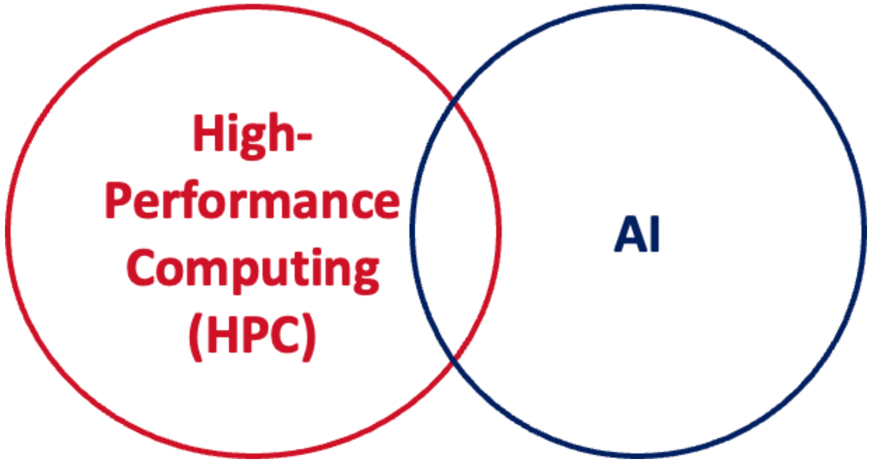
Weight-Sharing NAS with Architecture-Agnostic Intermediate Representation

Presenter: Mahdi Samani

Software Analytics & Pervasive Parallelism Lab

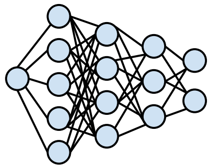
SwAPP



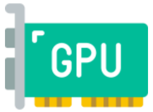
- The Laboratory for Software Analytics and Pervasive Parallelism (SwAPP)
  - Investigates the challenges that advance state-of-the-art in building reliable and efficient data-driven applications utilizing **AI/analytical methods and HPC**
- SwAPP lab is primarily focused on the **intersection of HPC (parallel computing & HPC) and AI (Data Science)** and includes
  - Efficient and Scalable Learning and Inference
  - High Performance Deep Learning
  - Software analytics (AI for HPC & Cyberinfrastructure)

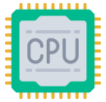
Challenges: Resource-Heterogeneous Machine Learning

Resources and Capabilities Vary Among Different Entities, Impacting Deep Neural Networks' (DNNs) Performance



Train/Deploy

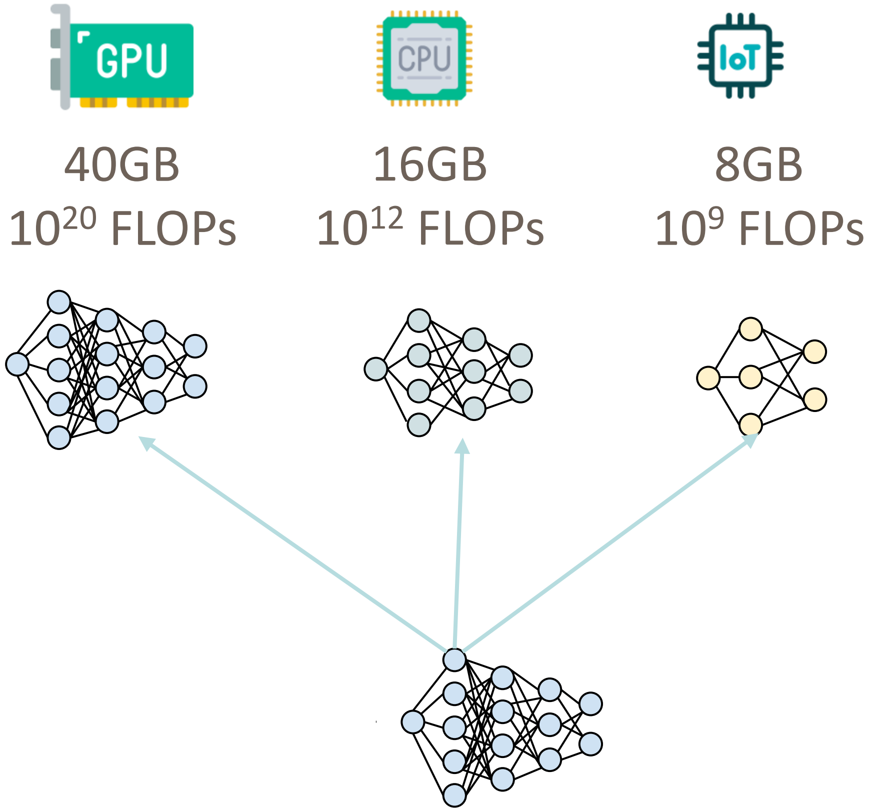




	GPU	CPU	IoT
	40GB 10^{20} FLOPs	16GB 10^{12} FLOPs	8GB 10^{9} FLOPs
Inference latency	10ms	70ms	2s

