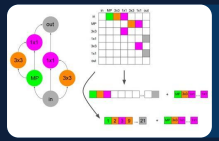
Neural Architecture Search: The Next Frontier

Colin White colin@abacus.ai

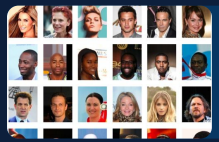
https://abacus.ai/publications

ABACUS.AI

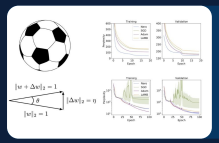
Research at Abacus.AI



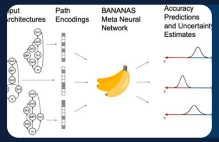
A Study on Encodings for Neural Architecture Search (NeurIPS 2020)



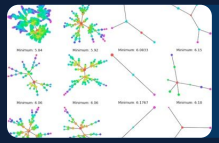
Intra-Processing Methods for Debiasing Neural Networks (NeurIPS 2020)



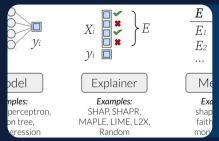
Learning by Turning: Neural Architecture Aware Optimisation (ICML 2021)



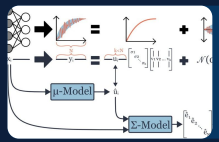
BANANAS: Bayesian Optimization with Neural Architectures for Neural Architecture Search (AAAI 2021)



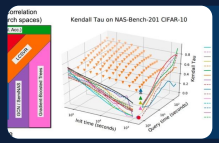
Exploring the Loss Landscape in Neural Architecture Search (UAI 2021)



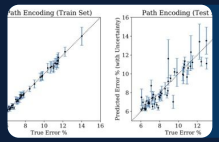
Synthetic Benchmarks for Scientific Research in Explainable Machine Learning (ICML 2021)



NAS-Bench-x11 and the Power of Learning Curves (CVPR 2021 Workshop)

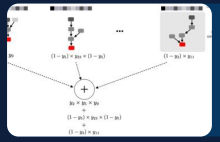


How Powerful are Performance Predictors in Neural Architecture Search? (ICLR Workshop)



Deep Uncertainty Estimation for Model-based Neural Architecture Search (NeurIPS 2019)

DECO: Debiasing through Compositional Optimization of Machine Learning Models (NeurIPS 2019)



Differentiable Functions for Combining First-order Constraints with Deep Learning via Weighted Proof Tracing (NeurIPS 2019)

Competitions

CVPR-NAS 2021

CVPR 2021 Unseen Data in Neural Architecture Search

2ND PLACE

https://abacus.ai/publications

Machine learning automation

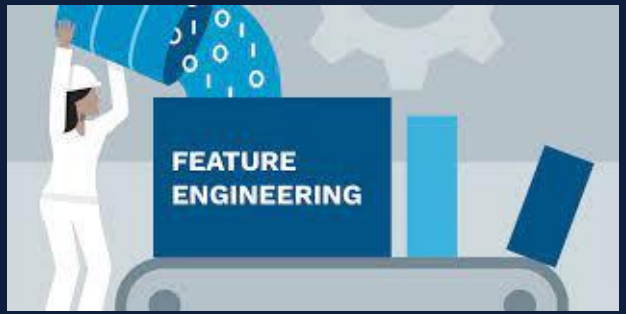
ABACUS.AI

1950s

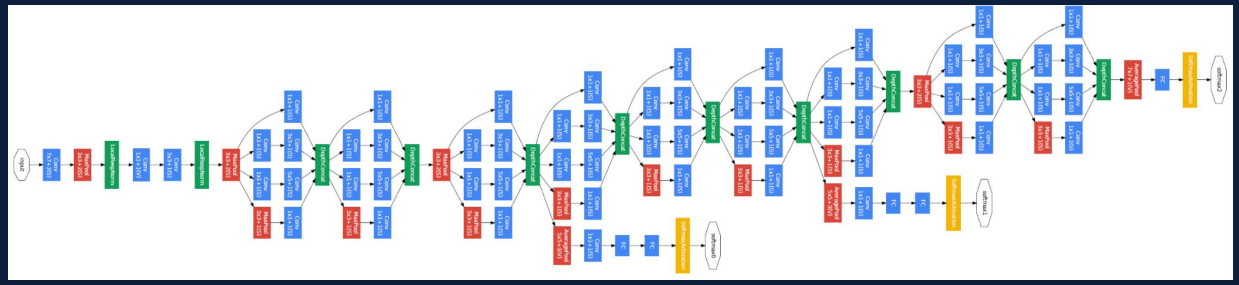
2013

2017

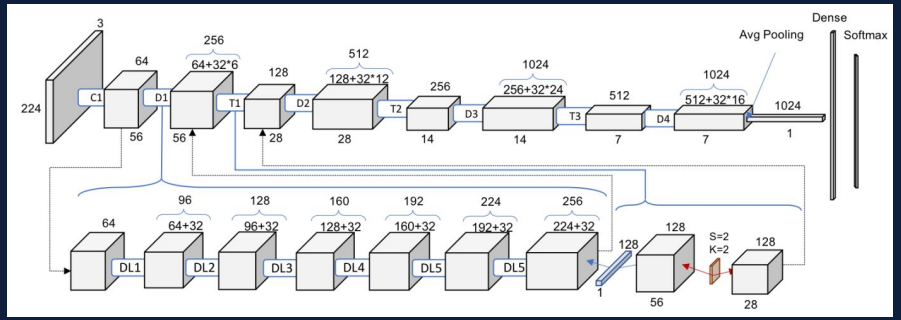
2022



Neural architecture search



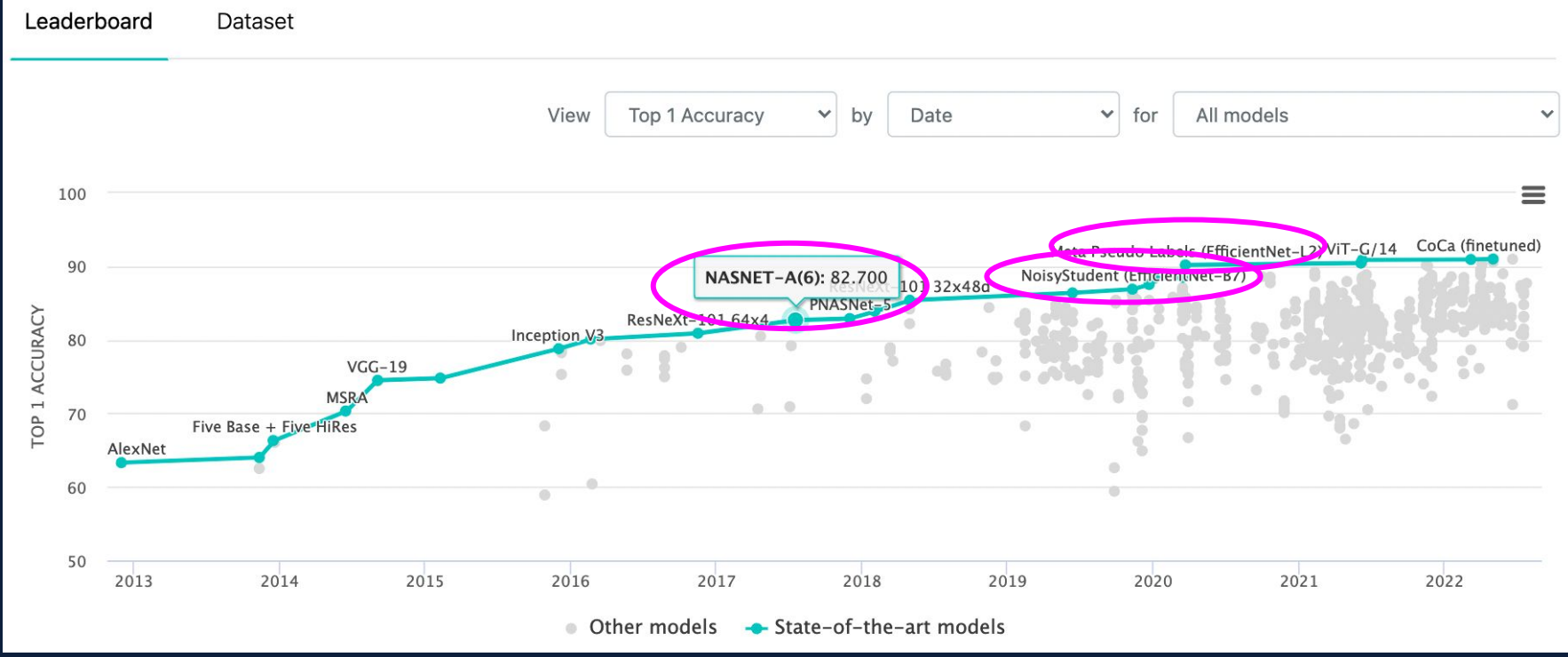
GoogLeNet (2014)



