Evaluating Efficient Performance Estimators of Neural Architectures

Xuefei Ning[1], Changcheng Tang[2], Wenshuo Li[1], Zixuan Zhou[1],
Shuang Liang[2], Huazhong Yang[1†], Yu Wang[1†]

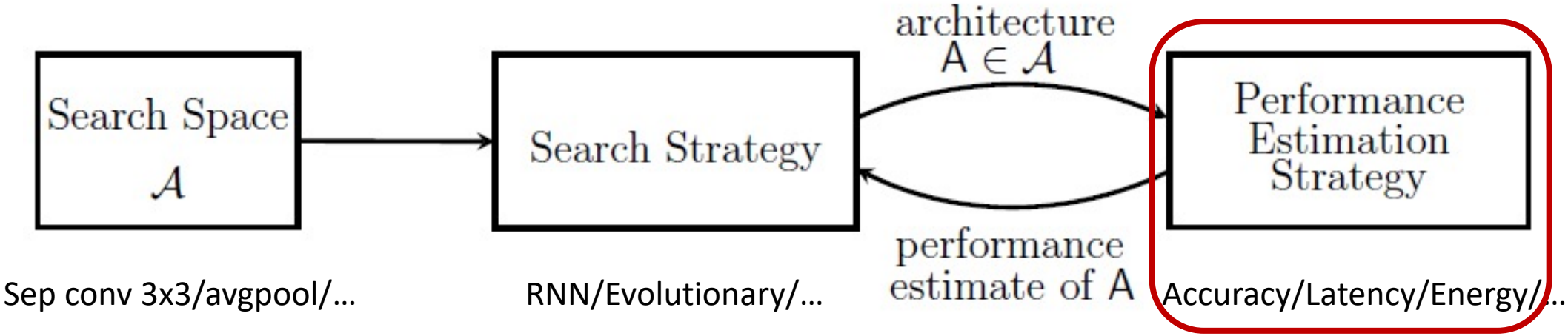
[1] Department of Electronic Engineering, Tsinghua University
[2] Novauto Technology Co. Ltd.
foxdoraame@gmail.com
†yanghz@tsinghua.edu.cn
†yu-wang@tsinghua.edu.cn

Menu

1. **Analysis Framework**
2. Evaluating One-shot Estimators (OSEs)
3. Improving One-shot Estimators (OSEs)
4. Evaluating Zero-shot Estimators (ZSEs)
5. Summary

What Does This Work Cares About?

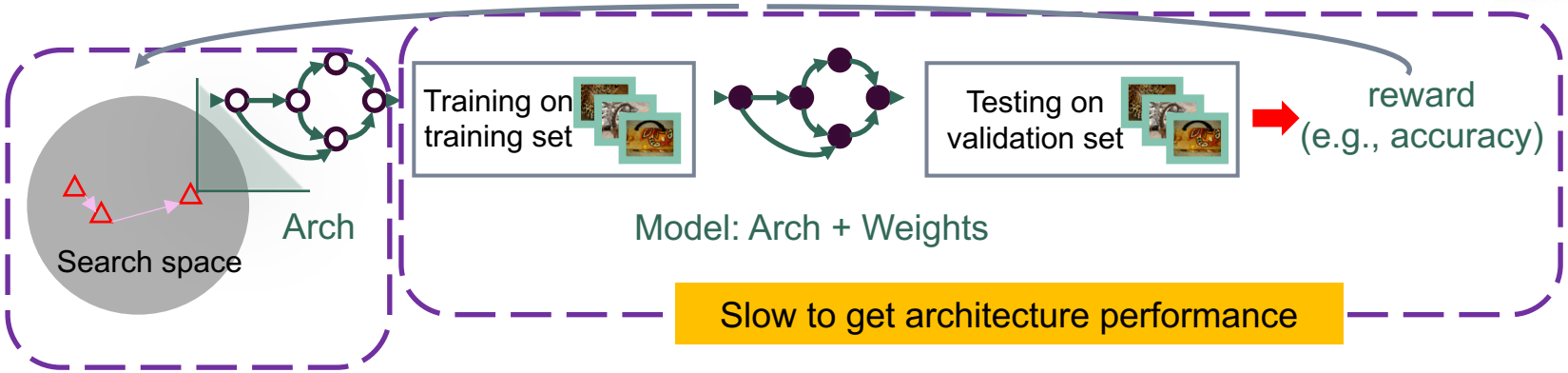
What is the quality of current efficient performance estimation strategies (estimators) ?



Thomas Elsken, et al., Neural Architecture Search: A Survey, JMLR 2019.

Analysis Target 1: One-Shot Estimators

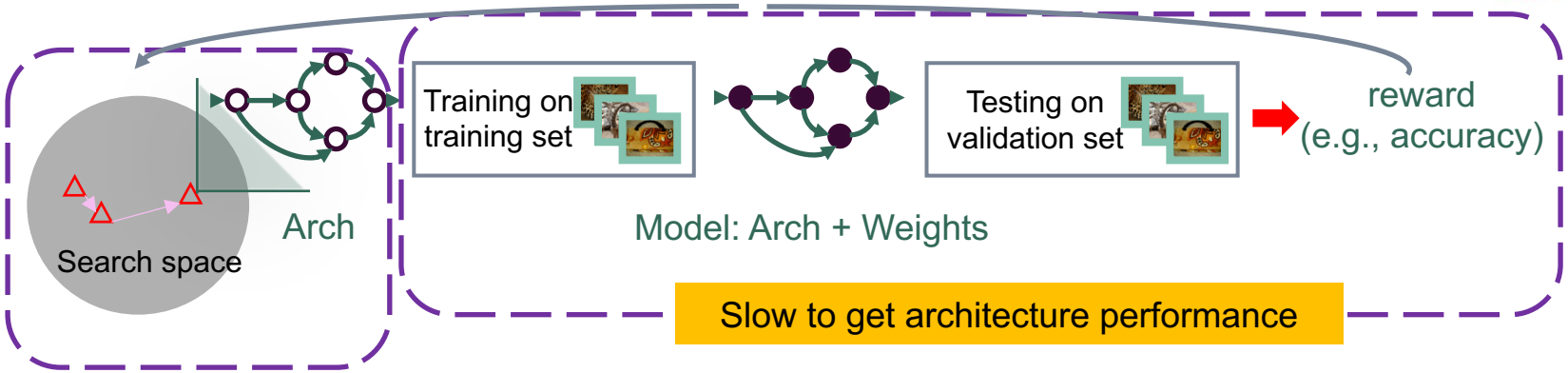
[Zoph et al., ICLR 2017] explore 13k architectures, each trained from scratch for 50 epochs: **~48k GPU hours! Expensive!**



Hieu Pham, et al., Efficient Neural Architecture Search via Parameter Sharing, ICML 2018.
Bender, Gabriel, et al. "Understanding and Simplifying One-Shot Architecture Search." ICML 2018.