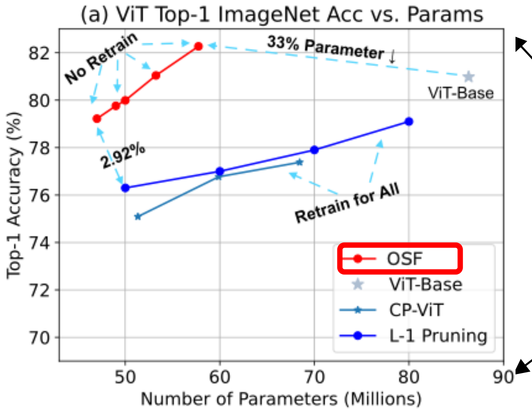
Research Objective

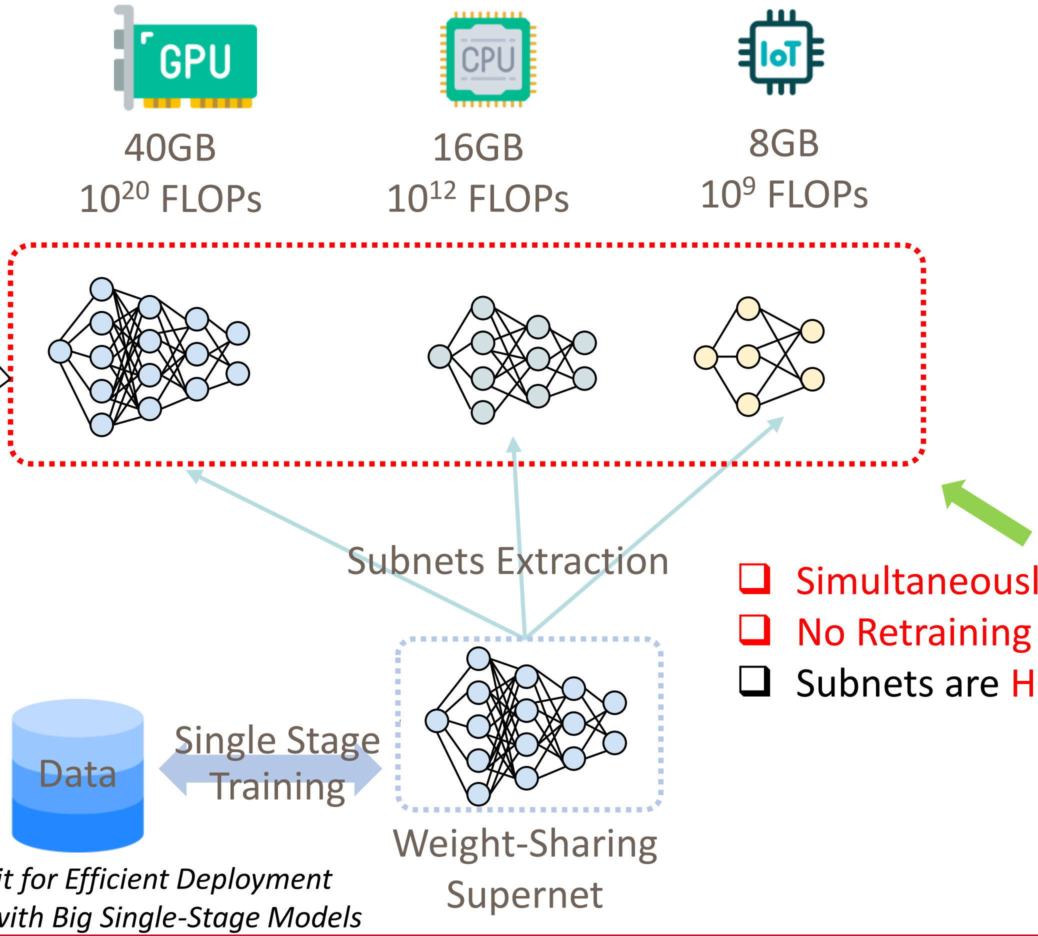
Goal: Specialize the DNNs and Improve the Resource Efficiency without Significant Compromise Model Performance



Neural Architecture Search With Supernet

A Well-trained Weight-Sharing Supernet can Generate a Huge Number of High-performance Subnets and Fit for a Wide-range of Constrains





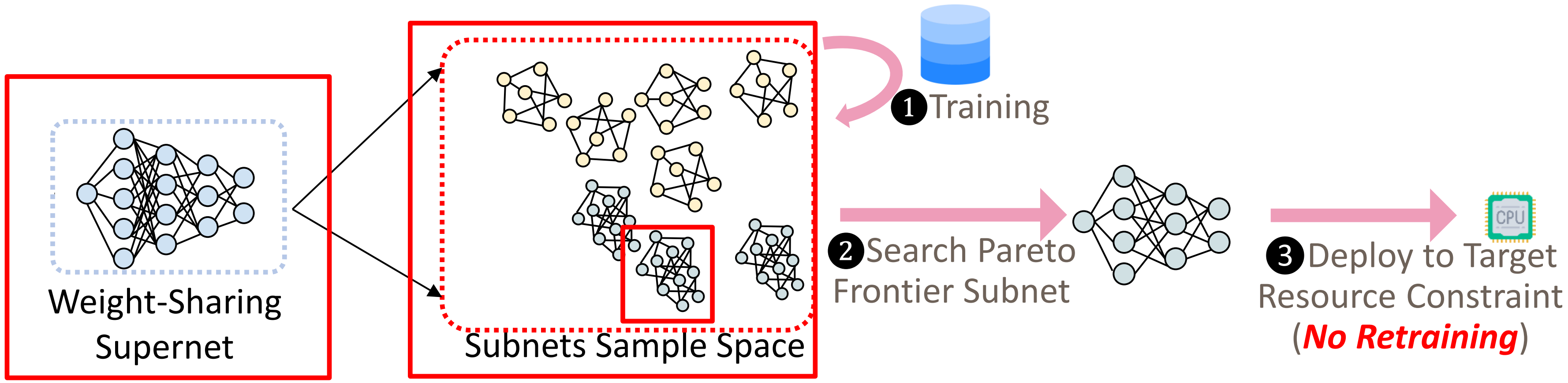
- ❑ Simultaneously delivery a set of Pareto Frontier models
- ❑ No Retraining or Finetuning!
- ❑ Subnets are High-performance

Cai et al. Once for All: Train One Network and Specialize it for Efficient Deployment
Yu et al. BigNAS: Scaling Up Neural Architecture Search with Big Single-Stage Models

Weight-Sharing Supernet Training Objective

Construct A Weight-Sharing Supernet Requires:

Jointly Train Potential Weight-Sharing Subnets and Minimize the Global Loss



Supernet Weights (Shared Weights)

