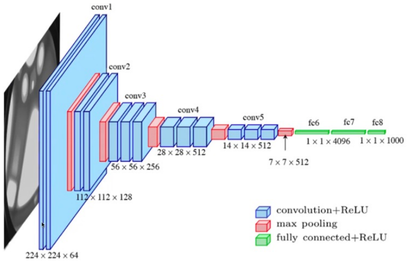
VGG-16
~30 GOPs

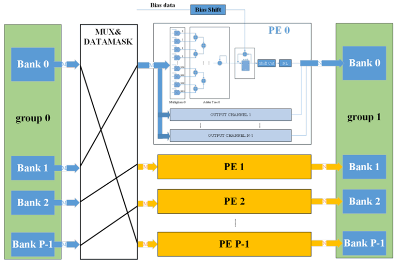
↓

~191ms

↑

Xilinx DPU
(400 GOPS)
~40% Utilization



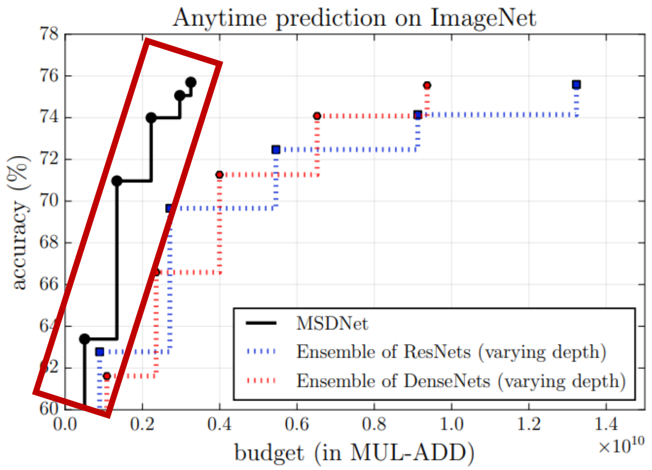


Compressed Dimension

Directions for Model Compression	Data Representation	Model Weight	Neural Architecture
Lower Computing Demand	**Model Quantization**		**NAS / Pruning / Factorization**
Improve Hardware Utilization	**Model Quantization**	**Weight Reuse**	**NAS / Pruning / Factorization**

Direction to Mitigate the Gap

$$System\ Performance: Task\ Performance, Latency$$

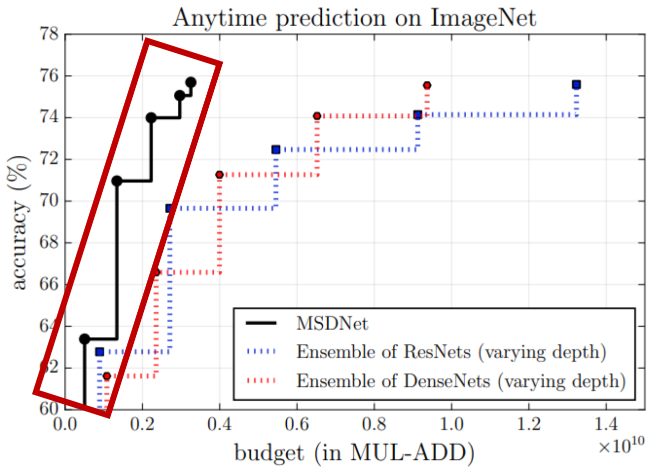


MSDNet[1] achieves better **trade-off** between accuracy and budget on ImageNet

[1] Huang et al., Multi-Scale Dense Convolutional Networks for Efficient Prediction, ICLR'18 oral

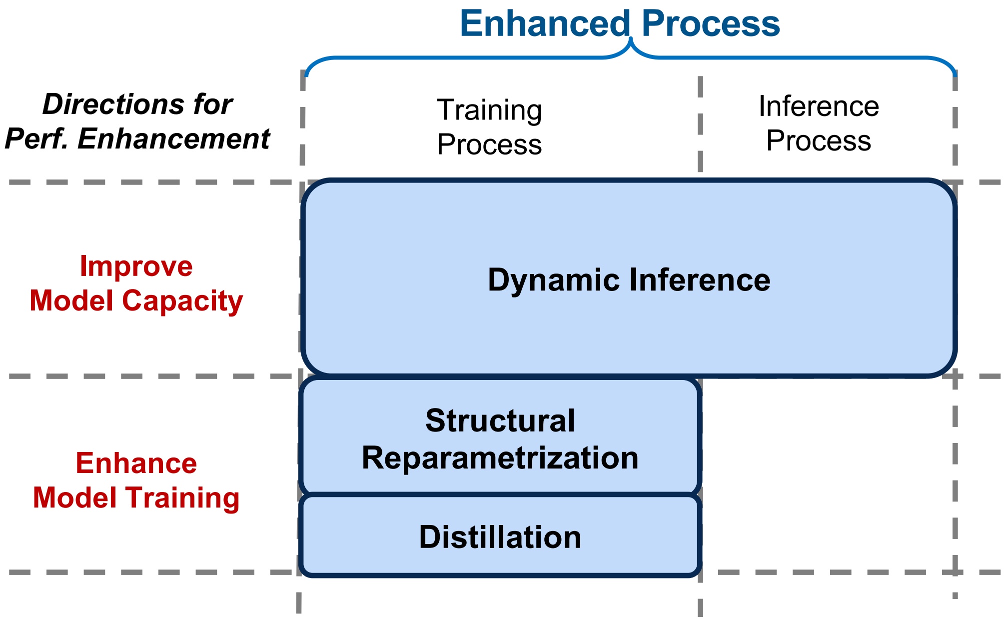
Direction to Mitigate the Gap

$\uparrow System\ Performance: Task\ Performance, \uparrow Latency$



MSDNet[1] achieves better **trade-off** between accuracy and budget on ImageNet

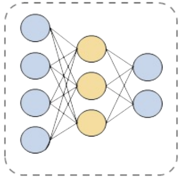
[1] Huang et al., Multi-Scale Dense Convolutional Networks for Efficient Prediction, ICLR'18 oral



Automated Model Compression

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

Varying Tasks

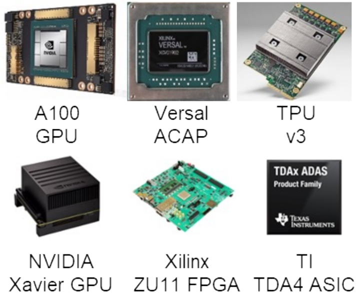


Neural Network

Expensive/Suboptimal

Purely Manual Model Compression

Varying Hardware



We need automated model compression!

目录 Contents

Model Compression

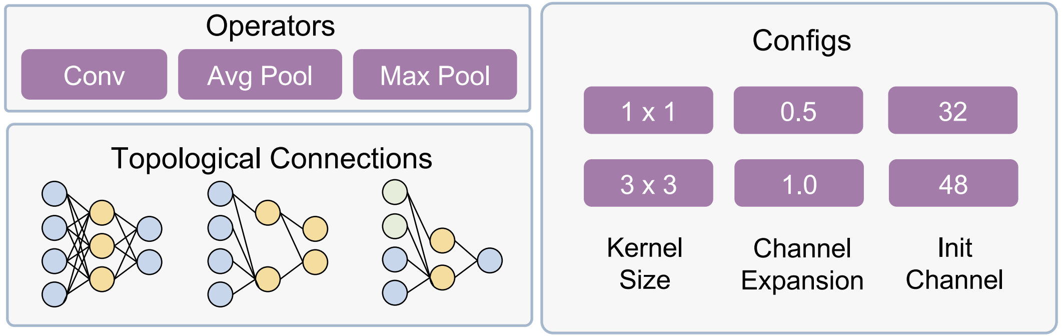
- Basic Problem Definition
  - Neural network architecture a; Search space S; Network parameters w
  - Valid task perf R_{val}; Train task loss L_{train}; Complexity function F; Budget B

$$\underset{a \in S}{\text{Maximize}} \ \boldsymbol{R}(w^*(a),\ a)$$ ➡ **Targets**: task perf, perf loss, …

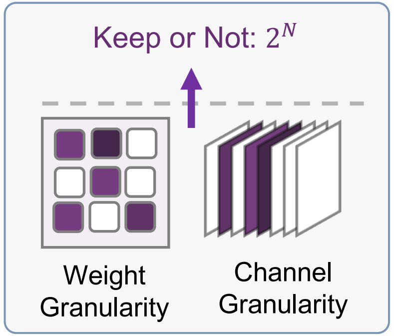
$$s.t.\ \ w^*(a) = \text{argmin}_w\, L_{train}(w, a)$$

$$\boldsymbol{F}(a) \leq \boldsymbol{B}$$ ➡ **Constraints**: latency, throughput, energy, …

- Optimize in the **Network Architecture** dimension: **NAS / Pruning**



NAS Search Space



Pruning Space

Model Compression

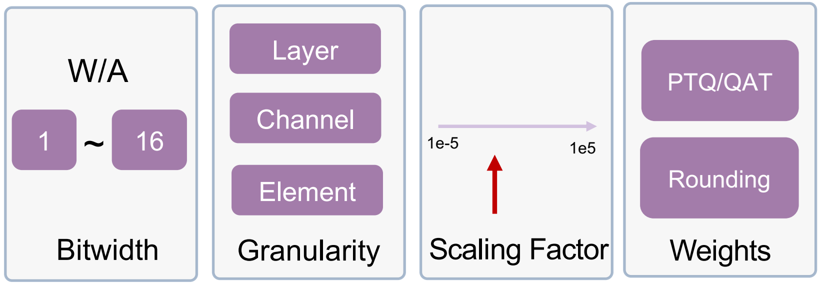
- Basic Problem Definition
  - Neural network architecture a; Search space S; Network parameters w
  - Valid task perf R_{val}; Train task loss L_{train}; Complexity function F; Budget B

$$\underset{a \in S}{\text{Maximize}} \; \boldsymbol{R}(w^*(a), \; a) \; \longrightarrow$$ **Targets**: task perf, perf loss, …

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

$$\boldsymbol{F}(a) \leq \boldsymbol{B} \; \longrightarrow$$ **Constraints**: latency, throughput, energy, …

- Optimize in the **Data Representation** dimension: **Quantization**

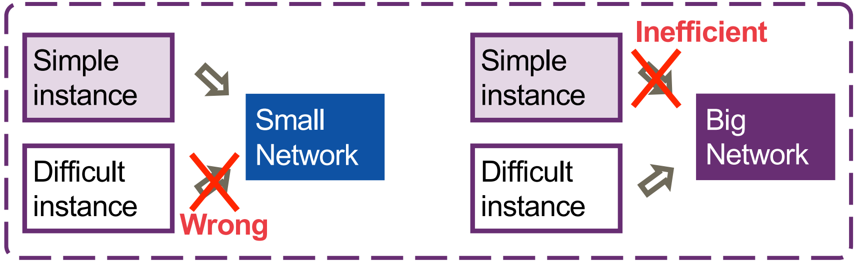


Quantization Space

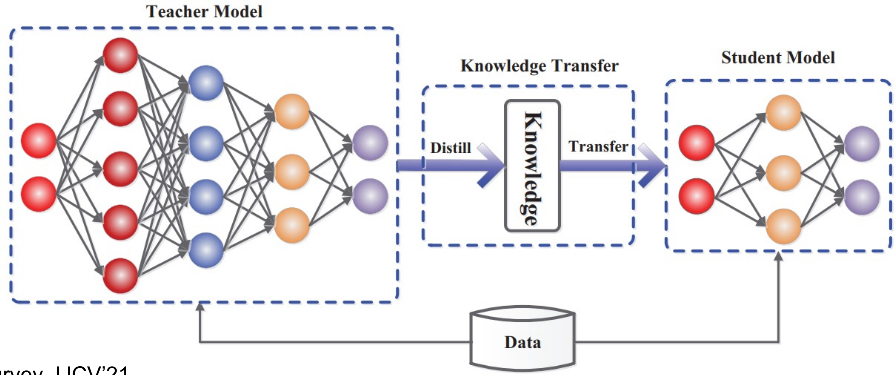
Enhanced Techniques

- Optimize the **model capacity**: **Dynamic Inference**

Better trade-off between accuracy & efficiency



- Enhance the **model training**: **Distillation**

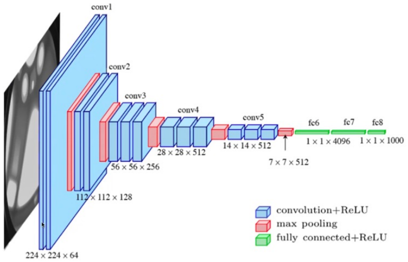


Gou et al., Knowledge Distillation: A Survey, IJCV'21

(Review) Direction to Mitigate the Gap

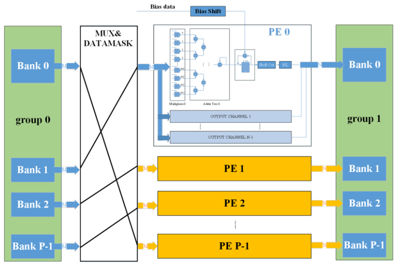
$$\downarrow Latency = \frac{Computing\ Demand\ \downarrow}{Computing\ Performace * Hardware\ Uilization\ \uparrow}$$

