RADARS: Memory Efficient Reinforcement Learning Aided Differentiable Neural Architecture Search

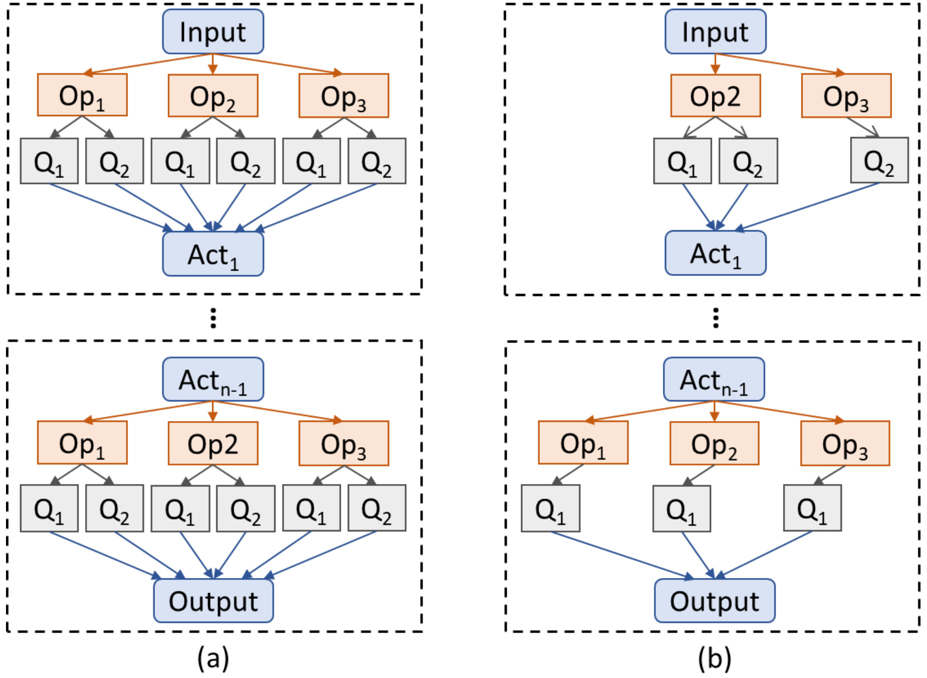
Presenter: Zheyu Yan,

Co-Authors: Weiwen Jiang, X. Sharon Hu, Yiyu Shi

UNIVERSITY OF NOTRE DAME

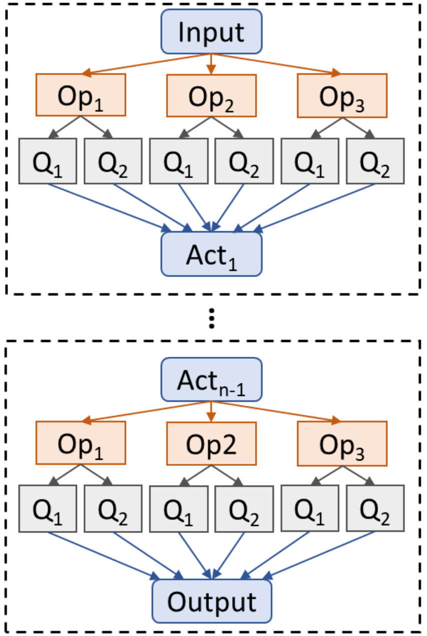
Outline

- Motivation
- Intuition
- RADARS
- Experimental Results

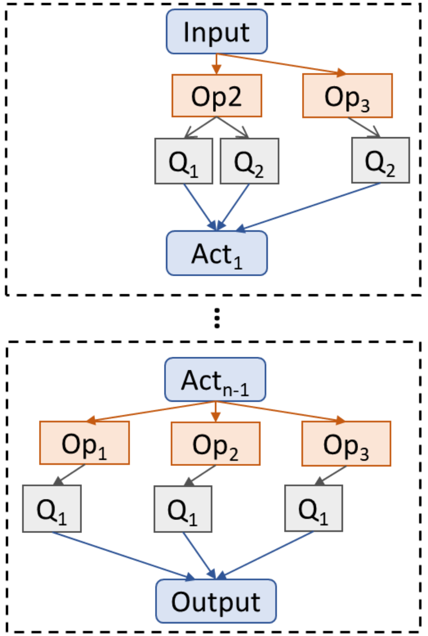


Outline

- **Motivation**
- Intuition
- RADARS
- Experimental Results



(a)



(b)

Motivations: Limitation of NAS Variants (1)

- **Neural architecture search (NAS)** is effective
- **Reinforcement learning (RL) based NAS** is time consuming
- **Differentiable NAS (DNAS)** is fast but memory hungry