DenseNet (2016)

Architectures are getting increasingly more specialized and complex

Image Classification on ImageNet



https://paperswithcode.com/sota/image-classification-on-imagenet

What is NAS?

What is neural architecture search?

Neural architecture search (NAS) is a technique for automatically designing neural networks. NAS algorithms typically use a reinforcement learning (RL) or evolutionary algorithm (EA) to search for an optimal neural network architecture.

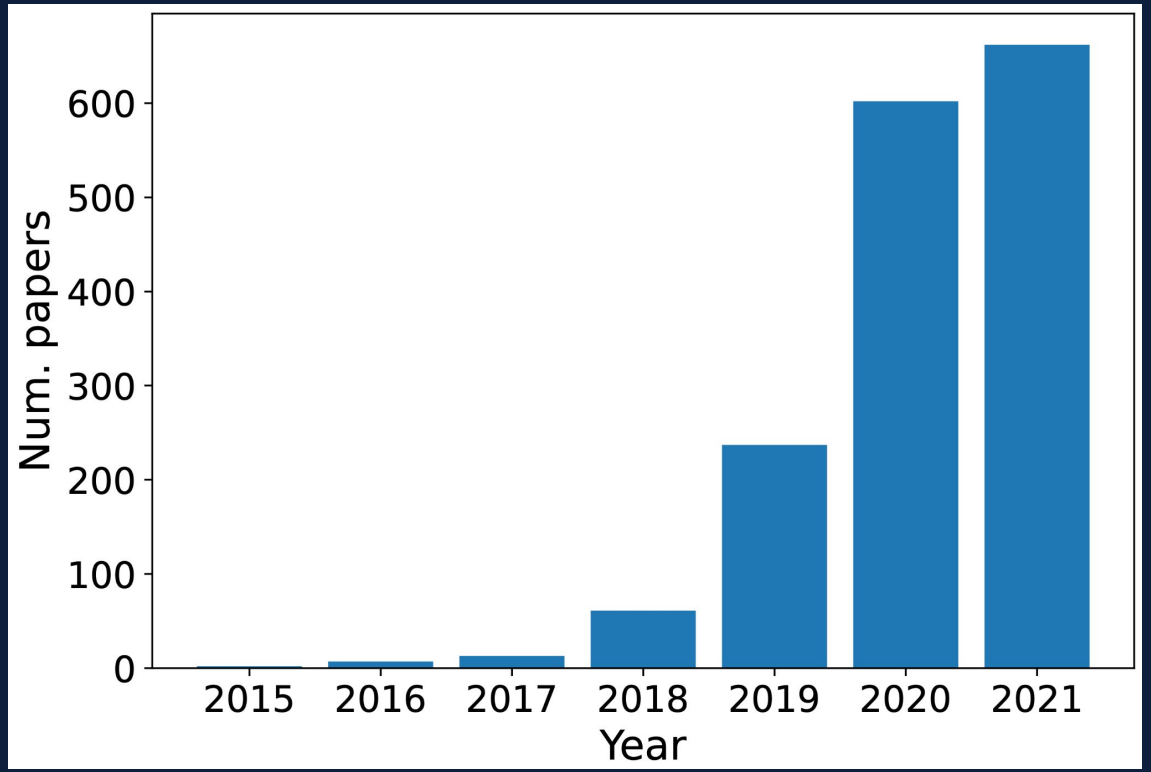
DALL·E My collection

Edit the detailed description

What is neural architecture search? Generate

Neural architecture search

NAS: the process of **automating** the design of **neural architectures** for a given dataset.

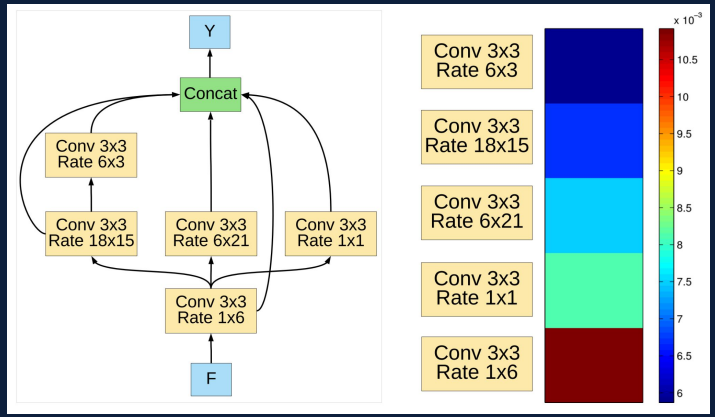


NAS on new datasets / tasks

Spherical	Omnidirectional Vision	
NinaPro DB5	Prosthetics Control	
FSD50K	Audio Classification	
Darcy Flow	PDE Solver	
PSICOV	Protein Folding	
Cosmic	Astronomy Imaging	
ECG	Medical Diagnostics	
Satellite	Earth Monitoring	
DeepSEA	Genetic Prediction	

[NAS-Bench-360 (2021)]

Graph neural networks . . . . .
Generative adversarial network
Dense prediction tasks . . . . .
Adversarial robustness . . . . .
Self-supervised learning for NAS



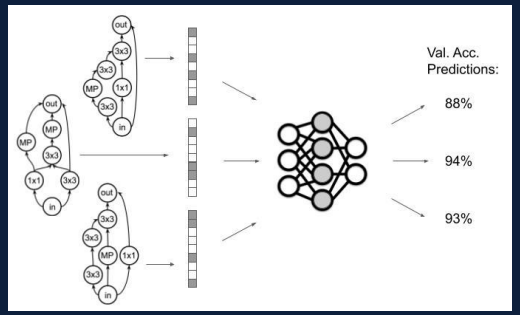
[Chen et al. 2018]

Roadmap

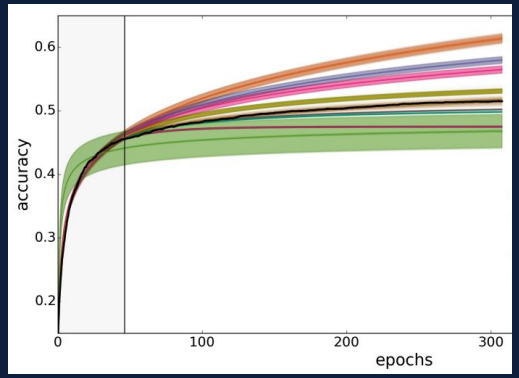
- Motivation and Introduction
- Performance Prediction
  - BANANAS
  - Learning curve extrapolation
  - Zero-cost proxies
- Recommender Systems
- De-biasing models

Performance Predictors

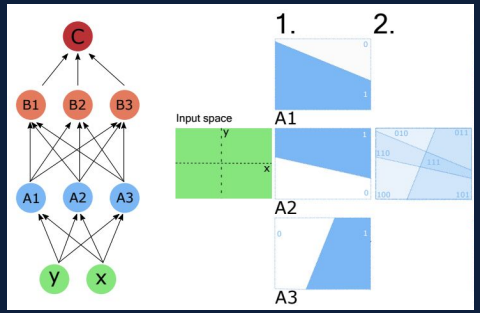
Predict the (relative) accuracy of an architecture, without fully training it.