Validation Loss

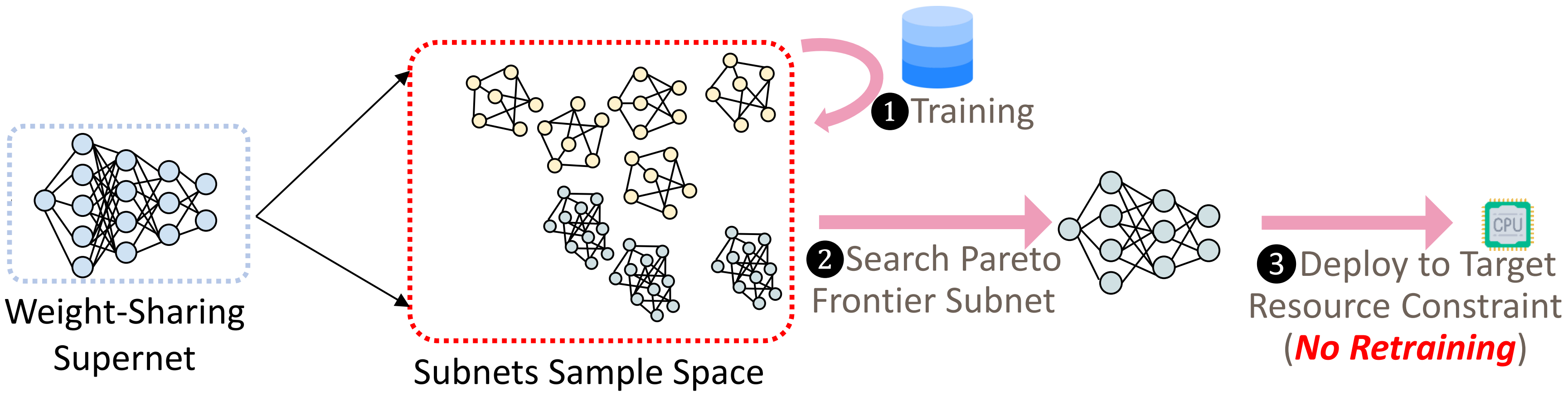
Subnets Architecture

$$W^* = \min_{W_0} \sum_{Subnet_i \in \mathcal{A}} L_{val}(F(W_0, Subnet_i))$$

Subnet Extraction Rules/Heuristics

Neural Architecture Search With Supernet

After Weight-Sharing Supernet is Well-trained, Search for Pareto Frontier Subnets for Target Resource Constraints



Evaluation Performance

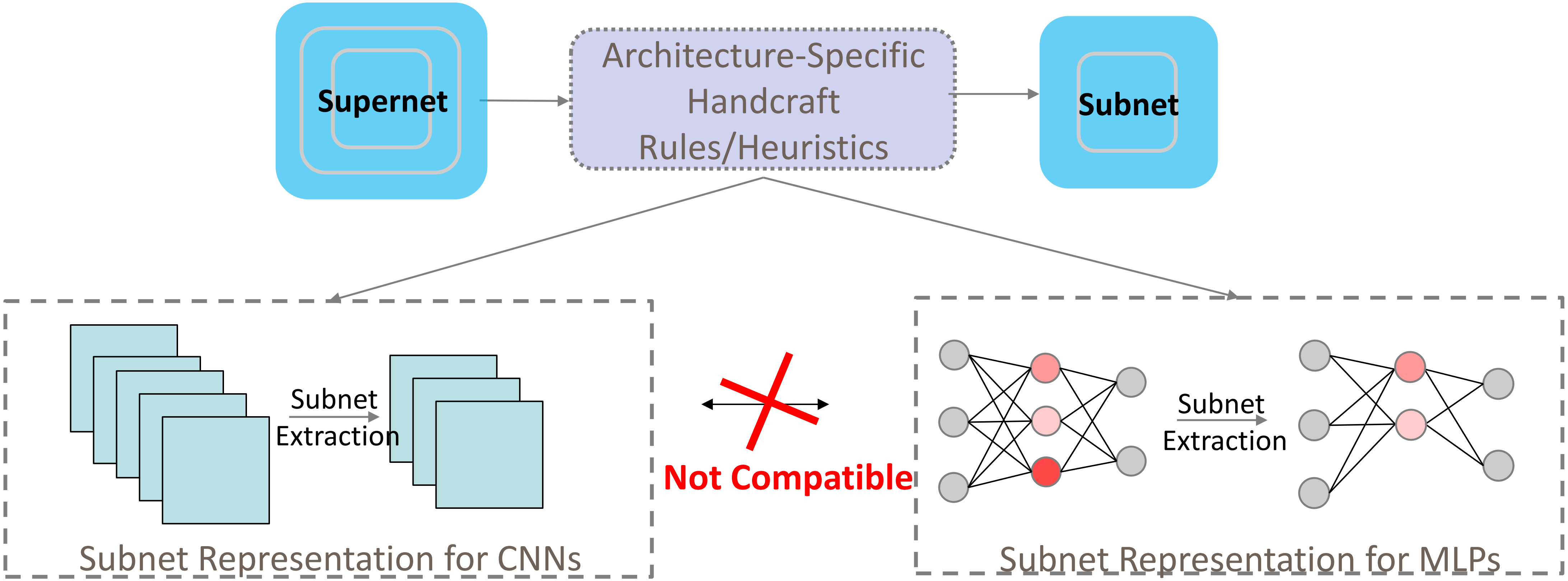
Resource Overhead

Resource Constraint

$$Subnet^* = \arg \max_{Subnet_i \in \mathcal{A}} \boxed{E_{val}}(F(W_o, Subnet_i)) \quad s.t. \quad \boxed{R(subnet_i)} \leq \boxed{\tau}$$

Subnet Representation Challenge

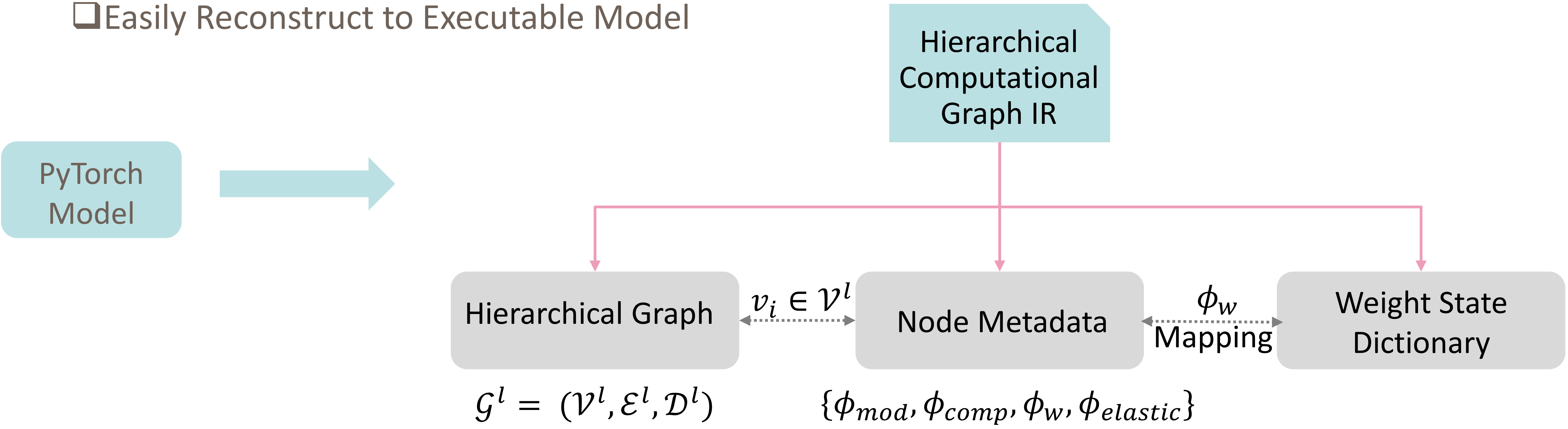
In weight sharing supernet, subnet representation requires **Architecture-Specific Handcraft** Rules/Heuristics



