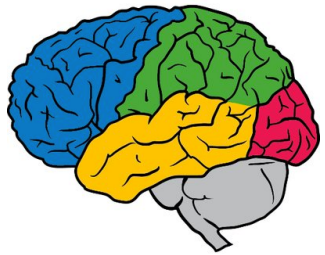
ECCV'20 ONLINE
23-28 AUGUST 2020

Neural Predictor for Neural Architecture Search

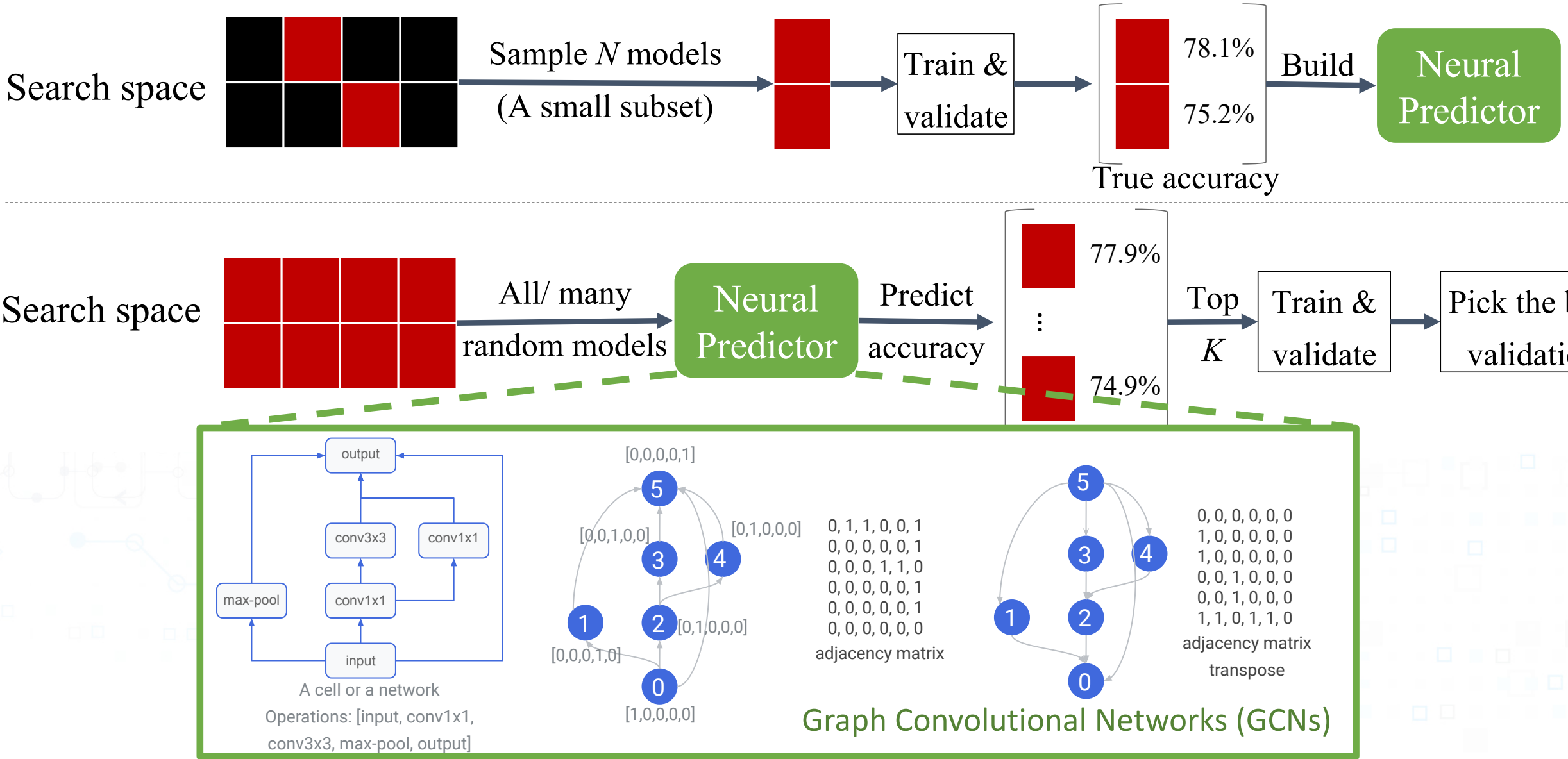
Wei Wen [1 2], Hanxiao Liu [1], Yiran Chen [2], Hai Li [2], Gabriel Bender [1], Pieter-Jan Kindermans [1]