Model-based



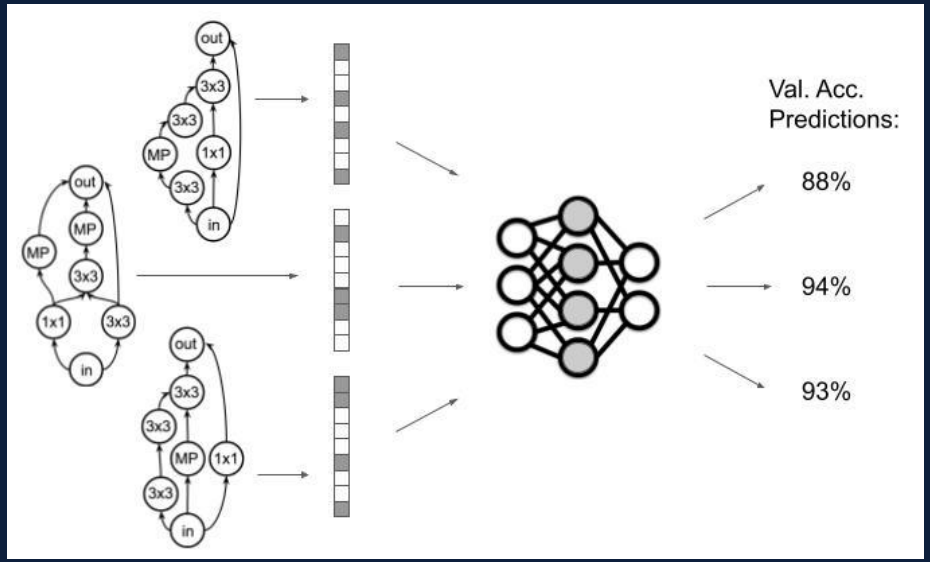
Learning curve extrapolation



Zero-cost proxies

Model-Based Predictors

Train a surrogate model



- 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]

“BO + Neural Predictor” Framework

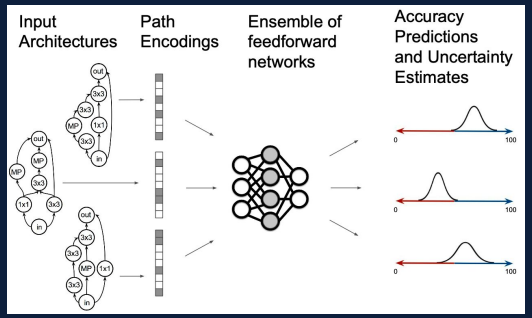
[NASGBO, 2019], [BONAS, 2019], [BANANAS, 2019]

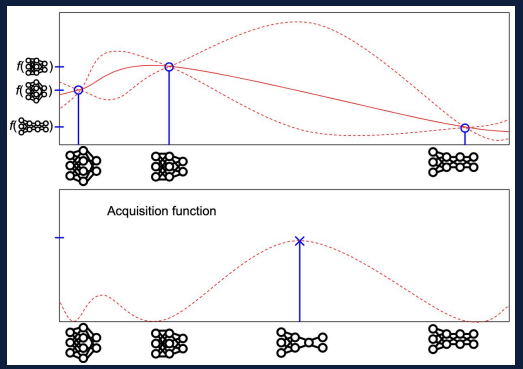
Algorithm 1 BANANAS

Input: Search space A, dataset D, parameters t_0, T, M, c, x, acquisition function ϕ, function $f(a)$ returning validation error of a after training.

1. Draw t_0 architectures $a_0, \ldots, a_{t_0}$ uniformly at random from A and train them on D.
2. For t from t_0 to T,
   - i. Train an ensemble of meta neural networks on $\{(a_0, f(a_0)), \ldots, (a_t, f(a_t))\}$.
   - ii. Generate a set of c candidate architectures from A by randomly mutating the x architectures a from $\{a_0, \ldots, a_t\}$ that have the lowest value of $f(a)$.
   - iii. For each candidate architecture a, evaluate the acquisition function $\phi(a)$.
   - iv. Denote a_{t+1} as the candidate architecture with minimum $\phi(a)$, and evaluate $f(a_{t+1})$.

Output: $a^* = \text{argmin}_{t=0,\ldots,T} f(a_t)$.





Train 10 arch.’s each iteration

“BO + Neural Predictor” Components

Algorithm 1 BANANAS

Input: Search space A, dataset D, parameters t_0, T, M, c, x, acquisition function ϕ, function $f(a)$ returning validation error of a after training.

1. Draw t_0 architectures $a_0, \ldots, a_{t_0}$ uniformly at random from A and train them on D.
2. For t from t_0 to T,
   i. Train an ensemble of meta neural networks on $\{(a_0, f(a_0)), \ldots, (a_t, f(a_t))\}$.
   ii. Generate a set of c candidate architectures from A by randomly mutating the x architectures a from $\{a_0, \ldots, a_t\}$ that have the lowest value of $f(a)$.
   iii. For each candidate architecture a, evaluate the acquisition function $\phi(a)$.
   iv. Denote a_{t+1} as the candidate architecture with minimum $\phi(a)$, and evaluate $f(a_{t+1})$.

Output: $a^* = \text{argmin}_{t=0,\ldots,T} f(a_t)$.

- Architecture encoding
- Uncertainty calibration
- Neural predictor architecture
- Acquisition optimization strategy
- Acquisition function

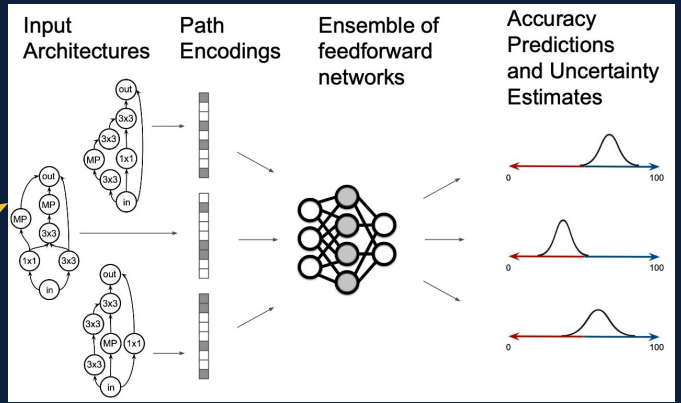
BANANAS

Algorithm 1 BANANAS

Input: Search space A, dataset D, parameters t_0, T, M, c, x, acquisition function ϕ, function $f(a)$ returning validation error of a after training.

1. Draw t_0 architectures $a_0, \ldots, a_{t_0}$ uniformly at random from A and train them on D.
2. For t from t_0 to T,
   i. Train an ensemble of meta neural networks on $\{(a_0, f(a_0)), \ldots, (a_t, f(a_t))\}$.
   ii. Generate a set of c candidate architectures from A by randomly mutating the x architectures a from $\{a_0, \ldots, a_t\}$ that have the lowest value of $f(a)$.
   iii. For each candidate architecture a, evaluate the acquisition function $\phi(a)$.
   iv. Denote a_{t+1} as the candidate architecture with minimum $\phi(a)$, and evaluate $f(a_{t+1})$.