VGG-16
~30 GOPs



~191ms

Xilinx DPU
(400 GOPS)
~40% Utilization

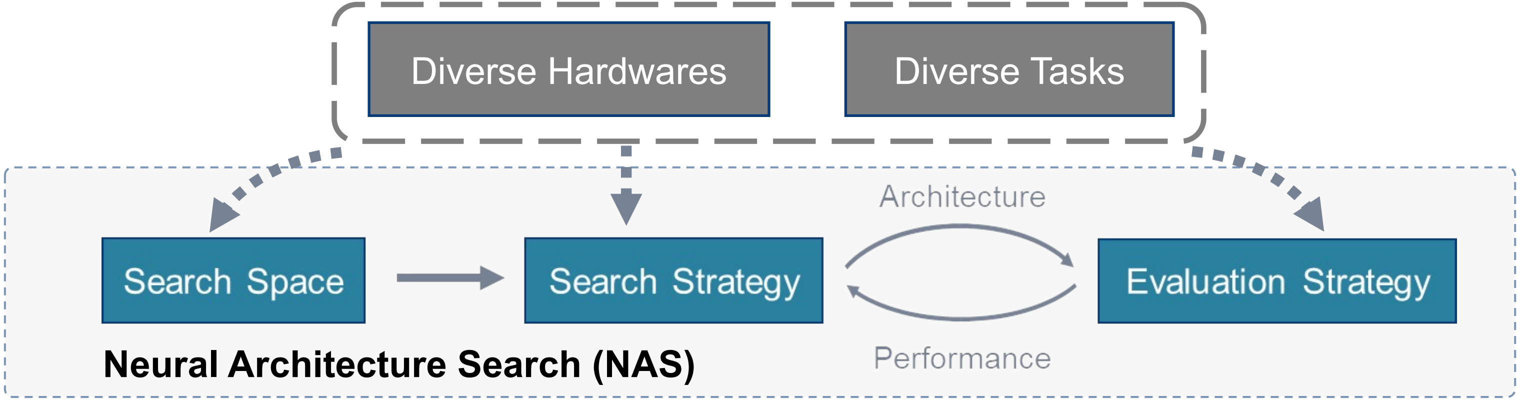


Compressed Dimension

Direction: Model Compression	Weight Representation	Model Weight	Neural Architecture
Lower Computing Demand	**Model Quantization**		**NAS / Pruning / Factorization**
Improve Hardware Utilization	**Model Quantization**	Weight Reuse	**NAS / Pruning / Factorization**

Neural Architecture Search

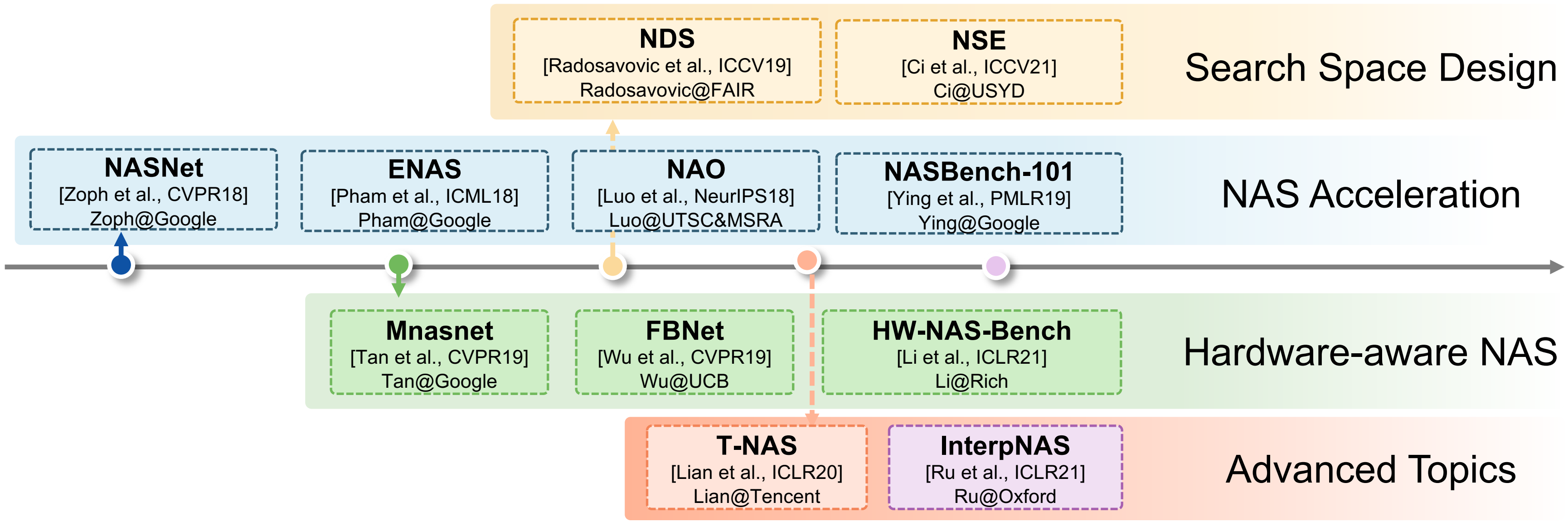
- Definition: Given the hardware-related constraints (e.g., latency, FLOPs), search for the network architecture to satisfy the constraints and maximize the accuracy
- Main Components:
  - Search Space: Define the searchable configurations, the space of candidate architectures
  - Evaluation Strategy: How to evaluate the architecture
  - Search Strategy: How to explore the search space



Refer to our websites https://sites.google.com/view/nas-nicsefc for more introduction on NAS.

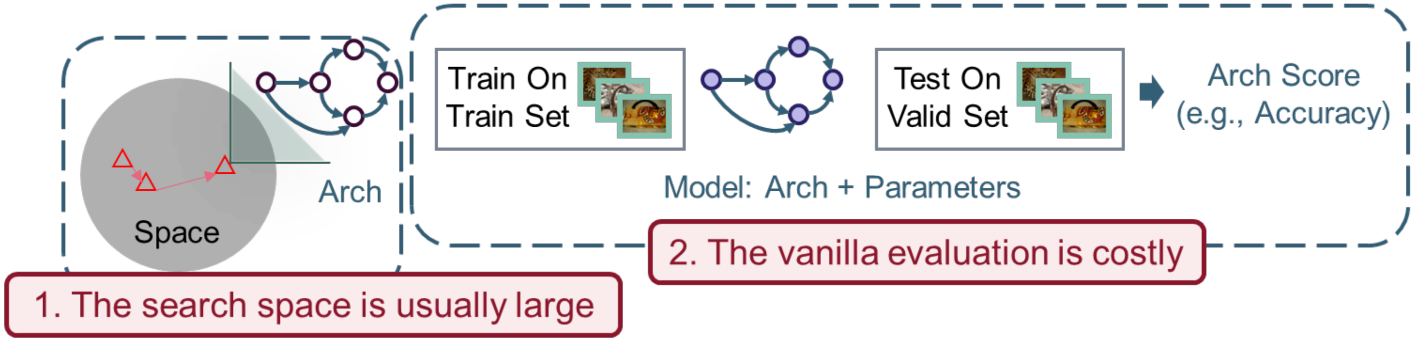
NAS Development Trends

- Trends
  - **Improve the tradeoff between NAS efficiency and performance**
  - Take the hardware system into consideration → co-design with system configurations
  - Manually design search space → Automatically design
  - …



Challenge

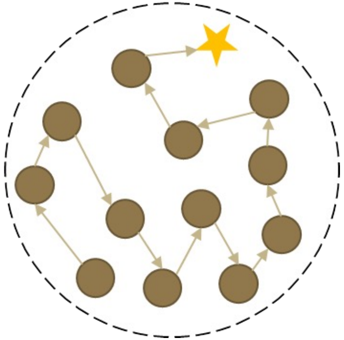
- Key Challenge: NAS process is slow
  - Large search space: Need to evaluate many architectures to explore sufficiently
  - Costly vanilla evaluation: The evaluation of each architecture is costly



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

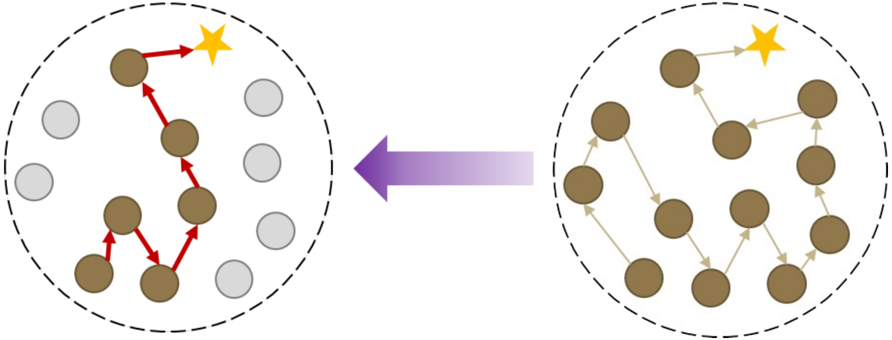
Zoph et al., "Neural architecture search with reinforcement learning", ICLR'17.
Zoph et al., "Learning Transferable Architectures for Scalable Image Recognition", CVPR'18.
Real et al.., "Regularized Evolution for Image Classifier Architecture Search", AAAI'19.

Direction towards Efficient NAS



Direction towards Efficient NAS

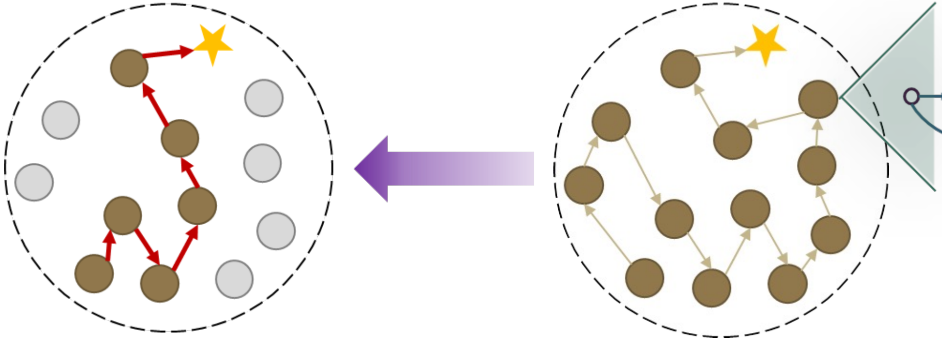
1. Improve the sample efficiency of the search strategy

