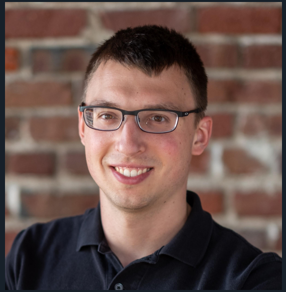
How Powerful are Performance Predictors in Neural Architecture Search?

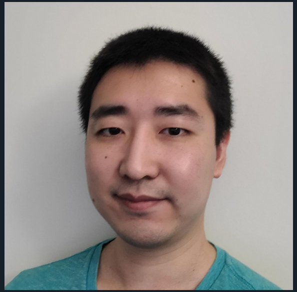
Colin White
Abacus.AI



Arber Zela
University of Freiburg

Binxin Ru
Oxford University

Yang Liu
Abacus.AI

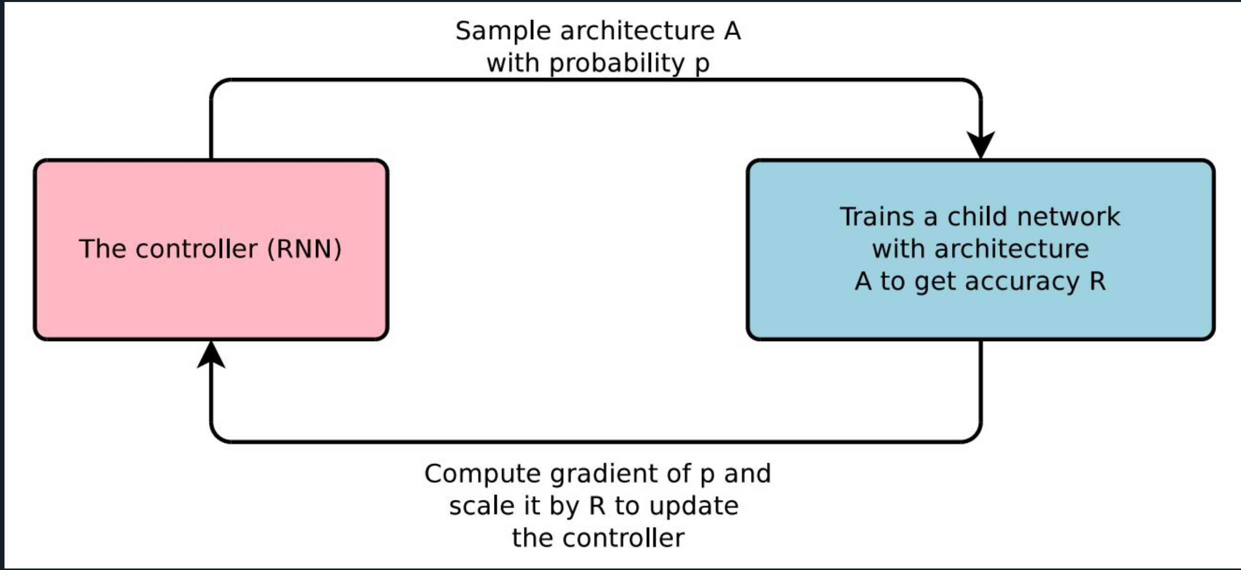


Frank Hutter
University of Freiburg
Bosch Center for AI

Slides (with hyperlinks): https://crwhite.ml

Performance prediction techniques

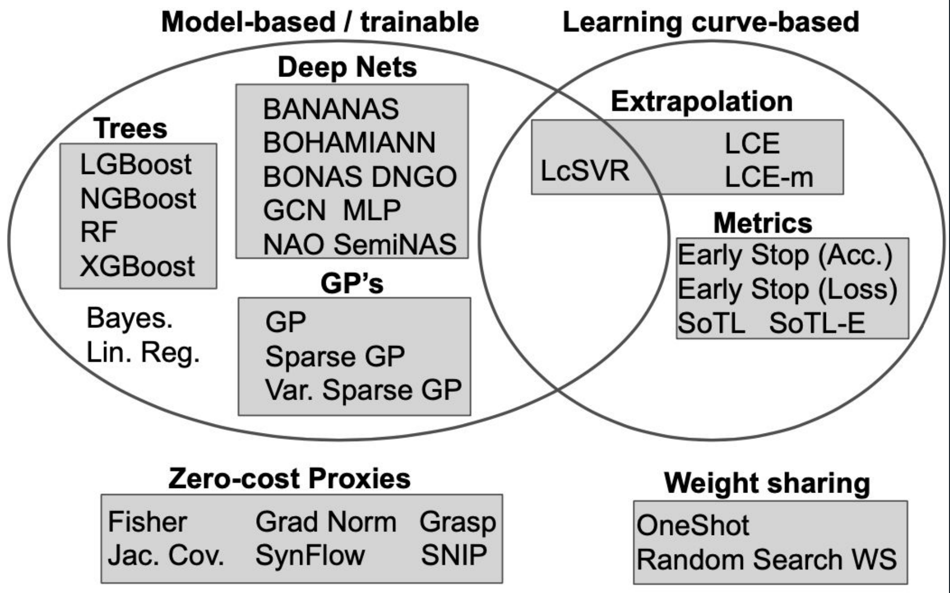
- Early NAS algos required fully training 1000s of architectures [Zoph and Le 2016]
- Recent algos use techniques to predict the final performance of architectures



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)



