On Redundancy and Diversity in Cell-based Neural Architecture Search

Xingchen Wan[1], Binxin Ru[1], Pedro M. Esperança[2], Zhenguo Li[2]
[1]University of Oxford [2]Huawei Noah's Ark Lab

Presented at the 10th International Conference on Learning Representations, 25 - 29 Apr 2022

Cell-based NAS

- Searching for *architecture cells* is a mainstream genre in NAS.
    - DARTS [Liu et al, 2019] search space (*right*) and its variants are dominant.
- Most search methods do not try to understand the cells found, nor to attribute performance to designs

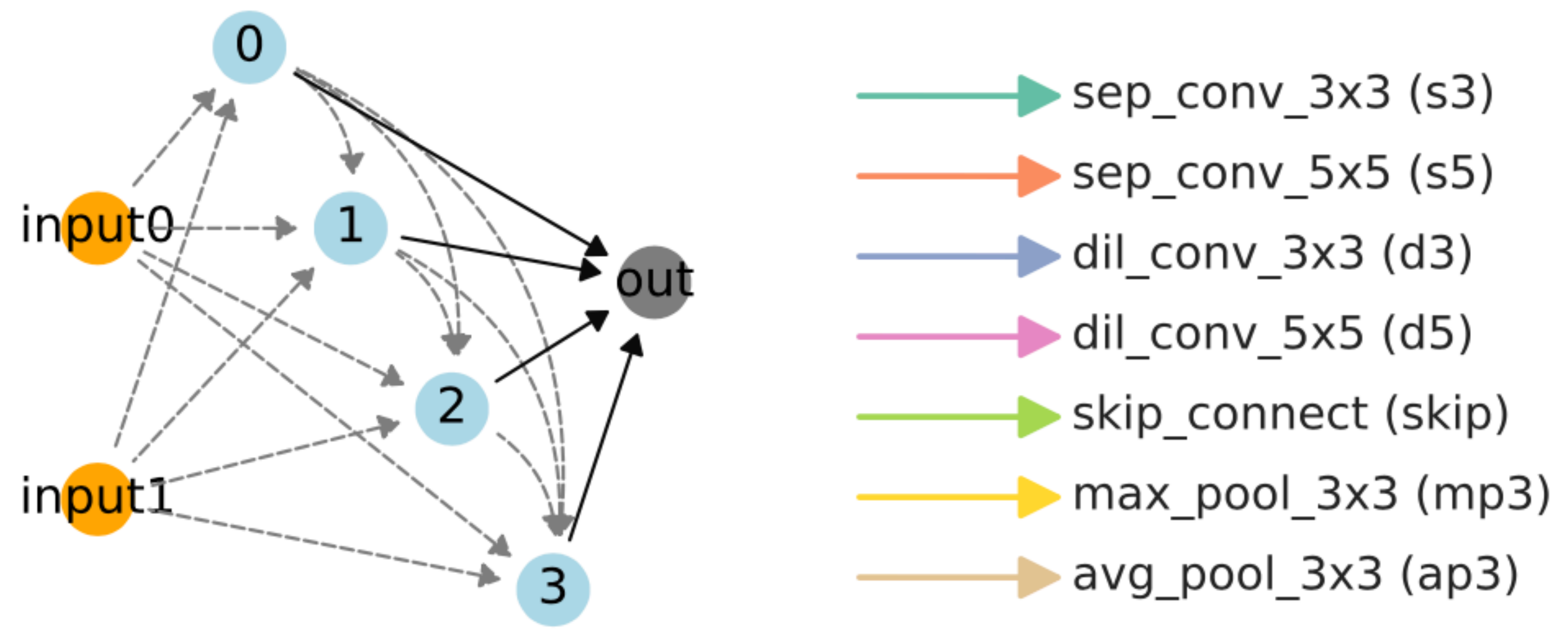


Illustration of the DARTS search space

[Liu et al, 2019] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *International Conference on Learning Representations (ICLR)*, 2019.

Dominance of Cell-based Search

In recent top conferences* ...

79%

Involving Cell-based Search

58%

Cell-based NAS **Only**

*: NeurIPS 2020, ICLR 2021, ICML 2021. Preliminary search could be incomplete; only papers mentioning NAS in abstract and/or titles are included.