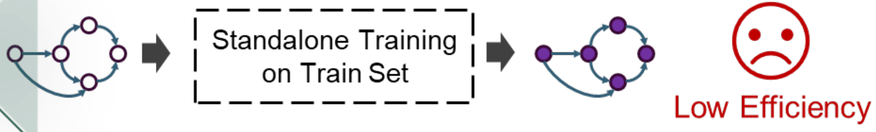


Direction towards Efficient NAS

1. Improve the sample efficiency of the search strategy

2. Accelerate the evaluation strategy

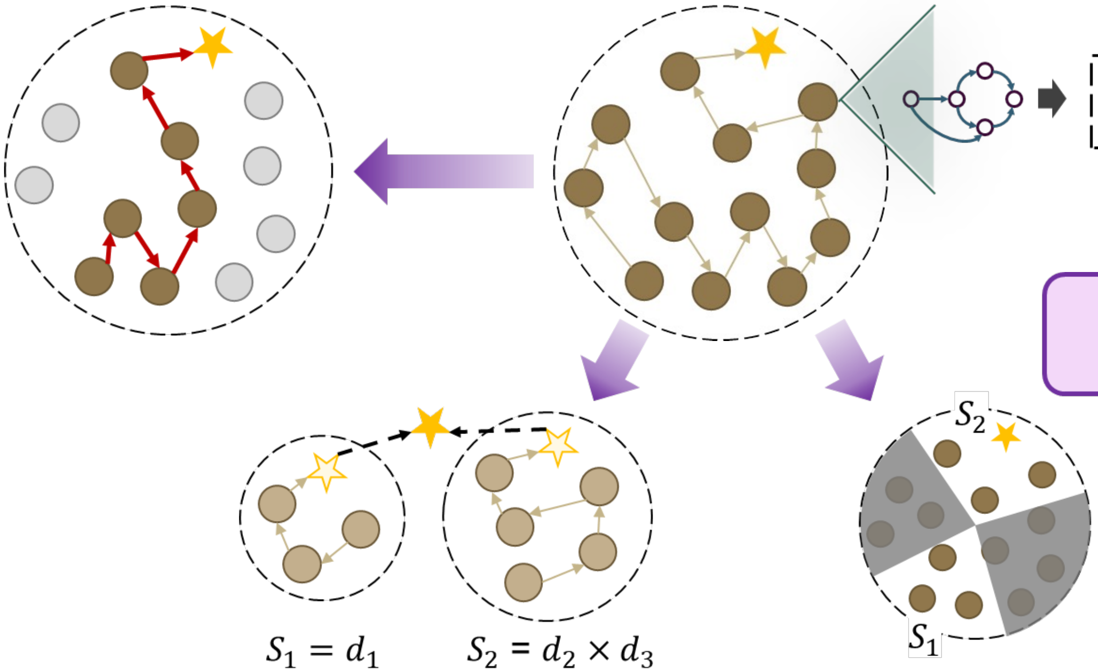


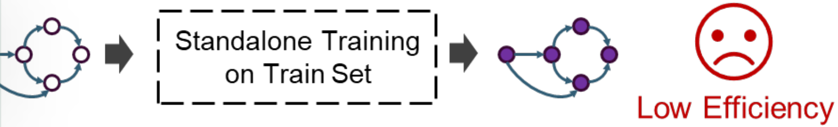


Direction towards Efficient NAS

1. Improve the sample efficiency of the search strategy

2. Accelerate the evaluation strategy



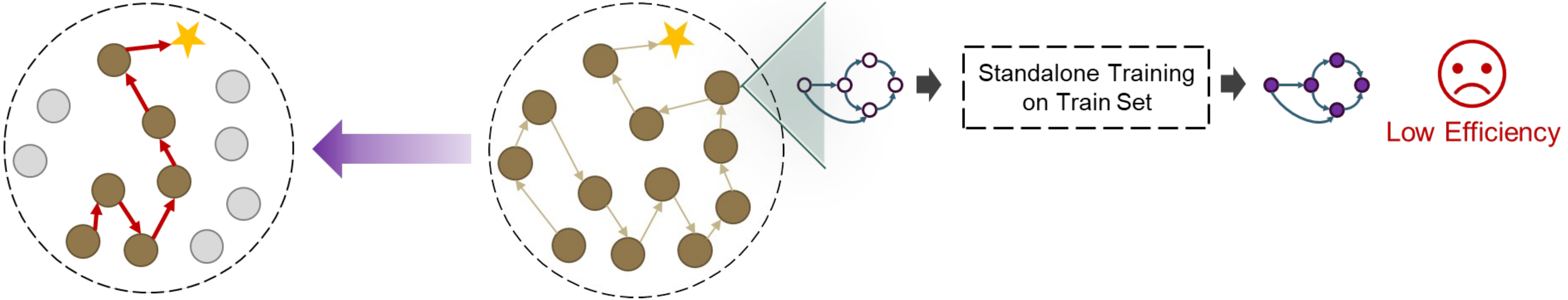


3. Factorize or partition the search space to reduce space complexity.

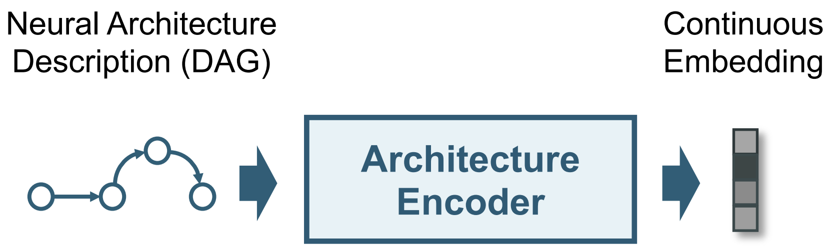
Our Solution towards Efficient NAS

1. Improve the sample efficiency of the search strategy

2. Accelerate the evaluation strategy



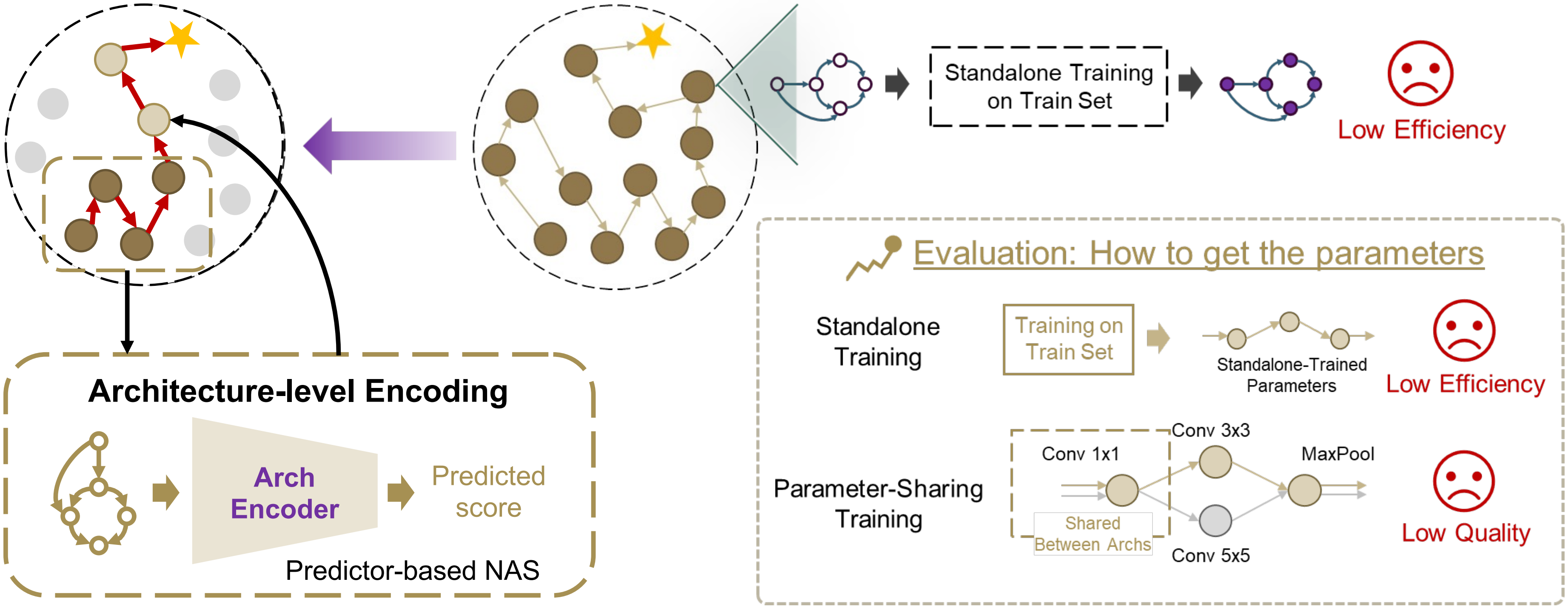
Architecture Modeling



Our Solution towards Efficient NAS

1. Improve the sample efficiency of the search strategy

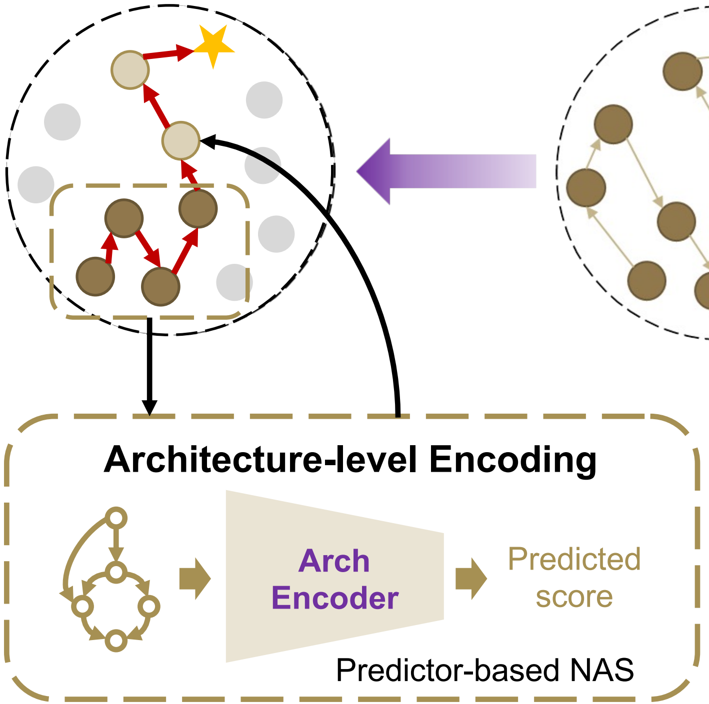
2. Accelerate the evaluation strategy

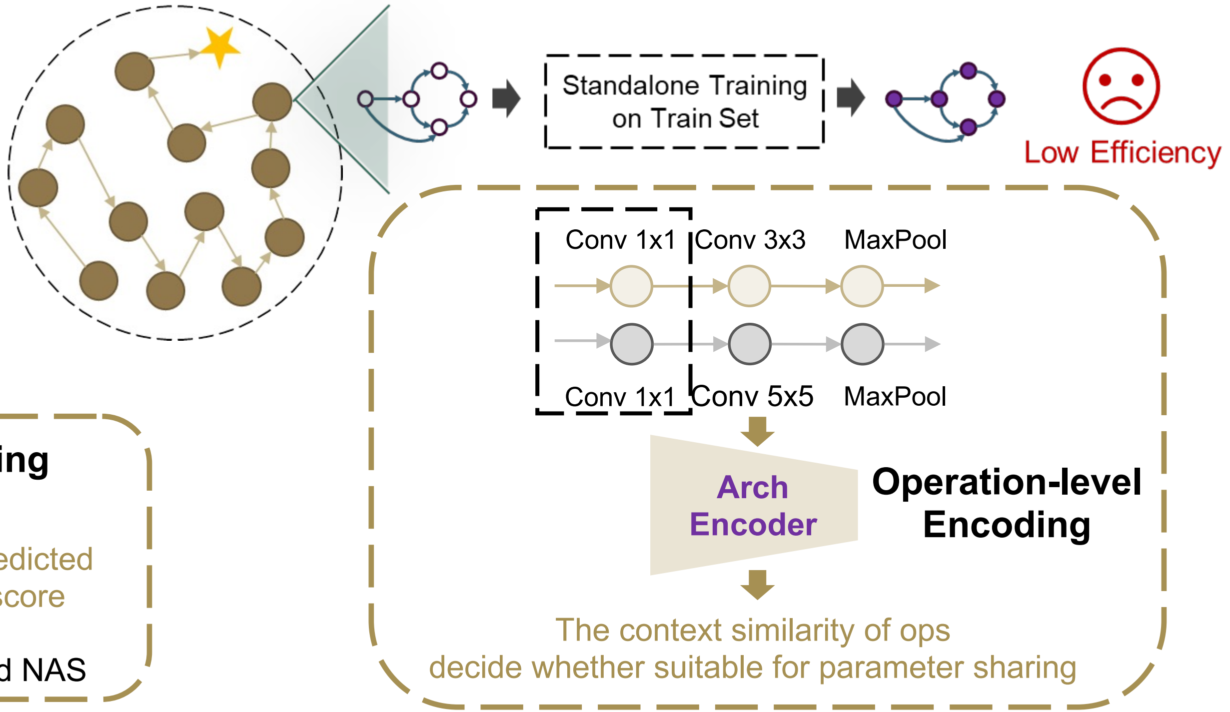


Our Solution towards Efficient NAS

1. Improve the sample efficiency of the search strategy

2. Accelerate the evaluation strategy

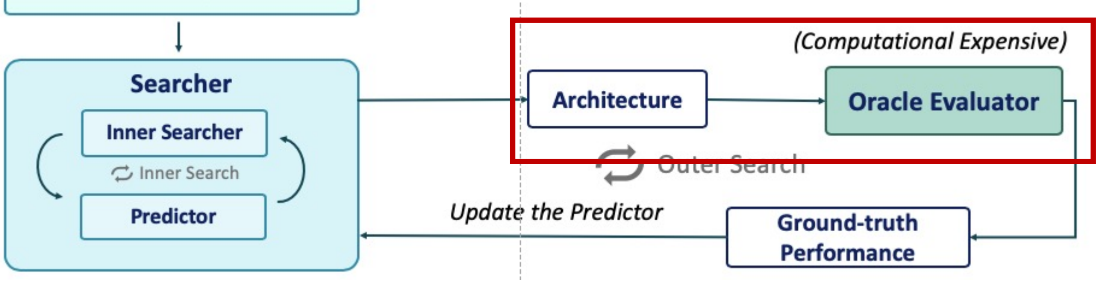




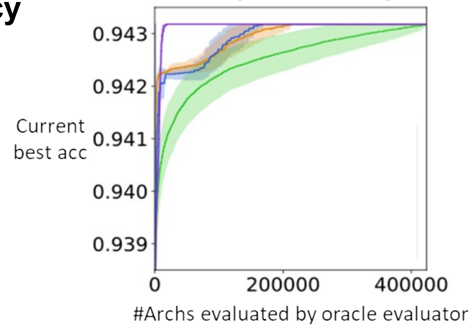
Enhance the Predictor-based NAS

- Predictor-based NAS

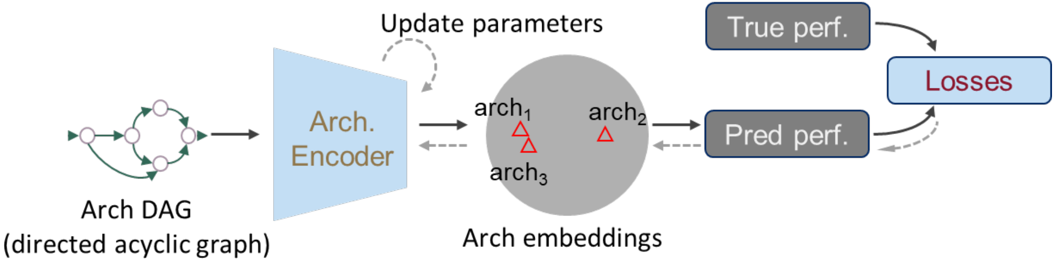
The predictor's fitness is vital to sample efficiency



Example plot of sample efficiency



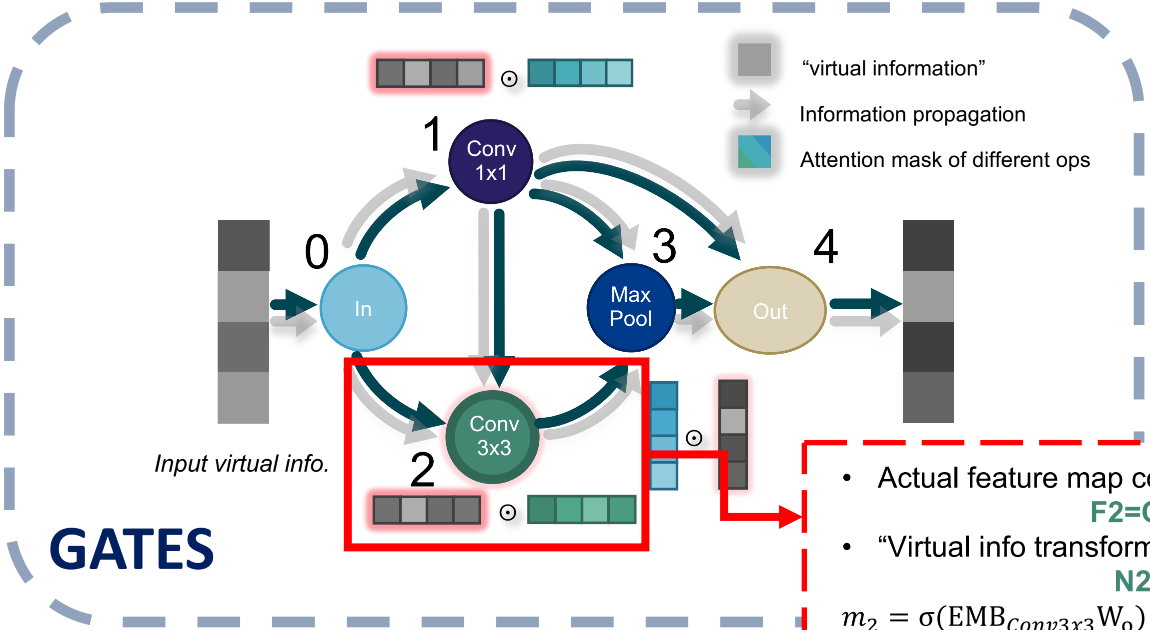
- Typical construction of the predictor



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

GATES: How to Encode

- **Mimic** the actual NN **information flow**
- Properly encode **computationally isomorphic** architectures to the same embedding



- Actual feature map computation:
  F2=Conv3x3(F0 + F1)
- "Virtual info transformation" in the encoding process:
  N2=$m_2 \odot$ (N0+N1);

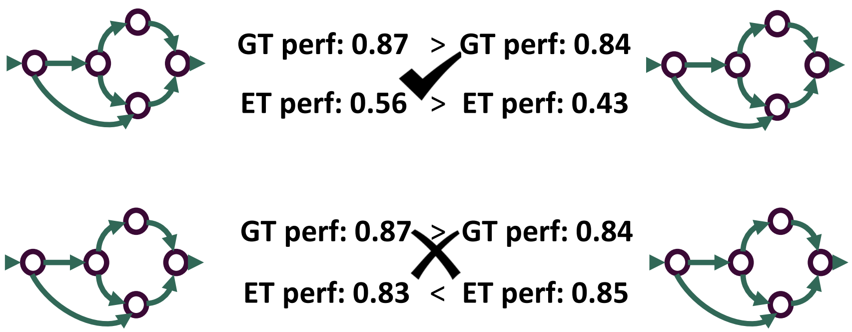
$m_2 = \sigma(\mathrm{EMB}_{Conv3x3}\mathrm{W_o})$ is the **attention mask** of Conv3x3

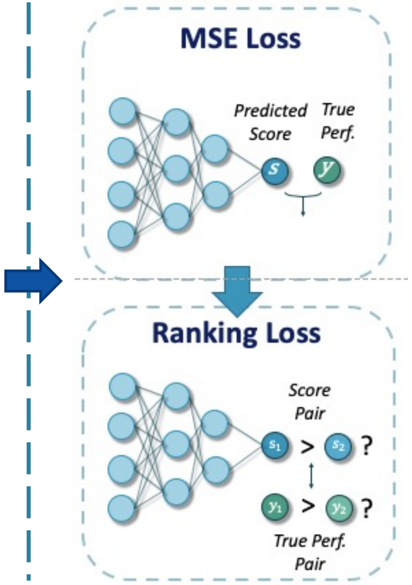
Ning et al., "A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS", ECCV'20.