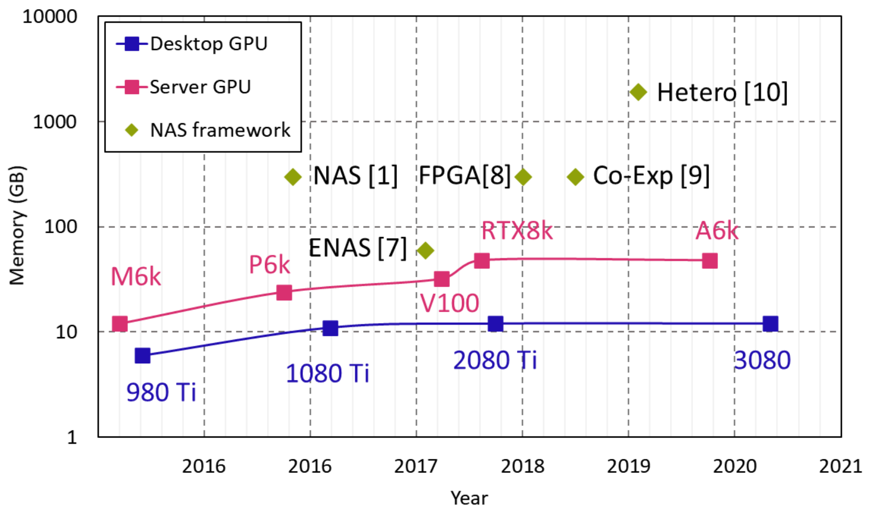


Fig. 1 Memory consumption for different NAS frameworks when mapped to DNAS scheme vs available GPU memory over time.

Motivations: Limitation of NAS Variants (2)

- **One-Shot DNAS** e.g., ProxylessNAS [1]
  - Activates **only one path** at each iteration of training-search
  - Reduces memory usage
  - Still needs to store the whole SuperNet in GPU memory or requires frequent swapping
- **RL-aided DNAS:** higher flexibility
- Why is DNAS memory hungry?
  The **search space** is large

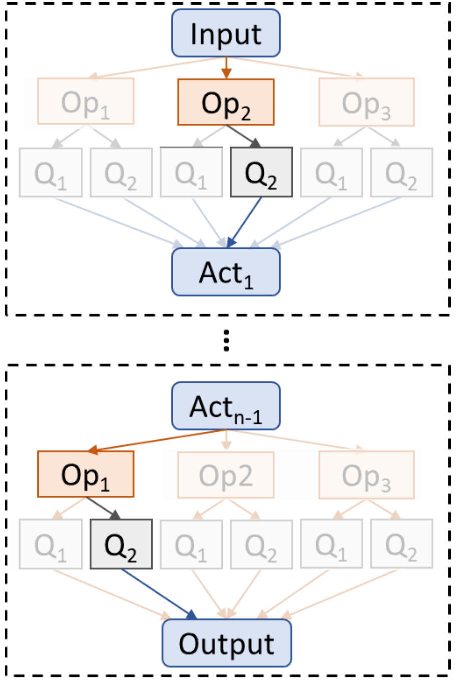
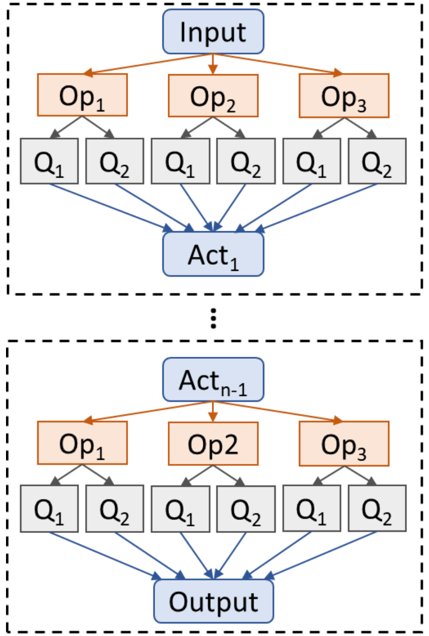


Fig. 2 Demonstration for One-Shot DNAS

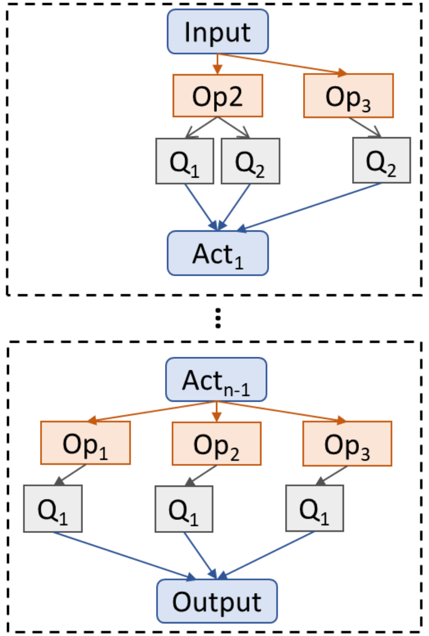
[1] Cai, Han et al. "Proxylessnas: Direct neural architecture search on target task and hardware." ICLR (2019).