Analysis Target 1: One-Shot Estimators

[Zoph et al., ICLR 2017] explore 13k architectures, each trained from scratch for 50 epochs: **~48k GPU hours! Expensive!**



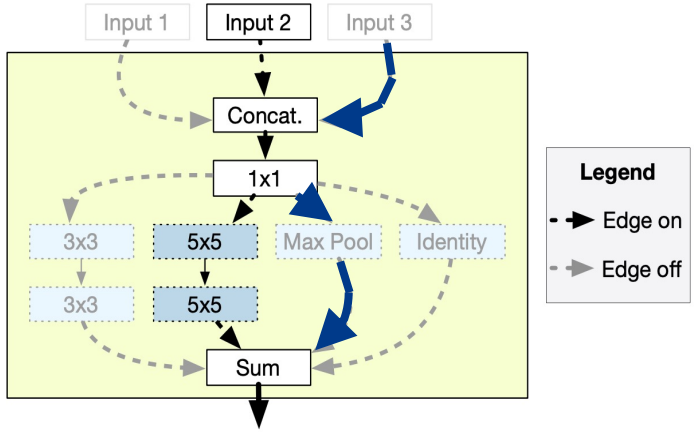
How to conduct **efficient performance estimation**? **One-Shot Estimator** (OSEs)

[Pham et al., ICML 2018]: **parameter sharing technique**

- Construct an over-parametrized network called **supernet**
- Each candidate architecture is evaluated using the corresponding subset of shared parameters, **without separate training**

Conv 1x1 shared between different architectures

- **Input 2 -> Concat -> 1x1 -> 5x5 -> 5x5**
- **Input 3 -> Concat -> 1x1 -> Max Pool**



Hieu Pham, et al., Efficient Neural Architecture Search via Parameter Sharing, ICML 2018.
Bender, Gabriel, et al. "Understanding and Simplifying One-Shot Architecture Search." ICML 2018.

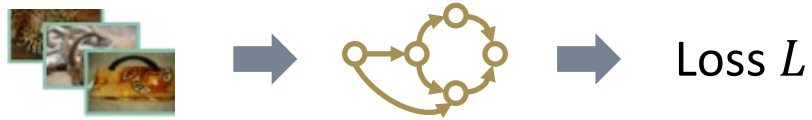
Analysis Target 2: Zero-Shot Estimators

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

Can we further reduce the training cost to **zero**? **Zero-Shot Estimator** (ZSEs)

[Abdelfattah, et al., ICLR 2021] measure the architecture's score by adding up parameter-wise sensitivity measures (from pruning literature).

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



$x_1, \ldots x_n$

randomly initialized θ

Abdelfattah M S, et al., Zero-cost proxies for lightweight NAS, ICLR 2021.
Mellor J, et al. Neural architecture search without training. arXiv preprint arXiv:2006.04647v1, 2020.
Mellor J, et al. Neural architecture search without training, ICML 2021.

Analysis Target 2: Zero-Shot Estimators

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

Can we further reduce the training cost to **zero**? **Zero-Shot Estimator** (ZSEs)

Architecture-level ZSEs measure the architecture's score (discriminability) by inference differences between different input images.

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

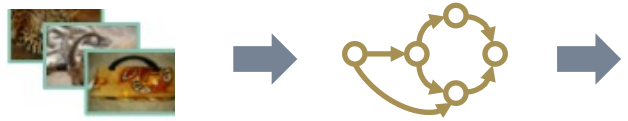
[Mellor, et al., arXiv 2020]

input gradients difference of different inputs

$$relu\_logdet(\alpha) = g(\boldsymbol{y}(x_1), \dots, \boldsymbol{y}(x_n))$$

[Mellor, et al., ICML 2021]

activation difference of different inputs



$x_1, \dots x_n$

randomly initialized θ

Loss L

Abdelfattah M S, et al., Zero-cost proxies for lightweight NAS, ICLR 2021.

Mellor J, et al. Neural architecture search without training. arXiv preprint arXiv:2006.04647v1, 2020.

Mellor J, et al. Neural architecture search without training, ICML 2021.

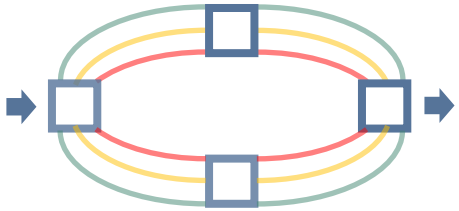
Why This Work?

OSEs are very widely applied.

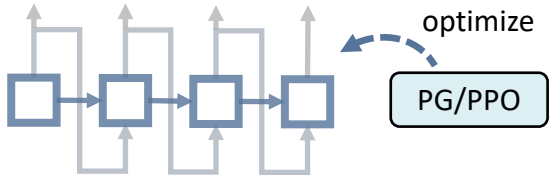
Despite their wide application, OSEs' quality lacks thorough study.

Various Search Strategies

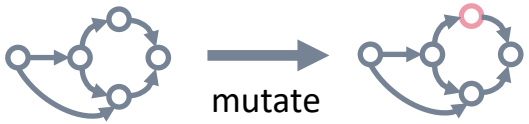
Differentiable



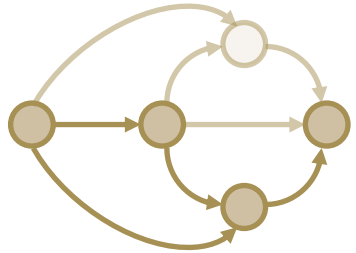
RL-learned RNN



Evolutionary

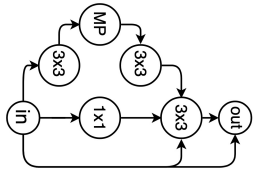


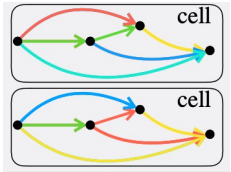
Various Applications

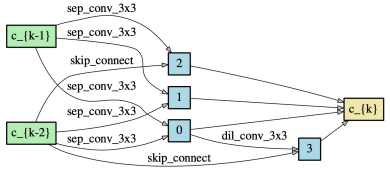


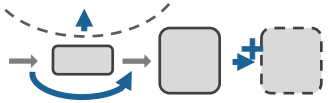
One-Shot Estimator (OSE)

Various Search Spaces









Why This Work?

ZSEs are ultra-efficient.

Are current ZSEs powerful enough for evaluating architectures on various search spaces?

Can the summation of parameter sensitivities reflects the architectures' ability?

What architectures do they overestimate or underestimate?

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

What should we do to further improve ZSEs?

Yu, et al., Evaluating the search phase of neural architecture search, ICLR 2020.
Zela, et al., Nasbench1shot1: Benchmarking and dissecting oneshot neural architecture search, ICLR 2020.

Why This Work?

ZSEs are ultra-efficient.

Are current ZSEs powerful enough for evaluating architectures on various search spaces?

Can the summation of parameter sensitivities reflects the architectures' ability?

What architectures do they overestimate or underestimate?

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

What should we do to further improve ZSEs?