DARTS & related Spaces

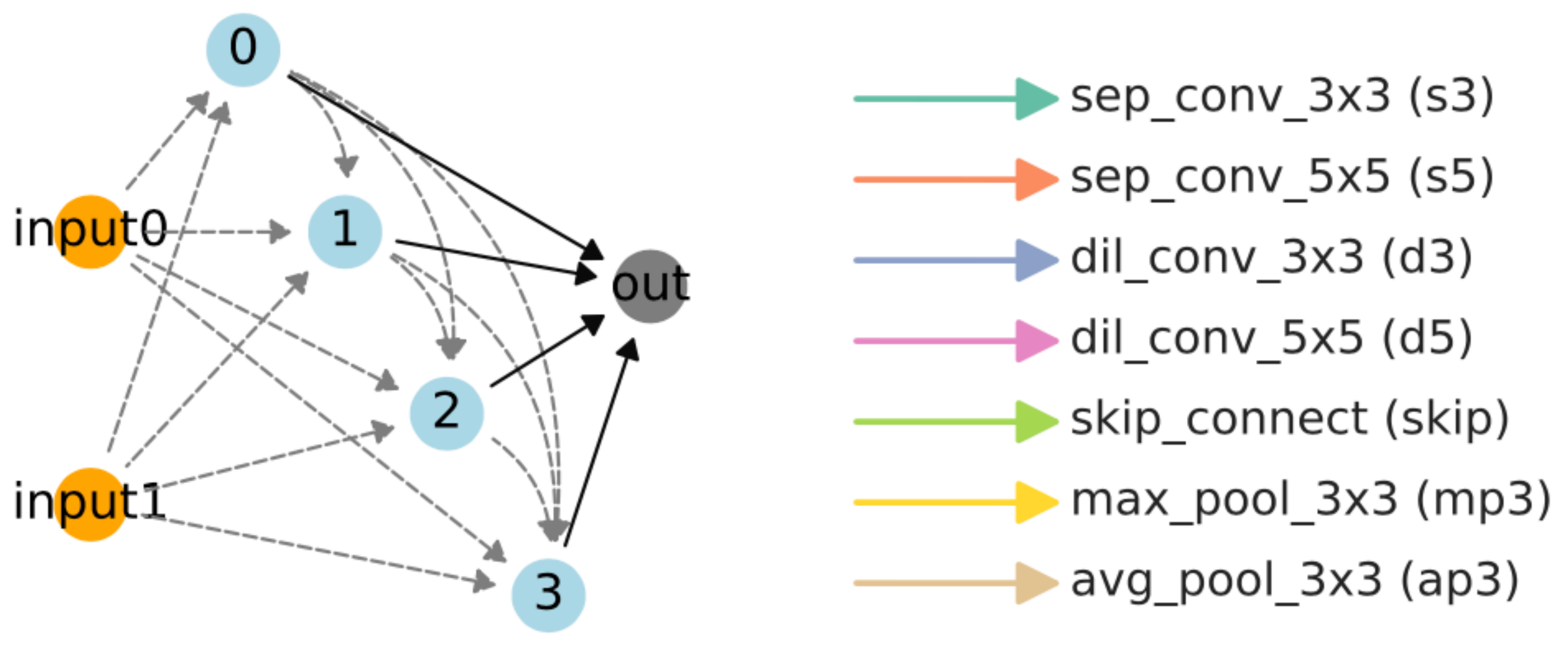


Illustration of the DARTS search space

- Takes the form of a directed acyclic graph (DAG)
- 2 inputs (connected to the previous cell and the cell before that), 4 intermediate nodes, 1 output node.
- *Operator* $o^{(i,j)}$ on the (i, j)-th node is selected from one of the *primitives*.
- Intermediate nodes aggregate: $x^j = \sum_{i<j} o^{(i,j)}(x^{(i)})$
- Up to 14 operators, but DARTS chooses 8
- Related spaces: NAS-Bench-201, DARTS with relaxed constraints/additional primitives etc.

[Liu et al, 2019] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *International Conference on Learning Representations (ICLR)*, 2019.

Cell-based Spaces are Nominally Large

- Number of possible cells: $\prod_{k=1}^{4} \frac{7^2 k(k+1)}{2} \approx 10^9$
- We typically search for two types of cells: *normal* and *reduce,* so possible number of architectures $\approx (10^9)^2 = 10^{18}$
- The large and complexity of search space motivates search
- Is the nominal complexity really informative?

Summary of Our Contributions

We present a post-hoc analysis to study the popular cell-based search spaces & architectures found by technically-diverse SoTA methods:

- Good (and bad) architectures often boil down to a small number of critical features.
- Many designs increase complexity but not performance.
- Astronomical nominal complexity of popular search spaces poorly reflects their actual diversity.
- With a few constraints, almost **any** random architectures can perform on par with the state of the art.

Operator-level Analysis

Operation Importance

- *Operators* — e.g. conv, skip, pool, are the simplest features.
    - 8 operators per DARTS cell — 16 in total since we usually search for normal and reduce cells separately
- *Operator Importance (OI)* quantifies the how important an operator is by measuring the sensitivity of performance to its perturbation.
- Consider cell α and edge $e_{(i,j)}$ currently assigned to primitive o_k, its OI is:

$$\mathrm{OI}(\alpha, e_{i,j} := o_k) = \frac{1}{|\mathcal{N}(\alpha, e_{i,j} := o_k)|} \sum_{m=1}^{|\mathcal{N}(\alpha, e_{i,j} := o_k)|} \Big[y(\alpha_m)\Big] - y(\alpha),$$

Neighbours of the original cell either by:

1. Assign the operator o_k to another primitive.
2. Change either end-node to another (allowed) node

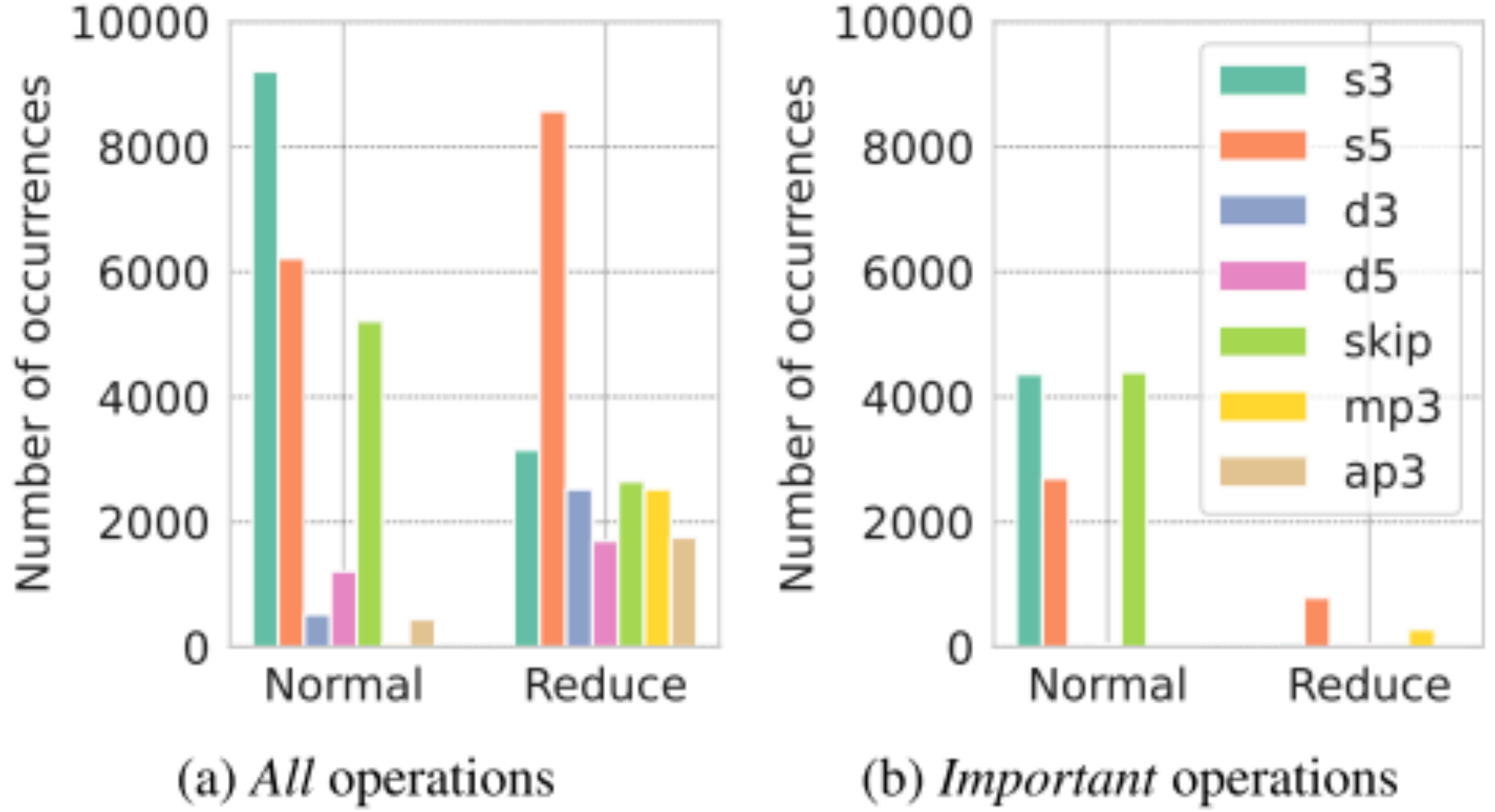
Operation Importance

- We study OI of all operators in top-5% performing architectures in NAS-Bench-301 [Siem et al, 2022]
- NAS-Bench-301: a benchmark defined over the entire DARTS space. Evaluated ~30K architectures, and then use an ensemble of surrogate to obtain *predicted* performance over the entire space.
- We use *predicted performance,* but **actually train** the architectures to verify any findings.