Outline

- Motivation
- **Intuition**
- RADARS
- Experimental Results



(a)



(b)

Intuitions: Redundant Search Space (1)

- Why is DNAS memory hungry?
  The **search space** is large
- Designers tend to offer **the same search space** (e.g., kernel size, # of channels, quantization precision) for different layers
- Different layers have different properties and require different hyperparameter sets

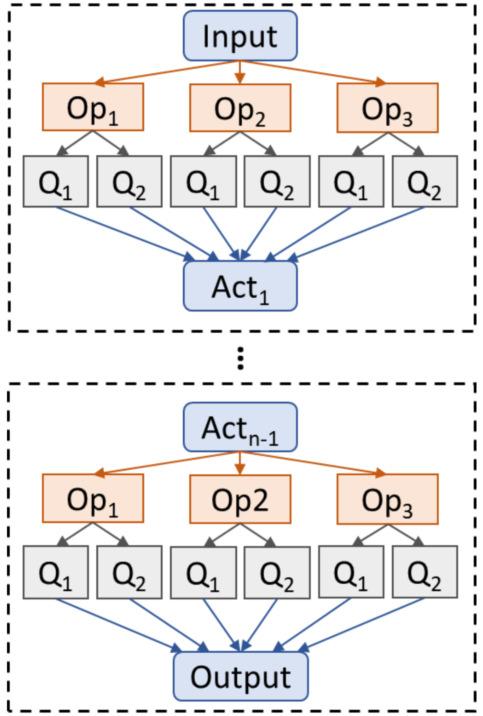


Fig. 3 Uniform search space for different layers

Intuitions: Redundant Search Space (2)

- Different layers have different properties and require different hyperparameter sets
- Some parameters are only more sensitive to **position**
- The **early episodes of RL-NAS** is a good approach to identify these differences

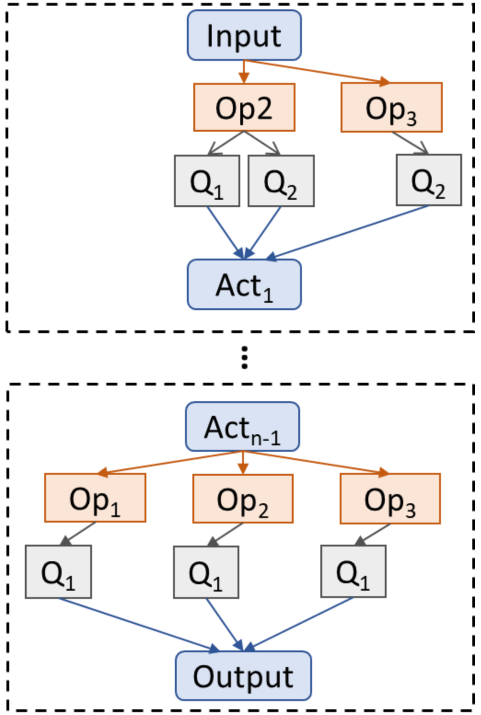
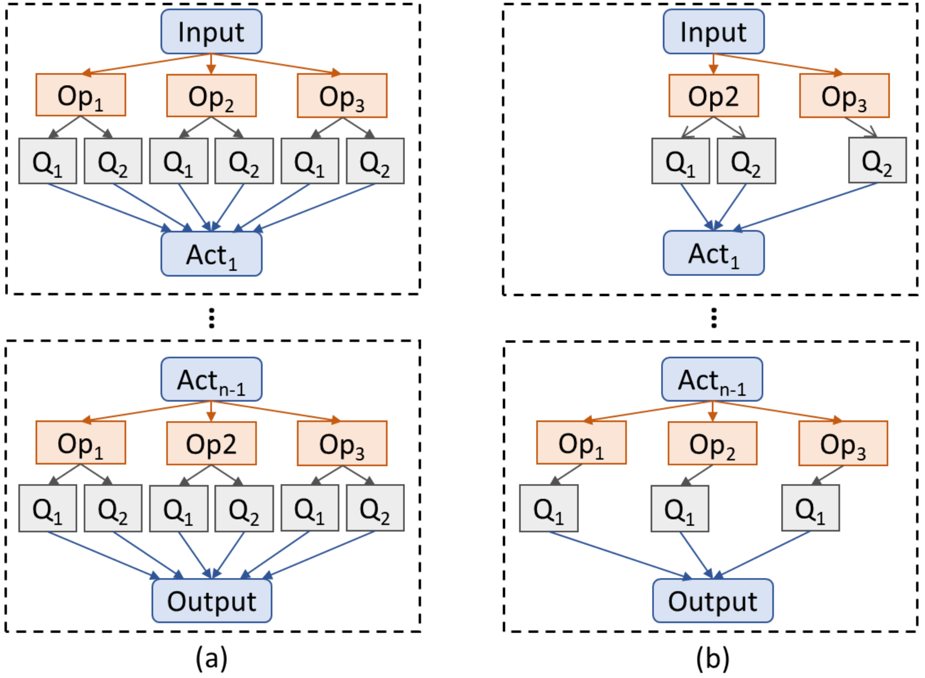


Fig. 4 Unique search space for different layers

Outline

- Motivation
- Intuition
- **RADARS**
- Experimental Results



RADARS: Pruning Search Space via RL

Goal: find a **subset of the original search space** that:

- Contains a solution that **satisfy** the performance requirement
- Fits in the **memory usage** constraint

Use this subset to find the optimal solution

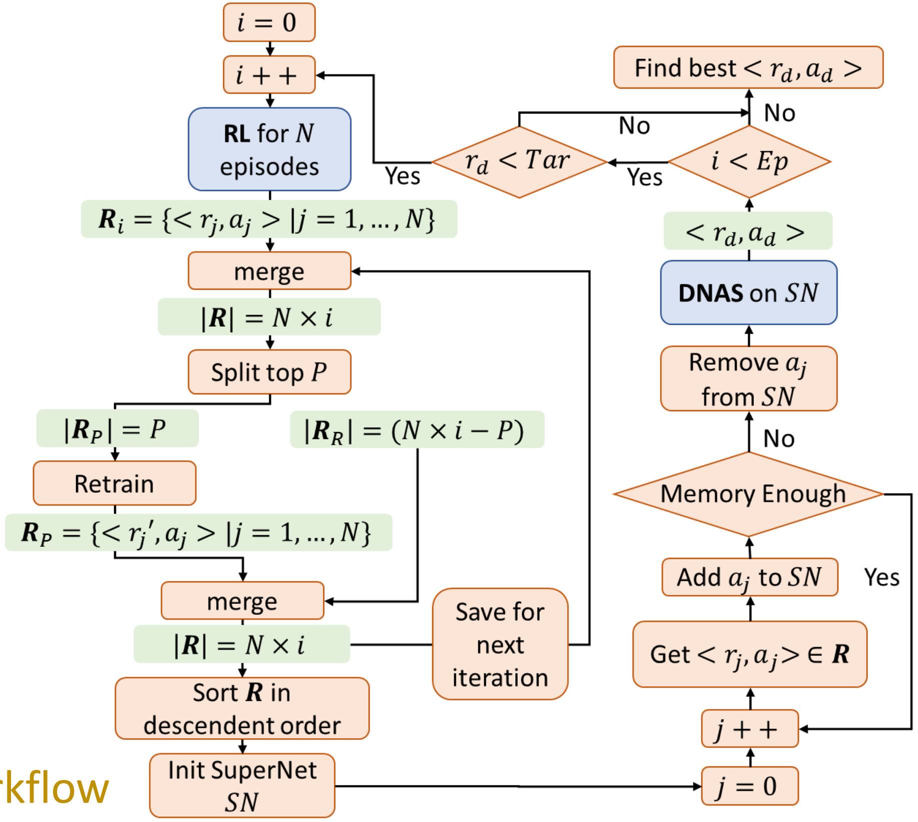
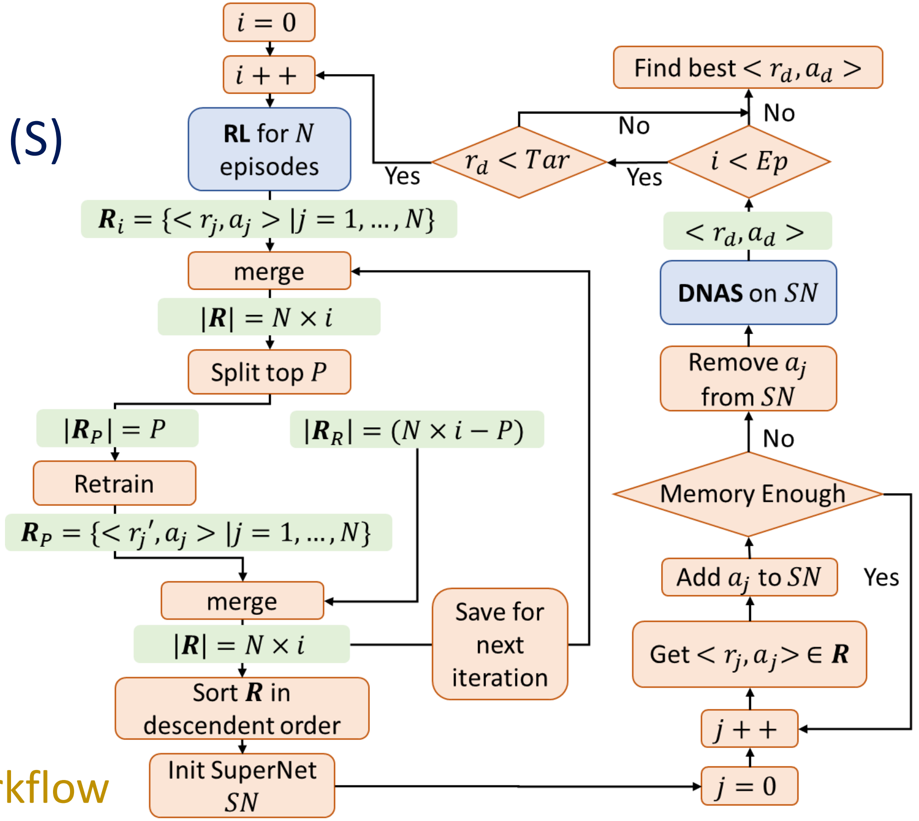