GATES: How to Train

- Use **ranking loss** to train the predictor
- Ranking loss are better surrogate of **ranking performance** than regression loss

The ranking of the estimated performances (ET perf) is more important





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

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

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

GATES: Results (on NAS-Bench-101)

- Ranking correlation (Kendall's Tau)
  - 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**

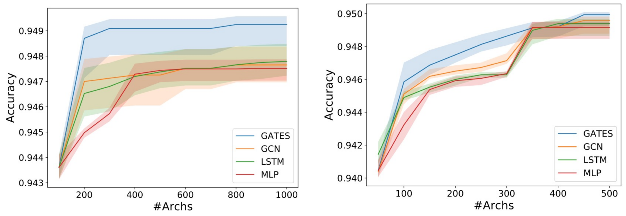
GATES outperform other encoders consistently, especially when there are few training samples

  - 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

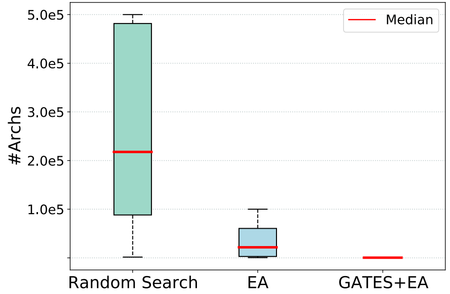
Ranking losses are better surrogate to ranking measures than regression losses

- 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

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

Verification on HW-SW Co-design

- Design **high accuracy and efficient** Process-In-Memory (PIM)-based system
- **Co-explore** the NN and PIM architectures based on **predictor-based NAS**

Method	NN accuracy	EDP $(ms \times mJ)$	Area (mm^2)	Search time (h)
NACIM [9]	73.9%	1.55	17.17	59
UAE [18]	83.0%	–	–	154
NAS4RRAM [10]	84.4%	–	–	289
CARS [16] (acc opt.)	88.0%	11.03	227.73	72
Gibbon (edp opt.)	84.6%	**0.24**	167.16	**7**
Gibbon (area opt.)	76.4%	1.00	**6.84**	**7**
Gibbon (acc opt.)	**88.3%**	14.33	186.32	**7**

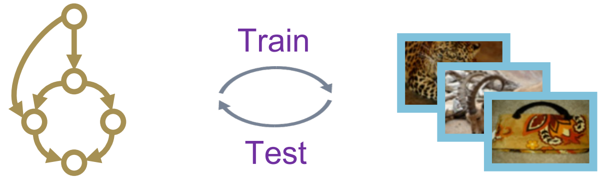
- **0.2~10.7%** accuracy promotion
- **2.51×** area reduction
- **6.48×** EDP reduction
- **8.4~41.3×** search efficiency improvement

Sun, Wang, Zhu, **Ning**+ et al., Gibbon: Efficient Co-Exploration of NN Model and Processing-In-Memory Architecture, DATE 22.

DELE: How to Train

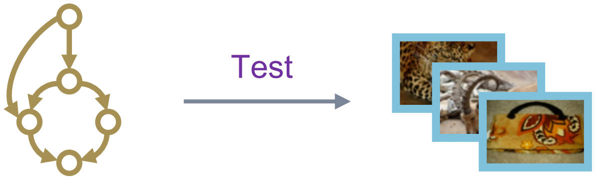
- Utilize **low-fidelity information** to help train the predictor
- More low-fidelity information **help learn better architecture representation**

Accurate but expensive



Ground-truth accuracy
(high-fidelity)

Less accurate but cheap



Zero-shot information
(low-fidelity)

Encoding Process

Training Process

Utilizing Low-fidelity Information

Zhao*, **Ning*** et al., “Dynamic Ensemble of Low-fidelity Experts: Mitigating NAS Cold-Start”, AAAI’23 Oral.

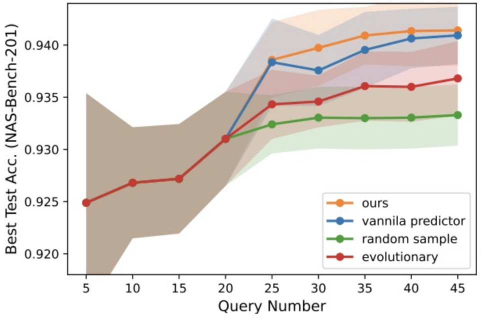
DELE: Results

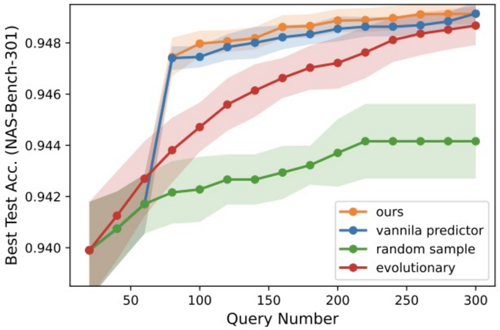
- Ranking correlation (Kendall's Tau)

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)

DELE achieves better ranking performance across different search spaces, encoders and training proportions.

- Sample efficiency

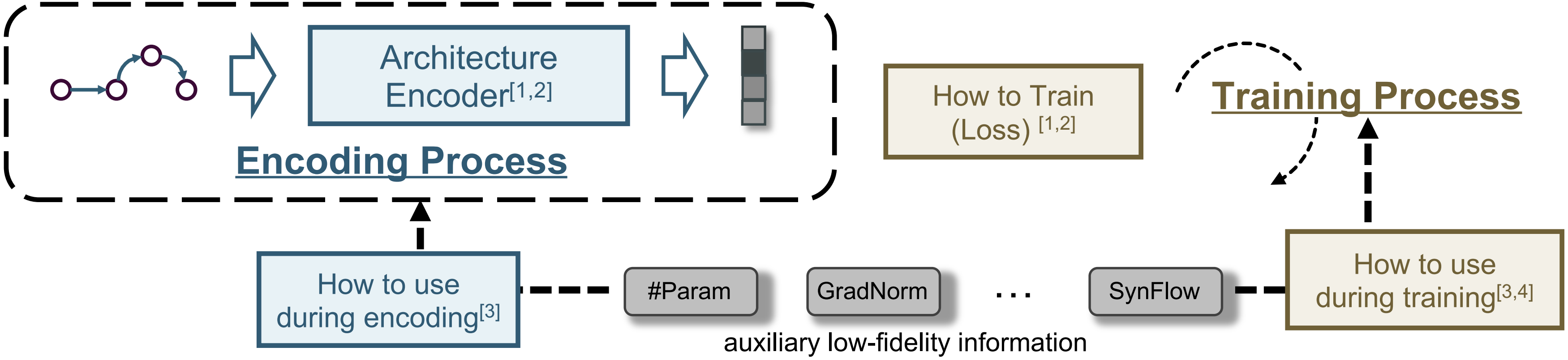




DELE discovers better architectures with less query number.

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

Summary of Architecture Encoding (Modeling)



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

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

GATES[1]
- Use encoding to rank architecture performance
- Maximize ranking ability i.e., Minimize ranking loss

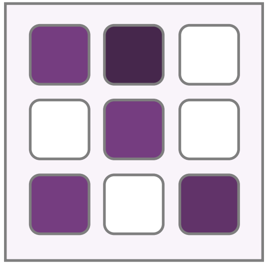
CLOSE[2]
- Use encoding to choose parameters for operations
- Minimize parameter-sharing loss, train jointly with supernet

DELE/GATES++[3,4]
- Low-fidelity information can be beneficial for modeling
- **Utilize low-fidelity information** in encoding and training

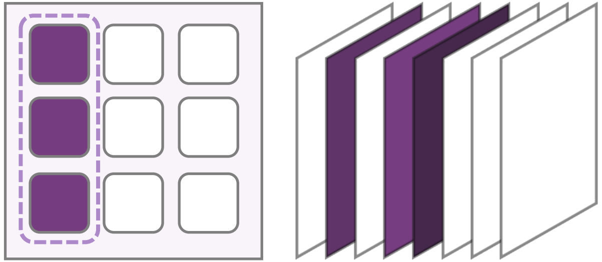
[1] **Ning** et al., "A Generic Graph-based Neural Architecture Encoding Scheme for Predictor-based NAS", ECCV'20.
[2] Zhou*, **Ning*** et al., "CLOSE: Curriculum Learning On the Sharing Extent Towards Better One-shot NAS", ECCV'22.
[3] **Ning** et al., "A Generic Graph-based Neural Architecture Encoding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

Neural Network Pruning

- Definition: Given the hardware-related constraints (e.g., latency, pruning ratio), prune the network structure to satisfy the constraints and minimize the accuracy loss
- Pruning Granularity: Structured Pruning / Unstructured Pruning



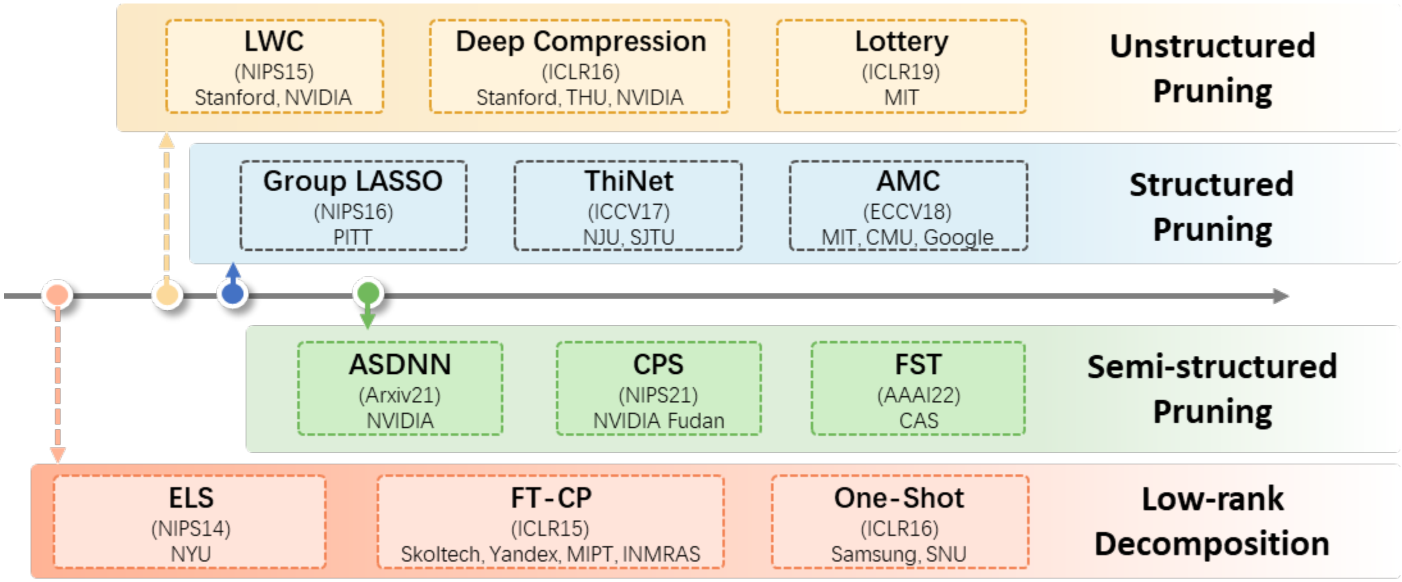
Unstructured Pruning
Granularity: Weight
(Example: Deep Compression [Han et. al. ICLR15])



Structured Pruning:
Granularity: Channel/Group
(Example: AMC [Han et. al. ECCV18])

Pruning Development Trends

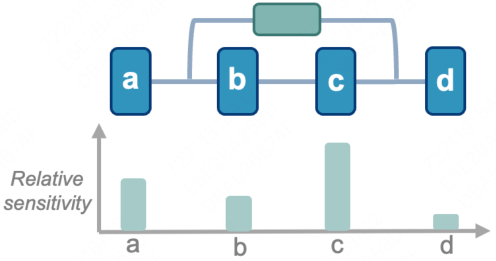
- Trends
  - Unstructured Pruning: Study on The Lottery Ticket Hypothesis
  - Structured/Semi-structured Pruning: Hardware-Software Co-design for better trade-off between sparsity and performance