[1] Google Brain, [2] Duke University

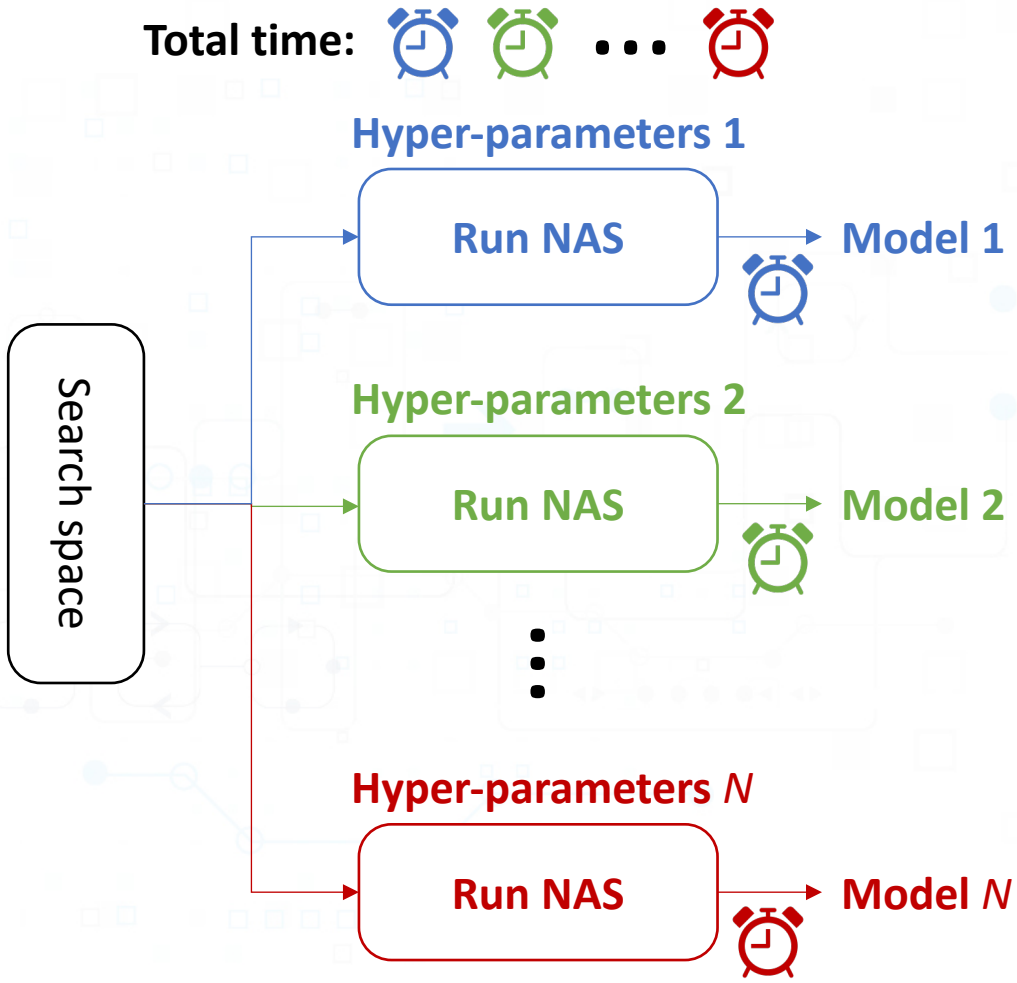


Duke
UNIVERSITY

Method Overview



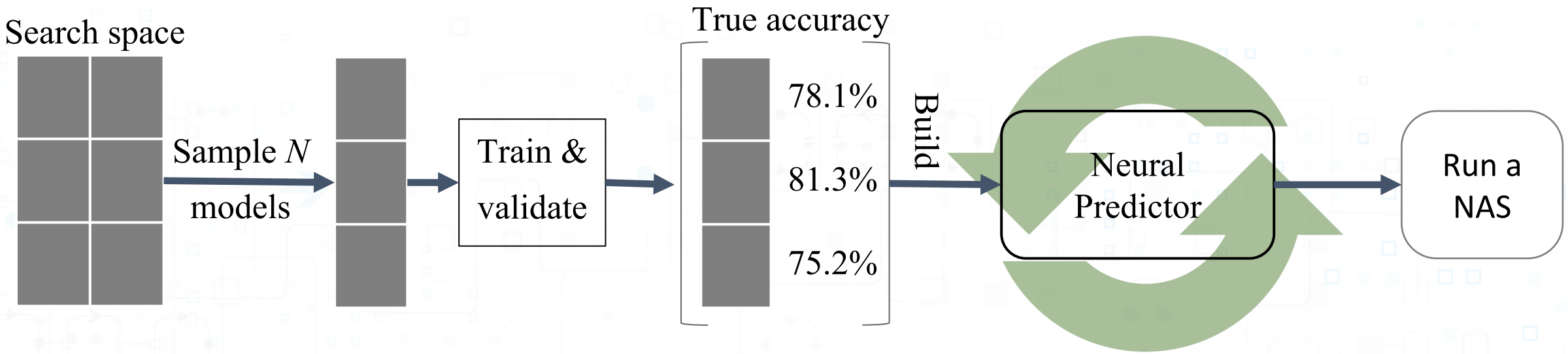
On hyper-parameter tuning of NAS methods