Fig. 5 RADARS workflow

RADARS: Pruning Search Space via RL

1. **Initialize** a **uniform** search space (S)
2. **RL** finds a set of architectures
3. Choose the **top P** of them
4. **Merge** top P candidates to the **subset** of S
5. **DNAS [2]** on this **subset**
6. **Iterate** the previous 5 steps if accuracy not high enough



$i = 0$
$i++$
RL for N episodes
$\boldsymbol{R}_i = \{< r_j, a_j > | j = 1, \dots, N\}$
merge
$|\boldsymbol{R}| = N \times i$
Split top P
$|\boldsymbol{R}_P| = P$
$|\boldsymbol{R}_R| = (N \times i - P)$
Retrain
$\boldsymbol{R}_P = \{< r_j', a_j > | j = 1, \dots, N\}$
merge
$|\boldsymbol{R}| = N \times i$
Sort $\boldsymbol{R}$ in descendent order
Init SuperNet SN
Save for next iteration
$j = 0$
$j++$
Get $< r_j, a_j > \in \boldsymbol{R}$
Add a_j to SN
Memory Enough
Yes
No
Remove a_j from SN
DNAS on SN
$< r_d, a_d >$
$i < Ep$
Yes
No
$r_d < Tar$
Yes
No
Find best $< r_d, a_d >$

Fig. 5 RADARS workflow

[2] Wu, Bichen, et al. "Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search." CVPR (2019).

RADARS: RL Exploration (1)

Adopting existing RL NAS frameworks

- **Controller:** reinforcement learning-based RNN controller. In each episode:
  - Generates one neural architecture
  - Receives a *reward* to generate better architectures
- **Trainer:** trains a neural architecture generated by controller
- **Evaluator:** evaluates the accuracy and hardware efficiency of a neural architecture and provide a *reward*

RADARS: DNAS Exploitation (1)

1. **Initialize** a **uniform** search space (S)
2. **RL** finds a set of architectures
3. Choose the **top P** of them
4. **Merge** top P candidates to create a **subset** of S
5. **DNAS** on this **subset**
6. **Iterate** the previous 5 steps if accuracy not high enough

For the K sorted architectures

- **Add** one architecture to DNAS search space
- **Build** a DNAS model using the new search space
- **Evaluate** the memory consumption of this model
- If **exceeds the memory** bound, break and remove the newly added architecture

RADARS: DNAS Exploitation (2)

For **each layer:**

- **Put candidates** in each architecture into the search space
- **Remove** the duplicated candidates

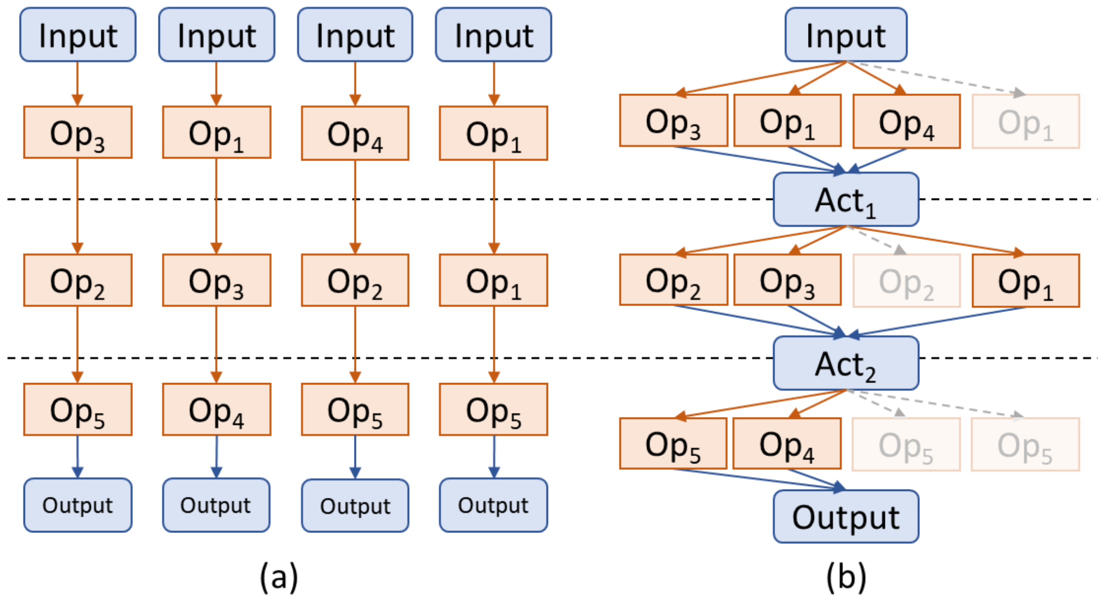
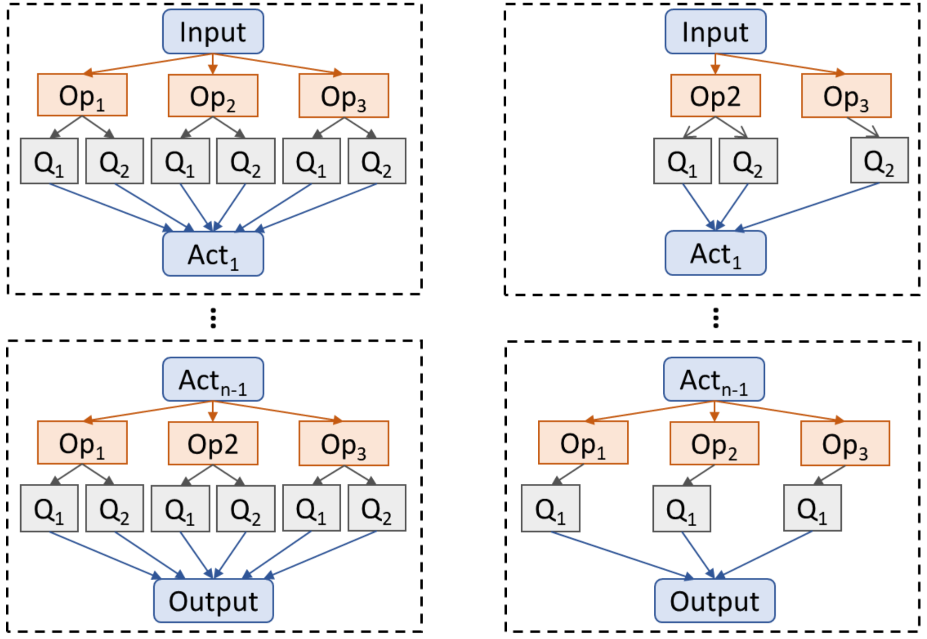