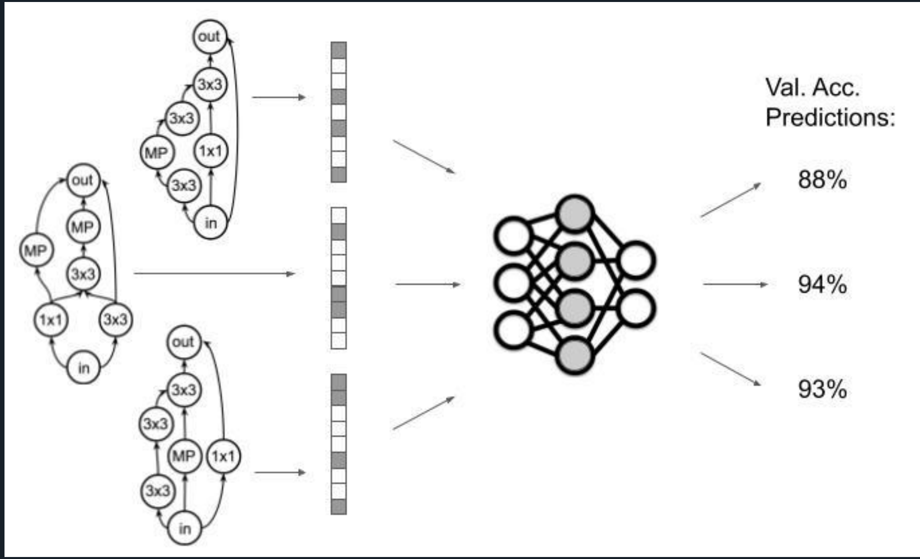
Outline

- 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

ABACUS.AI

Model-Based Predictors

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



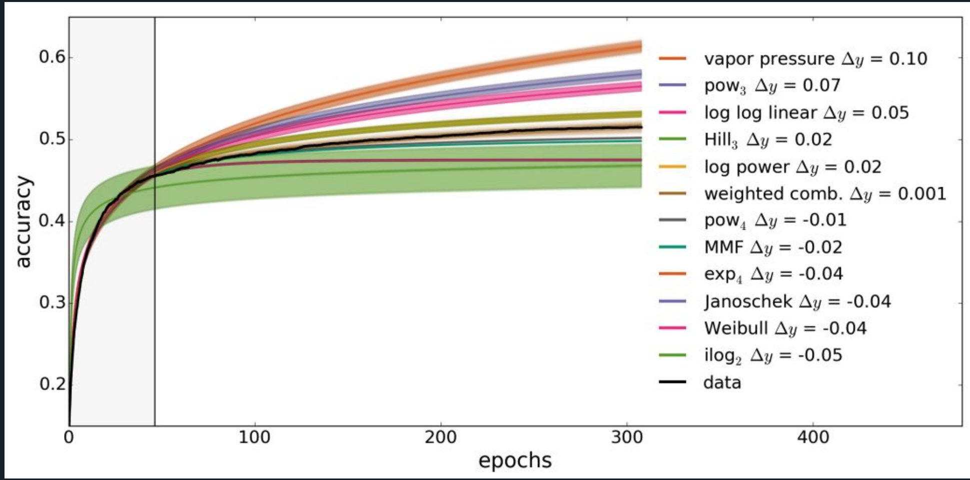
[White et al. 2019]

- 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]



[Elsken et al. 2018]

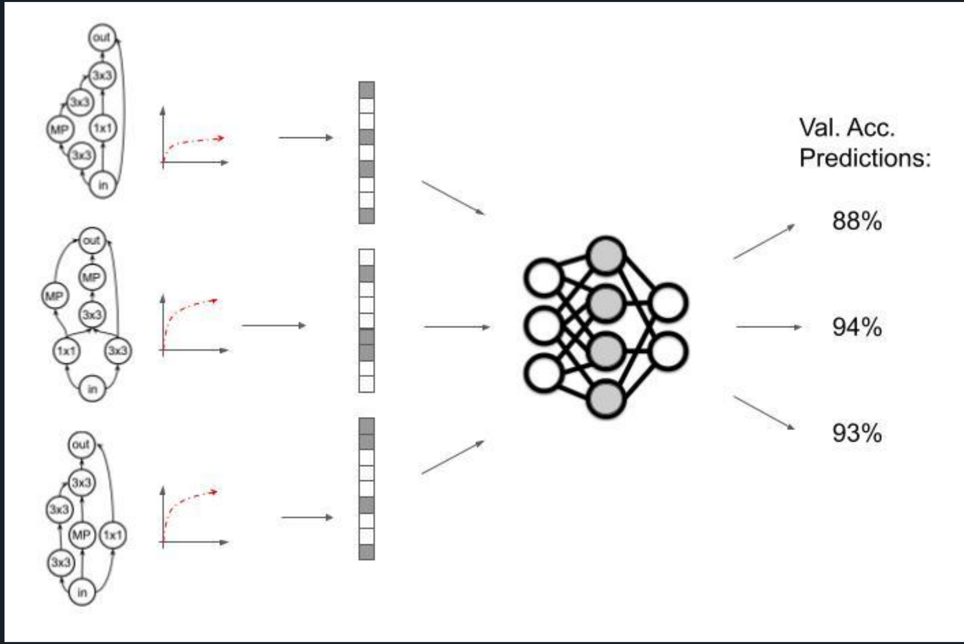
No init time, **high query time**

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

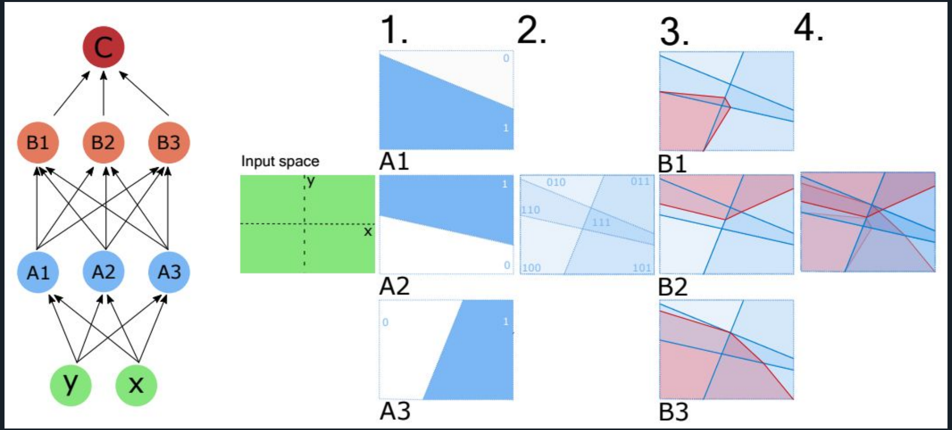


“Zero-cost” proxies

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



[Mellor et al. 2020]