NAS Method	Hyper-parameters (HPs)
RL	Architecture and training HPs of an agent (e.g. an LSTM)
Evolution	sample size, mutation probabilities, etc
Bayesian Optimization	Priors (e.g. GP kernels) and acquisition functions
Weight Sharing	Training HPs of a super-model (e.g. DARTS, one-shot)

In previous works, search time was compared for a single search under the best evaluated HPs

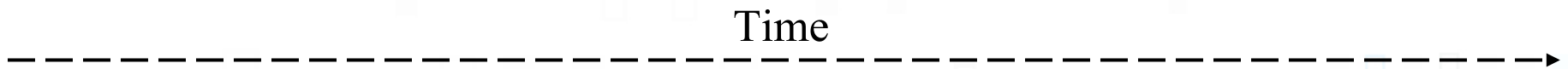
On hyper-parameter tuning of Neural Predictor



Hyper-parameter tuning without training new sets of models in the search space.

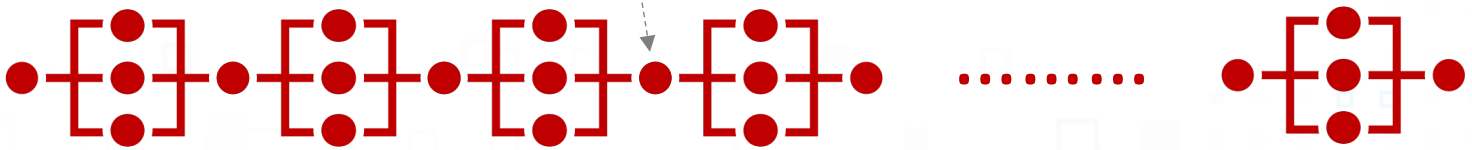
The only "hyper-parameter" that we might need to re-run the whole process is N (the number of models used to build a neural predictor). Discuss later.

On parallelizability of NAS methods



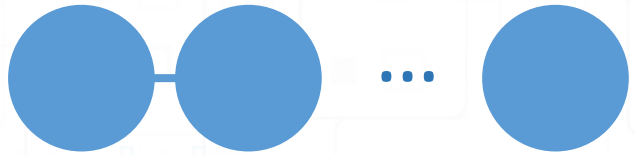
Synchronization of Reward/fitness/acquisition

NAS based on RL, Evolution or Bayesian Optimization

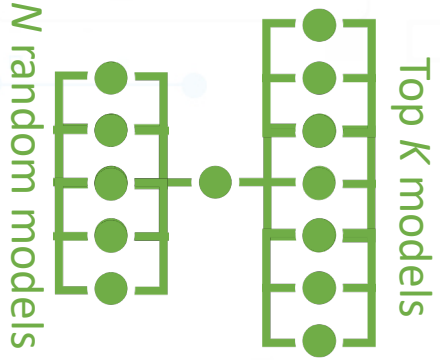


Train a super-model sequentially by an iterative optimizer (e.g. SGD)

Weight sharing (e.g. DARTS, one-shot)