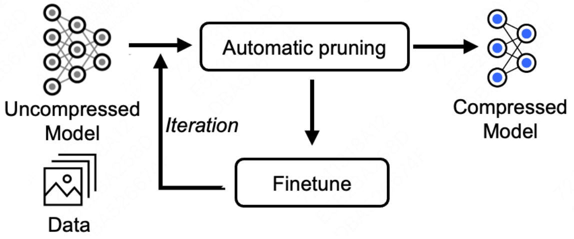
Our Work on Effective Pruning

Design **controllable** pruning schemes

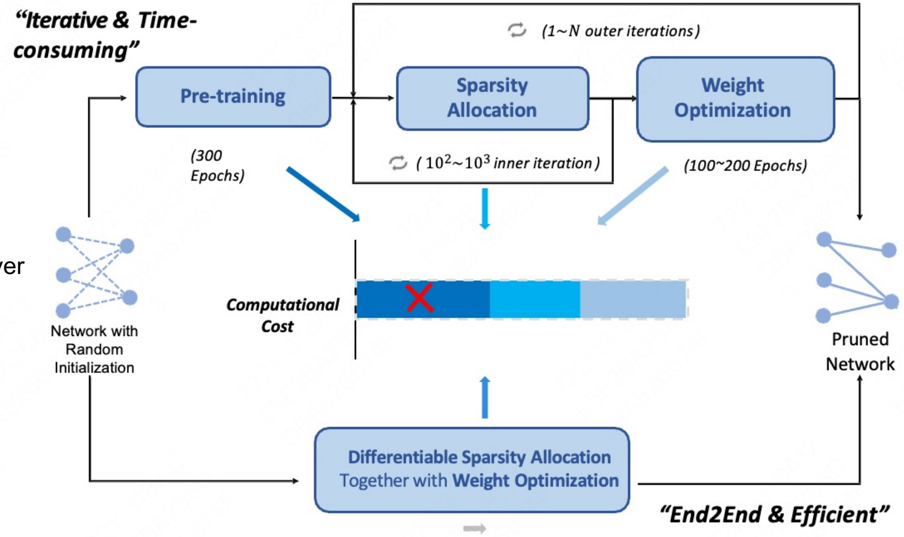
Programming-based iterative pruning[1,2]



1. Iteratively cut down resource consumption controllably
2. Each iter. is solved with a closed-form programming solver



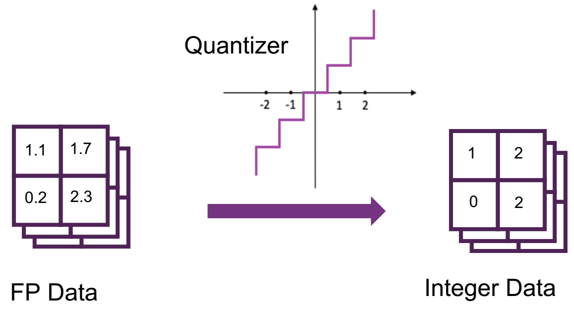
DSA: Differentiable Sparsity Allocation[3]



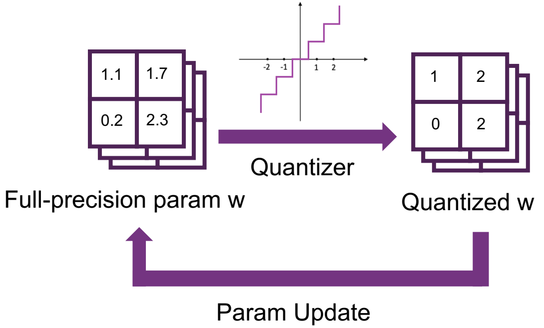
[1] Wang, **Ning** et al, Hardware Design and Software Practices for Efficient Neural Network Inference, *Low-Power Computer Vision*. Chapman and Hall/CRC 55-90. 2020.
[2] Shi*, **Ning***, Guo et al., Memory-Oriented Structural Pruning for Efficient Image Restoration, AAAI'23.
[3] **Ning** et al., DSA: More Efficient Budgeted Pruning via Differentiable Sparsity Allocation, ECCV'20.

Neural Network Quantization

- Definition: Given the hardware-related constraints (e.g., latency, bitwidth), quantize FP32 tensors (weights/activations) to satisfy the constraints and minimize the accuracy loss
- Quantization Methods: Post-Training Quantization / Quantization-Aware Training



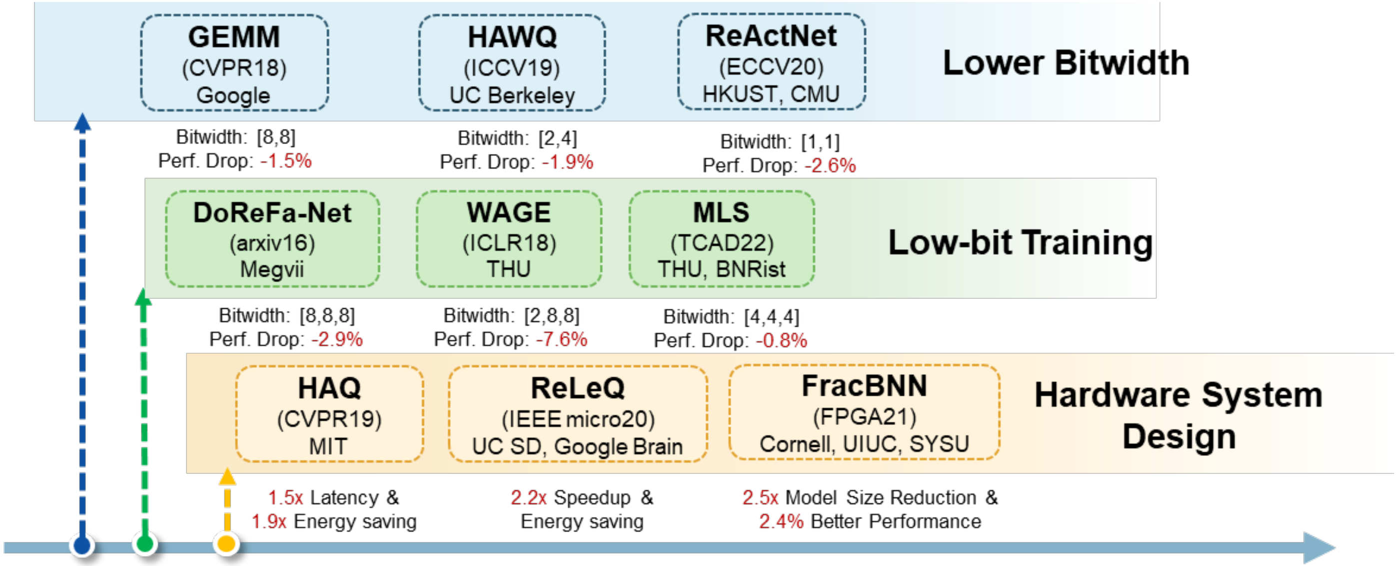
Post-Training Quantization
(Example: DFQ [Nagel et al. ICCV19])



Quantization-Aware Training
(Example: QT [Jacob et al. CVPR18])

Quantization Development Trends

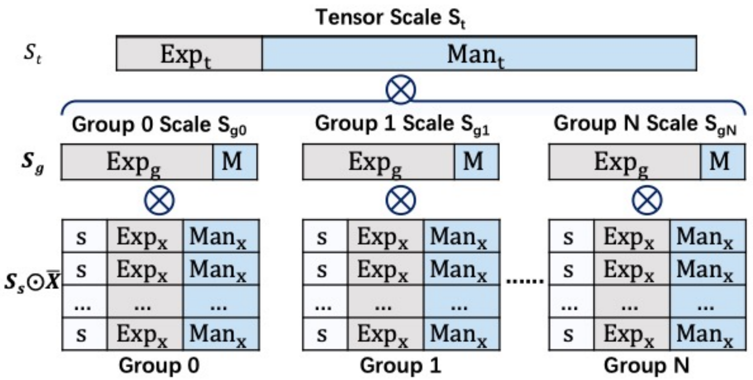
- Trends
  - Use lower bitwidth (8 → 4 → 2)
  - Low-bit inference → Training
  - Hardware-Algorithm co-design of the low-bit computing system

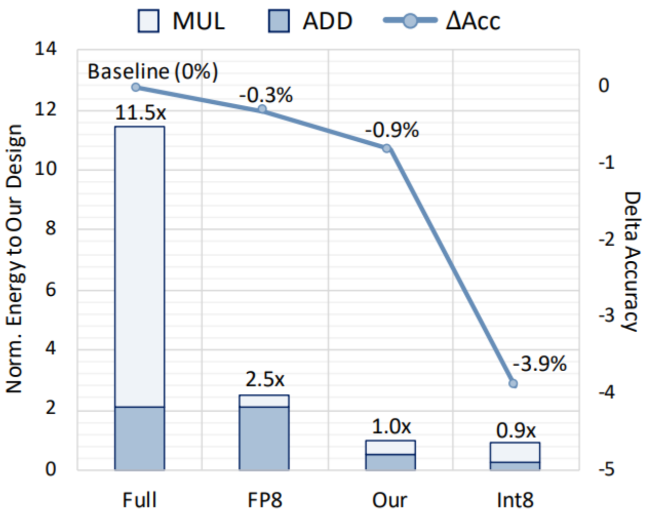


Our Work on Effective Quantization

Achieve efficient **low-bit training**

Special format design that (1) is **hardware-friendly**; (2) maintains **representation range**





- 4-/2-bit training within 1% accuracy loss
- 10.2× energy efficiency than FP32

[1] Zhong, **Ning** et al., Exploring the Potential of Low-bit Training of Convolutional Neural Networks, TCAD 22.

目录 Contents

目录 Contents

1. Why does efficient AI inference matters?
2. How to generally accelerate AI inference?
3. Research on specific applications
   - Low-level vision with pruning
   - Image generation with NAS
   - 3D perception with dynamic inference

目录 Contents

Background: Image Generation

- **Application field:** 2D image/3D assets generation task
  - Diverse Tasks & Efficiency Demands

"A shirt with the inscription: I love generative models"

Text-Image Generation

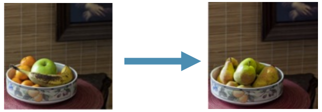


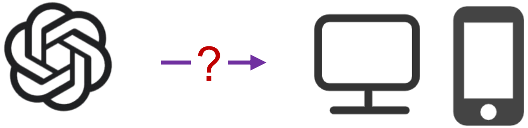
Image Editing

Image Super-resolution

3D Generation

Diverse Tasks

Low Memory: Deploying models on PC or Mobile Phone (**500M~1G runtime memory**) becomes popular in many application scenario.



Firm Real-time: The generation speed of the AI model should reach **1s/img**

- Stable Diffusion has been used as a plug-in to help paint in PhotoShop[1], which demands fast generation speed to satisfy the users' frequent requests.

Efficiency Demands

[1] Photoshop Stable Diffusion Plugin, https://github.com/AbdullahAlfaraj/Auto-Photoshop-StableDiffusion-Plugin

Application Challenges

- Challenges of Diffusion Models

Application Demands

- **Low Memory Cost: 500M~1G** runtime memory cost on edge devices
- **Low Latency**: Achieve **1s/img (firm real-time)** speed on edge devices

Future Trends

- **Larger-scale:** Larger resolution, more training data
- **More tasks & Multi-Modal :** More conditioning types and more data modal.

Efficiency of current methods

Nvidia A100

Sampler*	Memory (512x512)	Latency (512x512)
DDIM[2]	~12G	7~14s/img (100-200 NFE)
PNDM[3]		3.5~7s/img (50-100 NFE)
DPM-Solver[4]		1.4~3.5s/img (20-50 NFE)

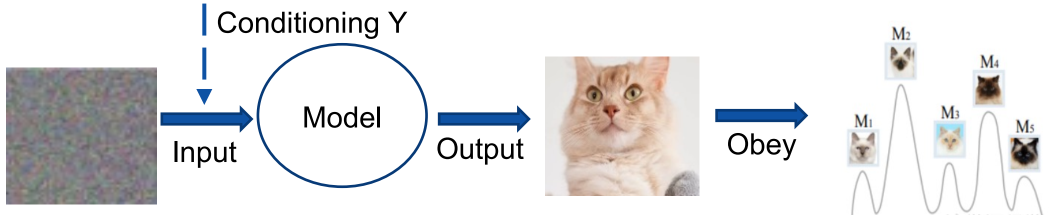
* Stable diffusion inference using different samplers.

Call for 10x~100x Efficiency Improvement

[1] Rombatch et al., High-Resolution Image Synthesis with Latent Diffusion Models, CVPR'22.
[2] Song et al., Denoising Diffusion Implicit Models, ICLR'21.
[3] Liu et al., Pseudo Numerical Methods for Diffusion Models on Manifolds, ICLR'22.
[4] Lu et al., DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps, NeurIPS'22.