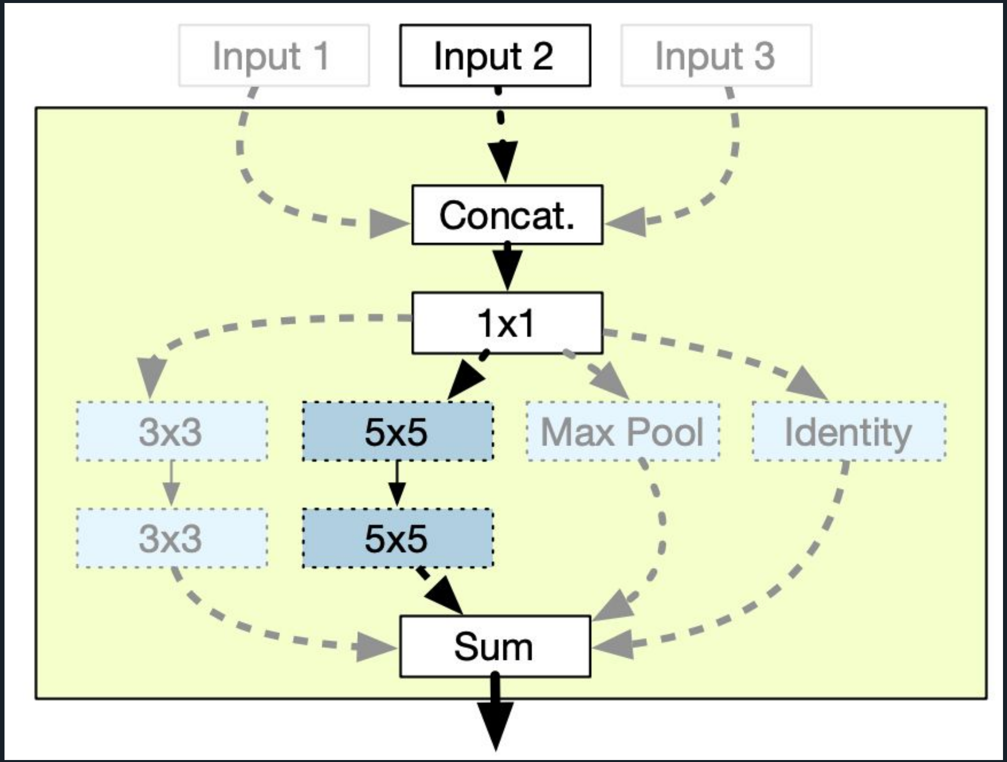
[Abdelfattah et al. 2021]

$$\texttt{snip} : \mathcal{S}_p(\theta) = \left| \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right|, \qquad \texttt{grasp} : \mathcal{S}_p(\theta) = -(H \frac{\partial \mathcal{L}}{\partial \theta}) \odot \theta, \qquad \texttt{synflow} : \mathcal{S}_p(\theta) = \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta$$

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]



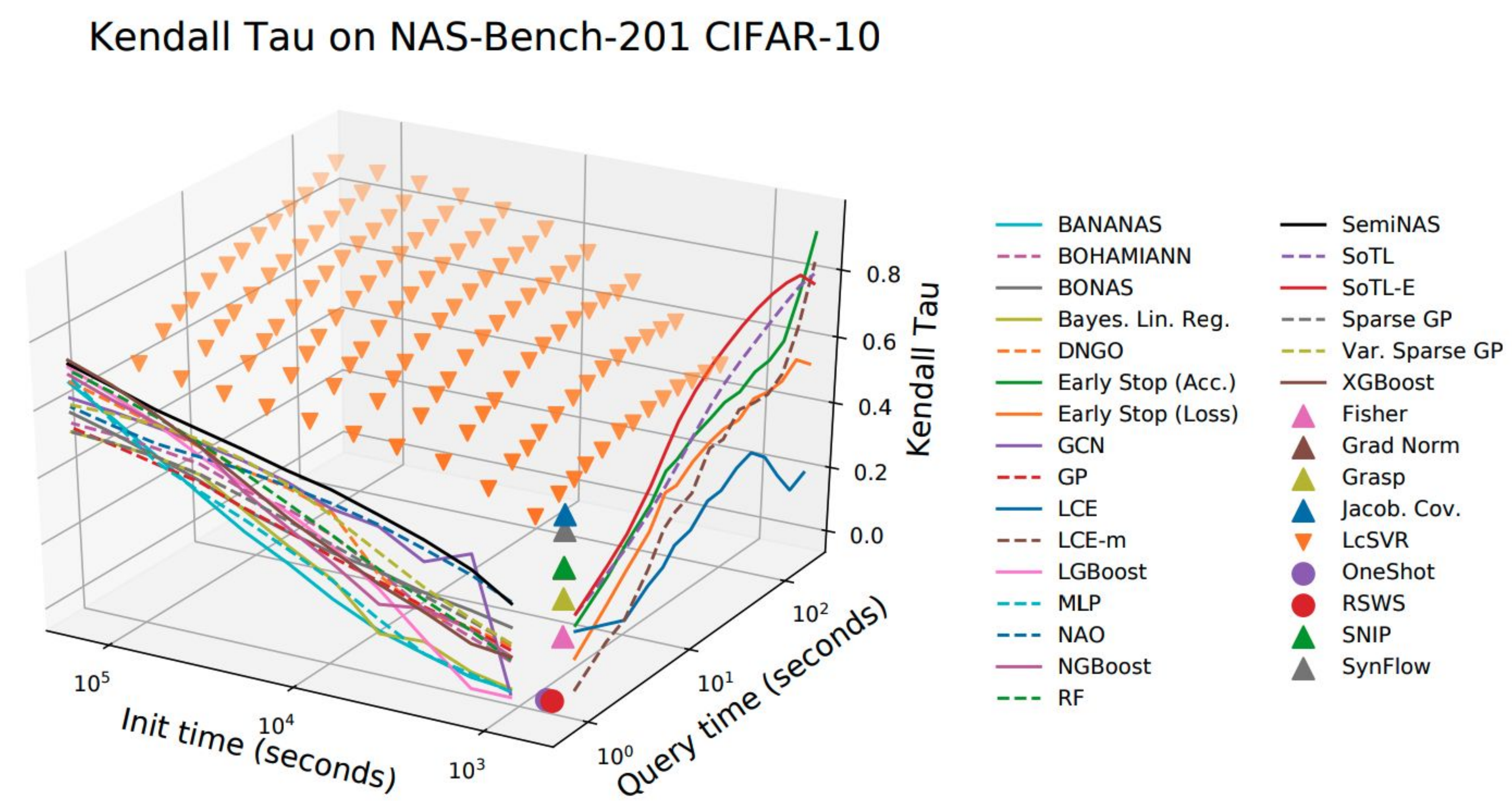
[Bender et al. 2018]