Neural Predictor

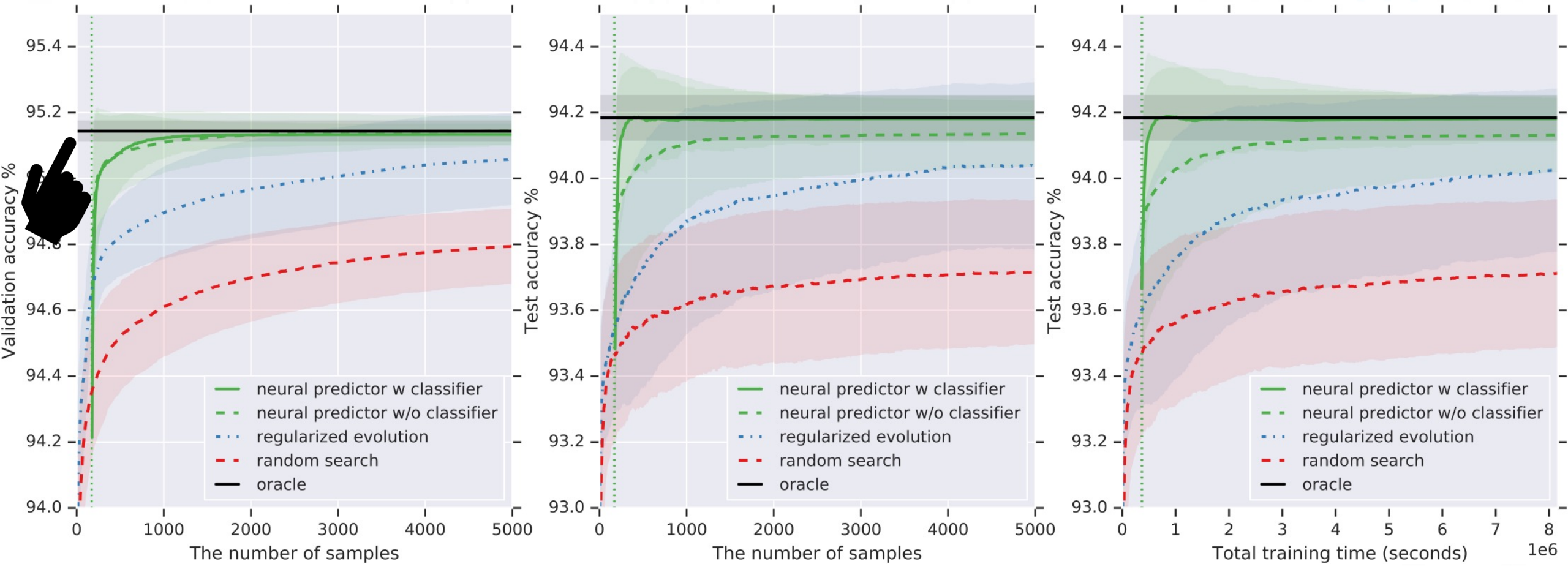


Comparison with other NAS methods

Method	Efficient	Hyper-parameter friendly	Fully parallelizable
Sampling- based (RL, evolution, Bayesian Optimization)	❌	❌	❌
Weight sharing	✅	❌	❌
Neural Predictor	✅	✅	✅