Background: Generative Model

- Generative model aims at learning the **data distribution**



Prior Distribution

Data Distribution X

2011
Auto-Regressive Model

2013
Variational Auto-Encoder

Low quality ☹

2014
Generative Adversarial Network

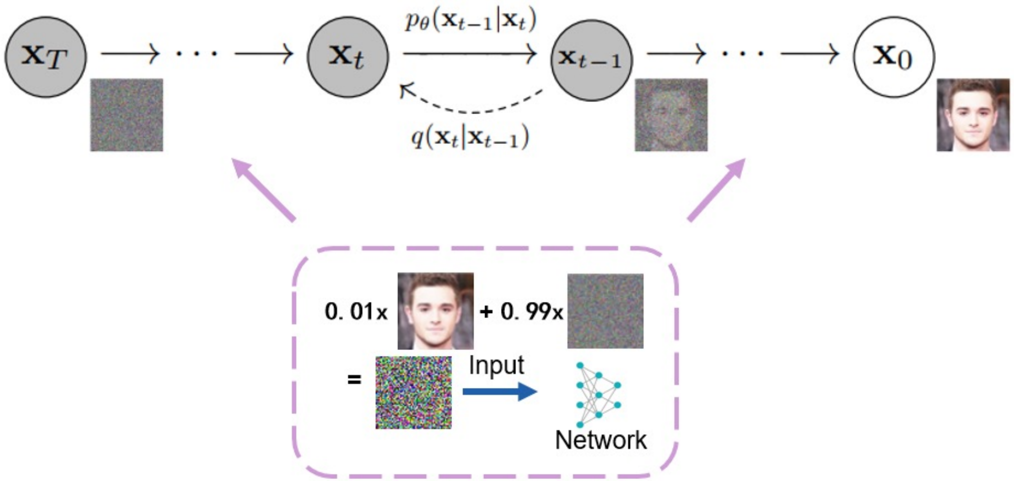
Mode Collapse
Unstable training ☹

2020
Diffusion Model

High quality
Stable training ☺

Background & Motivation

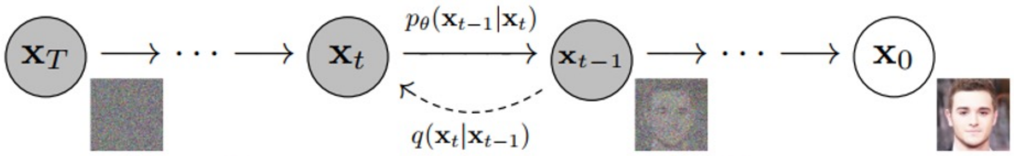
- Diffusion in a nutshell: learn to **iteratively denoise a gaussian noise** for generation



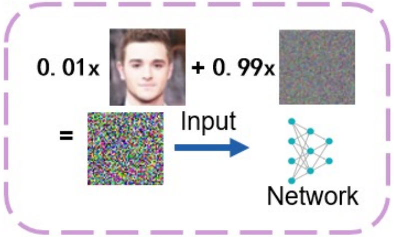
Sampler	NFE	Inference time per step (NVIDIA A100, BS=1)	Total Time (NVIDIA A100, BS=1)
DDIM[2]	100~200	~0.07s	7-14s/image
PNDM[3]	50~100		3.5~7s/image
DPM-Solver[4]	20~50		1.4~3.5s/image

Background & Motivation

- Diffusion in a nutshell: learn to **iteratively denoise a gaussian noise** for generation



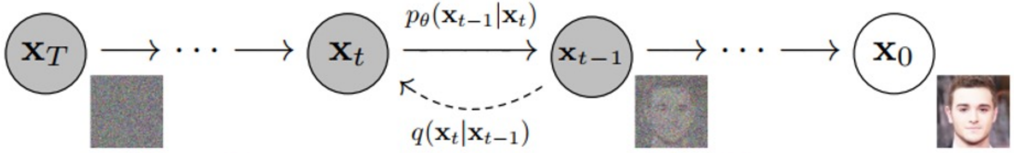
1. The number of sampling steps is large



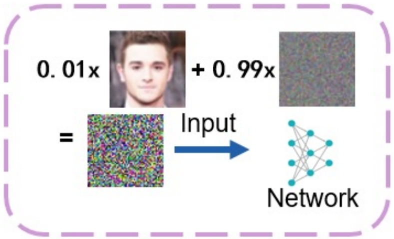
Sampler	NFE	Inference time per step (NVIDIA A100, BS=1)	Total Time (NVIDIA A100, BS=1)
DDIM[2]	100~200	~0.07s	7-14s/image
PNDM[3]	50~100		3.5~7s/image
DPM-Solver[4]	20~50		1.4~3.5s/image

Background & Motivation

- Diffusion in a nutshell: learn to **iteratively denoise a gaussian noise** for generation



1. The number of sampling steps is large

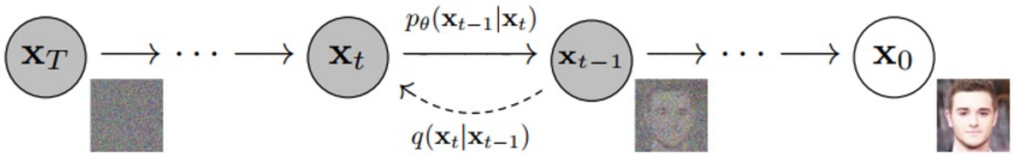


2. The inference speed for each step is slow

Sampler	NFE	Inference time per step (NVIDIA A100, BS=1)	Total Time (NVIDIA A100, BS=1)
DDIM[2]	100~200	~0.07s	7-14s/image
PNDM[3]	50~100		3.5~7s/image
DPM-Solver[4]	20~50		1.4~3.5s/image

Background & Motivation

- Diffusion in a nutshell: learn to **iteratively denoise a gaussian noise** for generation



1. The number of
samplin

Network

d
for each step is slow

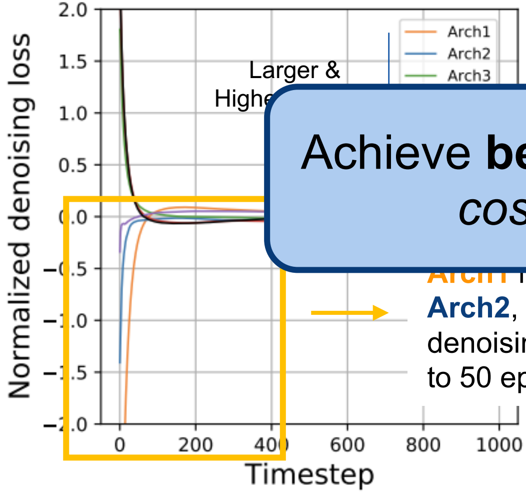
Our research goal is to **accelerate the inference speed** of the diffusion model

Sampler	NFE	Inference time per step (NVIDIA A100, BS=1)	Total Time (NVIDIA A100, BS=1)
DDIM[2]	100~200	~0.07s	7-14s/image
PNDM[3]	50~100		3.5~7s/image
DPM-Solver[4]	20~50		1.4~3.5s/image

Idea and Oracle Experiments

- Existing SOTA methods adopt the **same model** during **all the diffusion steps**
  - **Use different models in different steps to reduce the computation demand?**

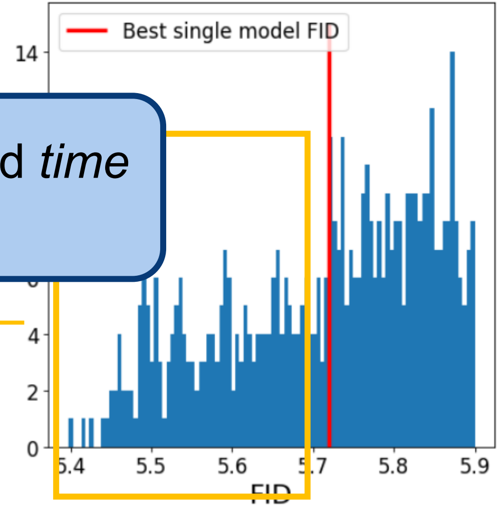
Observation1 **Small models outperform** large models at **some steps** in terms of denoising loss



Arch1 is smaller than **Arch2**, but obtains smaller denoising loss between 1 to 50 epochs.

Observation2 Some **mixtures of large and small models** get **better** performance than **only use large model**

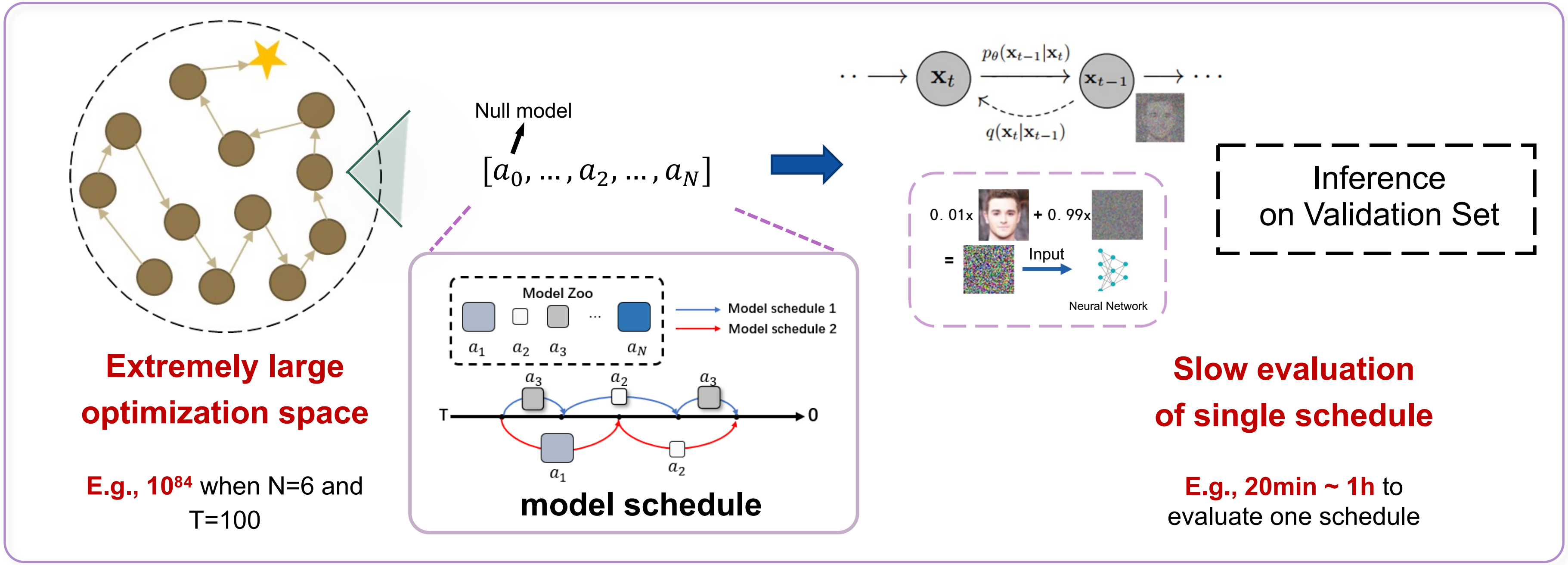
Mixing small and large models can possibly get better FID than only use large model



Achieve **better trade-off** between *quality* and *time cost* by optimizing **model schedule**

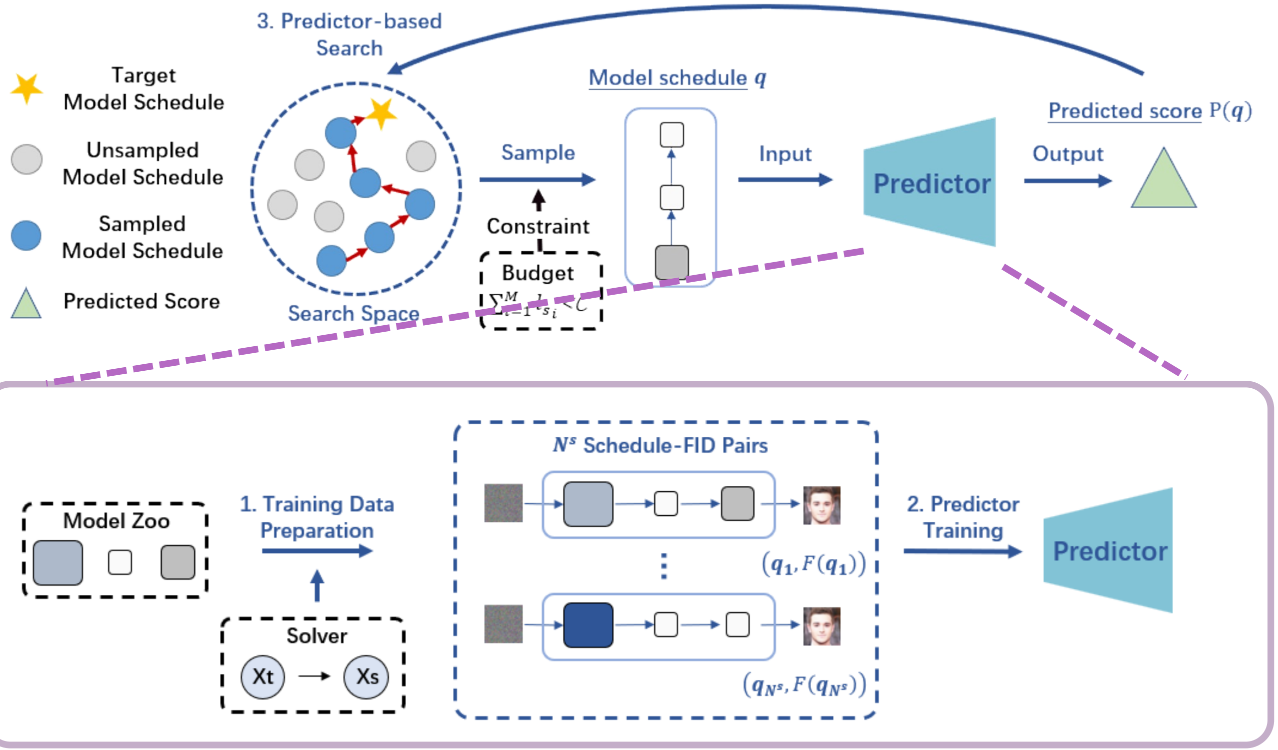
Problem Formulation and Challenges

- Formulation: Given a **pre-trained model zoo** and the **generation time constraint**, search for the **model schedule** to satisfy the constraint and <u>maximize the quality score (i.e., FID)</u>
- Challenges: **Slow search process**

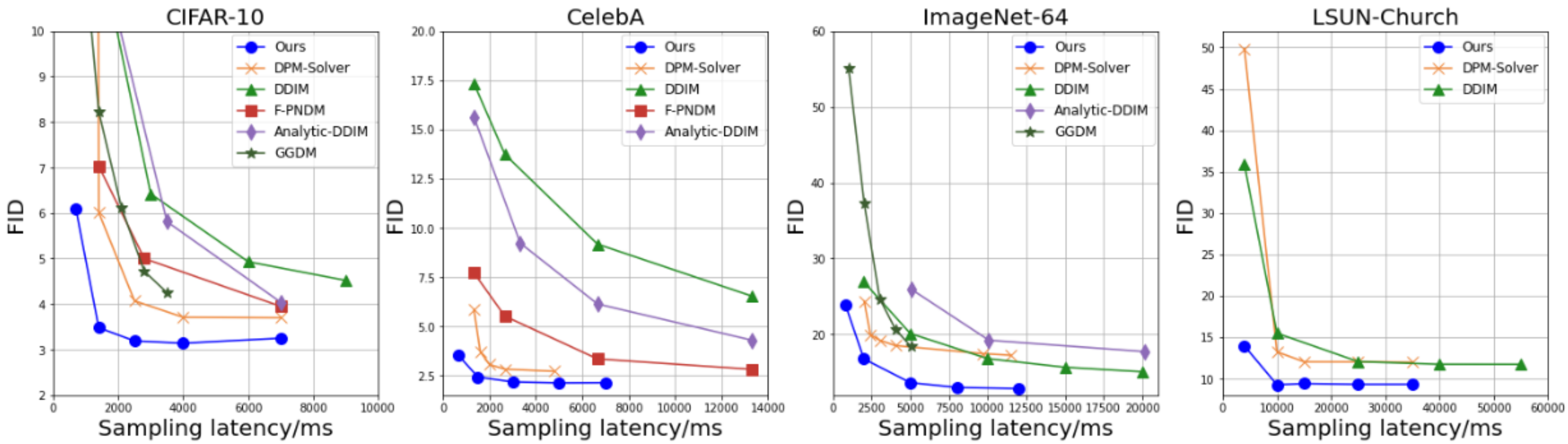


Method

- **Predictor-based search** for a good model schedule satisfying the constraint



- **Experimental Results**
  - Accelerate generation process by **2-5x** (on A100) without sacrificing the performance



Results

- **Experimental Results**
  - Accelerate generation process by **2-5x** (on A100) without sacrificing the performance
  - Achieve **better FID** than ***Stable Diffusion*** by optimizing the schedule

Budget/NFE	Baseline Type (1)	Baseline Type (2)	Ours
9	13.01	13.01	**12.90**
12	12.11	11.37	**11.34**
15	11.92	11.13	**10.72**
18	11.88	11.13	**10.68**
24	11.81	11.13	**10.57**

Table 3. FID on MS-COCO 256×256. All FIDs in the table are calculated between 30k images in validation set and 30k generated images guided with the same captions.