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

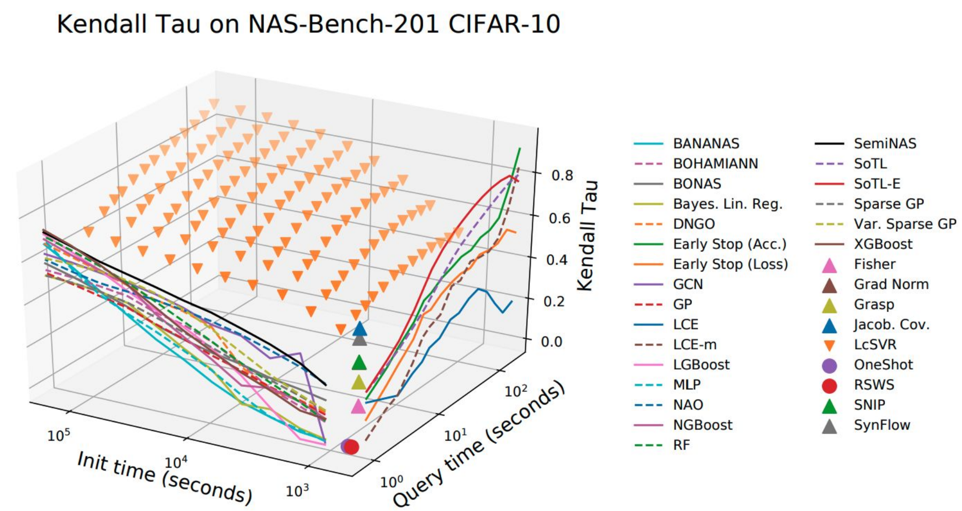
ABACUS.AI

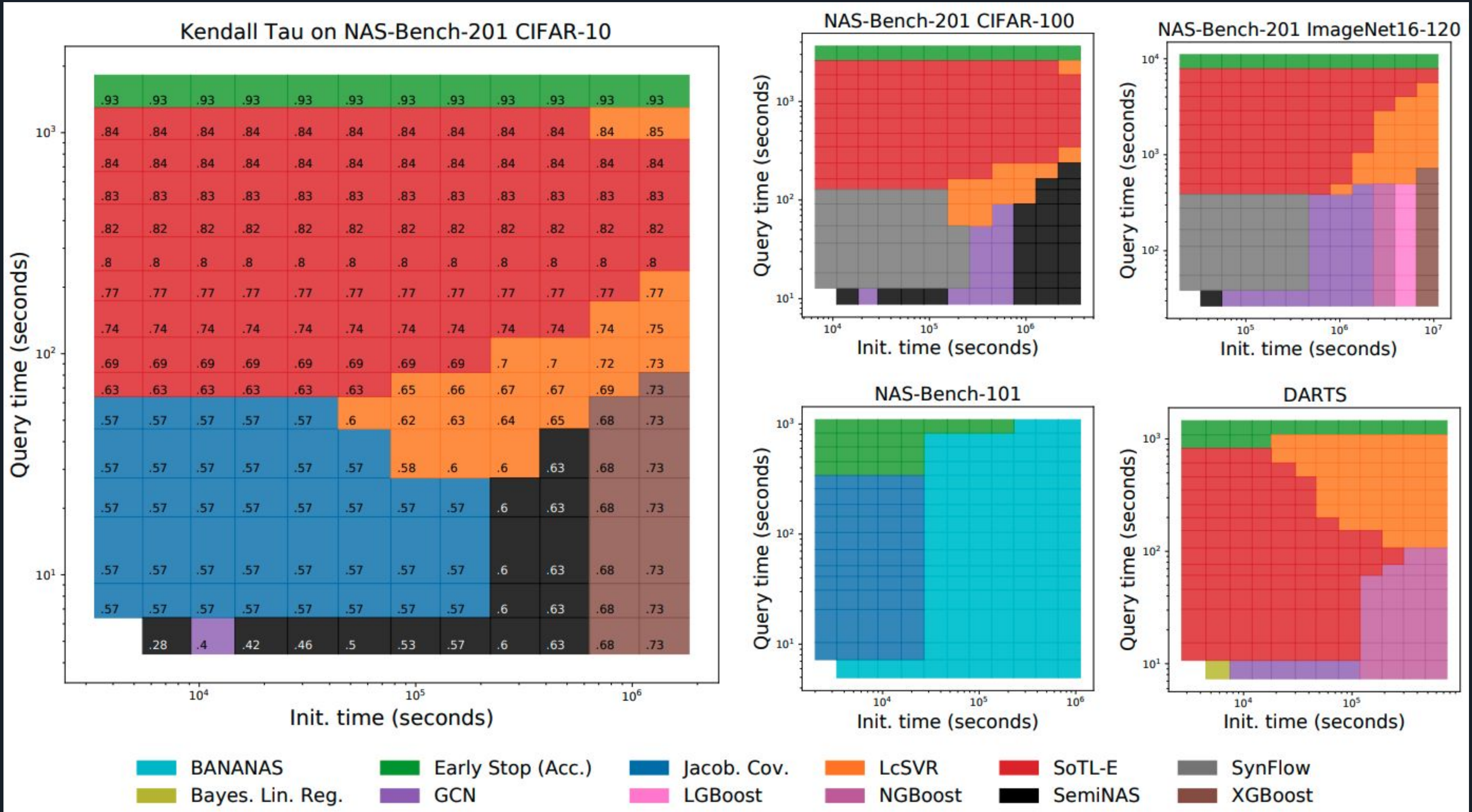


Kendall Tau on NAS-Bench-201 CIFAR-10
Kendall Tau
0.0
0.2
0.4
0.6
0.8
Init time (seconds)
10^5
10^4
10^3
Query time (seconds)
10^0
10^1
10^2
BANANAS
BOHAMIANN
BONAS
Bayes. Lin. Reg.
DNGO
Early Stop (Acc.)
Early Stop (Loss)
GCN
GP
LCE
LCE-m
LGBoost
MLP
NAO
NGBoost
RF
SemiNAS
SoTL
SoTL-E
Sparse GP
Var. Sparse GP
XGBoost
Fisher
Grad Norm
Grasp
Jacob. Cov.
LcSVR
OneShot
RSWS
SNIP
SynFlow