Hierarchical Computational Graph Intermediate Representation (IR)

We Propose Modeling DNNs as ***Hierarchical Graph Intermediate Representation (IR)***

- ❑ High-level Abstraction of DNNs
- ❑ Architecture-agnostic
- ❑ **Unify** and **Simplify** the Model Architecture Modification
- ❑ Easily Reconstruct to Executable Model

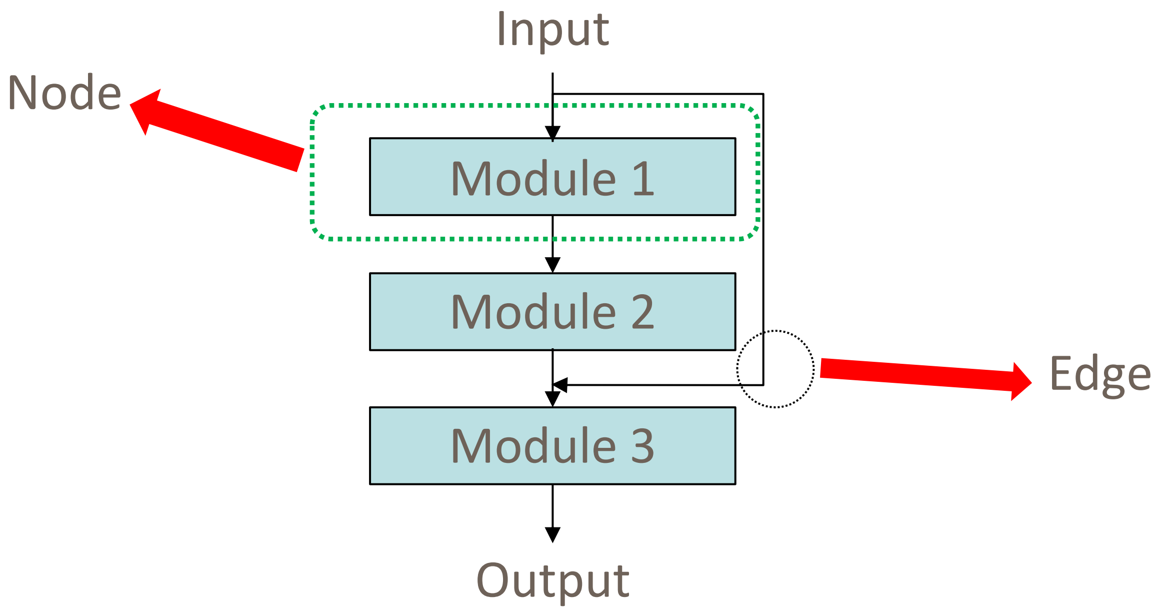


Topology Modeling

We Propose Modeling DNNs as ***Hierarchical Graph Intermediate Representation (IR)***

Node: Computational Module (nn.Module)

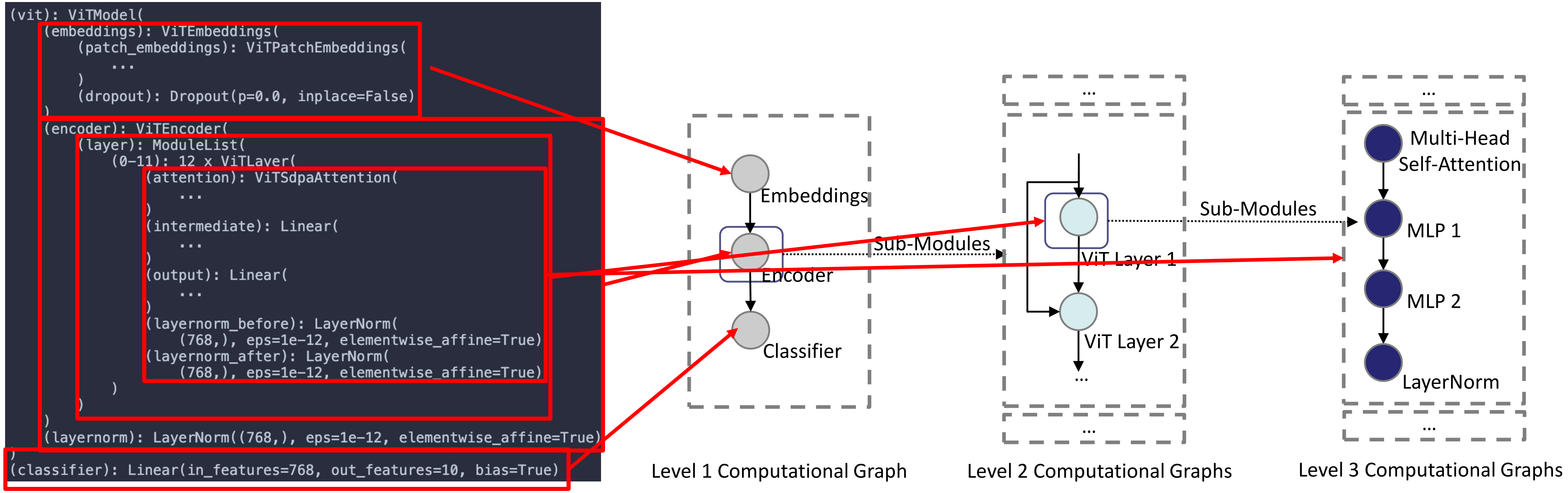
Edge: Model Forward-propagation Direction