Output: $a^* = \text{argmin}_{t=0,\ldots,T} f(a_t)$.



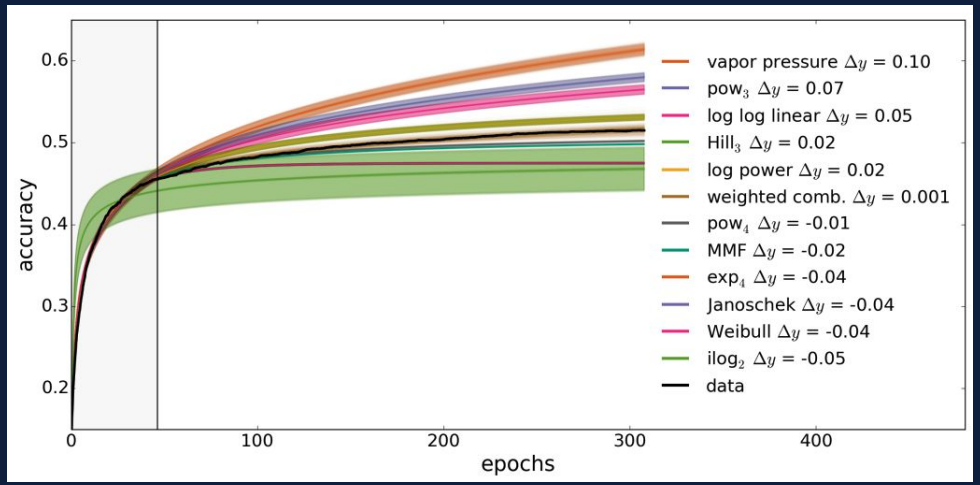
Path encoding, ensemble

Small mutations

Independent Thompson Sampling

[White, Neiswanger, Savani, AAAI 2021]

Learning curve based predictors



- Learning curve extrapolation
  - Fit partial learning curve to parametric model [Domhan et al. 2015]
  - Bayesian NN [Klein et al. 2017]
- LCE + Surrogate
  - SVR [Baker et al. 2017]
  - Full LC + Bayesian NN [Klein et al. 2017]

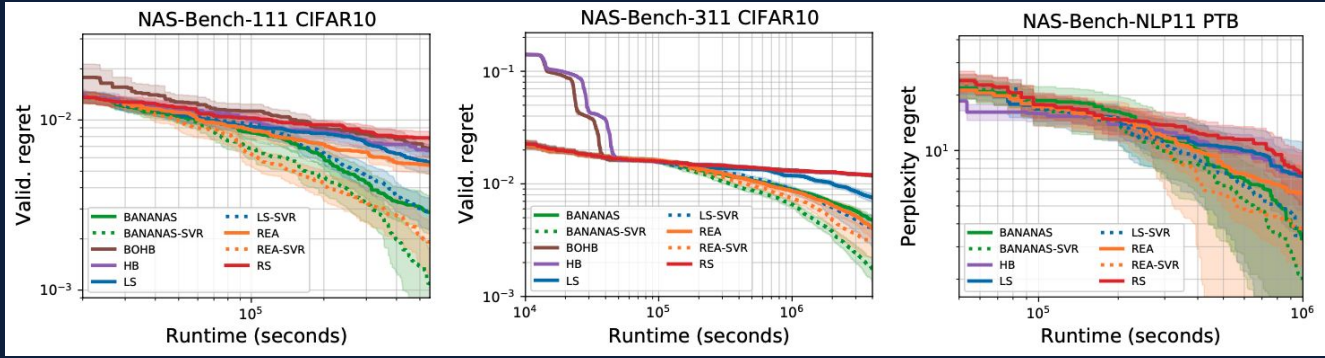
LCE Framework

Algorithm 1 Single-Fidelity Algorithm

```
initialize history
while t < t_max :
    arches = gen_candidates(history)
    accs = train(arches, epoch=E_max)
    history.update(arches, accs)
Return arch with the highest acc
```

Algorithm 2 LCE Framework

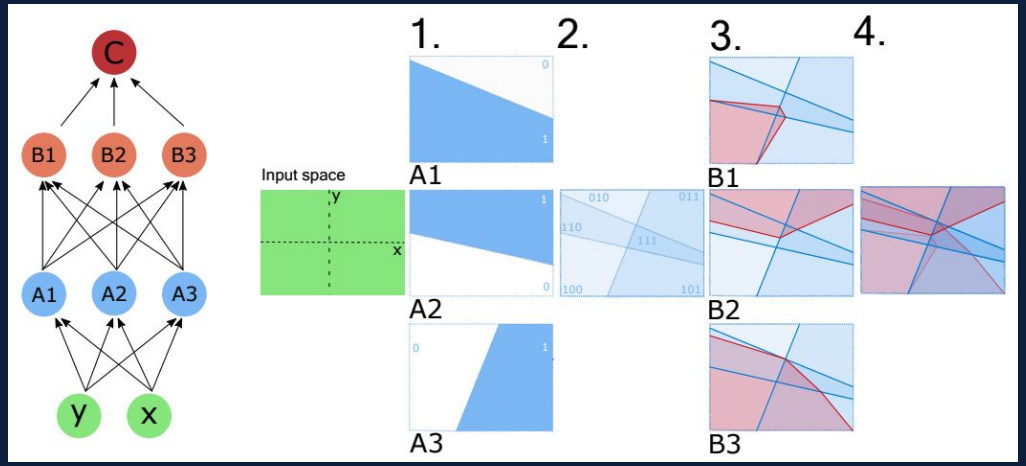
```
initialize history
while t < t_max :
    arches = gen_candidates(history)
    accs = train(arches, epoch=E_few)
    sorted_by_pred = LCE(arches, accs)
    arches = sorted_by_pred[:top_n]
    accs = train(arches, epoch=E_max)
    history.update(arches, accs)
Return arch with the highest acc
```