Current studies [Yu et al., ICLR 2020][Zela et al., ICLR 2020] that evaluate the quality of efficient performance estimators are not through in terms of

- **Analysis targets**: No extensive comparison of OSEs and ZSEs.
- **Analysis benchmarks**: Do not use benchmarks with different properties and sizes.
- **Analysis aspects**: Evaluation criteria are not thorough. Lack bias analysis (over-/under-estimation).

Yu, et al., Evaluating the search phase of neural architecture search, ICLR 2020.
Zela, et al., Nasbench1shot1: Benchmarking and dissecting oneshot neural architecture search, ICLR 2020.

Analysis Benchmarks

- Five NAS benchmarking search spaces with distinct sizes and properties

NAS-Bench-1shot1 (NB101)

[Zela ICLR 2020]

14.6k (without isomorphic counterparts)

op on node, 3 op types, 7 nodes, 9 edges

5 shared position

NAS-Bench-201 (NB201)

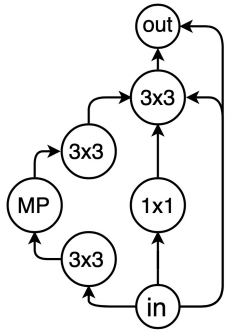
[Dong, ICLR 2020]

6.5k/15.6k (w.o./w. isomorphic counterparts)

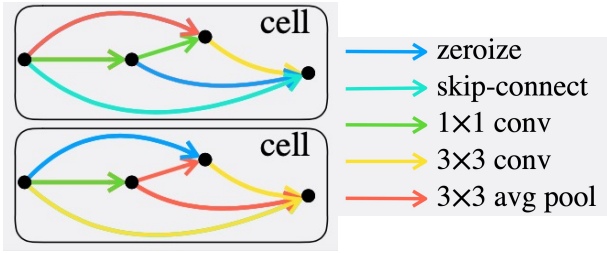
op on edge, 5 op types, 4 nodes, 6 edges

6 shared position

Op on node



Large proportion of isomorphic architectures



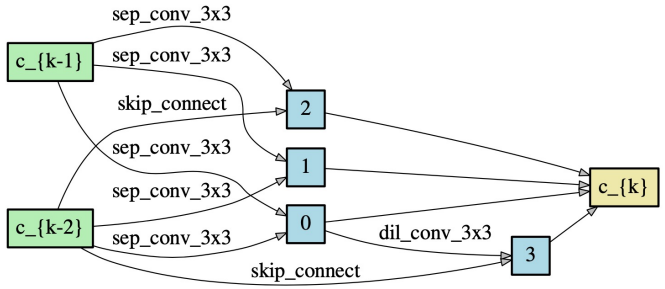
NAS-Bench-301 (NB301)

[Siems, ICLR 2021]

10^{18} (with isomorphic counterparts)

op on edge, 7 op types, 6 nodes, 8 edges

14 shared position



Largest search space, similar to DARTS

architecture difference is small, fewer very bad architectures

- Radosavovic et al., On network design spaces for visual recognition, ICCV 2019.
- Siems et al., Nas-bench-301 and the case for surrogate benchmarks for neural architecture search, ICLR 2021.
- Dong et al., One-shot neural architecture search via self-evaluated template network, ICLR 2020.

Analysis Benchmarks

- Five NAS benchmarking search spaces with distinct sizes and properties

NAS-Bench-1shot1 (NB101)

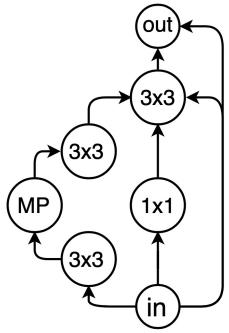
[Zela ICLR 2020]

14.6k (without isomorphic counterparts)

op on node, 3 op types, 7 nodes, 9 edges

5 shared position

Op on node



NAS-Bench-201 (NB201)

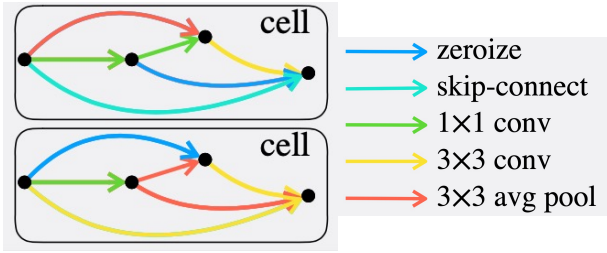
[Dong, ICLR 2020]

6.5k/15.6k (w.o./w. isomorphic counterparts)

op on edge, 5 op types, 4 nodes, 6 edges

6 shared position

Large proportion of isomorphic architectures



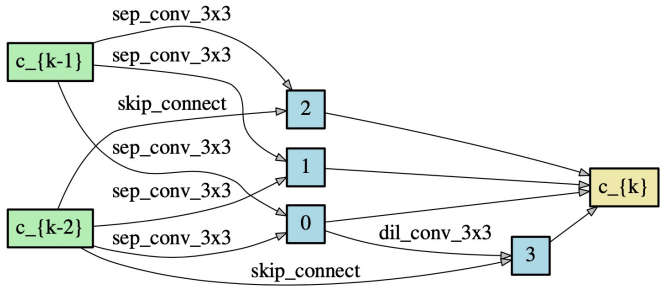
NAS-Bench-301 (NB301)

[Siems, ICLR 2021]

10^{18} (with isomorphic counterparts)

op on edge, 7 op types, 6 nodes, 8 edges

14 shared position



Largest search space, similar to DARTS

architecture difference is small, fewer very bad architectures

Non-topological Search Spaces

[Radosavovic, ICCV 2019]

NDS ResNet

1.26M

Decision types: depth, width

NDS ResNeXt-A

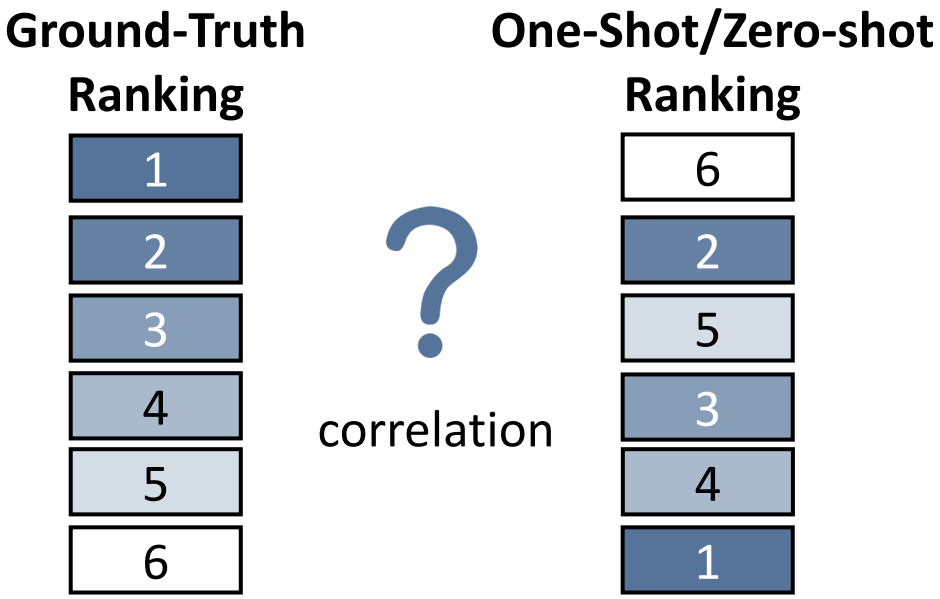
11.39M

Decision types: depth, width, bottleneck width ratio, conv group

	depth	width	ratio	groups	total
ResNet	1,24,9	16,256,12			1,259,712
ResNeXt-A	1,16,5	16,256,5	1,4,3	1,4,3	11,390,625