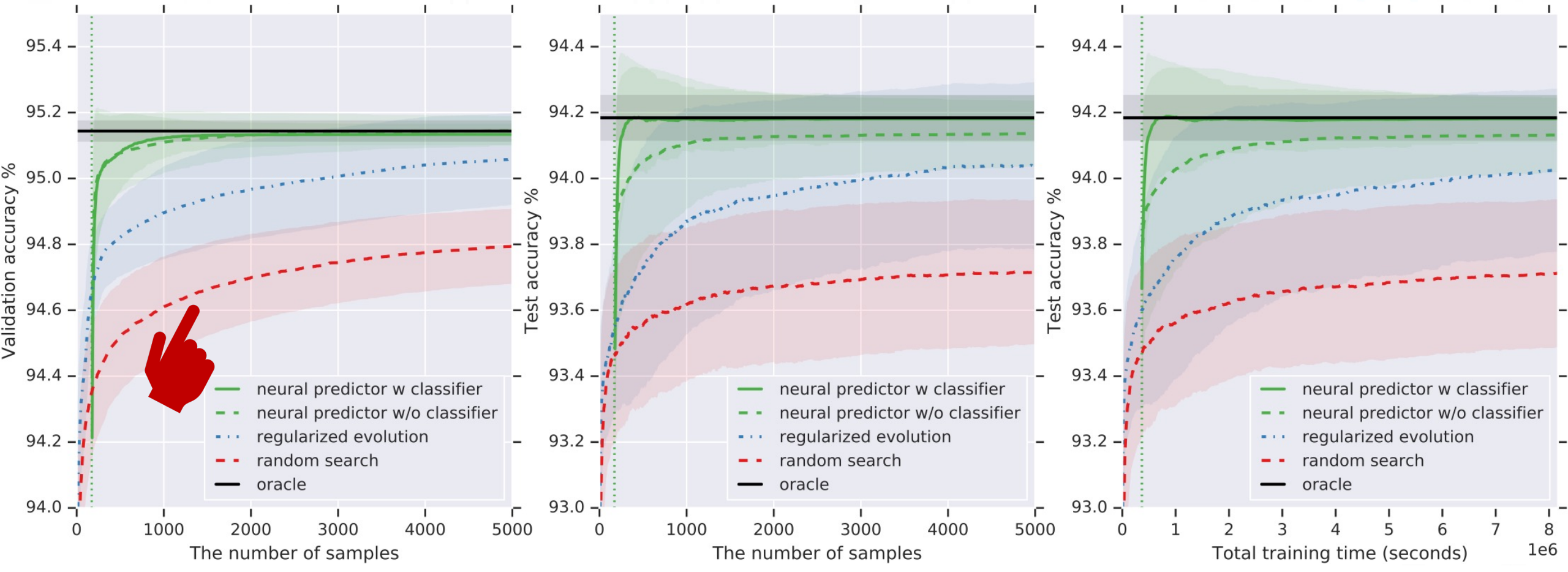
Results on NASBench-101

Oracle: an upper bound baseline, training all models and picking the best validation model.



Results on NASBench-101

Oracle: an upper bound baseline, training all models and picking the best validation model.
Random search: a lower bound baseline, training *M* random models and picking the best validation model.

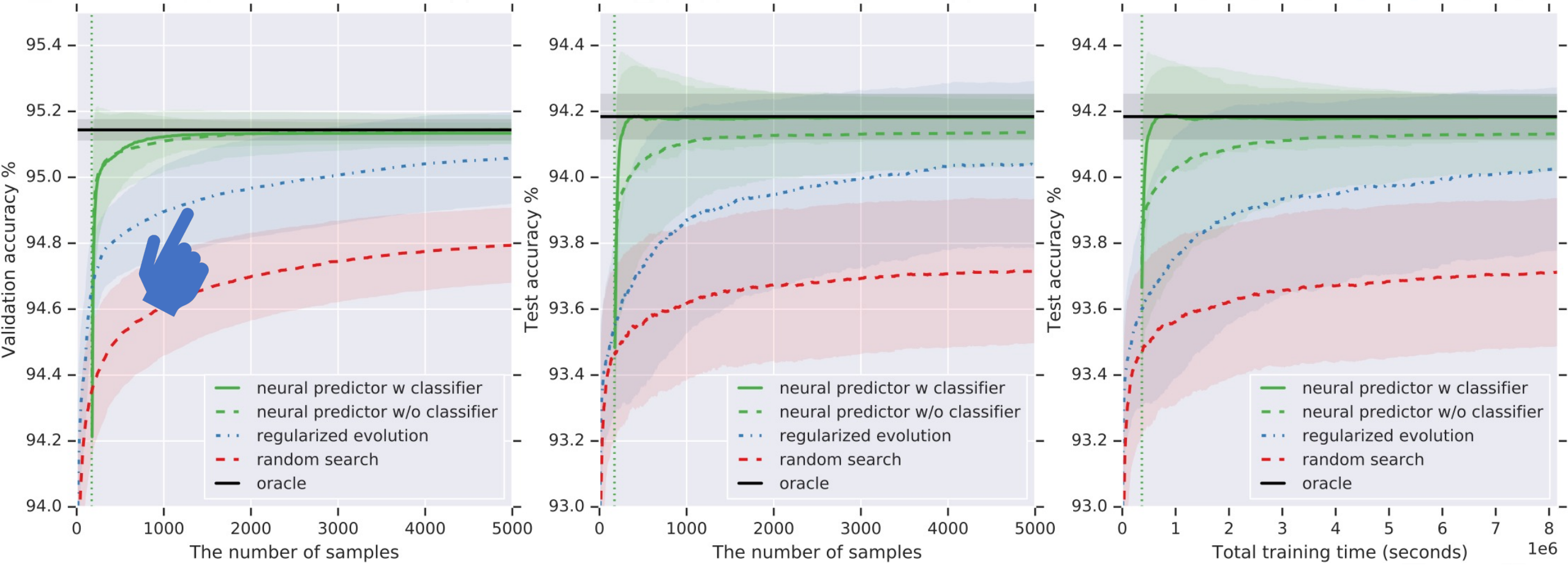


Results on NASBench-101

Oracle: an upper bound baseline, training all models and picking the best validation model.