Yan*, White*, Savani, Hutter, NeurIPS 2021

Zero-cost proxies

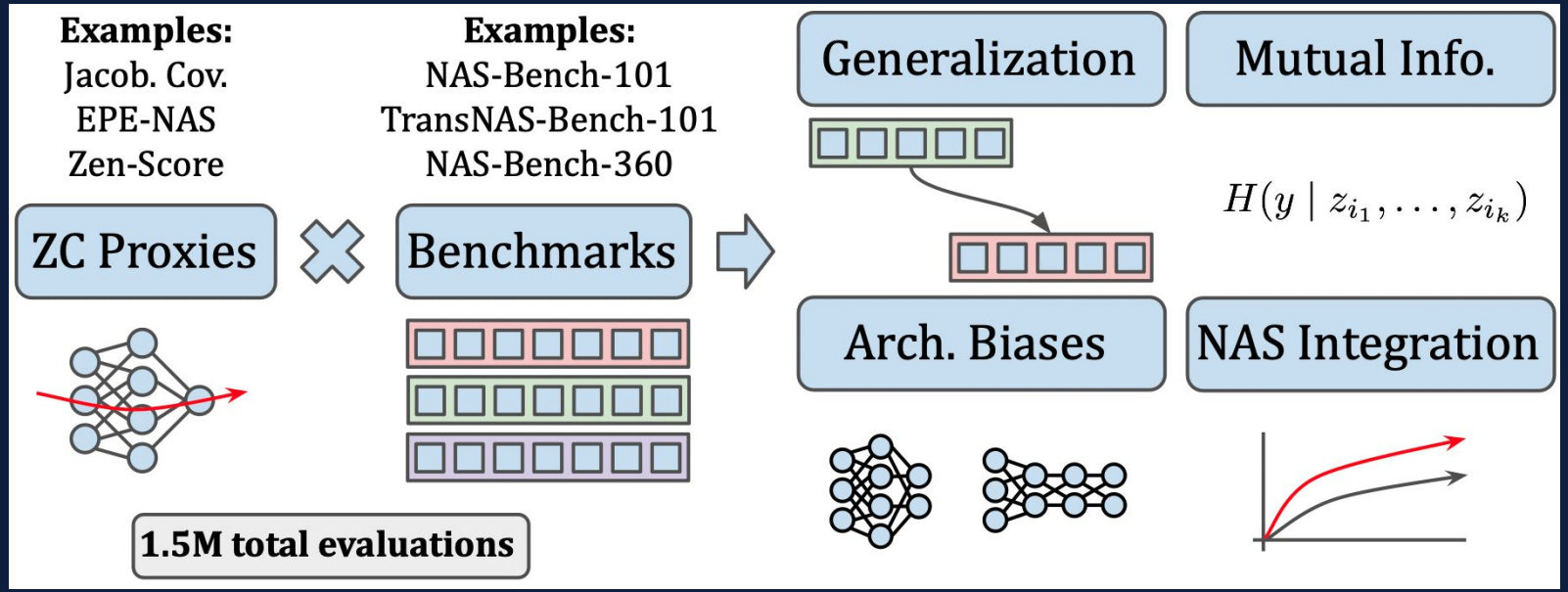
epe-nas [21]	Jacobian
fisher [42]	Pruning-at-init
flops [25]	Baseline
grad-norm [1]	Pruning-at-init
grasp [43]	Pruning-at-init
l2-norm [1]	Baseline
jacov [23]	Jacobian
nwot [23]	Jacobian
params [25]	Baseline
plain [1]	Baseline
snip [14]	Pruning-at-init
synflow [39]	Pruning-at-init
zen-score [16]	Piece. Lin.



[Mellor et al. 2020]

Compute an estimate in 5 seconds

NAS-Bench-Suite-Zero (28 tasks)



Krishnakumar*, White*, Zela*, Tu*, Safari, Hutter. NeurIPS Datasets Track 2022

https://github.com/automl/NASLib/tree/zerocost

Zero-cost proxies

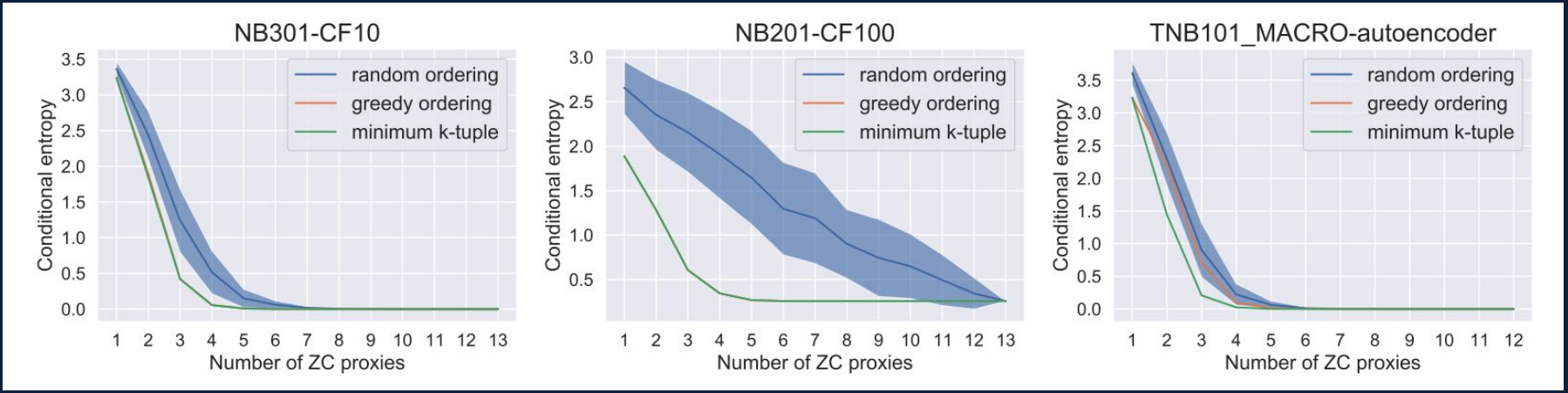
Table 4: Average ranking of each of the ZC proxies on each search space, and over all search spaces.

	fisher	grad_norm	grasp	jacob_cov	snip	synflow	flops	params
NATS-Bench TSS	6.0	6.0	5.0	4.0	5.67	**1.33**	4.0	4.0
DARTS	4.6	4.2	4.6	4.8	4.6	5.4	4.0	**3.8**
TransNAS-Bench-101	**2.75**	4.5	4.5	7.5	3.0	4.5	4.0	5.25
Overall	4.33	4.75	4.67	5.5	4.33	4.08	**4.0**	4.33

- Still do not consistently beat simple baselines
- Promising when used in conjunction with other NAS techniques

White, Khodak, Tu, Shah, Bubeck, Dey, ICLR Blog Track, 2022

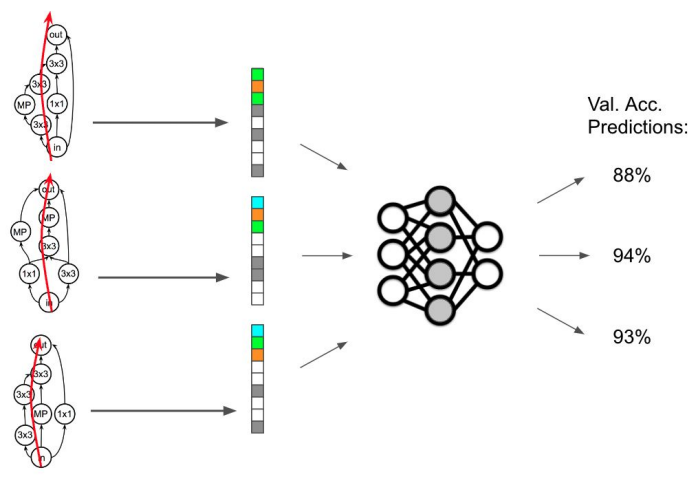
Complementary info in ZC proxies



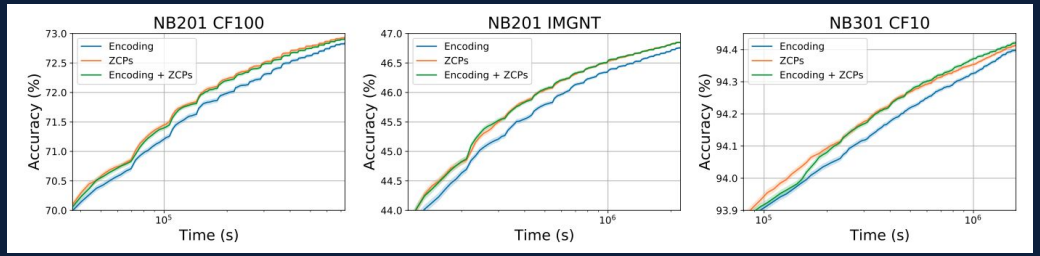
Conditional entropy $H(y \mid z_{i_1}, \ldots, z_{i_k})$ vs. k