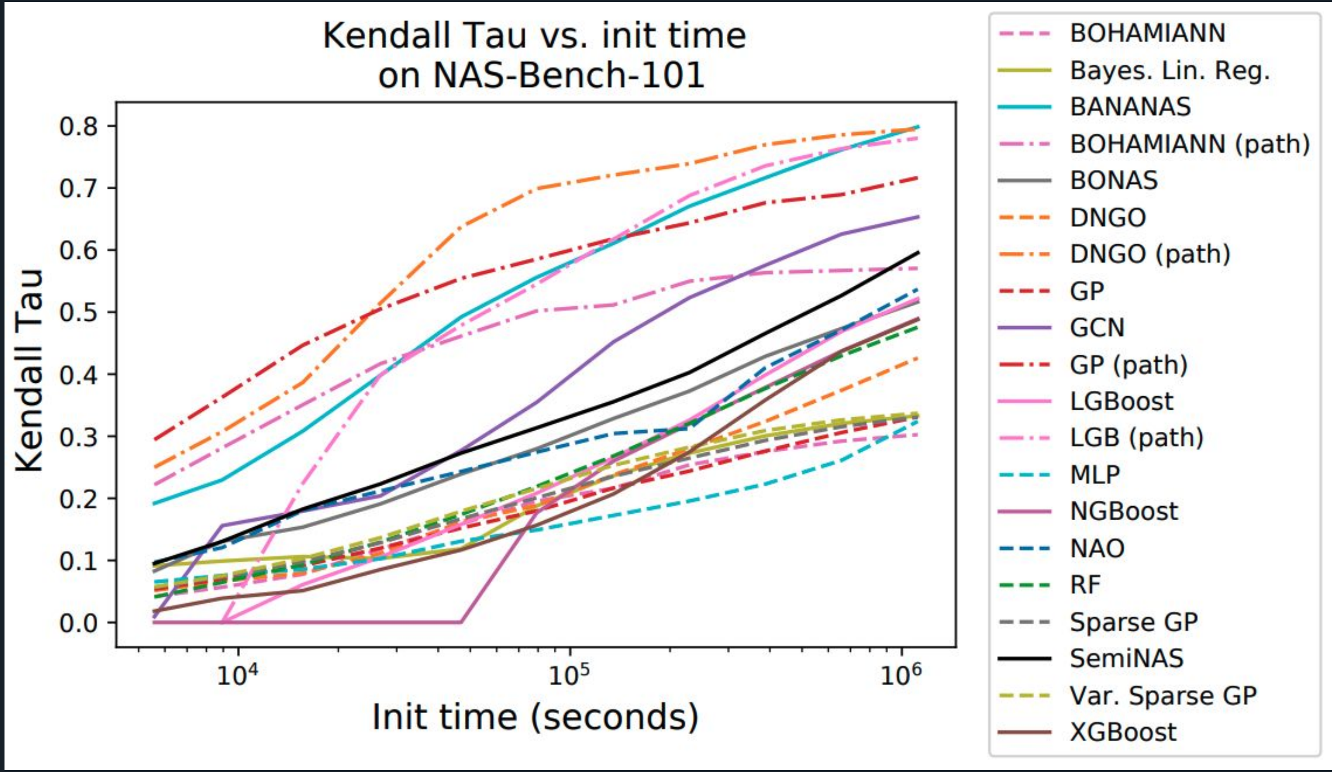
Notes on experiments

- 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

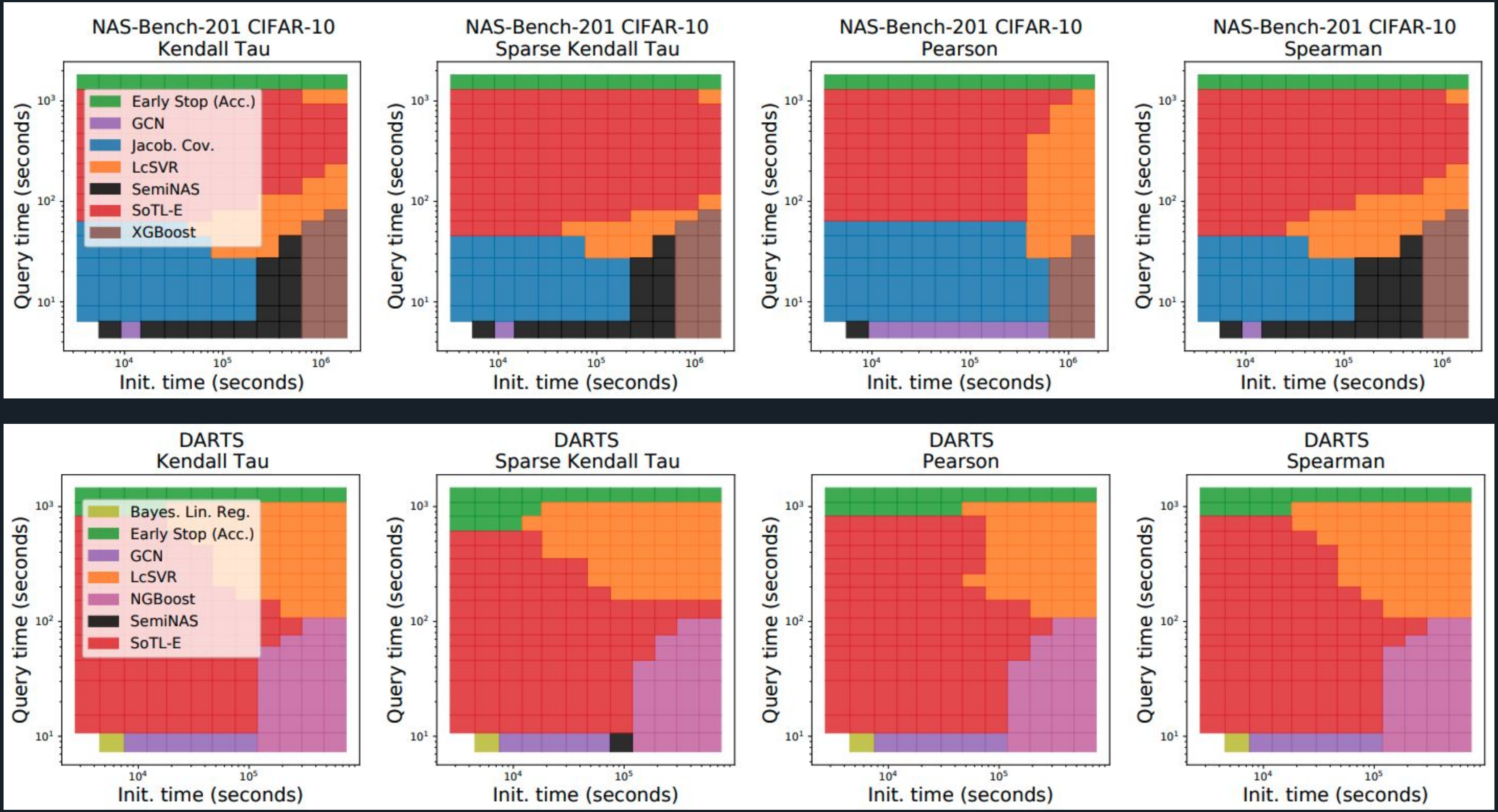



Kendall Tau on NAS-Bench-201 CIFAR-10
NAS-Bench-201 CIFAR-100
NAS-Bench-201 ImageNet16-120