[Siem et al, 2022] Julien Siems et al. NAS-Bench-301 and the case for surrogate benchmarks for neural architecture search. In ICLR, 2022

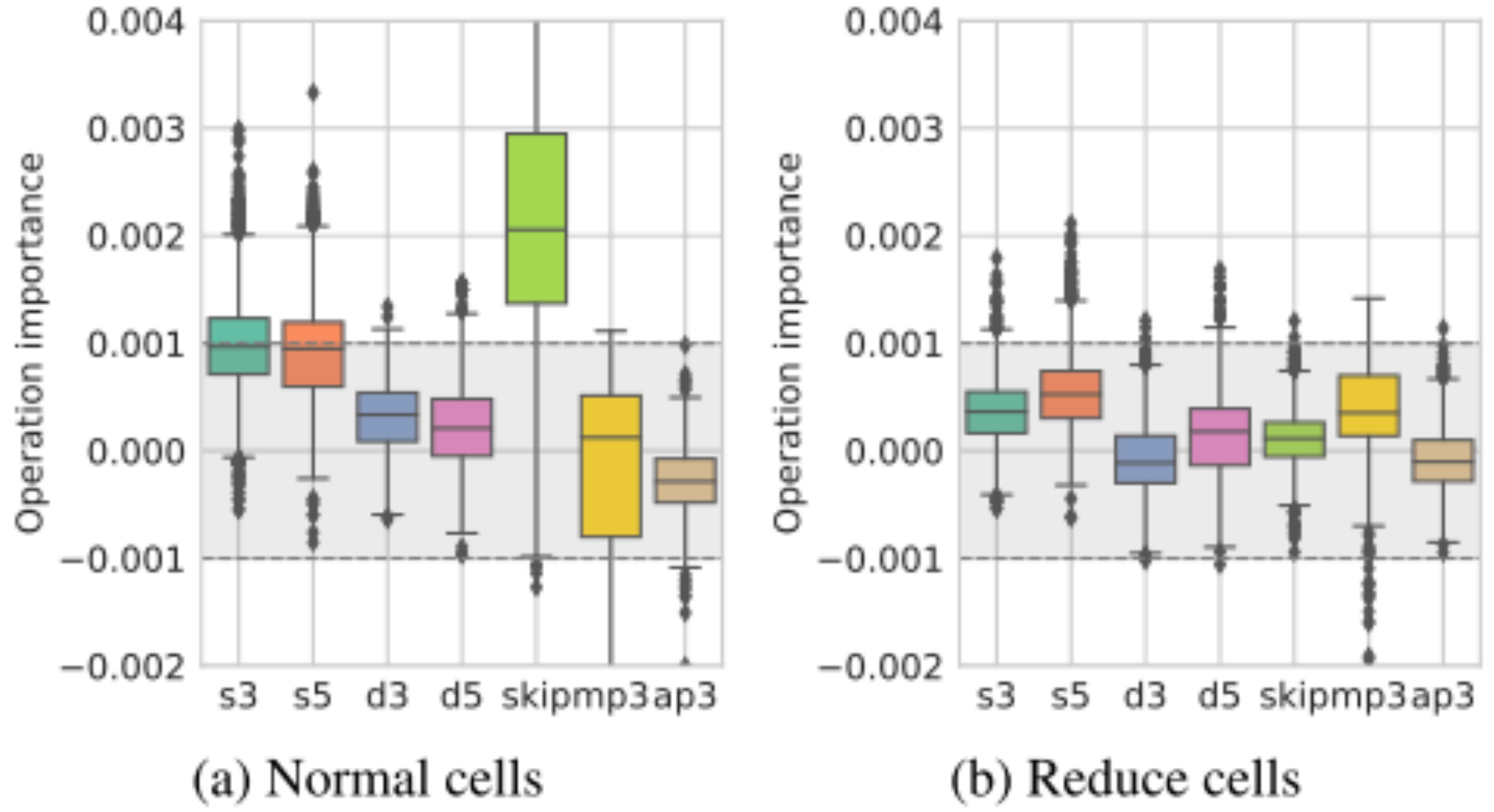
OI across cells & primitives

- *Important* operators: $|\mathrm{OI}| \geq 0.1\,\%$



(a) *All* operations (b) *Important* operations

Figure 3: Distribution of (a) all and (b) important operations by the primitive types of the top-performing archiectures, organised by primitive type.