Krishnakumar*, White*, Zela*, Tu*, Safari, Hutter. Preprint 2022

NAS integration



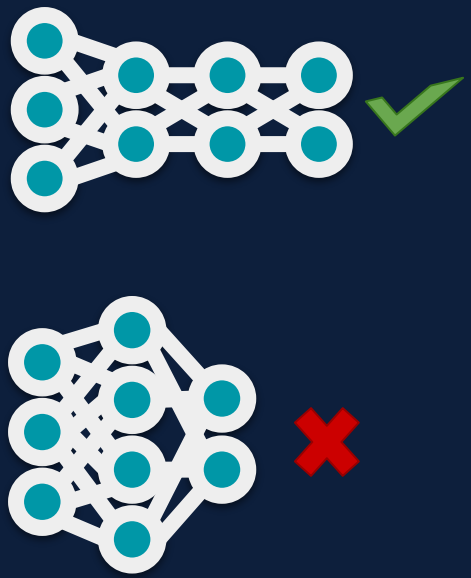
Features Benchmark	Encoding	ZC	Both	% Improvement (ZC)	% Improvement (Both)
NB101-CF10	0.546	0.708	0.718	29.67	31.50
NB201-CF10	0.622	0.905	0.906	45.50	45.66
NB201-CF100	0.640	0.907	0.908	41.71	41.87
NB201-IMGNT	0.683	0.879	0.883	28.70	29.28
NB301-CF10	0.314	0.405	0.465	28.98	48.09
TNB101_MACRO-AUTOENC	0.673	0.831	0.837	23.48	24.37
TNB101_MACRO-JIGSAW	0.809	0.706	0.809	-12.73	0.00
TNB101_MACRO-NORMAL	0.617	0.710	0.716	15.07	16.05
TNB101_MACRO-OBJECT	0.736	0.840	0.843	14.13	14.54
TNB101_MACRO-ROOM	0.683	0.589	0.707	-13.76	3.51
TNB101_MACRO-SCENE	0.832	0.891	0.899	7.09	8.05
TNB101_MACRO-SEGMENT	0.900	0.807	0.876	-10.33	-2.67
TNB101_MICRO-AUTOENC	0.714	0.754	0.803	5.60	12.46
TNB101_MICRO-JIGSAW	0.585	0.730	0.743	24.79	27.01
TNB101_MICRO-NORMAL	0.657	0.801	0.809	21.92	23.14
TNB101_MICRO-OBJECT	0.637	0.733	0.752	15.07	18.05
TNB101_MICRO-ROOM	0.582	0.843	0.844	44.85	45.02
TNB101_MICRO-SCENE	0.710	0.849	0.866	19.58	21.97
TNB101_MICRO-SEGMENT	0.767	0.886	0.897	15.51	16.95



Krishnakumar*, White*, Zela*, Tu*, Safari, Hutter. Preprint 2022

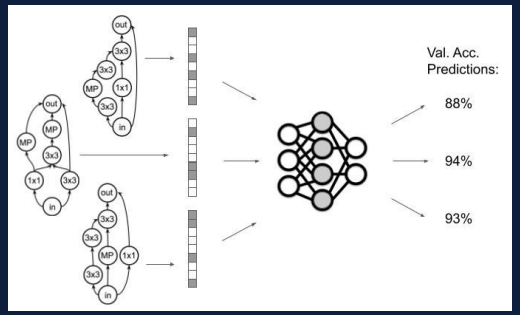
Removing biases in ZC proxies

$$f'(a) = f(a) \cdot \frac{1}{b(a) + C}$$

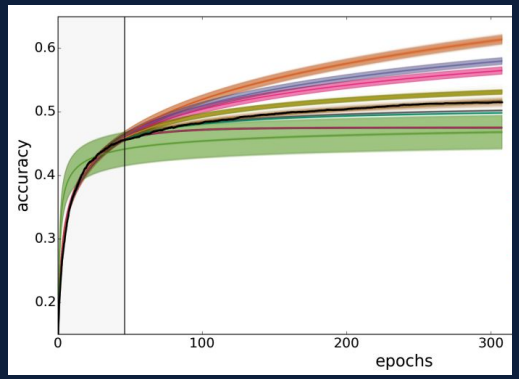


ZC proxy	dataset	bias metric	original bias	original perf.	new bias	new perf.	strategy
l2-norm	NB201-CF10	conv:pool	0.87	0.42	0.00	0.10	minimize
					0.37	0.11	equalize
					0.70	0.44	performance
nwot	NB301-CF10	conv:pool	0.78	0.49	0.00	0.03	minimize
					0.29	0.14	equalize
					0.78	0.49	performance
synflow	NB201-CF100	cell size	0.57	0.68	0.01	0.64	minimize
					0.35	0.71	equalize
					0.35	0.71	performance
synflow	NB201-IM	cell size	0.58	0.76	0.01	0.62	minimize
					0.43	0.76	equalize
					0.46	0.76	performance
flops	NB301-CF10	num. skip	-0.35	0.43	-0.01	0.06	minimize
					0.12	-0.05	equalize
					-0.35	0.43	performance