Results

- **Demonstration** of generated images with the same generation time

Previous SOTA Method

Our Method

Empirical Insights

Unconditional Generation

- Common
  - Low budgets: Small models & Sufficient steps
  - High budgets: Large models
- Varies with dataset
  - CIFAR-10/CelebA/ImageNet-64 (32x32, 64x64):
    - Larger models & Denser steps for lower t
  - LSUN-Church (256x256):
    - Smaller models & Sparser steps for lower t

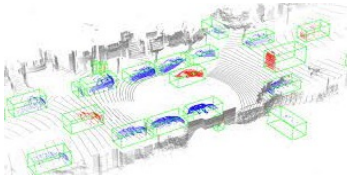
Conditional Generation with Stable Diffusion (512x512)

- Mixing checkpoints with the same complexity but different functionalities is still beneficial for the efficiency-quality trade-off
- 1-st solver is preferred for lower t

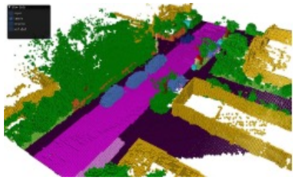
目录 Contents

Background: 3D Perception

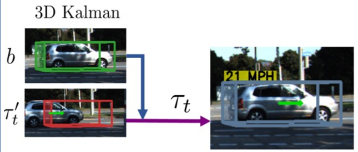
- **Application field:** 3D scene understanding for autonomous driving
  - Diverse Tasks & Efficiency Demands



3D Detection



3D Segmentation



3D Tracking

Line Detection
Map Segmentation
Trajectory Prediction
…

Diverse Tasks

Perception → Localization → Prediction → Control

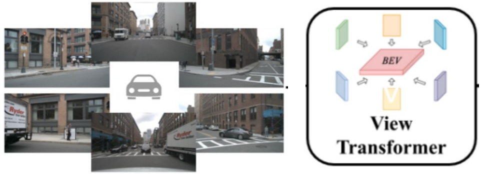
Real-time: According to different scenarios (e.g., APA, High-way, City) and functionalities, the **perception** system should reach **10~30Hz** (e.g., Tesla vision can reach 36Hz)

Efficiency Demands

Background: 3D Perception

- **Application field:** 3D scene understanding for autonomous driving
  - Different schemes

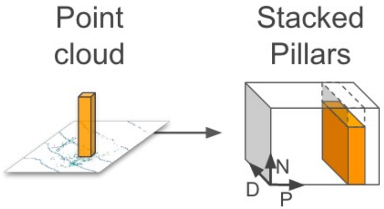
Camera



Multi-View Image

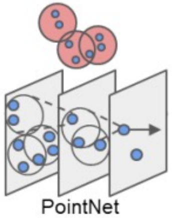
- *BEVDet*[1]
- *BEVFormer*[2]

Lidar



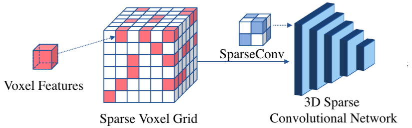
Pseudo 3D (Pillar)

- *PointPillar*[3]



Point

- *PointRCNN*[4]
- *3DSSD*[5]



Voxel

- *PVRCNN*[6]
- *CenterPoint*[7]

Good task performance!

[1] Huang et al., : BEVDet: High-performance Multi-camera 3D Object Detection in Bird-Eye-View, Arxiv.
[2] Li et al., BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers, ECCV22.
[3] Alex et al., PointPillars: Fast Encoders for Object Detection from Point Clouds, CVPR19.
[4] Shi et al., PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud, CVPR19.
[5] Yang et al., 3DSSD: Point-based 3D Single Stage Object Detector, CVPR20.
[6] Shi et al., PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection, CVPR20.
[7] Yin et al., Center-based 3D Object Detection and Tracking, CVPR21.

Application Challenges

- Application challenges of 3D Perception Methods

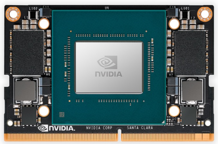
Application Demands

- **Low Latency:** Perception **10~30FPS**
- **Low Power:** Resource-constrained

Futu

- **Larger-scale:** Larger scene, more data
- **Time Dimension:** Process multiple frames in point cloud video
- **Multi-Modal:** Process multiple modality

Efficiency of current methods



Nvidia Xavier

Nvidia RTX2080

based)	
PVRCNN[6] *(Voxel-based)*	~3

	FPS
(Pillar-based)	~60
] *(Point-*	~8
PVRCNN[6] *(Voxel-based)*	~8
BEVDet[1] *(Image-based)*	~10

Our research goal is to **accelerate the inference speed** of the current 3D perception methods

Call for ~10x Efficiency Improvement

Idea

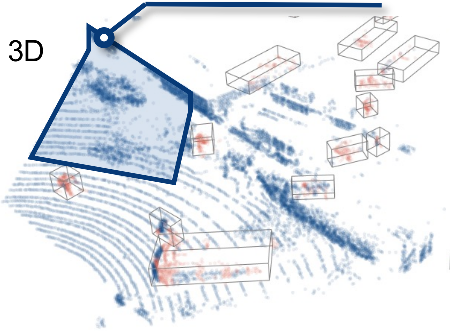
- Exploit the Spatial Data Redundancy with Adaptive Inference

Idea

3D detector needs to process large scale point cloud

Spatial redundancy exists for both the 3D and BEV space

Redundant *background points*

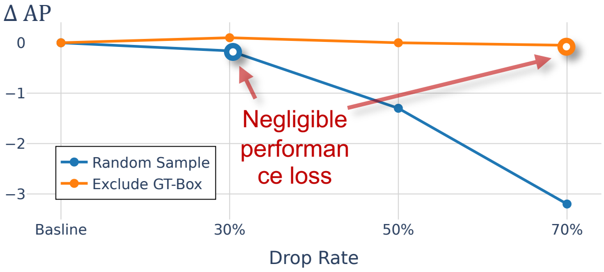


Redundant *background pixels*

(Only 5% nonzero pixels projected from 3D voxels)

Oracle

Validate the spatial redundancy exists, and correctly dropping redundant part brings negligible performance drop.



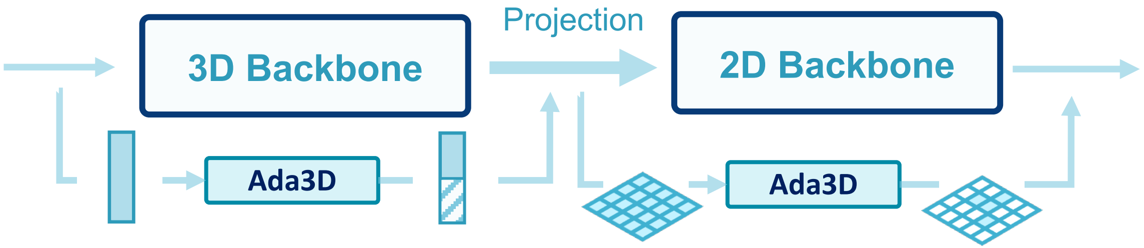
(Upper bound for adaptive inference)*

Idea

- Exploit the Spatial Data Redundancy with Adaptive Inference

Idea

- Reduce spatial redundancy by adaptively skipping computation and saving for redundant 3D/2D features during inference.
- A share lightweight predictor could effectively identify redundant features for each layer.

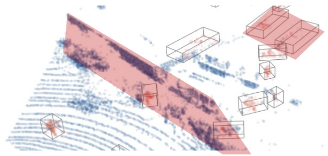


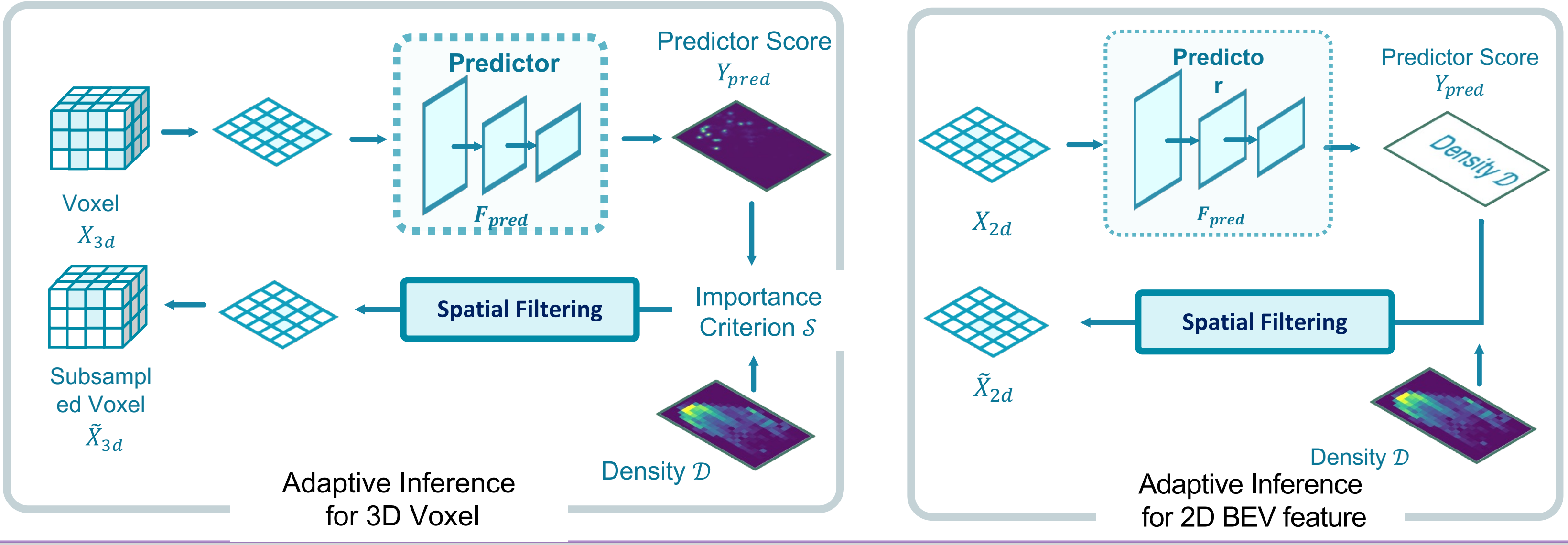
Reduce
redundant 3D voxels

Reduce
redundant 2D pixels

Methods

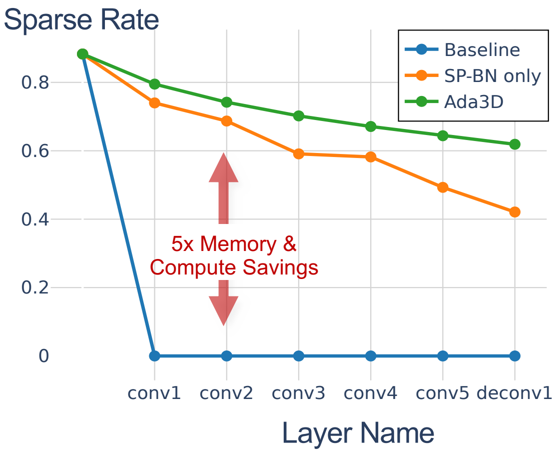
- **Adaptive inference**
  - Feature-based Importance Predictor
  - Density-guided Spatial Filtering

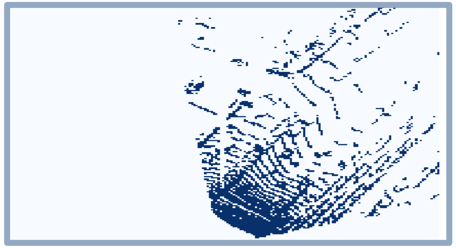

Local Point cloud have higher density



Methods

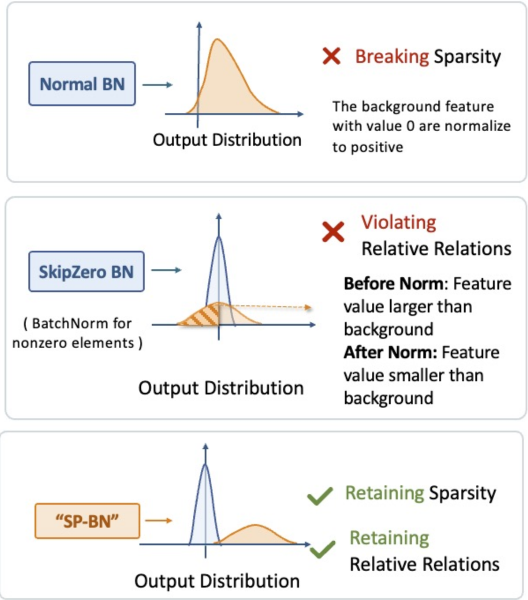
- **Sparsity-Preserving Batch Normalization**
  - Apply batch normalization for nonzero elements without mean substraction





(Only 5% nonzero pixels projected from 3D voxels)

Most existing literatures treat the 2D BEV space as dense features



SP-BN retains the sparsity of 2D BEV features

Experimental Results

- **Experiments on KITTI Test**
  - Ada3D achieves comparable performance with baseline methods with 5~10x FLOPs and memory compress rate.

