How Powerful are Performance Predictors in Neural Architecture Search?

Colin White
Abacus.AI

Arber Zela
University of Freiburg

Binxin Ru
Oxford University

Yang Liu
Abacus.AI

Frank Hutter
University of Freiburg
Bosch Center for AI

Slides (with hyperlinks): https://crwhite.ml
Performance prediction techniques

- Early NAS algs required fully training 1000s of architectures [Zoph and Le 2016]
- Recent algs use techniques to predict the final performance of architectures
Performance Predictors

A performance predictor is any technique which predicts the final accuracy or ranking of architectures, without fully training them.

- **Initialization**: performs any necessary pre-computation
- **Query**: take any architecture as input, and output predicted accuracy
- **(Update)**: similar to initialization
Outline

- Motivation
- Introduction to Performance Predictors
  - Model-based predictors
  - Learning curve based predictors
  - Zero-cost predictors
  - Weight sharing
- Experiments: 31 performance predictors
  - Stand-alone predictor experiments
  - OMNI
  - NAS experiments
- Conclusion
Model-Based Predictors

- Supervised learning - regression
  - \( X \) - the architecture encoding (e.g. one-hot adjacency matrix)
  - \( Y \) - validation accuracy of trained architecture

- Gaussian processes [Kandasamy et al. 2018], [Jin et al. 2018]
- Boosted trees [Luo et al. 2020], [Siems et al. 2020]
- GNNs [Shi et al. 2019], [Wen et al. 2019]
- Specialized encodings [White et al. 2019], [Ning et al. 2020]

High init time, low query time
Learning curve based predictors

- Learning curve extrapolation
  - Fit partial learning curve to parametric model [Domhan et al. 2015]
  - Bayesian NN [Klein et al. 2017]
- Training statistics
  - Early stopping (val acc) [Elsken et al. 2018]
  - Sum of training losses [Ru et al. 2020]

No init time, high query time
Hybrid model-based + LC predictors

Train a model, using partial learning curve + hyperparams, to predict final accuracy

- First and second derivatives as features, SVR [Baker et al. 2017]
- Full LC as features, Bayesian NN [Klein et al. 2017]

High init time, high query time
“Zero-cost” proxies

Compute a statistic of an architecture in 3-5 seconds

- Jacobian covariance [Mellor et al. 2020]
- Synaptic Flow [Abdelfattah et al. 2021]
  - SNIP [Lee et al. 2018]

Low init time, low query time

\[
\text{snip: } S_p(\theta) = \left| \frac{\partial L}{\partial \theta} \right| \odot \theta, \quad \text{grasp: } S_p(\theta) = -\left( H \frac{\partial L}{\partial \theta} \right) \odot \theta, \quad \text{synflow: } S_p(\theta) = \frac{\partial L}{\partial \theta} \odot \theta
\]
Weight Sharing

Train a set of shared weights that can be used by all architectures (the Supernetwork)

- OneShot [Bender et al. 2018]
- Random Search WS [Li & Talwalkar 2019]

Medium init time, low query time
Outline

- Motivation
- Introduction to Performance Predictors
  - Model-based predictors
  - Learning curve based predictors
  - Zero-cost predictors
  - Weight sharing
- Experiments: 31 performance predictors
  - Stand-alone predictor experiments
  - OMNI
  - NAS experiments
- Conclusion
Notes on experiments

● Three axes of comparison: initialization time, query time, correlation / rank correlation metrics
● Official implementation whenever possible
● Train/test data drawn u.a.r.
● Light hyperparameter tuning
  ○ Levels the playing field
  ○ Cross-validation is often used during NAS
Kendall Tau on NAS-Bench-201 CIFAR-10

- Query time (seconds)
- Init. time (seconds)

NAS-Bench-201 CIFAR-100

- Query time (seconds)
- Init. time (seconds)

NAS-Bench-201 ImageNet16-120

- Query time (seconds)
- Init. time (seconds)

NAS-Bench-101

- Query time (seconds)
- Init. time (seconds)

DARTS

- Query time (seconds)
- Init. time (seconds)

Legend:
- BANANAS
- Early Stop (Acc.)
- Jacob. Cov.
- LcSVR
- SoTL-E
- SynFlow
- GCN
- LGBboost
- NGBoost
- SemiNAS
- XGBoost
NAS-Bench-101: a more complex search space

- Path encoding performs very well
Mutation-based train/test sets

- Model-based predictors perform worse. Trees are comparatively better.
OMNI: The Omnipotent Predictor

- Combine best predictors from three families: SoTL + Jacob. Cov + NGBoost
- Consistent performance almost everywhere
- 20% improvement in most-competitive bottom row
OMNI Ablation

- Jacob. Cov + SoTL-E is consistent
- NGBoost needed for top performance in lower middle/right
NAS Experiments

Evol. NAS Framework, NAS-Bench-201 CIFAR10

Validation error (%)

Runtime (seconds)

BO Framework, NAS-Bench-201 CIFAR10

Validation error (%)

Runtime (seconds)

Evol. NAS Framework, NAS-Bench-201 ImageNet16-120

Validation error (%)

Runtime (seconds)

BO Framework, NAS-Bench-201 ImageNet16-120

Validation error (%)

Runtime (seconds)

- BANANAS
- BONAS
- BOHAMIAN
- DNGO
- MLP
- LGBStoost
- GCN
- GP
- NAO
- RF
- Sparse GP
- Var. Sparse GP
- XGBoost
- NGBoost
- OMNN(NGBoost)
- SemiNAS
- OMNN(SemiNAS)
So... How powerful are performance predictors?

- Largely the same trends across all experiments
- Combining predictors works the best
- Complex search spaces: specialized encodings (e.g. path encoding)
Conclusions & Future Work

● First large-scale study of performance predictors
● Four families, 31 total performance predictors
● OMNI achieves the best performance

Future work

● Zero-cost predictors that work on larger search spaces
● More sophisticated combinations of predictors + integration in NAS

Code: https://github.com/automl/NASLib

Full paper: https://arxiv.org/abs/2104.01177

Thanks!