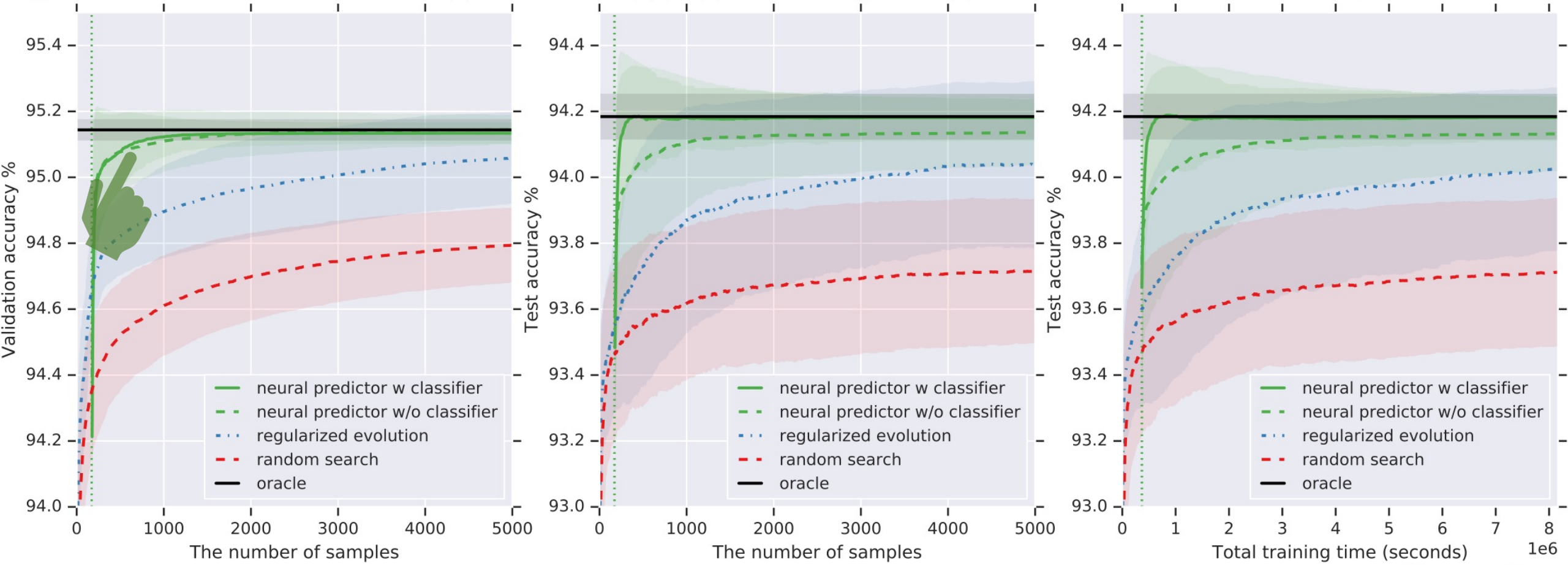
Random search: a lower bound baseline, training *M* random models and picking the best validation model.

Regularized evolution: a state-of-the-art baseline. The best method in the original NASBench-101 paper.