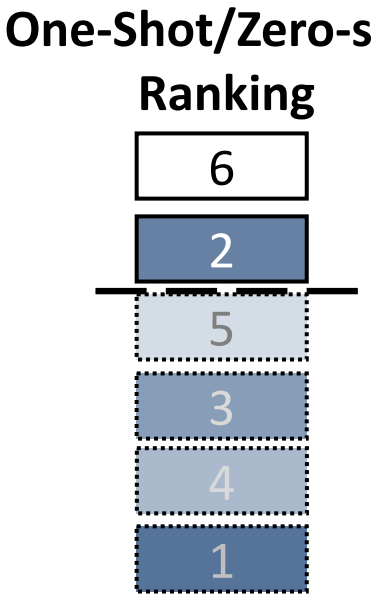
- Radosavovic et al., On network design spaces for visual recognition, ICCV 2019.
- Siems et al., Nas-bench-301 and the case for surrogate benchmarks for neural architecture search, ICLR 2021.
- Dong et al., One-shot neural architecture search via self-evaluated template network, ICLR 2020.

Analysis Aspects



1. Ranking correlation
   - Kendall's Tau (KD)
   - SpearmanR

Analysis Aspects



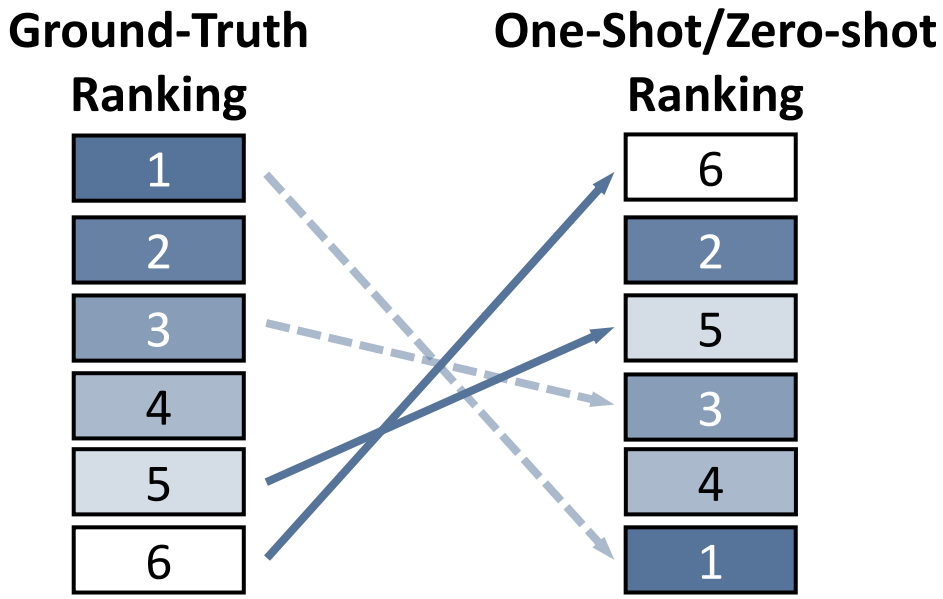
1. Ranking correlation
   - Kendall's Tau (KD)
   - SpearmanR
2. Distinguishing ability of top/bottom architectures
   - P@topK, P@bottomK
   - BestRanking@K, WorstRanking@K

Criteria that focus more on the top architectures

- P@topK: The proportion of the true top-K archs in the top-K archs according to the estimated scores
- BR@K: The best normalized ranking among the top-K archs according to the estimated scores

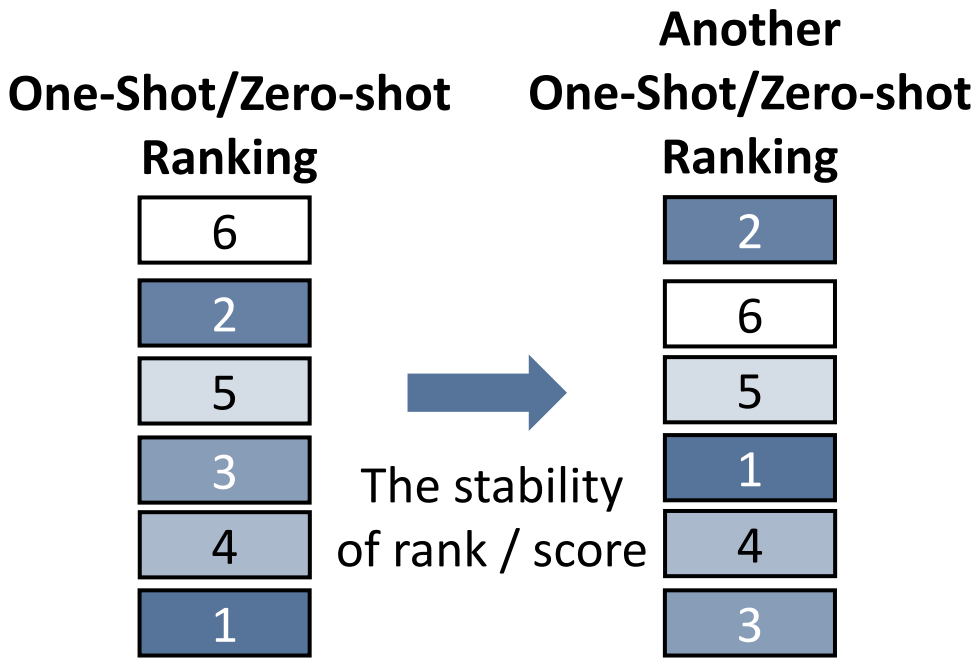
e.g., P@topK=50%, BestR@K=2/6 (K=$\frac{1}{3}$ in the example figure)

Analysis Aspects



1. Ranking correlation
   - Kendall's Tau (KD)
   - SpearmanR
2. Distinguishing ability of top/bottom architectures
   - P@topK, P@bottomK
   - BestRanking@K, WorstRanking@K
3. Bias
   - Which architectures are over- or under-estimated?