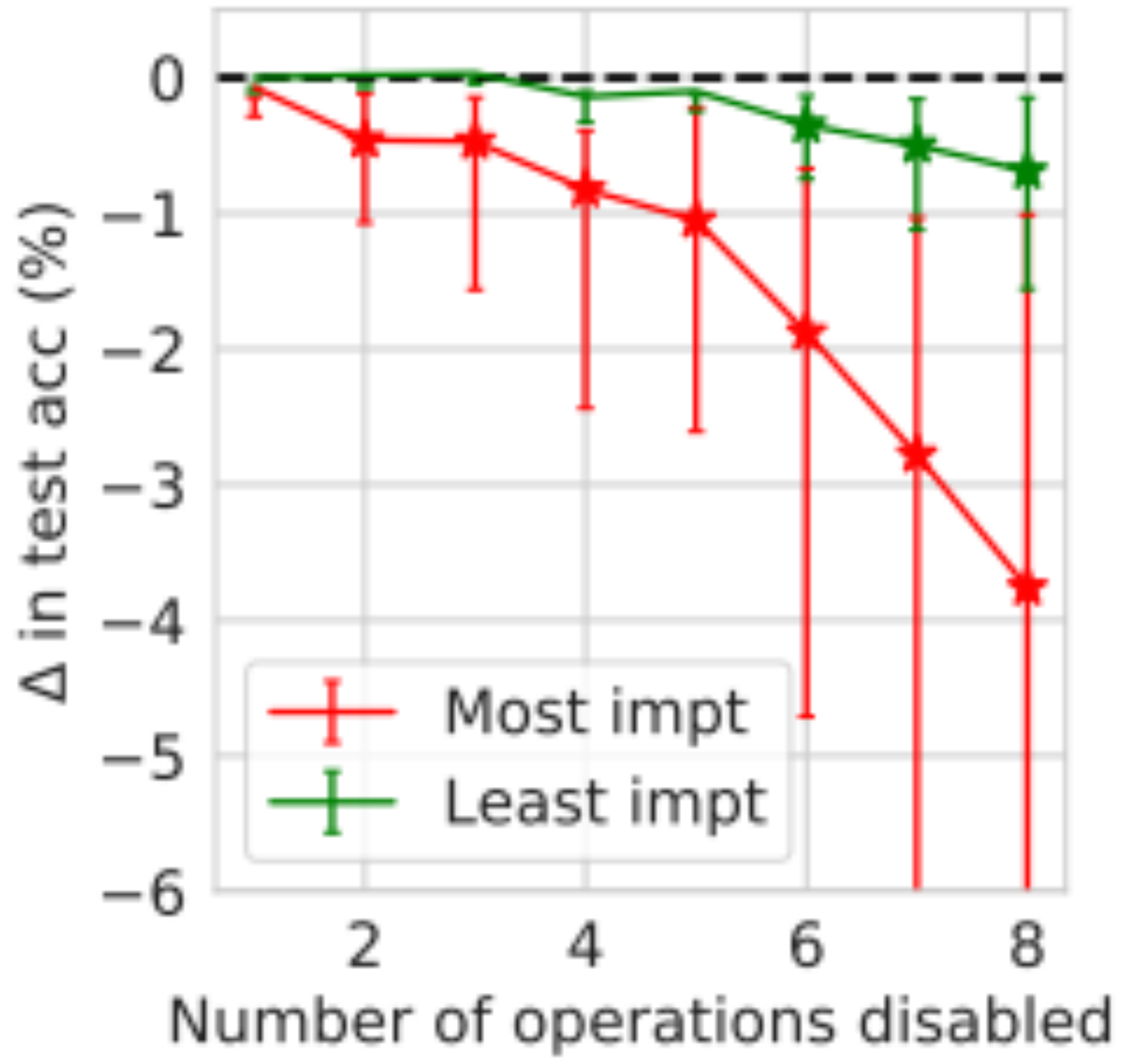


(a) Normal cells (b) Reduce cells

Figure 4: OI distribution in (a) normal and (b) reduce cells. The important operations are shown outside the gray shaded area.

Finding #1: Only a fraction of ops are critical

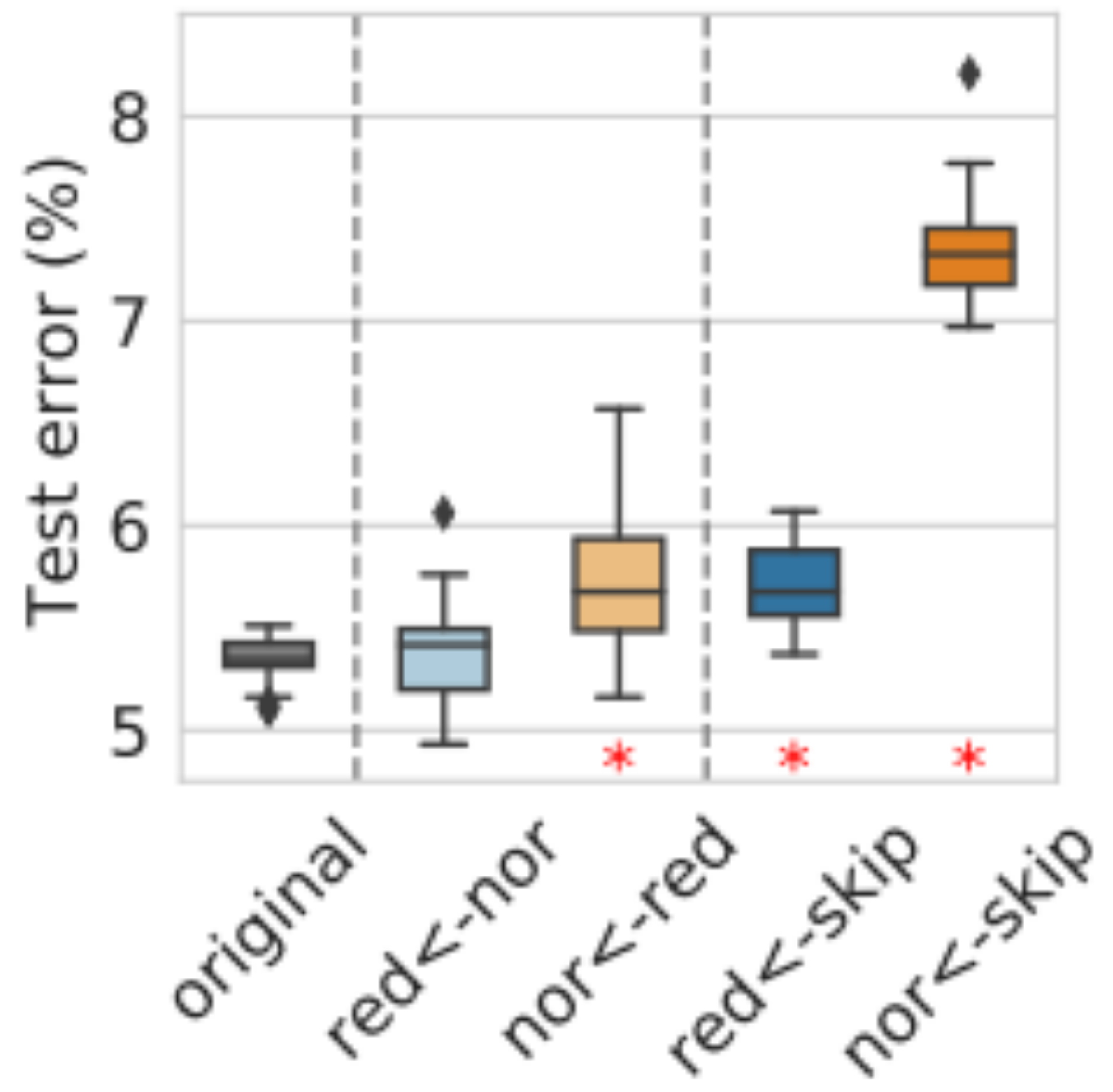
- Verification: select good-performing architectures by random, rank the 16 operators by OI, successively *disable* the operator (i.e. disconnect from the DAG) & *actually train from scratch*.



- Disable 6 low-OI operators ≈ 2 high-OI operators.
- Removing the important ops quickly degenerate performance.

Finding #2: Reduce cells are relatively unimportant

- Verification: draw another 30 random archs, for each cell construct 4 variants:
  1. Set reduce to normal (red <- nor)
  2. Set normal to reduce (nor <- red)
  3. Set reduce to all skip connections (red <- skip)
  4. Set normal to all skip connections (nor <- skip)



- Searching for one type of cell instead of two leads to no statistically significant deterioration of performance

Finding #3: Lots of redundant primitives

- Primitive pool is constructed from the prior knowledge
  - They are more or less useful in manually designed networks
- In NAS, it seems that we only need separable convolutions and skips!

Implications

- Search space in NAS in effect lies on a low-dimensional manifold
- Encoding NAS architectures exactly leads to high-dimensional vectors.
  - Difficult for methods like Gaussian Processes in particular
- Do we really lead to find an exact encoding?
  - What about approximate encoding and/or latent space representations?