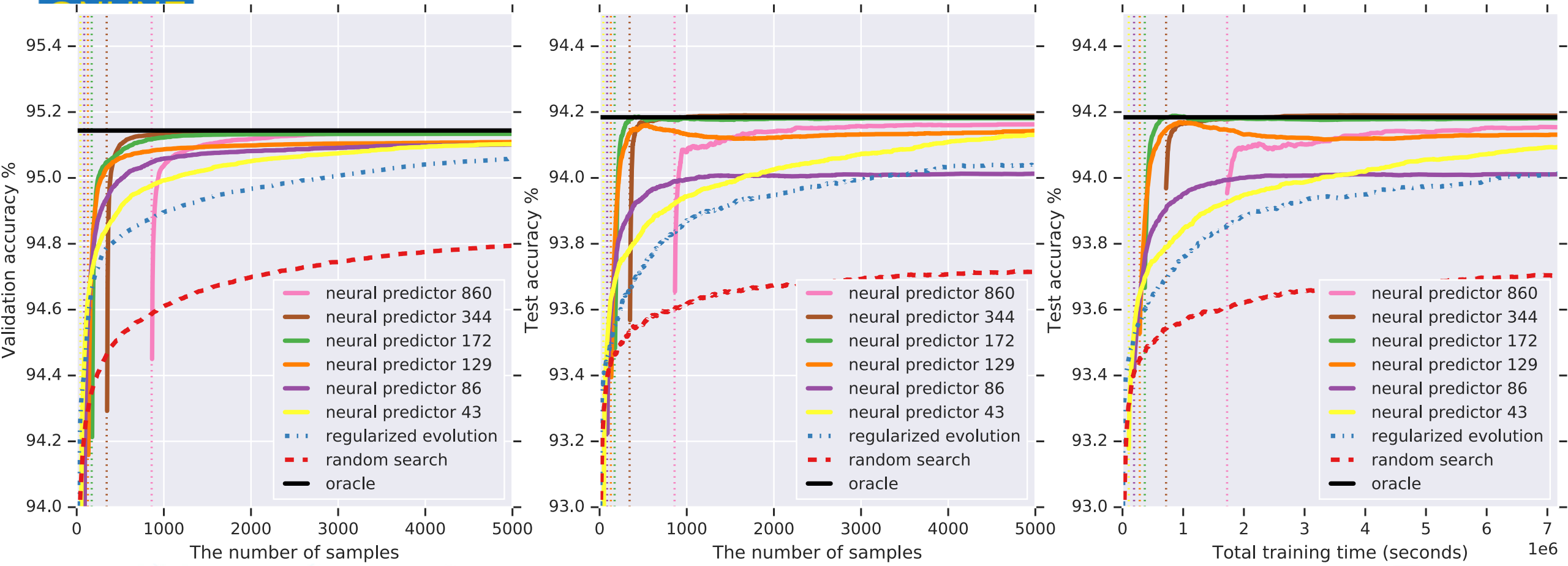


Results on NASBench-101

Oracle: an upper bound baseline, training all models and picking the best validation model.
Random search: a lower bound baseline, training *M* random models and picking the best validation model.
Regularized evolution: a state-of-the-art baseline. The best method in the original NASBench-101 paper.
Neural predictor: **>20x** as sample efficient as Regularized Evolution (*N*=172).