Analysis Aspects

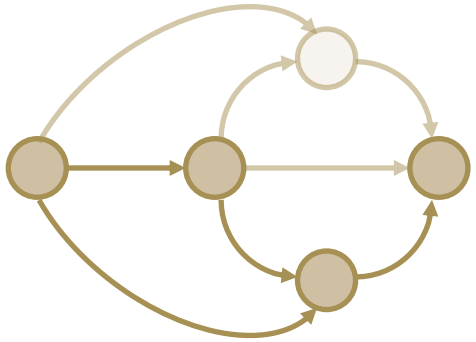


1. Ranking correlation
   - Kendall's Tau (KD)
   - SpearmanR
2. Distinguishing ability of top/bottom architectures
   - P@topK, P@bottomK
   - BestRanking@K, WorstRanking@K
3. Bias
   - Which architectures are over-under estimated?
4. Variance
   - E.g., How the ranking changes over time for OSE?

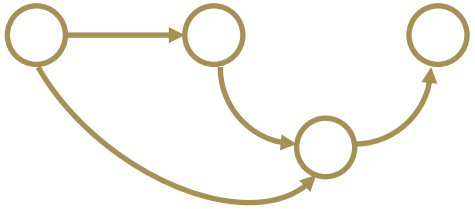
Analysis Framework

Analysis Targets

One-Shot Estimator (OSE)



Zero-Shot Estimator (ZSE)

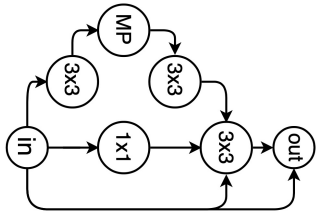


Analysis Aspects

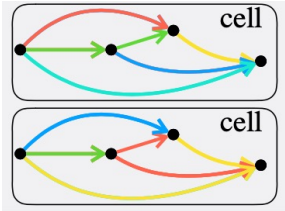
- Ranking correlation
- Distinguishing ability of top /bottom architectures
- Bias
  - Complexity-level
  - Operation-level
  - Architecture-level
- Variance (e.g., temporal variance of OSEs)

Analysis Benchmarks

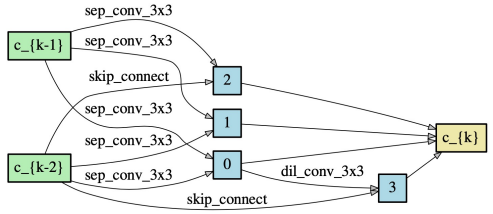
NB101



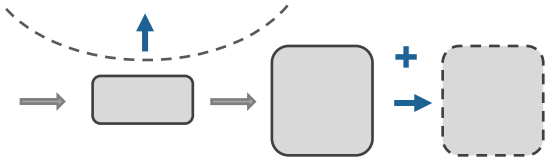
NB201



NB301



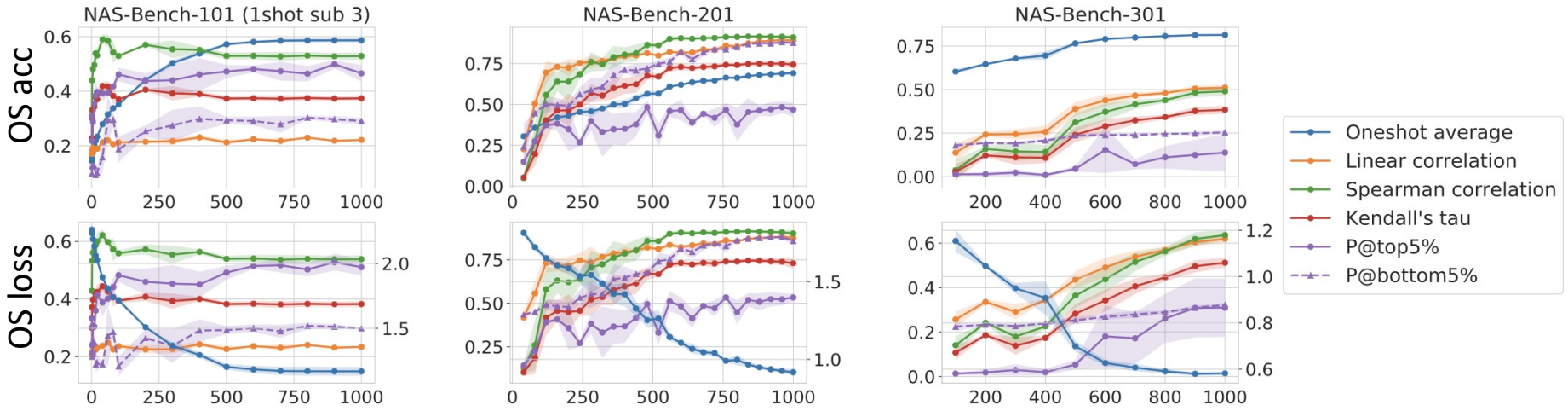
NDS ResNet & ResNeXt-A

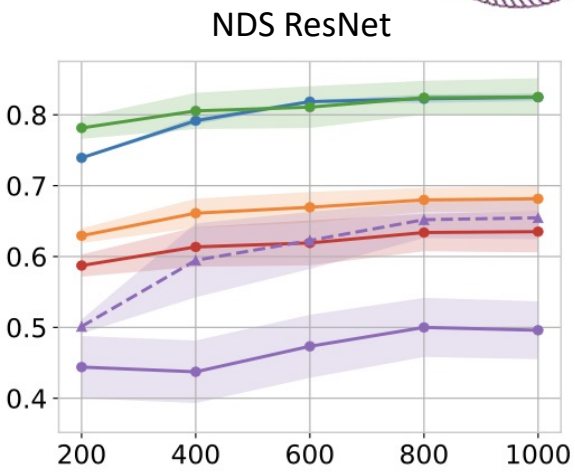


Menu

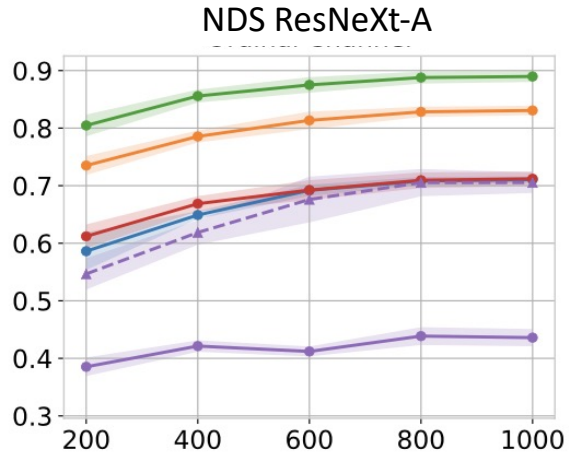
1. Analysis Framework
2. **Evaluating One-shot Estimators (OSEs)**
3. Improving One-shot Estimators (OSEs)
4. Evaluating Zero-shot Estimators (ZSEs)
5. Summary

How training time influences the ranking quality?