Topology Modeling

Pytorch model are defined as nested computational graph modules (i.e.,nn.Module), which can be modeling as hierarchical graph

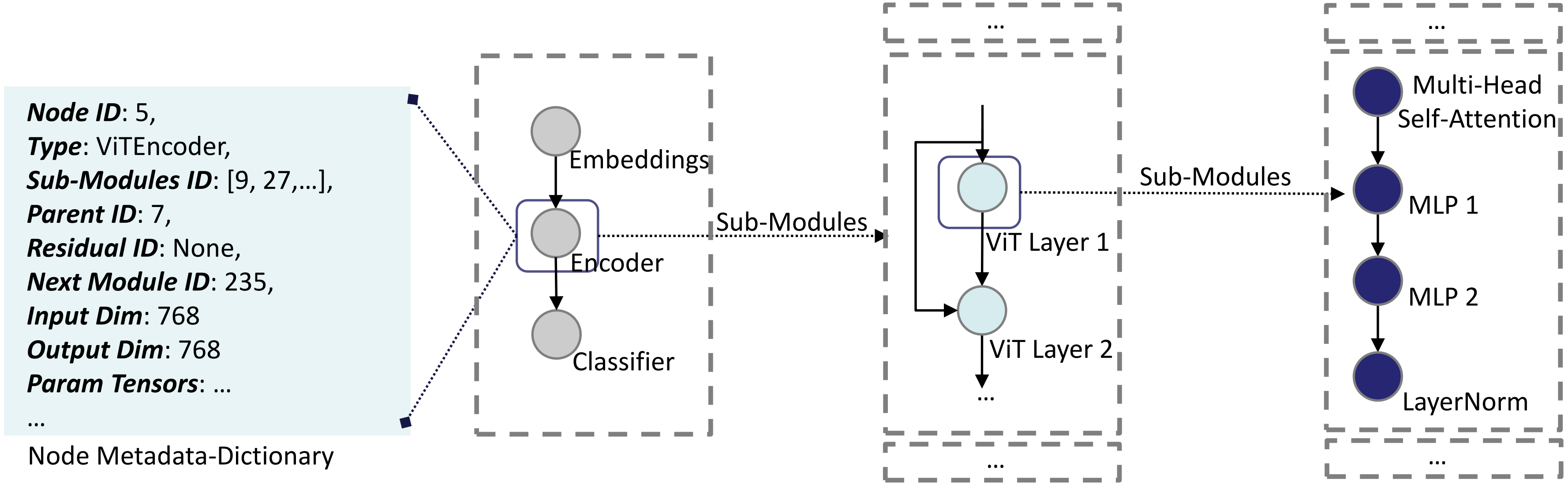
Take ViT as an example



Node Metadata for Module Reconstruction

Each node contains a metadata dictionary.

Modules can be ***Reconstructed*** via the Metadata Dictionary

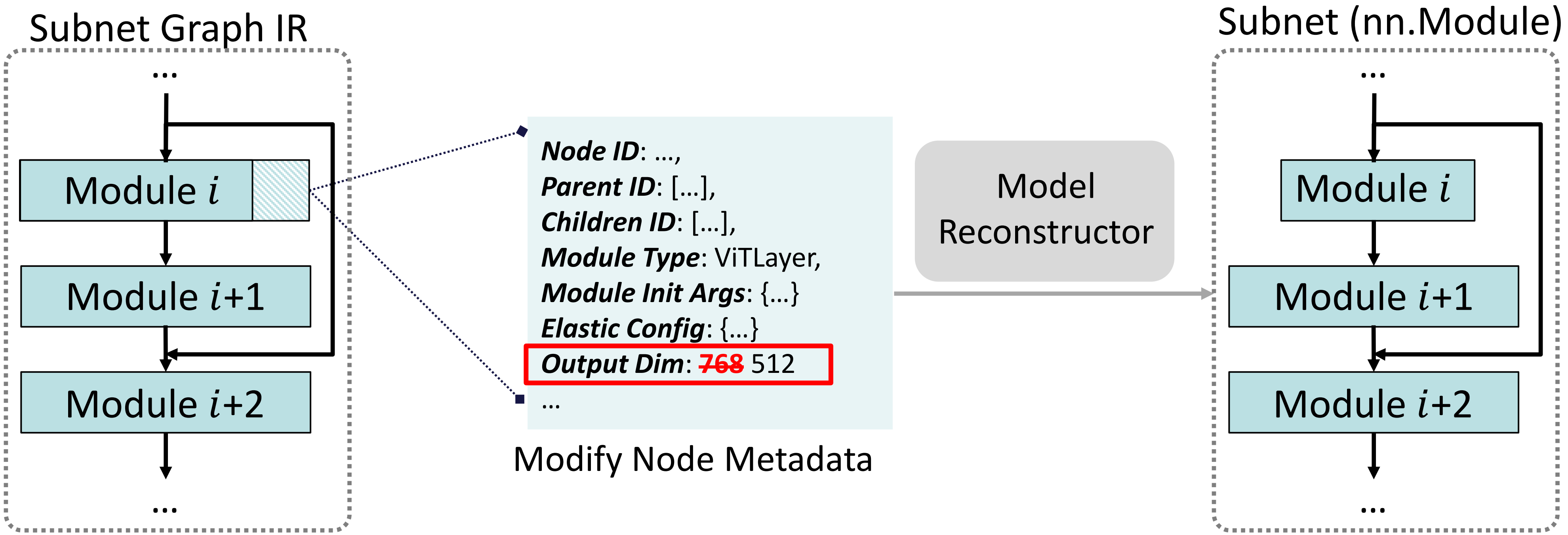


Level 1 Computational Graph   Level 2 Computational Graphs   Level 3 Computational Graphs

Subnet Extraction with IR

Two Types of Subnet Extraction in Weight-sharing Supernet:

(1) ***Structural Weights Pruning*** and (2) Module-wise pruning.