Subgraph-level Analysis

Mining Important Subgraphs

- Topologies play a part in determining network performance
- *Subgraphs* are often used in explaining [Ying et al, 2019]
- We use *Frequent Subgraph Mining (FSM)* to identify subgraphs, or *motifs,* over-represented in good architectures
- We use the *gSpan* algorithm [Yan et al, 2002] to identify a set of subgraphs with minimum *support* $\sigma \geq 0.05$

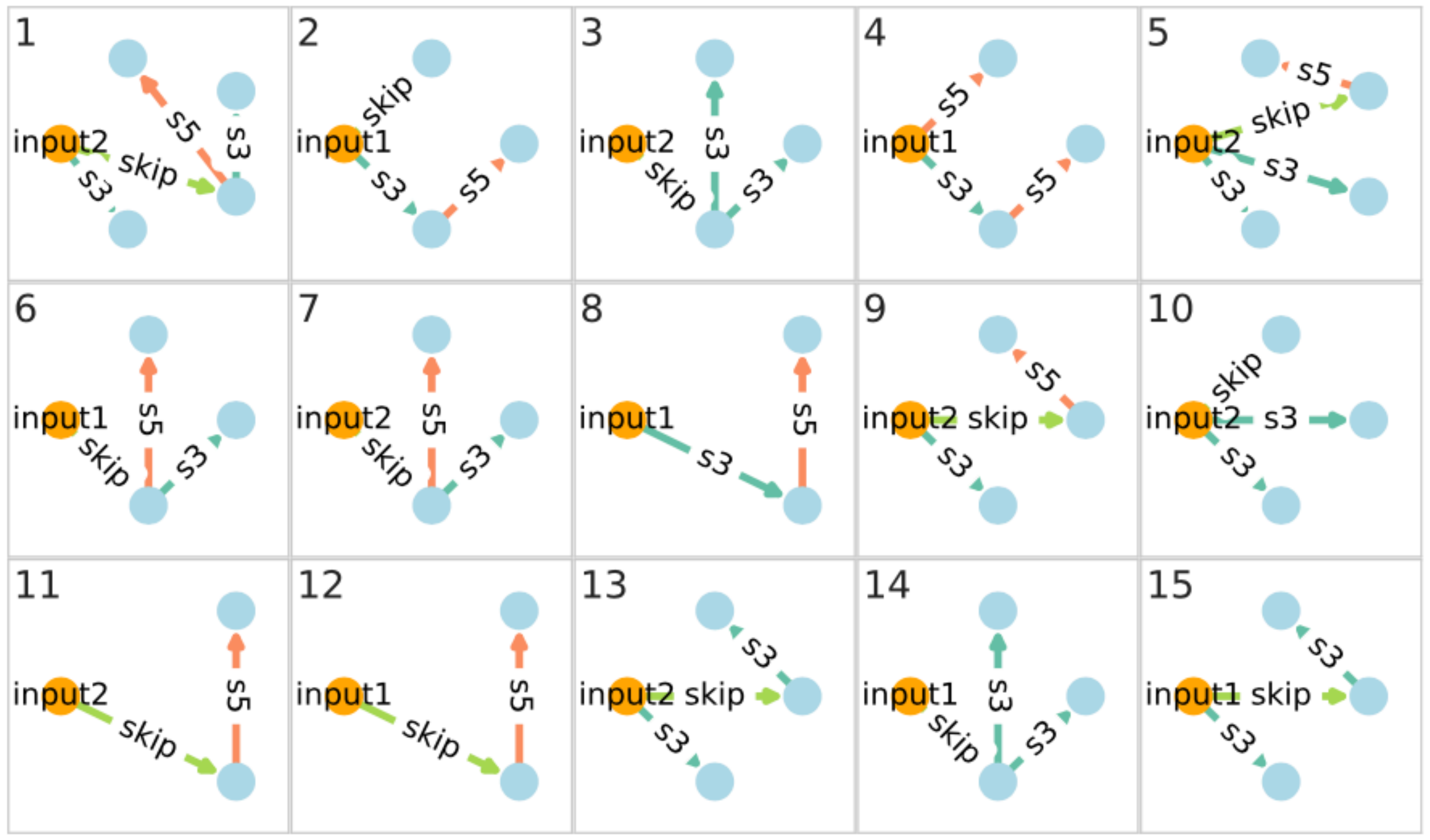
$$S_{g_i^f} = \frac{|\delta(g_i^f)|}{T} \geq \sigma \ \forall g_i^f \in \mathcal{G}^f, \text{ where } \delta(g_i^f) = \{G_j \,|\, g_i^f \subseteq G_j\}_{j=1}^{T}.$$

[Ying et al, 2019] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: A tool for post-hoc explanation of graph neural networks, 2019.
[Yan et al, 2002] Yan, Xifeng, and Jiawei Han. "gspan: Graph-based substructure pattern mining." *2002 IEEE International Conference on Data Mining, 2002. Proceedings.* IEEE, 2002.

Finding #4: Subgraphs found are boring

Almost all subgraphs are **separable conv + residual link** (i.e. skip connect with the input)!

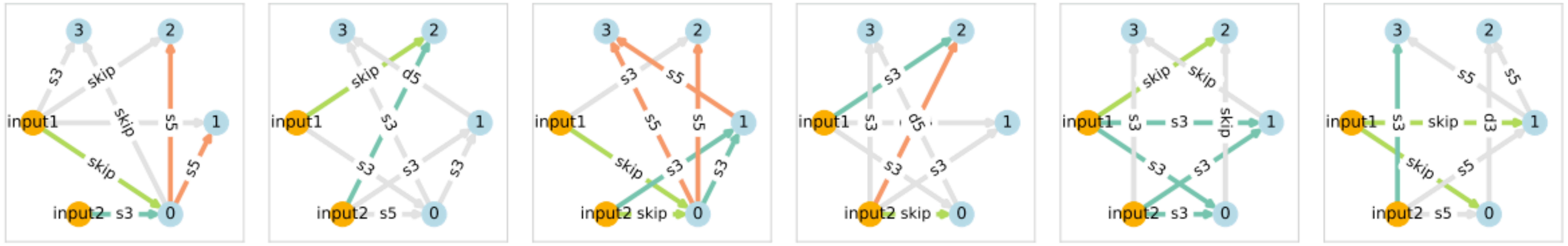
… and this is a combination that is known to work [Chollet, 2017]



Frequent subgraphs mined in the top-performing architectures in DARTS space

[Chollet 2017] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *CVPR*, pp. 1251–1258, 2017.

Patterns in the SOTA architectures discovered



(a) White et al. (2021) (b) Chen et al. (2021b) (c) Li et al. (2021) (d) Ru et al. (2021) (e) Wang et al. (2021c) (f) Chen et al. (2021a)

Figure 9: Normal cells of various SoTA (left to right: BANANAS, DRNAS, GAEA, NAS-BOWL, DARTS_PT and TE-NAS) architectures with the important operations highlighted (the connections to output are omitted since they are all identical across the DARTS search space). Note all cases considered are consistent with the residual link + separable convolution patterns identified, even though the cells and search methods are very different and except for BANANAS, none of the methods here was used to generate the NB301 training set.