Fig. 6 Demonstration of merging a set of architectures to a DNAS search space

Outline

- Motivation
- Intuition
- RADARS
- **Experimental Results**



(a) (b)

Experiments: Evaluation Metrics

Normalized arithmetic intensity

- Similar to FLOPS, not a good metric for latency
- Targeting accurate **energy consumption** analysis

$$\text{AOPS} = \text{MACOPS} \times \text{Activation bits} \times \text{Weight bits}$$

$$\text{Reward} = \alpha \times \text{Acc} + (1 - \alpha) \times (1 - (\text{AOPS} - \beta)/\gamma)$$

Experiments: CIFAR-10 (1)

- Search quantized CNN for CIFAR-10
- Baselines: quantNAS (RL) [3], One-Shot [1], DNAS [2]
- Search space shown below

Hyper-Parameter type	Settings
Block type	Quantized Convolution
# of convolution layers	6
# of channels per layer	[64, 64, 128, 128, 256, 256]
Stride	[1, 2, 1, 2, 1, 2]
Kernel size choices	(1, 3, 5, 7)
# of integer bits choices	(1, 3)
# of fraction bits choices	(1, 3, 6)

[3] Qing Lu et al, "On neural architecture search for resource-constrained hardware platforms," ICCAD, 2019.

Experiments: CIFAR-10 (2)

- RADARS gets 3.41% lower test error and 39% lower arithmetic intensity, with 2.5× search time reduction than RL
- DNAS takes 10x more memory
- One-Shot takes 26x more time

Model	Top-1 (%)	AOPS (G)	Time (h)	Memory (GB)
QuantNAS [6]	84.92	4.09	22.9	1.430
One-Shot [5]	OOT	OOT	>240	1.537
DNAS [4]	OOM	OOM	OOM	112.0
RADARS (ours)	88.33	2.50	9.15	10.43

Experiments: CIFAR-10 (3)

- Each point represents an architecture
- A solution is better as it moves towards the bottom-left corner
- RADARS can significantly push forward the Pareto frontier