1. **Observations**
   1. On NB201/NB301/NDS ResNet/NDS ResNeXt-A
      - More sufficient training leads to ranking quality improvements
      - OSEs are better at distinguishing bad architectures than good architectures (P@top<P@bottom)
   2. On NB101-1shot1, bad architectures are harder to distinguish (P@top>P@bottom), longer training does not help after 20 epochs
   3. On NB301, using loss as the OS score is much better than using accuracy.
2. **Provide very strong baselines for future one-shot improvement techniques**

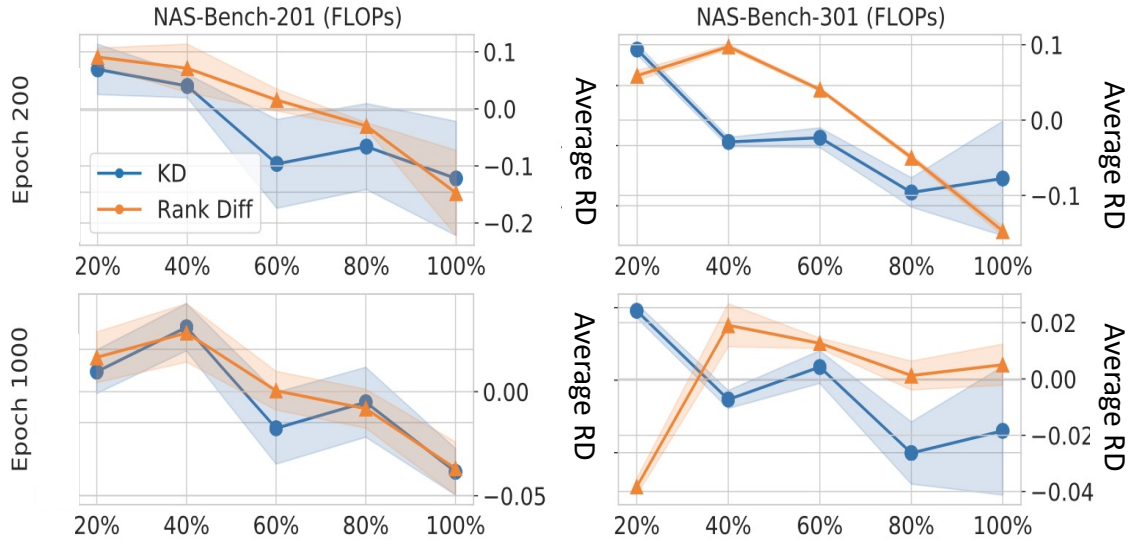


How the OSEs are biased?

Complexity-level bias

Method: Partitioning architectures into 5 complexity groups

Rank Diff of a_i: $(r_i - n_i)$. Positive RD indicates overestimation.

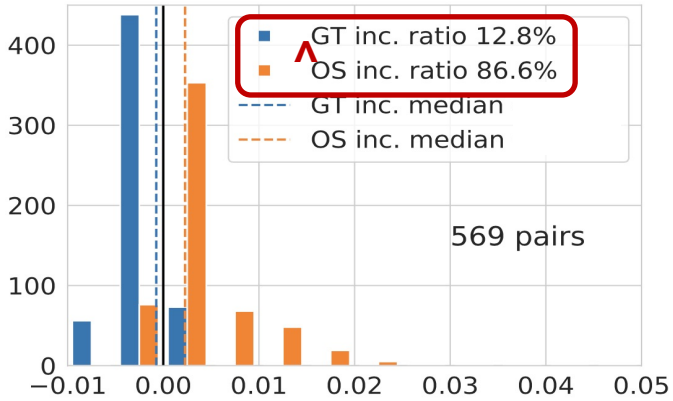


The average RD in each complexity group

- In the early stages of supernet training, due to the insufficient training, smaller architectures tend to be overestimated.

Op-level bias

Method: Mutate one op in architectures to another op, observe the change histogram of OS score/GT score of the mutation arch pairs (edit distance=1). Larger OS increase ratio than GT increase ratio indicates overestimation of the target operation.



The changing histogram of mutating skip-connect to DilConv

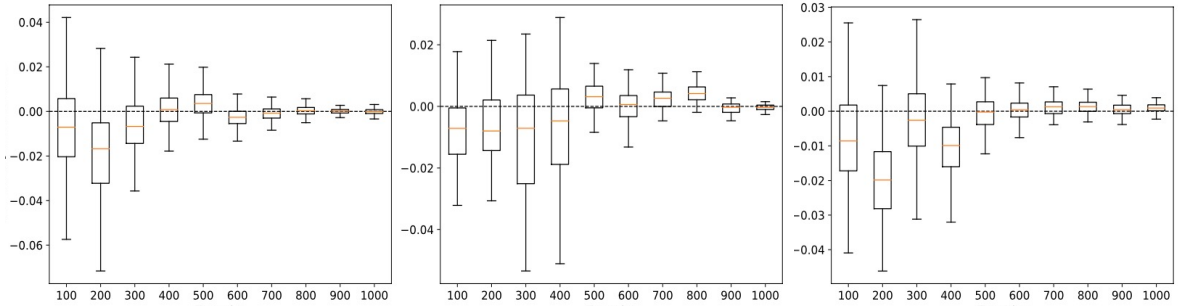
- For example, on NB301, OSE underestimates skip-connect and overestimate DilConv.

Is the OSE ranking/accuracy stable? (temporal variance)

Accuracy stability & forgetting

Method: **Forgetting value (FV)** defined as:
the accuracy of a_i after 1 epoch – the accuracy of a_i after it has just been trained in the supernet.
Negative FV indicates the existence of accuracy forgetting.

The changes of FV as the training goes on:



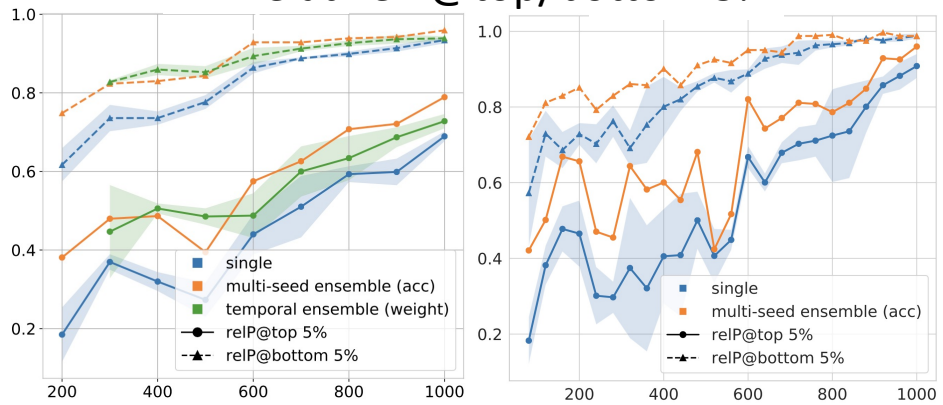
NB101 NB201 NB301

- Early stage: The forgetting phenomenon exists
- Later stage: LR decay -> More stable, and positive transfer exists

Ranking stability

Method: **Relative P@topK/bottomK** defined as:
Evaluating the similarity of two adjacent checkpoints, the one is used as the GT, and calculate the P@topK/bottomK
Larger values indicates better stability.