NAS-Bench-101
DARTS
Query time (seconds)
Init. time (seconds)
BANANAS
Bayes. Lin. Reg.
Early Stop (Acc.)
GCN
Jacob. Cov.
LGBoost
LcSVR
NGBoost
SoTL-E
SemiNAS
SynFlow
XGBoost

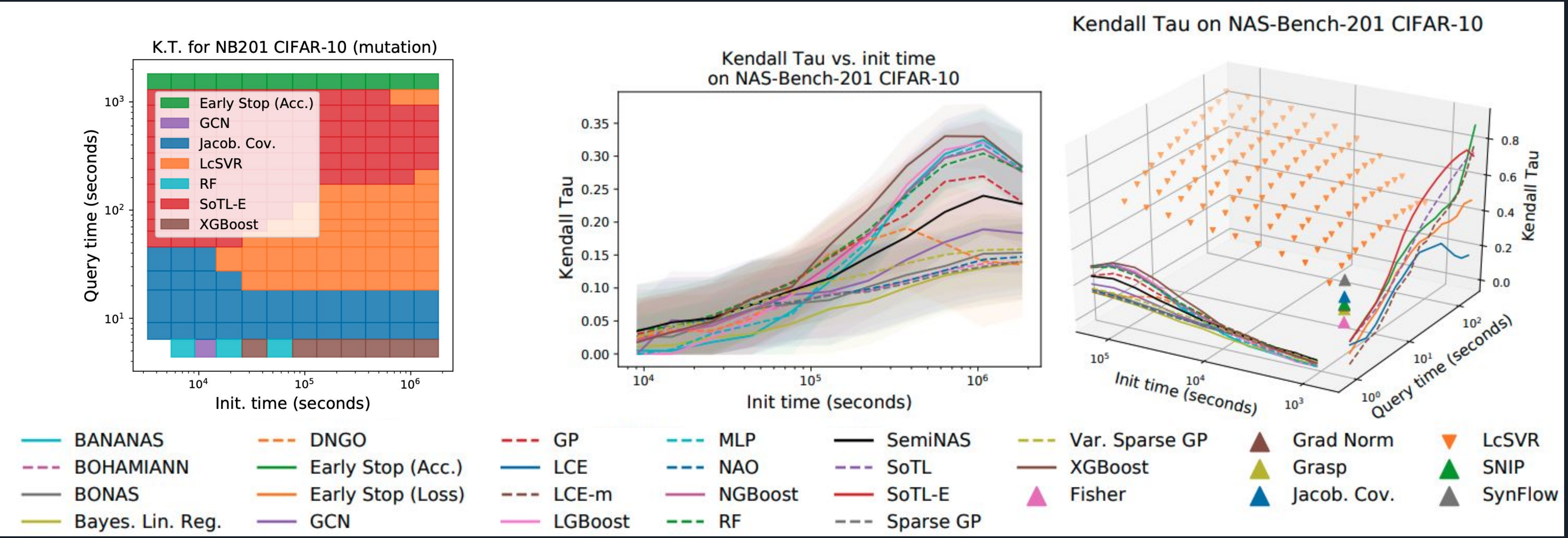
NAS-Bench-101: a more complex search space