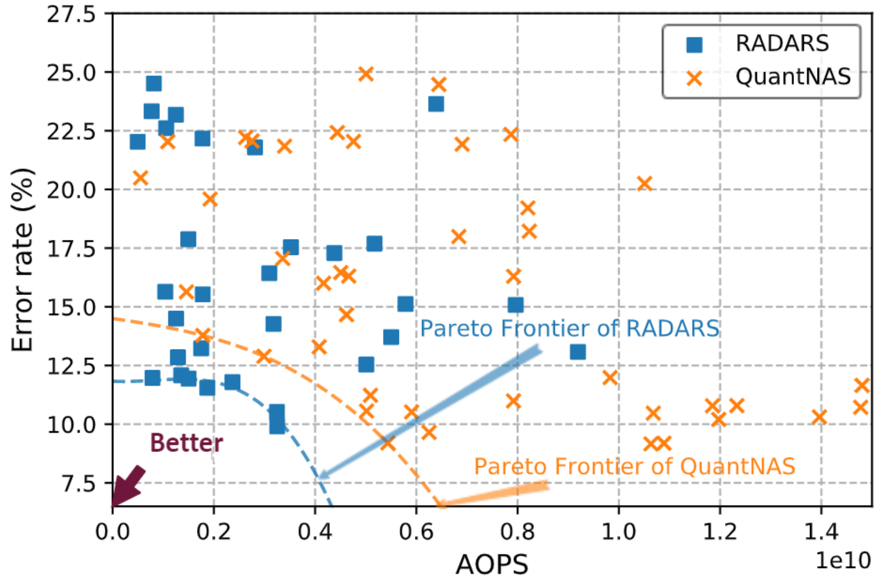
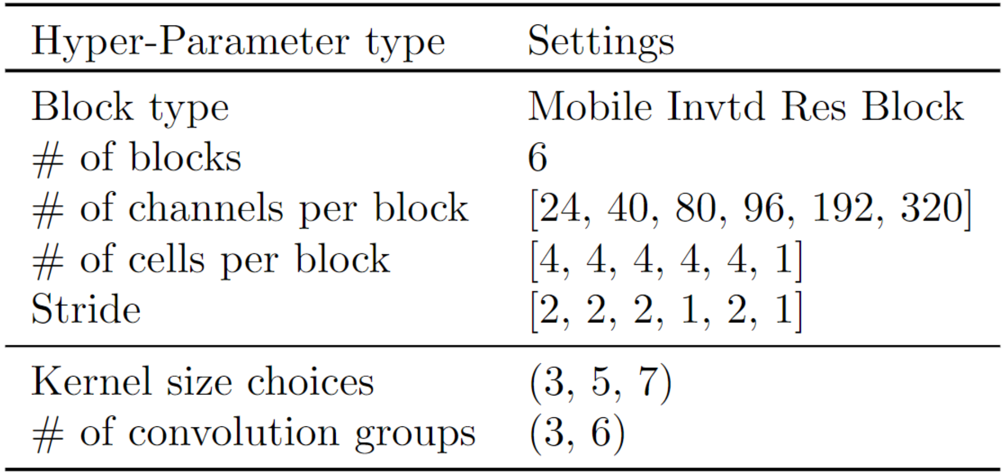


Fig. 7 Performance of RADARS and QuantNAS. Each point represents an architecture

Experiments: ImageNet (1)

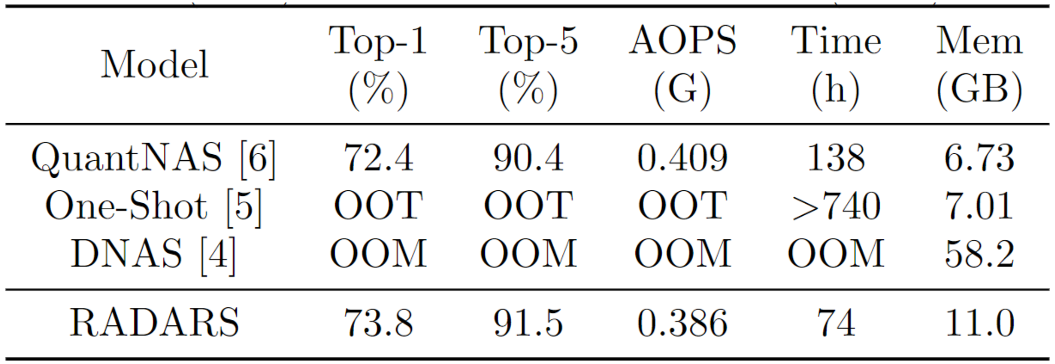
- Search MobileNet for ImageNet
- Baselines: quantNAS (RL) [3], One-Shot [1], DNAS [2]
- Search space shown below



Hyper-Parameter type	Settings
Block type	Mobile Invtd Res Block
# of blocks	6
# of channels per block	[24, 40, 80, 96, 192, 320]
# of cells per block	[4, 4, 4, 4, 4, 1]
Stride	[2, 2, 2, 1, 2, 1]
Kernel size choices	(3, 5, 7)
# of convolution groups	(3, 6)

Experiments: ImageNet (2)

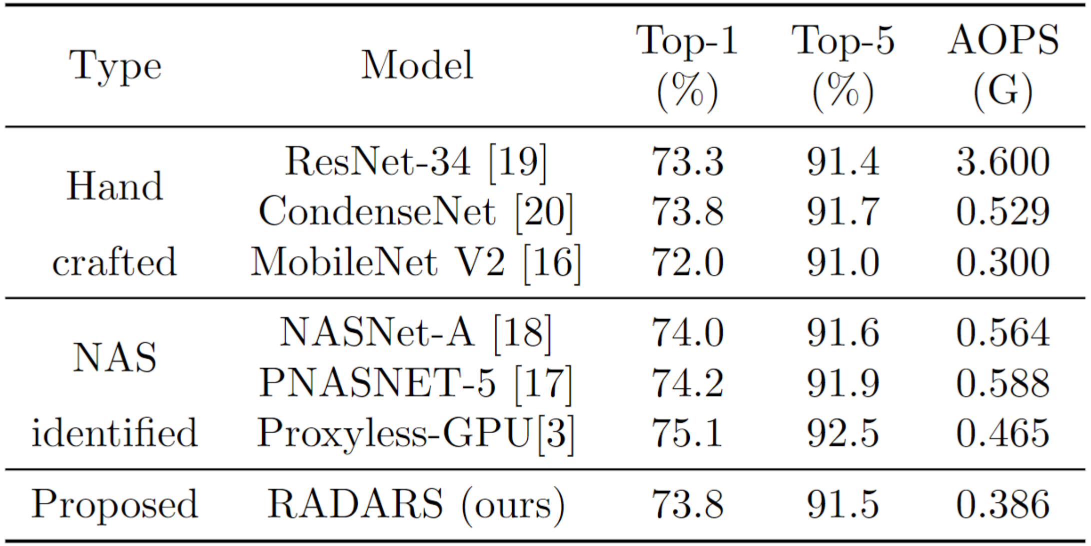
- RADARS achieves 1.4% higher top-1 accuracy than QuantNAS using only 54% of the search time than RL
- DNAS takes 5x more memory
- One-Shot takes 10x more time