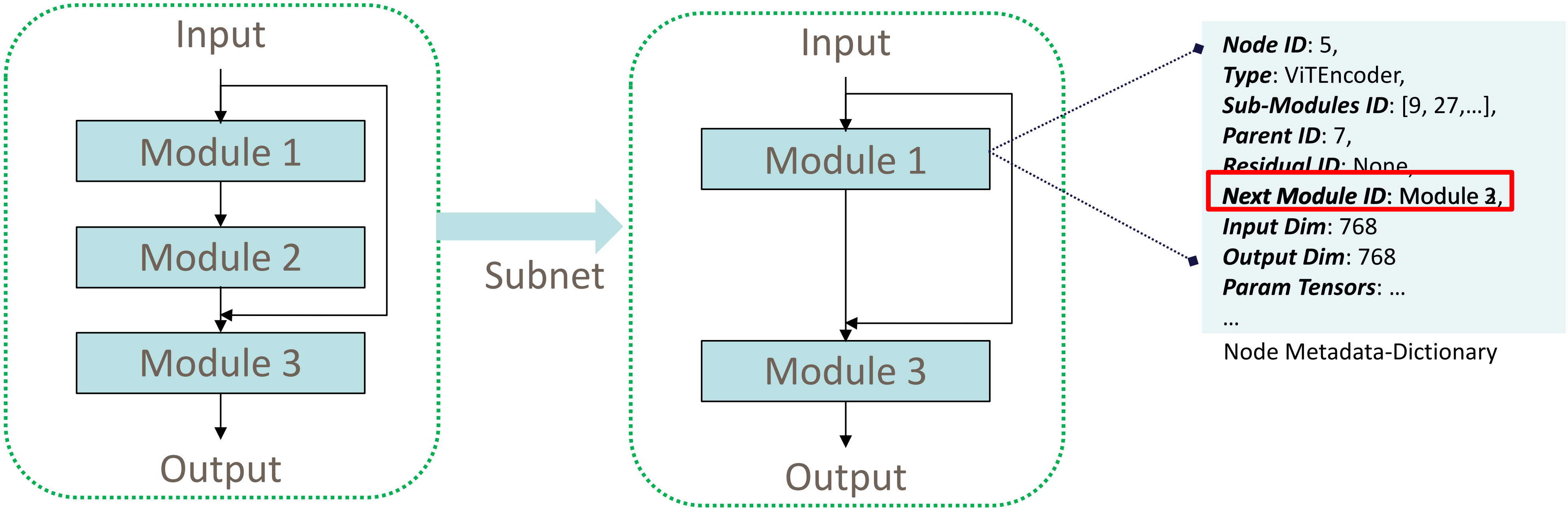


Subnet Extraction with IR

Two Types of Subnet Extraction in Weight-sharing Supernet:

(1) Structural Weights Pruning and (2) ***Module-wise pruning.***

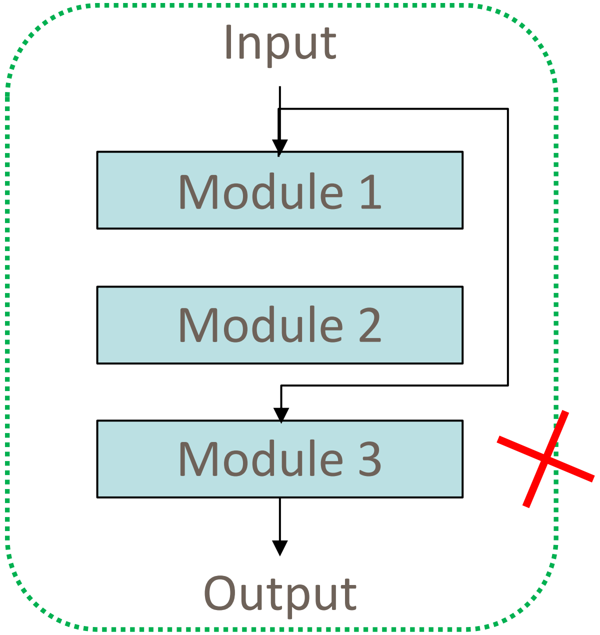
Graph IR achieves Module-wise Pruning Simply via ***Edge Contraction***



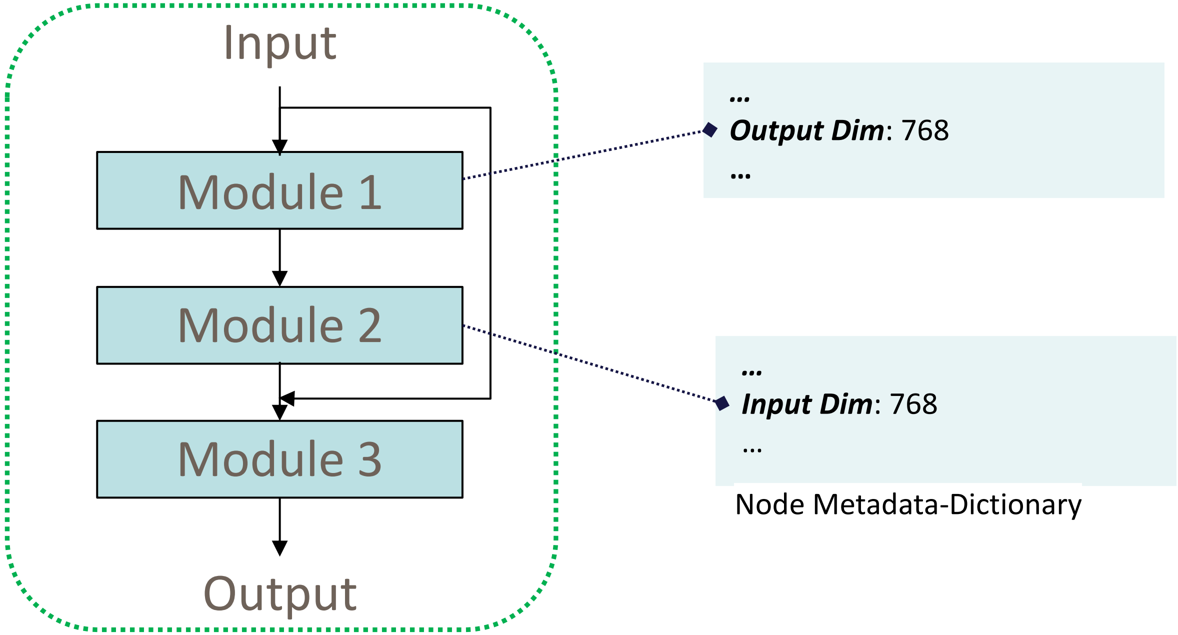
Subnet Extraction with IR

To Guarantee the Subnets are Executable, Two Conditions Must Satisfied:

(1) ***Node Connectivity*** (2)***Dimensions compatibility***



Isolated Nodes Exists

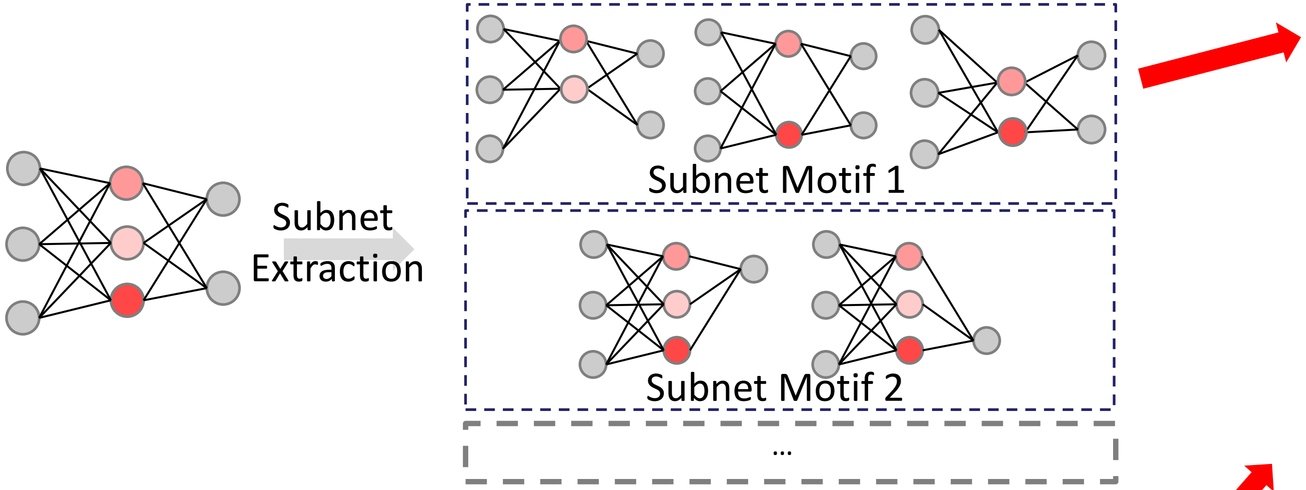


Output Dim Match Next Node’s Input Dim

Efficient Subnet Sample Strategy With IR

We noticed that there exists Many of Subnets in the Sample Space are Redundant
We call it Subnet Motifs.

Subnet Motifs Definition: *Subnets have the same topology*



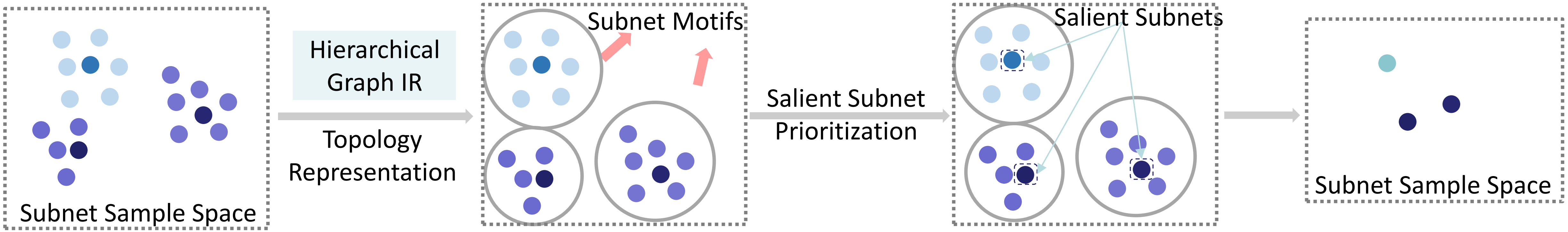
We only need the best performance subnets in a motif

$$Subnet^* = \arg \max_{Subnet_i \in \mathcal{A}} E_{val}(F(W_o, Subnet_i)) \quad s.t. \; \boxed{R(subnet_i) \leq \tau}$$

Our Objective: Pre-identify Subnet in Each Motif, and only Focus on one Salient Subnet in a Motif

Efficient Subnet Sample Strategy

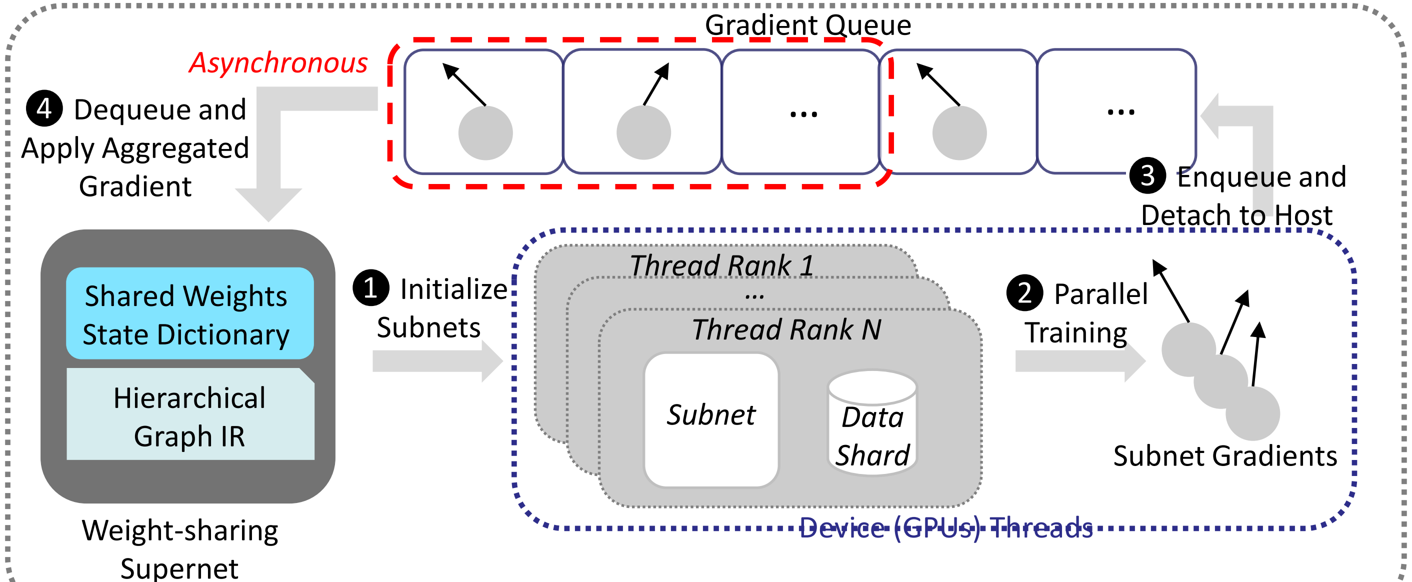
We noticed that there exists Many of Subnets in the Sample Space are Redundant
We call it Subnet Motifs.
Subnet Motifs Definition: Subnets have the same topology