- Path encoding performs very well


NAS-Bench-201 CIFAR-10
Kendall Tau
NAS-Bench-201 CIFAR-10
Sparse Kendall Tau
NAS-Bench-201 CIFAR-10
Pearson
NAS-Bench-201 CIFAR-10
Spearman
Early Stop (Acc.)
GCN
Jacob. Cov.
LcSVR
SemiNAS
SoTL-E
XGBoost
Query time (seconds)
Init. time (seconds)
DARTS
Kendall Tau
DARTS
Sparse Kendall Tau
DARTS
Pearson
DARTS
Spearman
Bayes. Lin. Reg.
Early Stop (Acc.)
GCN
LcSVR
NGBoost
SemiNAS
SoTL-E
Query time (seconds)
Init. time (seconds)

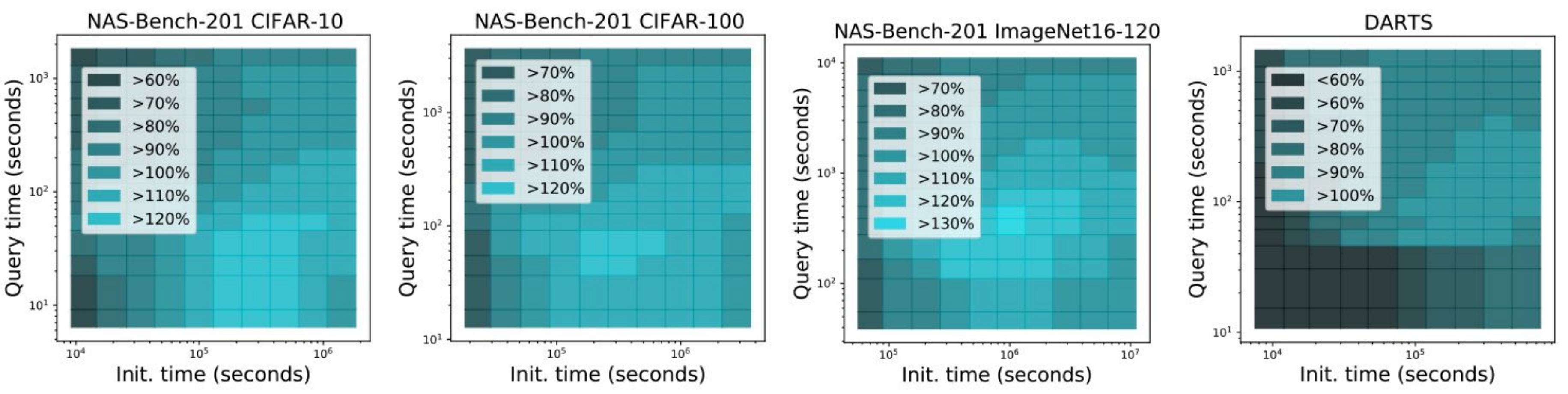
Mutation-based train/test sets



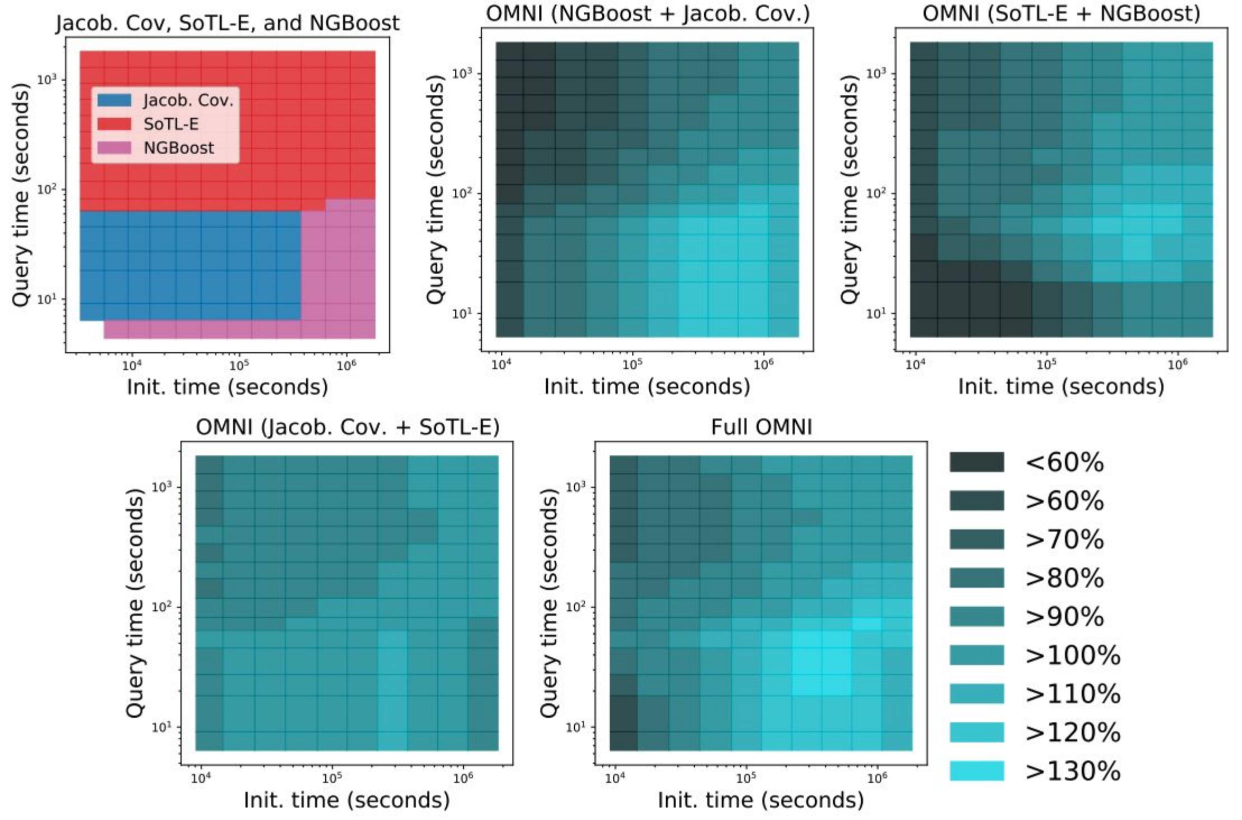
- Model-based predictors perform worse. Trees are comparatively better