Model	Top-1 (%)	Top-5 (%)	AOPS (G)	Time (h)	Mem (GB)
QuantNAS [6]	72.4	90.4	0.409	138	6.73
One-Shot [5]	OOT	OOT	OOT	>740	7.01
DNAS [4]	OOM	OOM	OOM	OOM	58.2
RADARS	73.8	91.5	0.386	74	11.0

Experiments: SOTA on ImageNet

- Handcrafted models and NAS models with different search spaces
- RADARS achieves comparable performances



Type	Model	Top-1 (%)	Top-5 (%)	AOPS (G)
Hand crafted	ResNet-34 [19]	73.3	91.4	3.600
	CondenseNet [20]	73.8	91.7	0.529
	MobileNet V2 [16]	72.0	91.0	0.300
NAS identified	NASNet-A [18]	74.0	91.6	0.564
	PNASNET-5 [17]	74.2	91.9	0.588
	Proxyless-GPU[3]	75.1	92.5	0.465
Proposed	RADARS (ours)	73.8	91.5	0.386

Conclusions

- We propose RADARS, an RL-aided DNAS framework
- RADARS explores large hardware-aware neural network search spaces in a memory efficient manner
- Experiments on CIFAR-10 and ImageNet demonstrate the superiority of RADARS over the state-of-the-art.

References

[1] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.

[2] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *ICLR*, 2018.

[3] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *ICLR*, 2018.

[4] Y. Li, C. Hao, X. Zhang, X. Liu, Y. Chen, J. Xiong, W.-m. Hwu, and D. Chen, "Edd: Efficient differentiable dnn architecture and implementation co-search for embedded ai solutions," *DAC*, 2020.

[5] X. Dong and Y. Yang, "One-shot neural architecture search via self-evaluated template network," in *ICCV*, 2019, pp. 3681–3690.

[6] Q. Lu, W. Jiang, X. Xu, Y. Shi, and J. Hu, "On neural architecture search for resource-constrained hardware platforms," *ICCAD*, 2019.

[7] W. Jiang, X. Zhang, E. H.-M. Sha, L. Yang, Q. Zhuge, Y. Shi, and J. Hu, "Accuracy vs. efficiency: Achieving both through fpga-implementation aware neural architecture search," in *DAC*, 2019, pp. 1–6.

[8] W. t. Jiang, "Hardware/software co-exploration of neural architectures," *TCAD*, vol. 39, no. 12, pp. 4805–4815, 2020.

[9] W. Jiang, Q. Lou, Z. Yan, L. Yang, J. Hu, X. S. Hu, and Y. Shi, "Device-circuit-architecture co-exploration for computing-in-memory neural accelerators," *IEEE Transactions on Computers*, vol. 70, no. 4, pp. 595–605, 2020.

[10] L. t. Yang, "Co-exploration of neural architectures and heterogeneous asic accelerator designs targeting multiple tasks," in *DAC*. IEEE, 2020, pp. 1–6.

[11] Z. Yan, D.-C. Juan, X. S. Hu, and Y. Shi, "Uncertainty modeling of emerging device based computing-in-memory neural accelerators with application to neural architecture search," in *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2021, pp. 859–864.

[12] A. Vahdat, A. Mallya, M.-Y. Liu, and J. Kautz, "Unas: Differentiable architecture search meets reinforcement learning," in *CVPR*, 2020, pp. 11 266–11 275.

[13] Y. Zhao, L. Wang, Y. Tian, R. Fonseca, and T. Guo, "Few-shot neural architecture search," in *ICML*. PMLR, 2021, pp. 12 707–12 718.

[14] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*. Ieee, 2009, pp. 248–255.

[16] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *CVPR*, 2018, pp. 4510–4520.

[17] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *ECCV*, 2018, pp. 19–34.

[18] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[20] G. Huang, S. Liu, L. Van der Maaten, and K. Q. Weinberger, "Condensenet: An efficient densenet using learned group convolutions," in *CVPR*, 2018, pp. 2752–2761.

Thanks & Questions