Skips only matter when they are residual

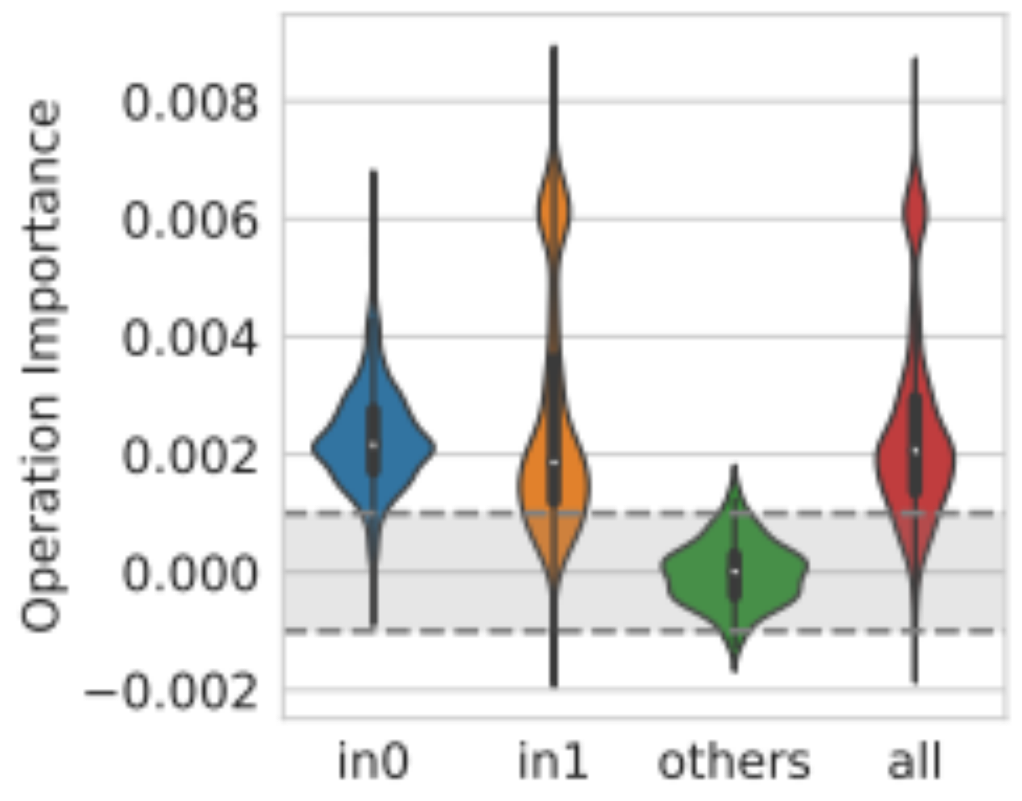


Figure 8: *`skips` are only useful when they form residual links:* `in0` and `in1` denote the residual links formed with either inputs, `others` denote the skip connections not forming residual links and `all` is the overall distribution of OI of skip connections.

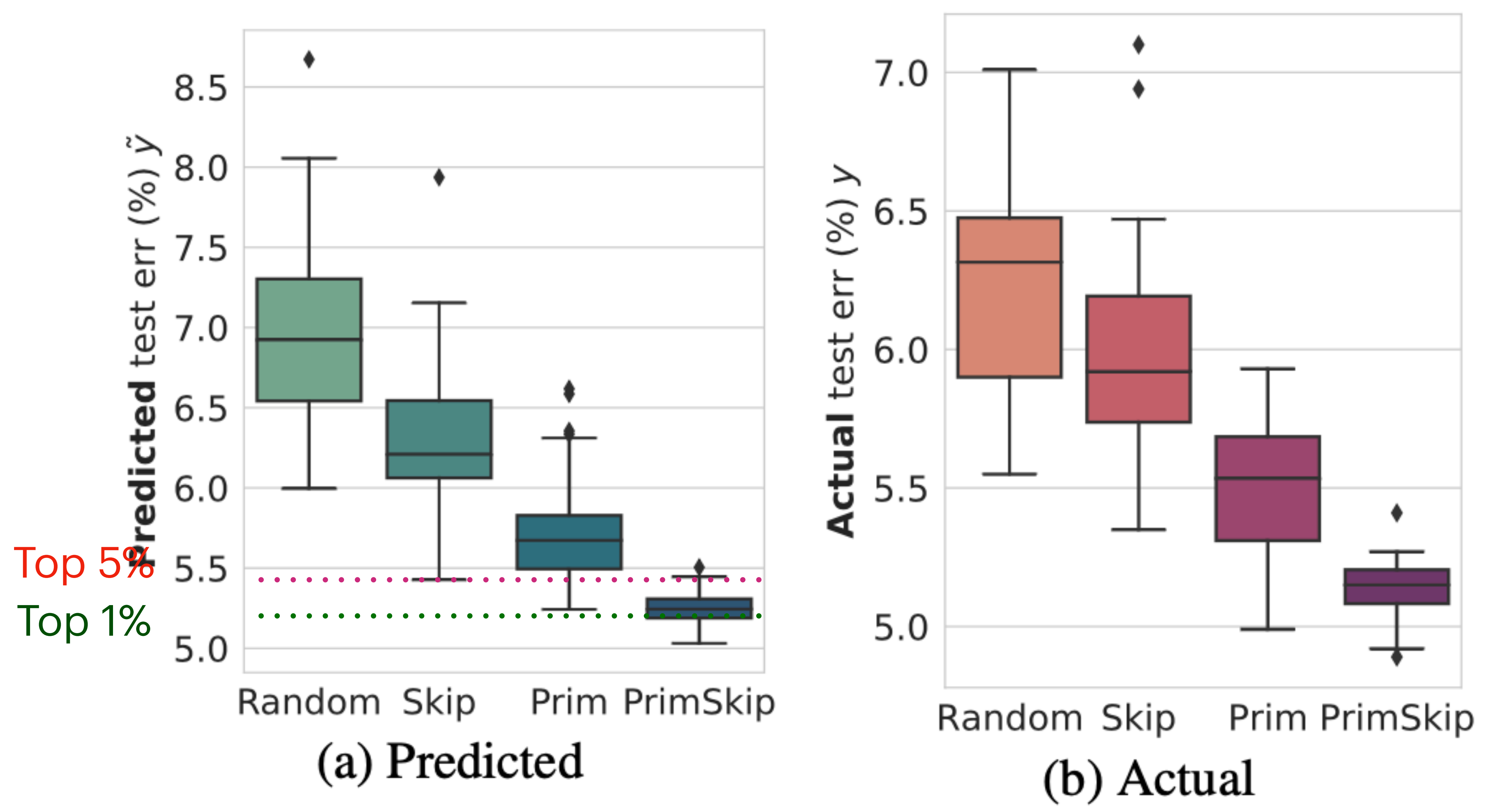
- Residual connections (skip involving inputs) drive the high-OI of skips.
- Skip connections do not just benefit optimisation of NAS supernets but also actively contribute to generalisation if they posit as residual connections.
- Skip is the only primitive whose exact position in the cell matters!
  - Could be an alternative explanation why it's difficult for search methods to learn this relation.