The robust hyperparameter *N*

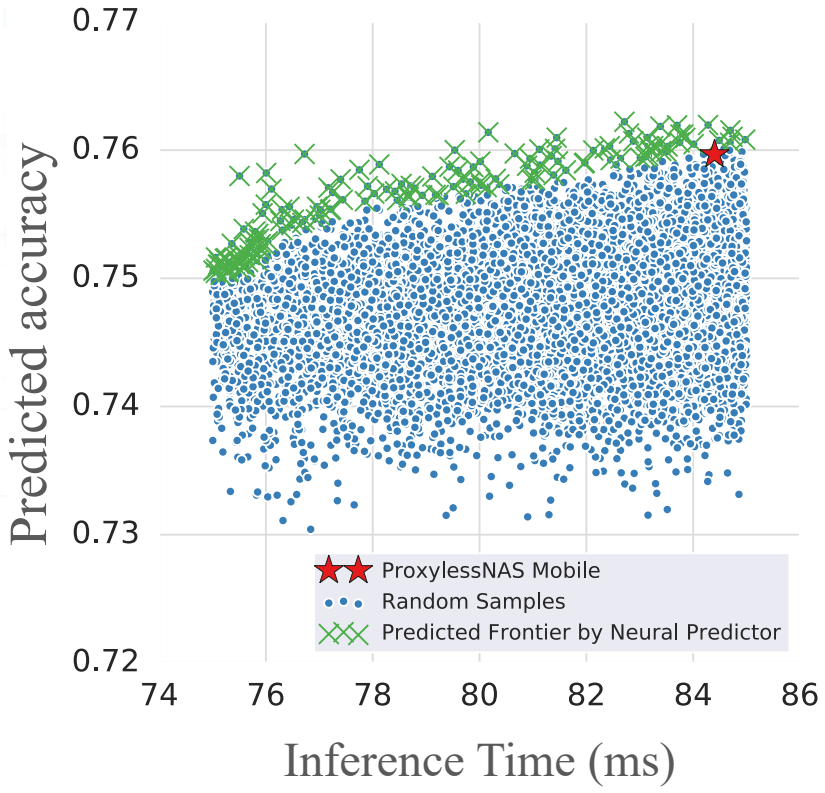


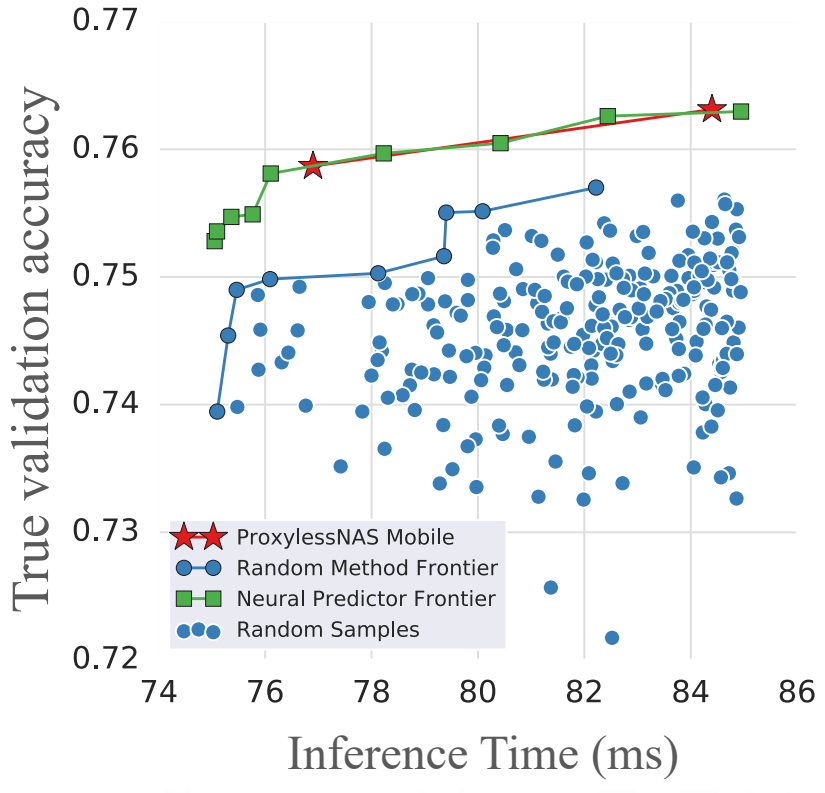
Given a total number of models (*N*+*K*) that we can train, what's an optimal *N* (the dataset size building the predictor)?

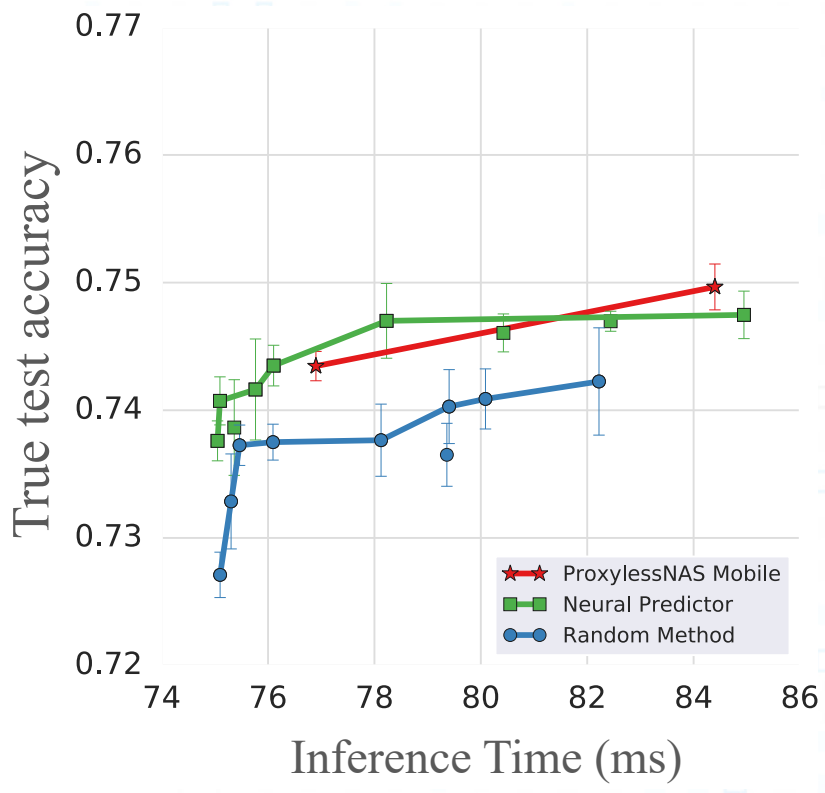
- If *N* is too small, the predictor is not accurate;
- If *N* is too large, a small top-*K* models can be trained, which cannot tolerate prediction noise in the predictor.
- However, all *N*s outperforms regularized evolution.

Results on ImageNet

- Using the ***same search space*** of ProxylessNAS (for fast mobile models)
- Our result comes from the ***single (first)*** search: no tuning N, just set *N*=120 beforehand
- Accuracy is on-par but ours is fully parallelizable and hyper-parameter friendly







Conclusion and takeaways

- Neural Predictor is
  - Simple
  - Sample efficient
  - Friendly to hyper-parameter tuning
  - Full parallelizable.
- Using Graph Convolutional Networks as the predictor is more effective (check ablation study in the paper)
- NAS reproducibility and fair comparison rules that we followed and the community should follow
  1. Search by validation accuracy and use test accuracy for final report only
  2. Use the same search space for comparison
  3. Comparing the search time of a single run is NOT sufficient. The cost of hyper-parameter tuning should be discussed


Thanks!