Relative P@top/bottom 5%



NB301 NB201

- Blue line: In later training stage, the ranking is more stable
- Score averaging of multiple supernets' scores (Orange) and temporal averaging of multiple checkpoints' weights (green) is beneficial for ranking stability

Menu

1. Analysis Framework
2. Evaluating One-shot Estimators (OSEs)
3. **Improving One-shot Estimators (OSEs)**
4. Evaluating Zero-shot Estimators (ZSEs)
5. Summary

Analyzing the problems of OSEs

- The random "sampling and training" scheme leads to temporal instability of supernet ranking (especially in the early stage)

Direction 1: Improving the stability

- Improper sampling leads to the bias of supernet ranking
  1. Some architectures (e.g. with larger complexity) need higher sampling probability to match their relative performances in standalone training.
  2. Architectures are sampled from an unfair distribution, i.e., some architectures have undesirable higher equivalent probabilities.

Direction 2: Improving the sampling fairness

- The underlying reason is parameter sharing of many architectures, whose desired weights might be very different

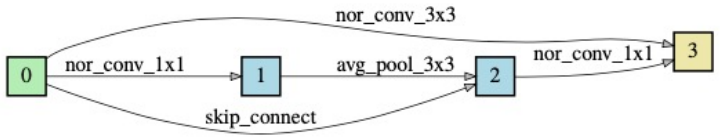
Direction 3: Reducing the sharing extent

Example of Direction 2: Improving the sampling fairness

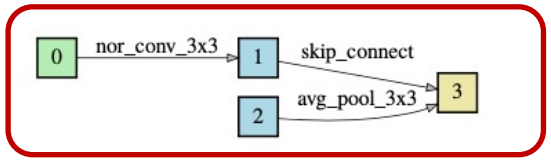
Deisomorphic sampling can improve the sampling fairness

Iso sampling

Rank 1: 92.55%, 1



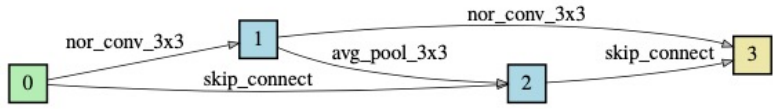
Rank 2: 91.96%, **31**



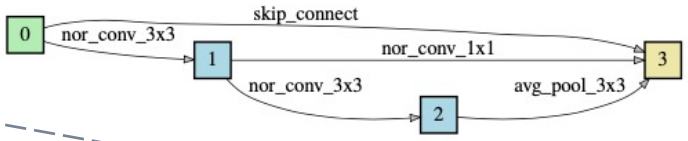
Vanilla sampling will overestimate simple architectures with many isomorphic counterparts (e.g., a degenerated arch with one single convolution, GT accuracy **91.96%**)

Deiso sampling

Rank 1: 93.43%, 1



Rank 2: 93.67%, **1**

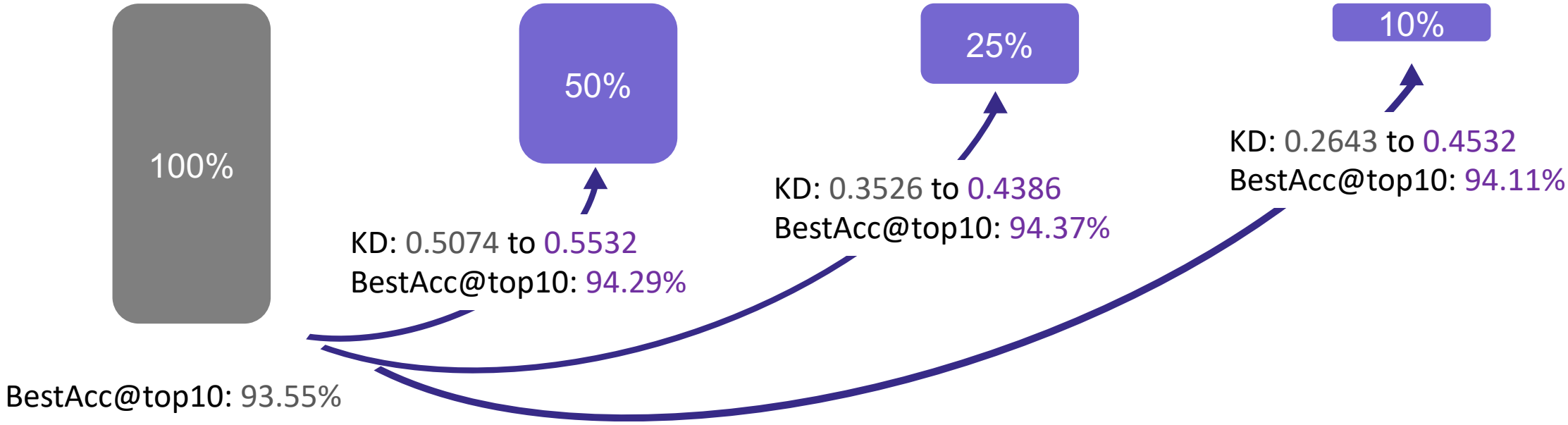


GT accuracy, **number of isomorphic counterparts**

Deisomorphic sampling can mitigate this improper overestimation, thus **improve the distinguishing ability of top architectures**, and help identify a better architecture with GT accuracy **93.67%**

Example of Direction 3: Reducing the sharing extent

One-shot SS pruning leads to higher ranking quality in the remaining SS



- The ranking quality on the remaining sub-SS significantly improves
- Help find better architectures
  - BestAcc@top10: Best ground-truth accuracy in top-10 architectures ranked by the OS score

Menu

1. Analysis Framework
2. Evaluating One-shot Estimators (OSEs)
3. Improving One-shot Estimators (OSEs)
4. **Evaluating Zero-shot Estimators (ZSEs)**
5. Conclusion

Evaluation of Parameter-level ZSEs

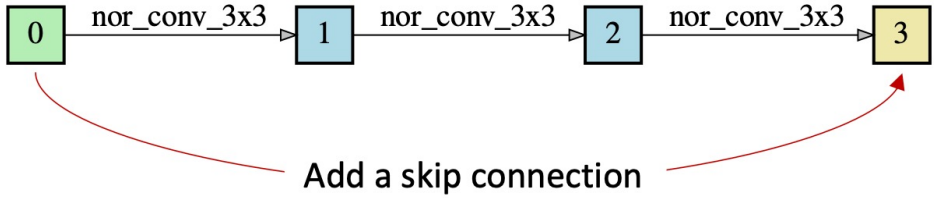
- [Abdelfattah, et al., ICLR 2021] add up parameter-wise sensitivity measures (One type of sensitivity measure corresponds to one type of ZSE)

$$plain: \mathcal{S}(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta; \quad snip: \mathcal{S}(\theta) = |\frac{\partial \mathcal{L}}{\partial \theta} \odot \theta|; \quad grasp: \mathcal{S}(\theta) = -(H\frac{\partial \mathcal{L}}{\partial \theta}) \odot \theta;$$

$$synflow: \mathcal{R} = \mathbb{1}^T(\prod_{\theta_i \in \theta} |\theta_i|)\mathbb{1}, \mathcal{S}(\theta) = \frac{\partial \mathcal{R}}{\partial \theta} \odot \theta; \quad fisher: \mathcal{S}(z) = \sum_{z_i \in z} (\frac{\partial \mathcal{L}}{\partial z} z)^2.$$