Table 1: **Performance comparison of Ada3D and other methods on KITTI *test* set.** The “Ada3D-B” and “Ada3D-C” are centerpoint models optimized by Ada3D with different drop rates.

Mehod	*FLOPs Opt.*	*Mem Opt.*	mAP *(Mod.)*	3D Car (IoU=0.7)			3D Ped. (IoU=0.5)			3D Cyc. (IoU=0.5)		
				Easy	*Mod.*	*Hard*	*Easy*	*Mod.*	*Hard*	*Easy*	*Mod.*	*Hard*
VoxelNet [30]	-	-	49.05	77.47	65.11	57.73	39.48	33.69	31.50	61.22	48.36	44.37
SECOND [21]	-	-	57.43	84.65	75.96	68.71	45.31	35.52	33.14	75.83	60.82	53.67
PointPillars [8]	-	-	58.29	82.58	74.31	68.99	51.45	41.92	38.89	77.10	58.65	51.92
SA-SSD [4]	-	-	-	88.75	79.79	74.16	-	-	-	-	-	-
TANet [12]	-	-	59.90	84.39	75.94	68.82	53.72	44.34	40.49	75.70	59.44	52.53
Part-A^2 [17]	-	-	61.78	87.81	78.49	73.51	53.10	43.35	40.06	79.17	63.52	56.93
SPVCNN [20]	-	-	61.16	87.80	78.40	74.80	49.20	41.40	38.40	80.10	63.70	56.20
PointRCNN [16]	-	-	57.95	86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53
3DSSD [24]	-	-	55.11	87.73	78.58	72.01	35.03	27.76	26.08	66.69	59.00	55.62
IA-SSD [28]	-	-	60.30	88.34	80.13	75.10	46.51	39.03	35.60	78.35	61.94	55.70
CenterPoint [25]	-	-	59.96	88.21	79.80	76.51	46.83	38.97	36.78	76.32	61.11	53.62
CenterPoint-Pillar [25]	-	-	57.39	84.76	77.09	72.47	44.07	37.80	35.23	75.17	57.29	50.87
CenterPoint (Ada3D-B)	**5.26×**	**4.93×**	**59.85**	87.46	79.41	75.63	46.91	39.11	36.43	76.09	61.04	53.73
CenterPoint (Ada3D-C)	**9.83×**	**8.49×**	**57.72**	82.52	74.98	69.11	43.66	38.23	34.80	75.27	59.96	52.14

Experimental Results

- **Experiments on KITTI Val**
  - Ada3D-A: Improve the performance with 80% 2D reduction
  - Ada3D-B: Without performance loss, reduce 40% 3D features, 80% 2D features, compress FLOPs and memory cost by 5x
  - Ada3D-C: With moderate performance loss, reduce 60% 3D features. 90% 2D features, improve model FLOPs and memory by **an order of magnitude**

Table 3: **Ablation studies and quantitve efficiency improvements of different Ada3D models on KITTI *val*.** "IP" stands for "importance predictor", "DG" for "density-guided spatial filtering", "SP-BN" for "sparsity preserving batch normalization". The "FLOPs" and "Mem." calculates the normalized resource consumption of the optimized model.

Method	*Technique*			*FLOPs*		*Mem.*		mAP	Car Mod.	Ped. Mod.	Cyc. Mod.
	IP	DG	SP-BN	*3D*	*2D*	*3D*	*2D*	(Mod.)	(IoU=0.7)	(IoU=0.5)	(IoU=0.5)
CenterPoint	-	-	-	1.00	1.00	1.00	1.00	66.1	79.4 (-)	53.4 (-)	65.5 (-)
CenterPoint (SP-BN)	-	-	✓	1.00	0.49	1.00	0.45	66.0	79.1 (-0.3)	53.3 (-0.1)	65.6 (+0.1)
CenterPoint (Ada3D-A)	✓	✓	✓	1.00	0.22	1.00	0.25	66.4	79.5 (+0.1)	53.6 (+0.2)	66.1 (+0.6)
CenterPoint (Ada3D-B)	✓	✓	✓	0.66	0.18	0.68	0.17	66.1	79.1 (-0.3)	54.0 (+0.6)	65.3 (-0.3)
CenterPoint (Ada3D-B w.o. DG)	✓	-	✓	0.64	0.18	0.66	0.16	65.1	78.8 (-0.6)	51.6 (-1.8)	64.9 (-0.6)
CenterPoint (Ada3D-C)	✓	✓	✓	0.39	0.08	0.43	0.07	65.4	77.6 (-1.8)	53.5 (+0.2)	65.1 (-0.4)

Experimental Results

- **Experiments on nuScenes and ONCE**
  - With less than 1% mAP loss, compress the FLOPs and memory cost by 2~4x
  - Achieve higher compression rate with less performance loss than model-level compression methods (e.g. SPSS-pruning & Channel Scaling)

Table 1: **Performance comparison of Ada3D on ONCE *val* set.** The results are taken from the OpenPCDet [5] implementation.

Method	*FLOPs Opt.*	*Mem. Opt.*	mAP	Veh.	Ped.	Cyc
PointRCNN [3]	-	-	28.74	52.09	4.28	29.84
PointPillar [1]	-	-	44.34	68.57	17.63	46.81
SECOND [6]	-	-	51.89	71.16	26.44	58.04
PVRCNN [2]	-	-	53.55	77.77	23.50	59.37
CenterPoint [7]	-	-	63.99	75.69	49.80	66.48
CenterPoint *(with Ada3D)*	2.32×	2.61×	62.68	73.43	49.09	65.53

Table 2: **Performance comparison of Ada3D on Nuscenes *val* set.** The "SPSS-Conv" model applies pruning for the 3D sparse convolution only, and the "CenterPoint-Mini" uses the 2D backbone with half the usual width.

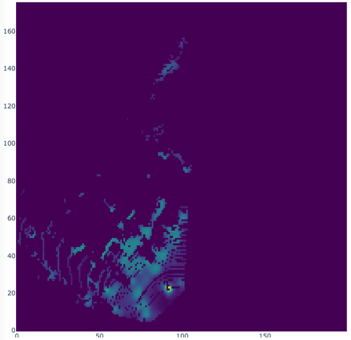
Method	*FLOPs Opt.*	*Mem. Opt.*	mAP	NDS
PointPillar [8]	-	-	44.63	58.23
SECOND [21]	-	-	50.59	62.29
CenterPoint-Pillar [25]	-	-	50.03	60.70
CenterPoint [25] *(voxel=0.1)*	-	-	55.43	64.63
CenterPoint-Ada3D *(voxel=0.1)*	2.32×	2.61×	54.80	63.53
CenterPoint [25] *(voxel=0.075)*	-	-	59.22	66.48
SPSS-Conv [11] *(voxel=0.075)*	1.14×	1.14×	57.80	65.69
CenterPoint-Mini [25] *(voxel=0.075)*	2.78×	2.78×	57.19	64.08
CenterPoint-Ada3D *(voxel=0.075)*	3.34×	3.96×	58.62	65.68

Outperform model-level compression methods

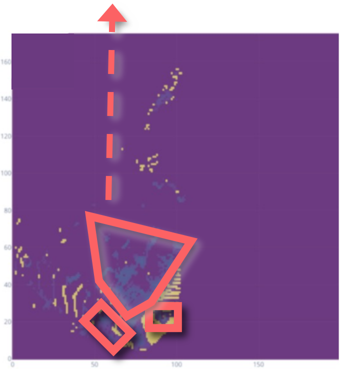
Qualitative Results

- **Visualization of 2D & 3D heatmap**
  - The predictor identifies the voxels/pixels inside the bounding box
  - The spatial filtering avoids dropping inbox data

Importance-Predictor:
(Filtered out boxes)



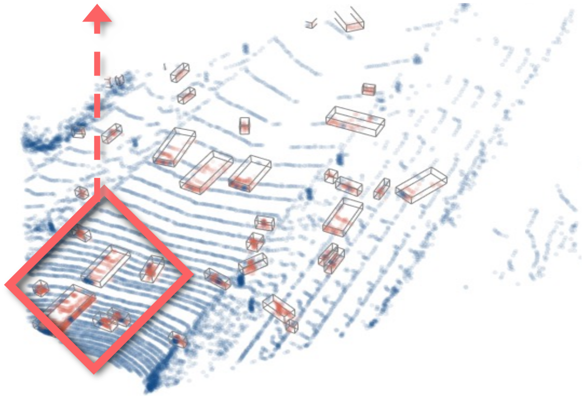
Predicted Heatmap



Dropped Input

Bounding Boxes

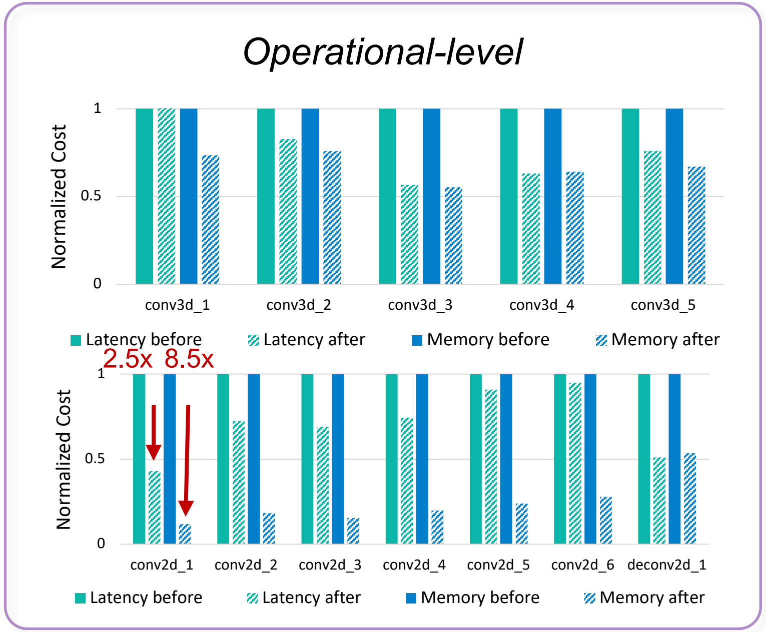
Inbox voxels have large predictor scores
(Red presents larger importance, Blue means lower)

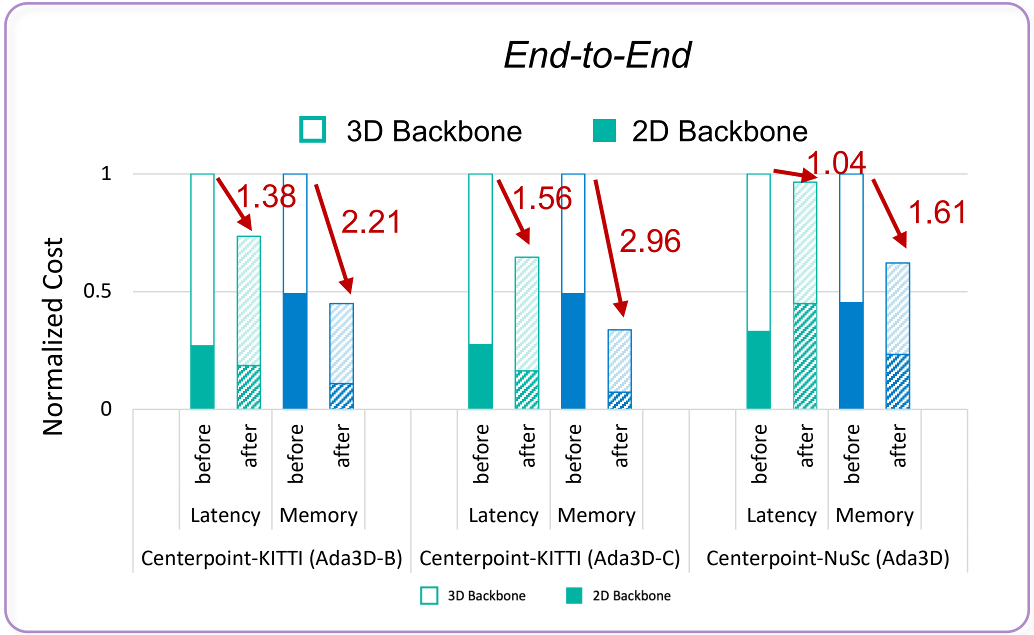


Predicted Heatmap

Hardware Experiments

- **Hardware Profiling on RTX3090 GPU**
  - Ada3D-B achieves 1.38x latency and 2.21x GPU (RTX 3090) peak memory optimization



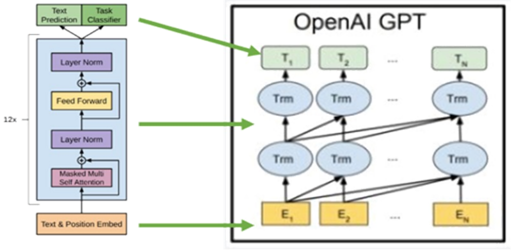


Summary

Real-time Requirement

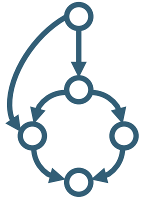
E.g., Video Conference ~30FPS
Mobile Phone / PC (1~10 TOPS)

High Latency

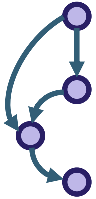


E.g., GPT-3: 175B, 664TOPs, 32s/seq
NVIDIA A100 (19.5 TFLOPS)

Research Direction: Model Compression



→ **NAS / Pruning** → **Model Quantization** →

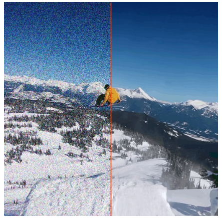


How to **efficiently search**

How to **controllably prune**

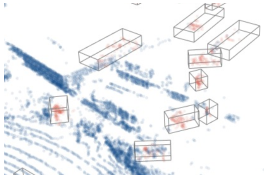
How to **efficiently quant**

Apply to practical scenarios



Low-level Vision
2x peak mem. reduce

Image Generation
2~5x speedup



3D Perception
1.38x speedup & 2.21x peak mem. reduce

Thank You !

Xuefei Ning foxdoraame@gmail.com
Department of Electronic Engineering, Tsinghua University

2023.03.16

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

Thanks to our students: Zixuan Zhou, Tianchen Zhao, Shiyao Li, Xiangsheng Shi, Lidong Guo, Junbo Zhao, Enshu Liu