Our Idea: We prioritize the most salient subnets (based on weight tensor magnitude) in the given subnet Motif

Fork-join parallel training

Weight-sharing Nature Raises Write-After-Write dependency when training multiple subnets in parallel



(c) Parallel Weight-Sharing Subnet Training

if we train multiple subnets concurrently, the weights shared by the subnets can be overwritten by the latest trained subnet

Results – Architecture Agnostic Compression –ViT

With Proposed IR, our method adapt to a wide range of neural architecture type without specialization rules. – Result on ViT

Table 1. Image classification results on ImageNet benchmarks.

Method	#Param	#Param ↓	Val Acc.	Δ Acc.	FLOPs	FLOPs ↓
ViT-B [7]	86.6	-	80.98	-	17.6G	0%
DeiT-B [25]	86.6	0%	81.84	+0.86	17.6G	0%
AutoFormer-B [3]	54M	37.6%	82.40	+1.42	11.0G	37.5%
T2T-ViT-24 [34]	64M	26%	82.30	+1.32	13.8G	21.6%
ViT-Slim [2]	52.6M	39.2	82.40	+1.42	10.6G	39.8%
SAViT [4]	42	40%	82.75	+1.77	10.6G	39.8%
VTP-B [35]	47M	45.4%	80.70	-0.28	10G	43.2%
PS-ViT-B [24]	86.6	0%	81.5	+0.52	9.8G	44.3%
PreNAS [27]	54M	37.6%	82.6	+1.62	11G	37.5%
UVC [32]	N/A	N/A	80.57	-0.41	8G	54%
OSF (Ours)	57M	33.7%	82.27	+1.29	10.02G	**42%**
	53M	38.8%	81.04	+0.06	8.7G	**50%**

We compared with ViT specialized compression method, pruning method, and AutoML methods.
OSF shows competitive compression performance on ViT with State-of-the-art!

Results – Architecture Agnostic Compression – CNN

With Proposed IR, our method adapt to a wide range of neural architecture type without specialization rules. – Result on CNN

Table 2. CNN Model Image classification results on ImageNet benchmarks.

Method	#Param	#Param ↓	Val Acc.	Δ Acc.	FLOPs	FLOPs ↓
ResNet-50 [10]	97.8MB	-	76.13	-	3.8G	0%
ResNet-18 [10]	44.7 MB	54.3%	69.75	-6.38	1.81G	52.4%
AutoPruner [13]	68.5MB	30%	73.05	-3.08	2.64G	35%
Meta-Pruning [18]	48.9MB	50%	73.4	-2.73	1.9G	50%
SFP [11]	68.5MB	30%	77.37	+1.24	2.2G	42%
AutoSlim [29]	51.8MB	47%	74.00	-2.13	1.9G	50%
GNN-RL [33]	48.7MB	50%	74.28	-1.85	1.78G	53%
EagleEye [15]	48.9MB	50%	74.20	-1.93	1.9G	50%
FPGM [12]	68.5MB	30%	74.83	-1.30	1.8G	53%
NISP [31]	55MB	44%	75.24	-0.89	2.1G	44 %
ThiNet-50 [19]	N/A	N/A	71.01	-5.12	3.41G	11 %
PFP-B [16]	N/A	N/A	65.65	-10.48	1.03G	73%
DepGraph [8]	N/A	N/A	75.83	-0.30	1.86G	51%
DMCP [23]	N/A	N/A	76.23	+0.10	2.8G	26%
ATO [28]	N/A	N/A	76.07	-0.06	1.48G	61%
OSF (Ours)	40MB	59.1%	76.13	0	1.33G	65%

We compared with CNN specialized compression method, pruning method, and AutoML methods.
OSF shows competitive compression performance on ViT with State-of-the-art!

Results – Architecture Agnostic Compression – SAM

With Proposed IR, our method adapt to a wide range of neural architecture type without specialization rules. – Result on Segment Anything

Table 3. Image segmentation task with Segment Anything

Method	Dataset	#Param	#Param ↓	mIoU	Δ mIoU.
SAM [14]	COCO	90M	-	69.20	-
OSF		47M	47.8%	75.30	+6.10
OSF		49M	45.6%	75.44	+6.24
SAM [14]	SA1B	90M	-	73.00	-
OSF		44M	51.1%	74.67	+1.67
OSF		53M	41.1%	77.24	+4.24

Results – Architecture Agnostic Compression – NLP

With Proposed IR, our method adapt to a wide range of neural architecture type without specialization rules. – Result on Question Answering

Table 4. Question-answering benchmark results on BERT Architecture (Encoder-only Transformer)

Param Group	Method	#Param	#Param ↓	F1	Δ F1	FLOPs ↓	Latency	Latency ↓
>150M	BERT-L [6]	334M	-	89.49	-	-	16.28ms	-
	OSF (Ours)	166M	168M	89.73	+0.24	49%	7.88ms	8.40ms
100 - 150M	BERT-B [6]	109M	225M	84.45	-5.04	72%	6.10ms	10.18ms
	RoBERTa [17]	124M	210M	90.28	+0.79	65%	9.25ms	7.03ms
	OSF(Ours)	104M	230M	89.51	+0.02	72%	8.00ms	8.28ms
	OSF (Ours)	93M	241M	89.50	+0.01	76%	7.94ms	8.34ms
<100M	DistillBERT [22]	77M	257M	82.25	-7.24	89%	2.84ms	13.44ms
	OSF (Ours)	75M	259M	88.74	-0.75	83%	4.48ms	11.80ms
	OSF(Ours)	45M	289M	80.72	-8.77	93%	2.79ms	13.49ms

Thank you

If you have question, feel free to reach out to:

yusx@iastate.edu