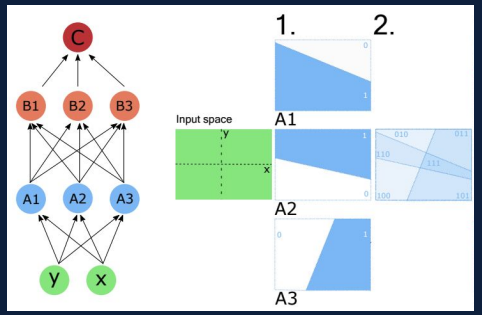
Performance predictor families



Model-based

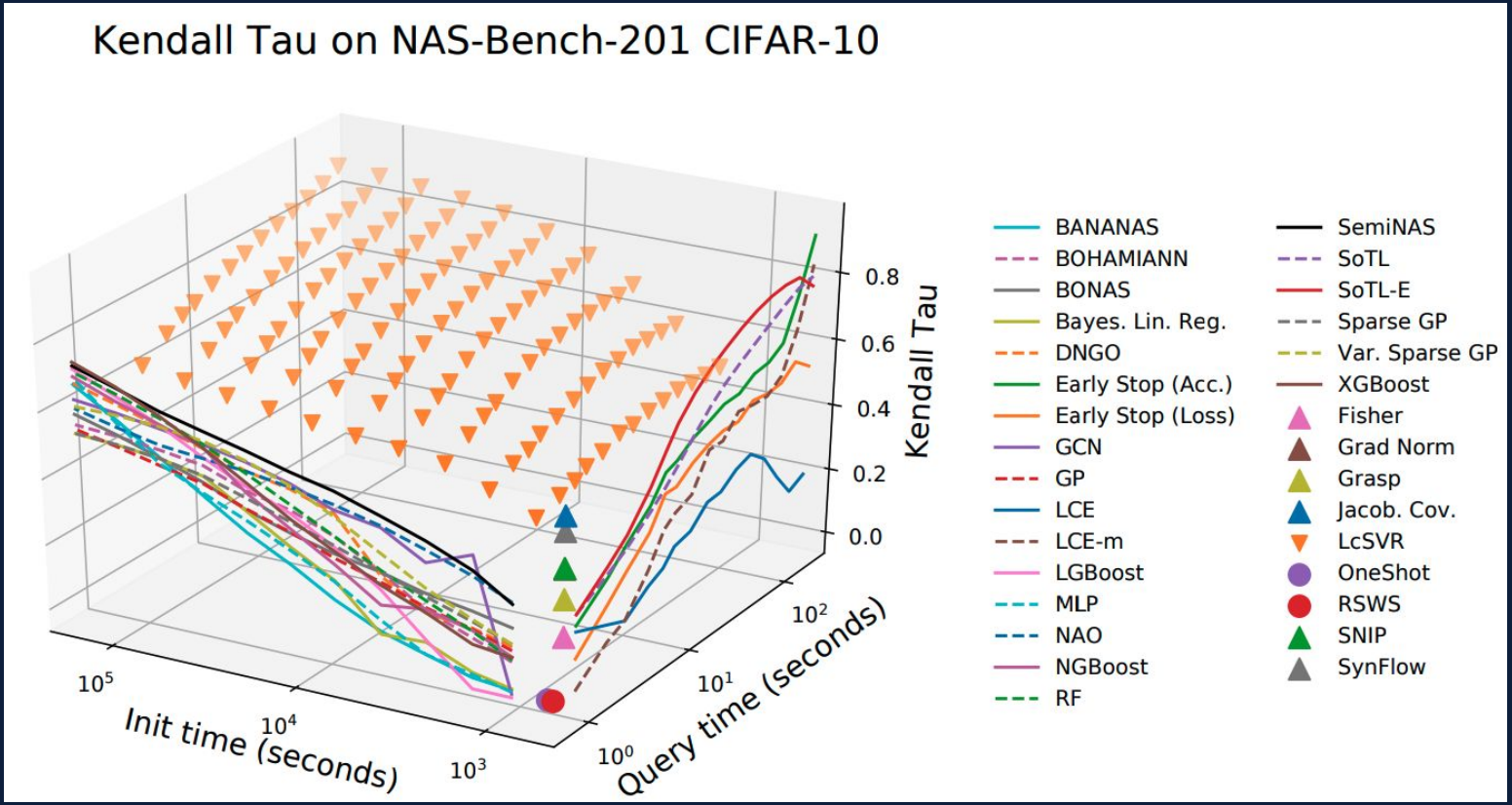


Learning curve extrapolation

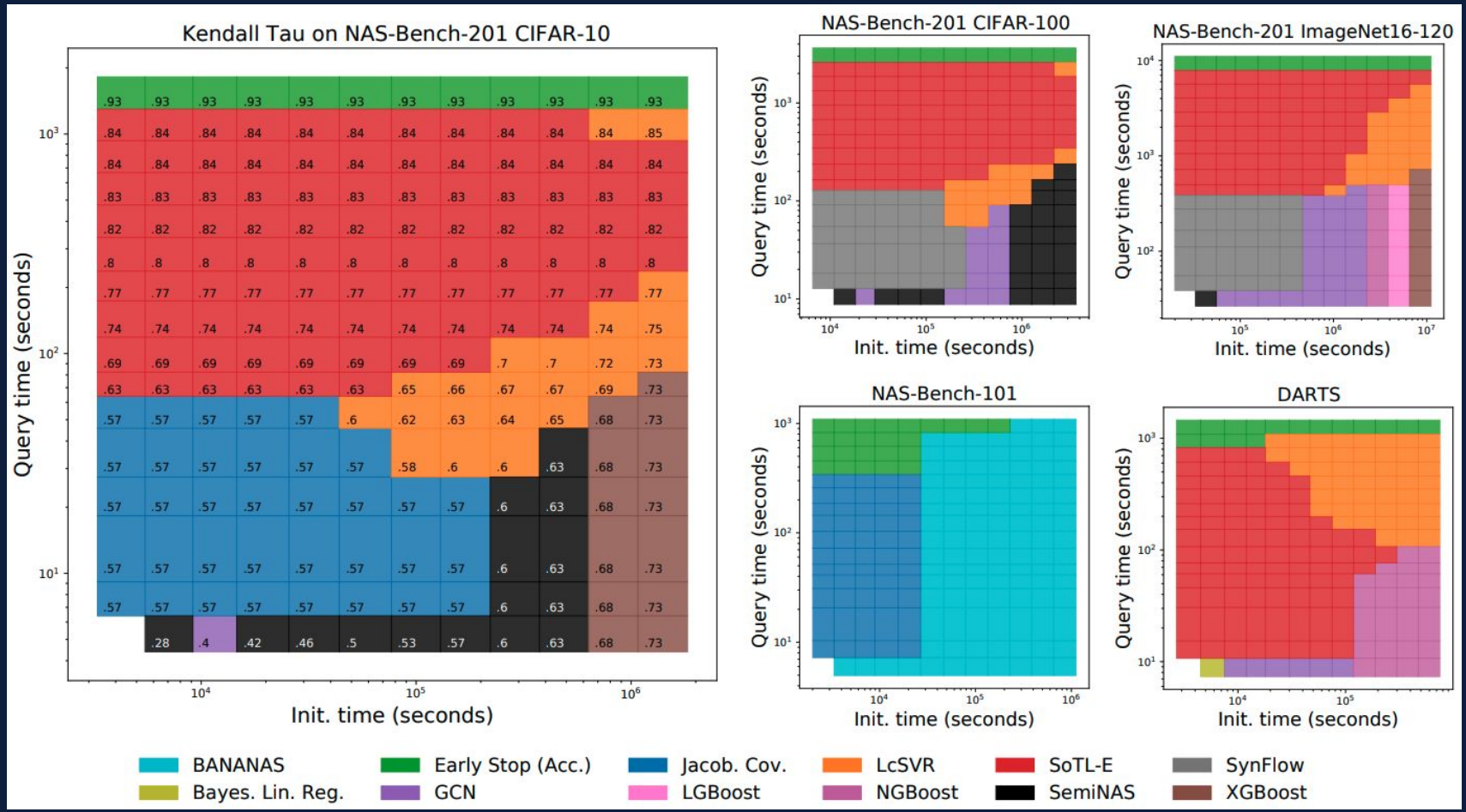


Zero-cost proxies

Performance predictors

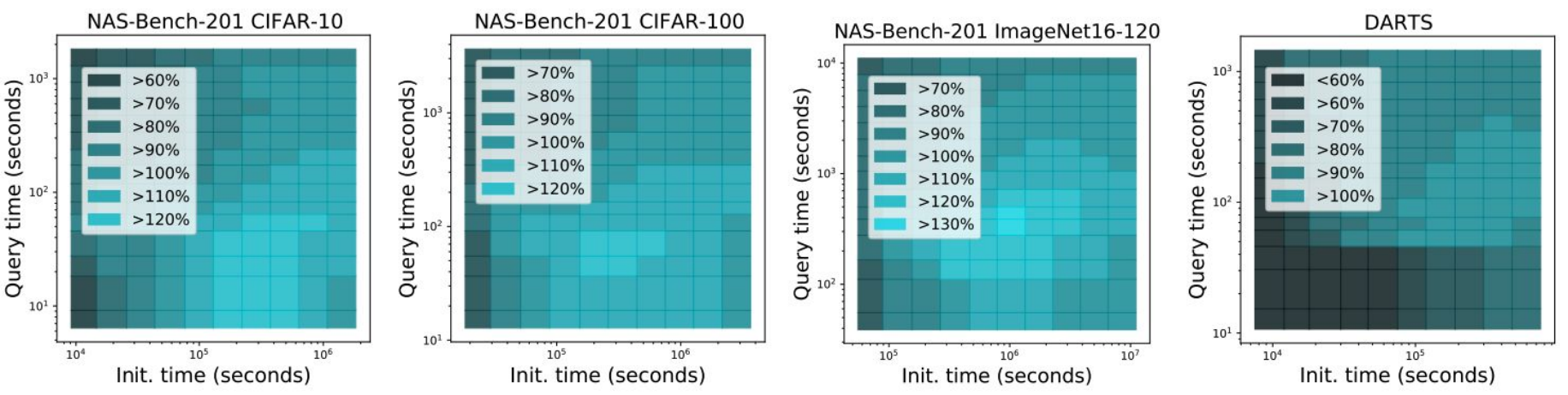


White, Zela, Ru, Liu, Hutter, NeurIPS 2021


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

OMNI: The Omnipotent Predictor

Combine best predictors from each family

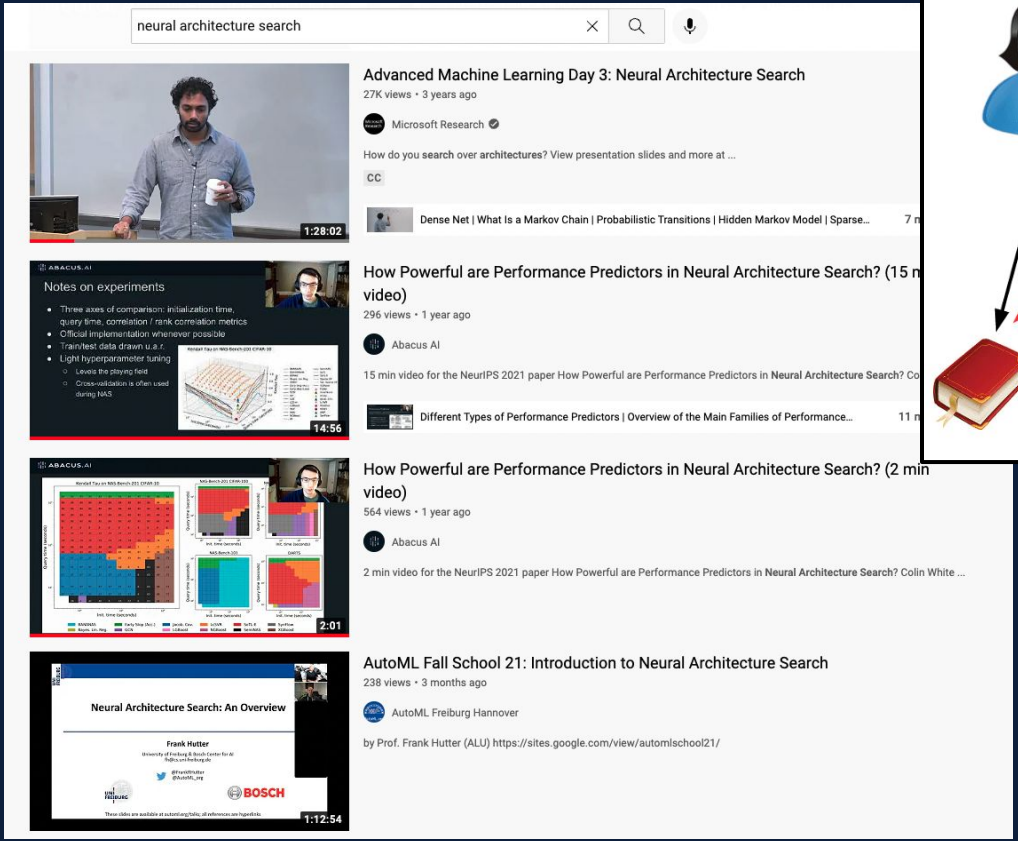


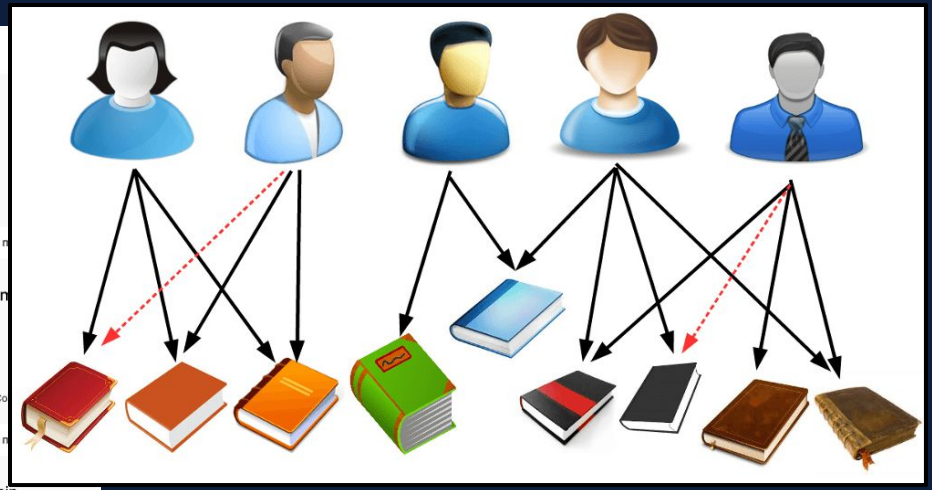
White, Zela, Ru, Liu, Hutter, NeurIPS 2021

Roadmap

- Motivation and Introduction
- Performance Prediction
  - BANANAS
  - Learning curve extrapolation
  - Zero-cost proxies
- Recommender Systems
- De-biasing models

Recommender Systems





A Worrying Analysis of Recommender Systems

Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches

Maurizio Ferrari Dacrema
Politecnico di Milano, Italy
maurizio.ferrari@polimi.it

Paolo Cremonesi
Politecnico di Milano, Italy
paolo.cremonesi@polimi.it

Dietmar Jannach
University of Klagenfurt, Austria
dietmar.jannach@aau.at

ABSTRACT

Deep learning techniques have become the method of choice for researchers working on algorithmic aspects of recommender systems. With the strongly increased interest in machine learning in general, it has, as a result, become difficult to keep track of what represents the state-of-the-art at the moment, e.g., for top-n recommendation tasks. At the same time, several recent publications point out problems in today's research practice in applied machine learning, e.g., in terms of the reproducibility of the results or the choice of the baselines when proposing new models.

In this work, we report the results of a systematic analysis of algorithmic proposals for top-n recommendation tasks. Specifically, we considered 18 algorithms that were presented at top-level research conferences in the last years. Only 7 of them could be reproduced with reasonable effort. For these methods, it however turned out that 6 of them can often be outperformed with comparably simple heuristic methods, e.g., based on nearest-neighbor or graph-based techniques. The remaining one clearly outperformed the baselines but did not consistently outperform a well-tuned non-