Observations

- ZSEs based on parameter-level analysis is not suitable for ranking architectures, their ranking qualities cannot even surpass those of using #Params or #FLOPs. For example, as for KD:
  - NB201: *synflow* 0.574 V.S. *plain* 0.311 V.S. #Param 0.606 V.S. One-shot 0.766
  - NB301: *synflow* 0.201 V.S. *plain* 0.394 V.S. #Param 0.515 V.S. One-shot 0.534
- Have clearly improper bias: e.g., prefer linear architecture without skip connection (gradient explosion)



snip/grad_norm/fisher scores: 3063/1922/454.8 -> 36.66/18.74/0.019

GT acc: 88.52%->93.99%

Evaluation of Architecture-level ZSEs

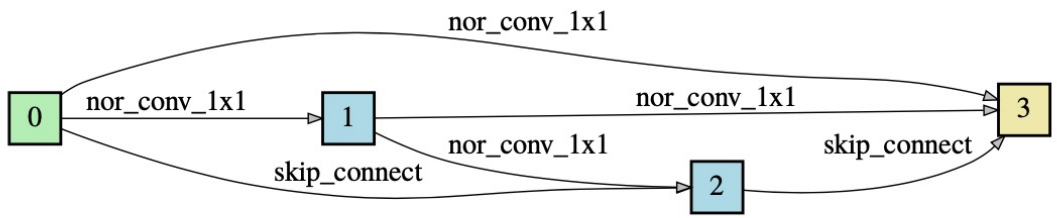
- ***jacob_cov*** [Mellor, et al., arXiv 2020] and ***relu_logdet*** [Mellor, et al., ICML 2021] measure the architecture's discriminability using the differences of input gradients and activations when inputting different images

Observations

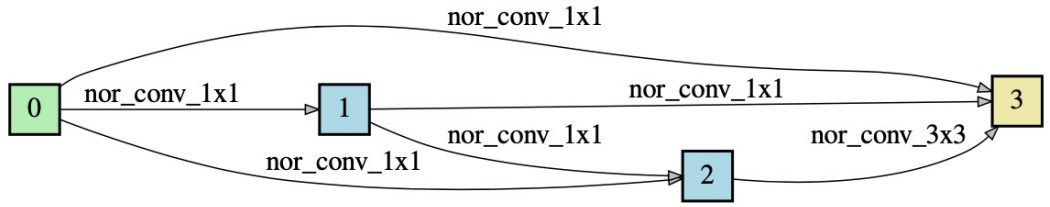
- Relatively good overall ranking correlations. But they have an improper preference for smaller kernel sizes, thus have **poor P@topKs**. For example, the P@topK comparison on NB201 is
  - *jacob_cov* 8.7% V.S. *relu_logdet* 15.8% V.S. #Param 28.2% V.S. One-shot 53.3%
- The best-performing ZSE is different on different search spaces. Generally speaking,
  - *relu_logdet* seems to behave best on topological search spaces.
  - *synflow* is better on the two non-topological search spaces.
- Most ZSEs are not sensitive to the input distribution (using random gaussian / uniform noises as the input results in very similar architecture ranking)

Top-1 ranked architecture

jacob_cov



relu_logdet



Menu

1. Analysis Framework
2. Evaluating One-shot Estimators (OSEs)
3. Improving One-shot Estimators (OSEs)
4. Evaluating Zero-shot Estimators (ZSEs)
5. **Summary**

Summary

- What does this work do and reveal?
  - An analysis framework and analyzing tools of multi-aspect properties of efficient performance estimators
    - Should be used in future development of efficient architecture performance estimators
  - Evaluation of OSEs' ranking quality
    - Give out strong baselines.
    - How the training sufficiency / validation data size influence the ranking quality? Longer training helps; With sufficient training, more validation data help.
  - Bias (complexity-level, op-level) and temporal variances of OSEs
    - In early training stages, the complexity-level bias and variance is large. Sufficient training can mitigate this.
    - Skip-connect is underestimated on NB201/NB301.
  - Evaluation of ZSEs' ranking quality
    - Comparison with OSEs and Param/FLOPs? A lot worse than OSE (especially P@topK), even worse than Param/FLOPs.
    - Does ZSEs benefit from OS training? No, ZSEs using high-order gradients get worse and worse.
    - Does ZSEs' results depend on the input data? The dependency of ZSEs' results on input data is small.
  - Bias of ZSEs (complexity-level, op-level, arch-level)
    - Clearly improper arch/op-level biases

Suggestions and Future Directions for OSEs

- **Longer training** makes one-shot estimations better
- **Using one-shot loss** instead of accuracy helps in **large search spaces with smaller inter-architecture differences**
- One should **use enough validation data for OSEs**, instead of merely several batches as in ZSEs
- **Using temporal ensemble** helps reduce the ranking instability, and brings non-negative improvements on the ranking quality in different search spaces
- In search space with isomorphic architectures, **augmenting the sampling strategy to improve the sampling fairness** is essential to avoid overestimating simple architectures. Using multiple MC sample does not seem to be beneficial on the benchmark search spaces. Affine operation should not be used in batch normalization (BN) during supernet training
- **Search space pruning based on one-shot scores** is beneficial for improving the ranking quality of OSE estimations, especially for good architectures.
- Picking channels according to L1 norm of weights is not necessary when searching for the channel decision (non-topological SSes). **Using the simplest ordinal picker is slightly better**

Suggestions and Future Directions for ZSEs

- relu_logdet is the best ZSE on topological SSes, its score is highly-correlated with the ReLU number (0.6 on NB101, 0.88 on NB201/NB301), and it is not very good on non-topological SSe**s,** where the only architectural decision that can influence the ReLU number is depth. **Is there a general ZSE for different search spaces?**
- ZSEs based on parameter-level sensitivity analysis is not reasonable and have clear bias. **Future ZSEs should conduct estimation using architectural-level analysis.**
- According to the prominent bias of existing ZSEs, **some structural knowledge can be incorporated into ZSE voting ensembles** (e.g., receptive field analysis seems promising for improving jacob_cov or relu_logdet).
- ZSEs have small variance w.r.t. different weight initialization & data distribution & data size. **Should we make ZSEs utilize the input data information better, and how?**
- Currently, the voting of different ZSEs does not bring improvements
- All ZSEs perform poorly on good architectures (Low P@topKs). **Can we develop ZSEs that can distinguish good architectures better?**
- In future development of ZSEs, researchers should **compare to #Params and #FLOPs**, since they are very competitive baseline “ZSEs”.

Thanks for listening!

Welcome to check our paper and code

https://github.com/walkerning/aw_nas

Discussions are welcome

- Xuefei Ning foxdoraame@gmail.com
- Yu Wang yu-wang@tsinghua.edu.cn