Generate architectures from motifs

Random sampling, but with one or both of the 2 constraints:

- `Skip`: Cells must contain residual link
- `Prim`: Apart from the residual link, ops can only be separable conv

Normal and reduce cells sampled using the same rules



(a) Surrogate-predicted and (b) actual test errors of **randomly sampled** DARTS architectures with/without the constraints

(Constrained) random archs perform similarly to SoTA

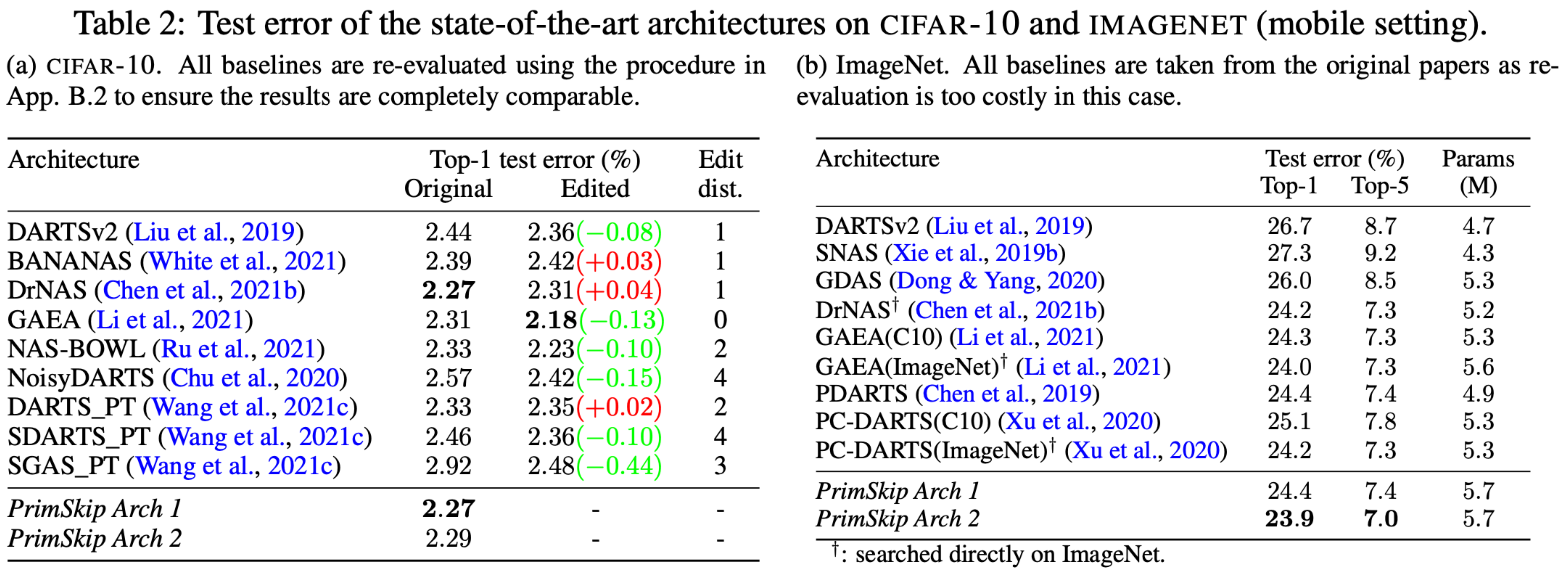


Table 2: Test error of the state-of-the-art architectures on CIFAR-10 and IMAGENET (mobile setting).

(a) CIFAR-10. All baselines are re-evaluated using the procedure in App. B.2 to ensure the results are completely comparable.

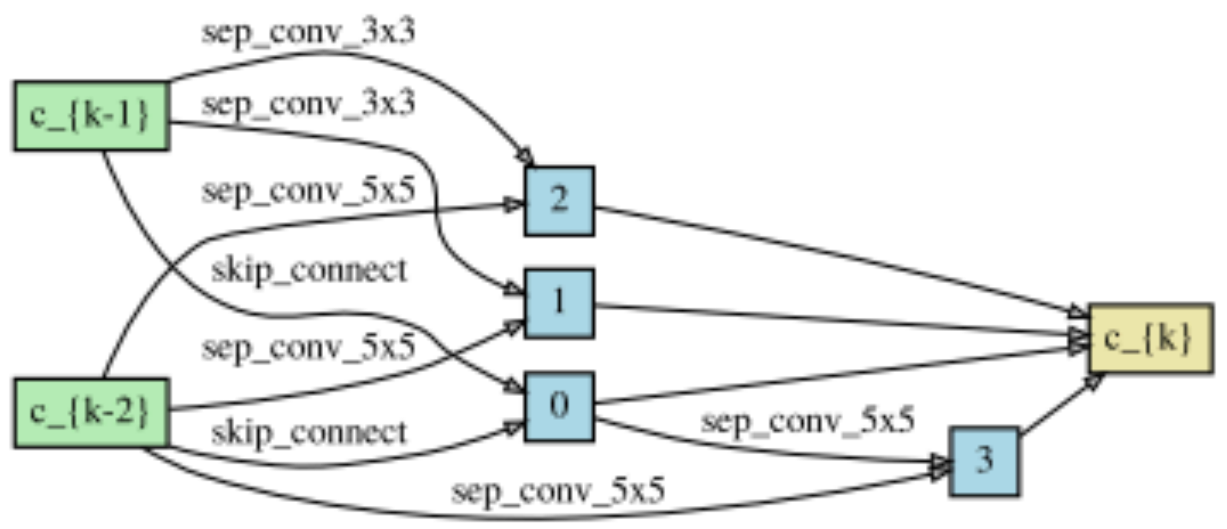
Architecture	Top-1 test error (%) Original	Top-1 test error (%) Edited	Edit dist.
DARTSv2 (Liu et al., 2019)	2.44	2.36(−0.08)	1
BANANAS (White et al., 2021)	2.39	2.42(+0.03)	1
DrNAS (Chen et al., 2021b)	**2.27**	2.31(+0.04)	1
GAEA (Li et al., 2021)	2.31	**2.18**(−0.13)	0
NAS-BOWL (Ru et al., 2021)	2.33	2.23(−0.10)	2
NoisyDARTS (Chu et al., 2020)	2.57	2.42(−0.15)	4
DARTS_PT (Wang et al., 2021c)	2.33	2.35(+0.02)	2
SDARTS_PT (Wang et al., 2021c)	2.46	2.36(−0.10)	4
SGAS_PT (Wang et al., 2021c)	2.92	2.48(−0.44)	3
PrimSkip Arch 1	**2.27**	-	-
PrimSkip Arch 2	2.29	-	-

(b) ImageNet. All baselines are taken from the original papers as re-evaluation is too costly in this case.