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

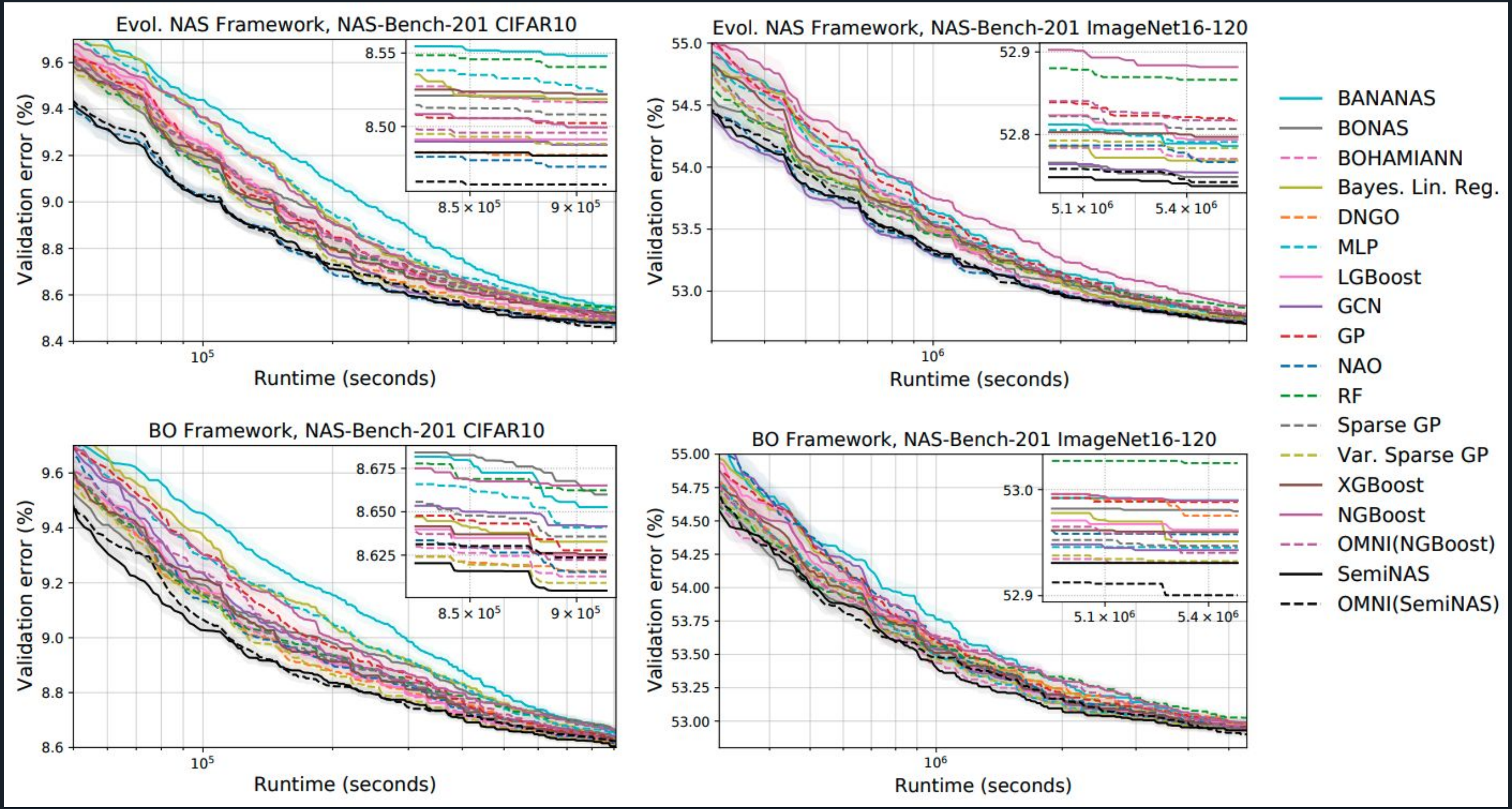


OMNI Ablation



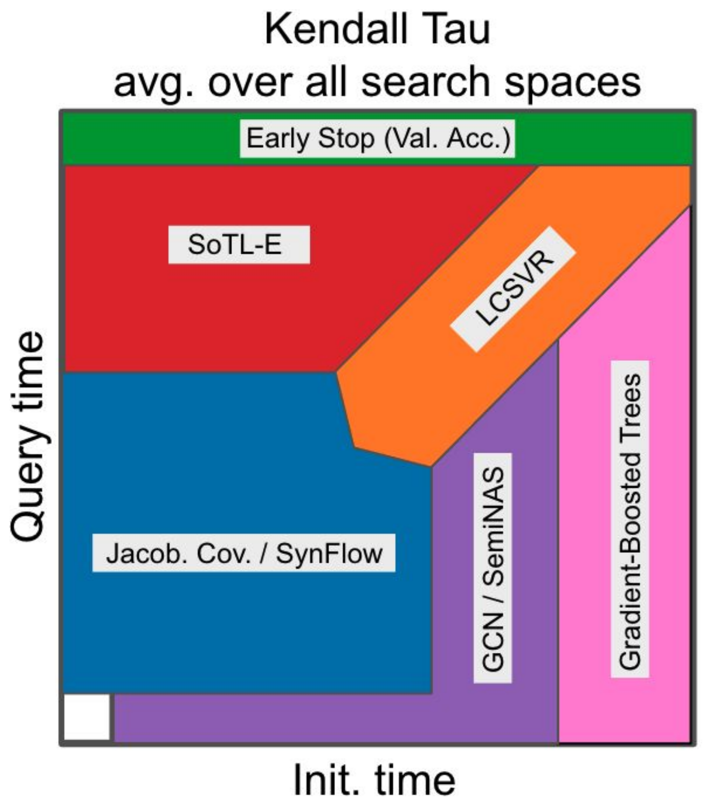
- Jacob. Cov + SoTL-E is consistent
- NGBoost needed for top performance in lower middle/right

NAS Experiments



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!

ABACUS.AI