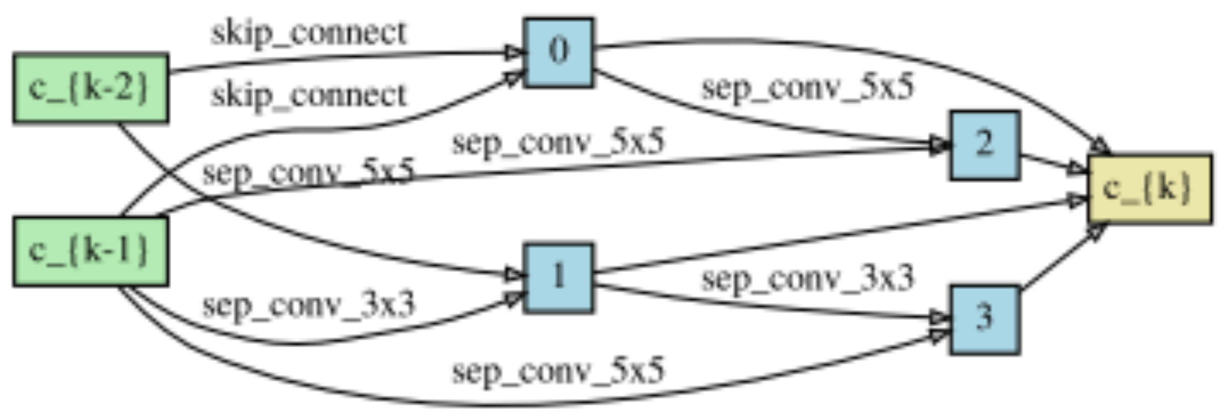
Architecture	Test error (%) Top-1	Test error (%) Top-5	Params (M)
DARTSv2 (Liu et al., 2019)	26.7	8.7	4.7
SNAS (Xie et al., 2019b)	27.3	9.2	4.3
GDAS (Dong & Yang, 2020)	26.0	8.5	5.3
DrNAS† (Chen et al., 2021b)	24.2	7.3	5.2
GAEA(C10) (Li et al., 2021)	24.3	7.3	5.3
GAEA(ImageNet)† (Li et al., 2021)	24.0	7.3	5.6
PDARTS (Chen et al., 2019)	24.4	7.4	4.9
PC-DARTS(C10) (Xu et al., 2020)	25.1	7.8	5.3
PC-DARTS(ImageNet)† (Xu et al., 2020)	24.2	7.3	5.3
PrimSkip Arch 1	24.4	7.4	5.7
PrimSkip Arch 2	**23.9**	**7.0**	5.7

†: searched directly on ImageNet.

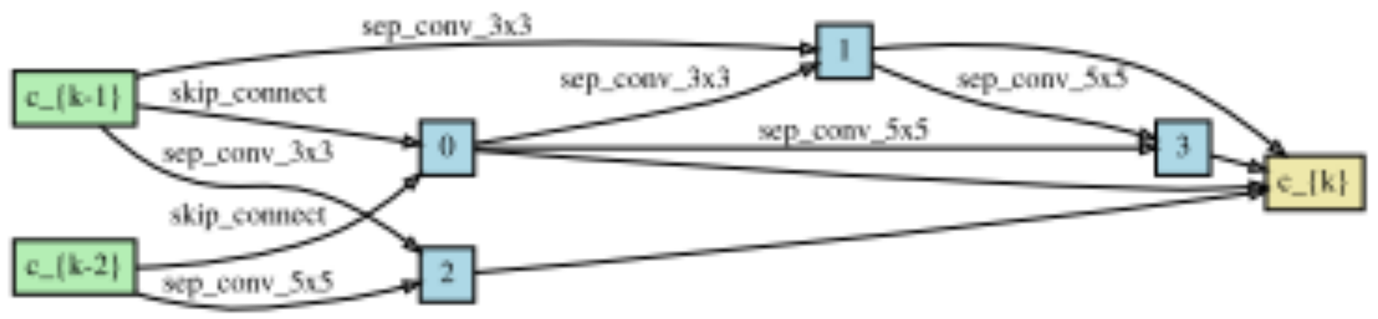
Selected architectures



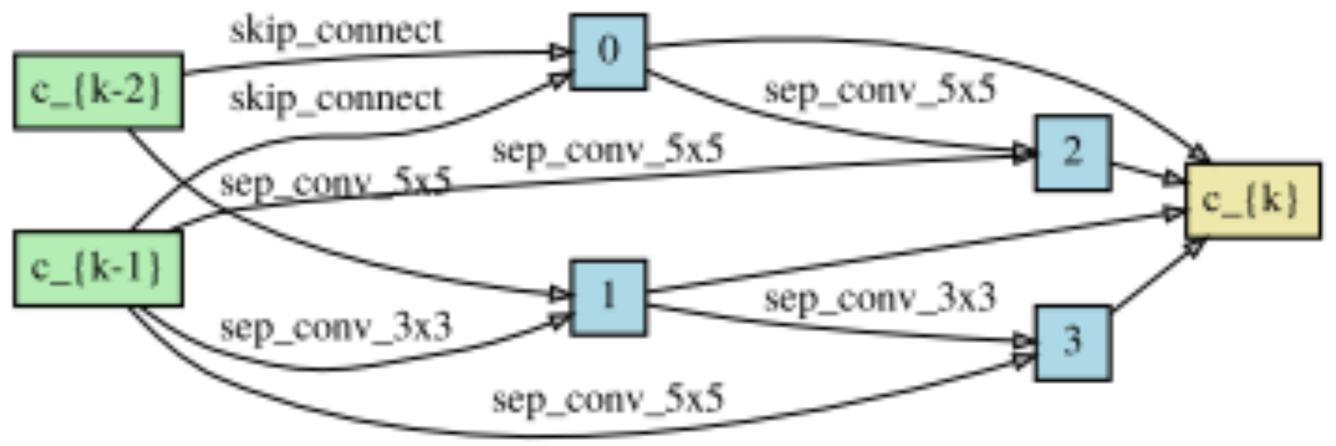
(a) Normal



(b) Reduce



(a) Normal



(b) Reduce

Implications

- Despite the visual differences, the good-performing architectures in DARTS are similar to each other.
- They may be collectively viewed as variants to **classical architectures**
- Should we still use existing cell-based search spaces as the key (or only) venue to develop new search methods?
  - If they **are** representative of actual search tasks, we don't actually need to search (if the architectures found via search are similar to ResNet).
  - If they **aren't**, then we should instead come up with more realistic search spaces

Summary & Suggestions

Key Takeaways & Suggestions

- Nominal complexity does **not** necessarily reflect architecture diversity & performance often hinges on simple (and often known) patterns.
- If good performance only depends on such simple factors, should we *only* rely on current cell-based benchmarks?
- One cause could be we search **within** cells only — should we also focus on the patterns **between** the cells? e.g. NAGO [Ru et al, 2020]
- Lower-level search? e.g. AutoML-Zero [Real et al, 2020]
- NAS benchmarks beyond cell-based search spaces only?

[Ru et al, 2020] Binxin Ru, Pedro Esperanca, and Fabio Maria Carlucci. "Neural architecture generator optimization." *Advances in Neural Information Processing Systems* 33 (2020)
[Real et al, 2020] Esteban Real et al. "Automl-zero: Evolving machine learning algorithms from scratch." *International Conference on Machine Learning*. PMLR, 2020.

Conclusion

- Paper: https://openreview.net/pdf?id=rFJWoYoxrDB
- Code: https://github.com/xingchenwan/cell-based-NAS-analysis
- Contact: Xingchen Wan <xwan@robots.ox.ac.uk>