1 INTRODUCTION

Within only a few years, deep learning techniques have started to dominate the landscape of algorithmic research in recommender systems. Novel methods were proposed for a variety of settings and algorithmic tasks, including top-n recommendation based on long-term preference profiles or for session-based recommendation scenarios [36]. Given the increased interest in machine learning in general, the corresponding number of recent research publications, and the success of deep learning techniques in other fields like vision or language processing, one could expect that substantial progress resulted from these works also in the field of recommender systems. However, indications exist in other application areas of machine learning that the achieved progress—measured in terms of accuracy improvements over existing models—is not always as strong as expected.

Lin [25], for example, discusses two recent neural approaches in the field of information retrieval that were published at top-level conferences. His analysis reveals that the new methods do *not* significantly outperform existing baseline methods when these

Family	*Method*	*Description*
Non-personalized	TopPopular	Recommends the most popular items to everyone [18]
Nearest-Neighbor	UserKNN	User-based k-nearest neighbors [58]
	ItemKNN	Item-based k-nearest neighbors [61]
Graph-based	$P^3\alpha$	A graph-based method based on random walks [16]
	$RP^3\beta$	An extension of $P^3\alpha$ [54]
Content-Based and Hybrid	ItemKNN-CBF	ItemKNN with content-based similarity [43]
	ItemKNN-CFCBF	A simple item-based hybrid CBF/CF approach [50]
	UserKNN-CBF	UserKNN with content-based similarity
	UserKNN-CFCBF	A simple user-based hybrid CBF/CF approach
Non-Neural Machine Learning	iALS	Matrix factorization for implicit feedback data [33]
	pureSVD	A basic matrix factorization method [18]
	SLIM	A scalable linear model [36, 52]
	$EASE^R$	A recent linear model, similar to auto-encoders [63]

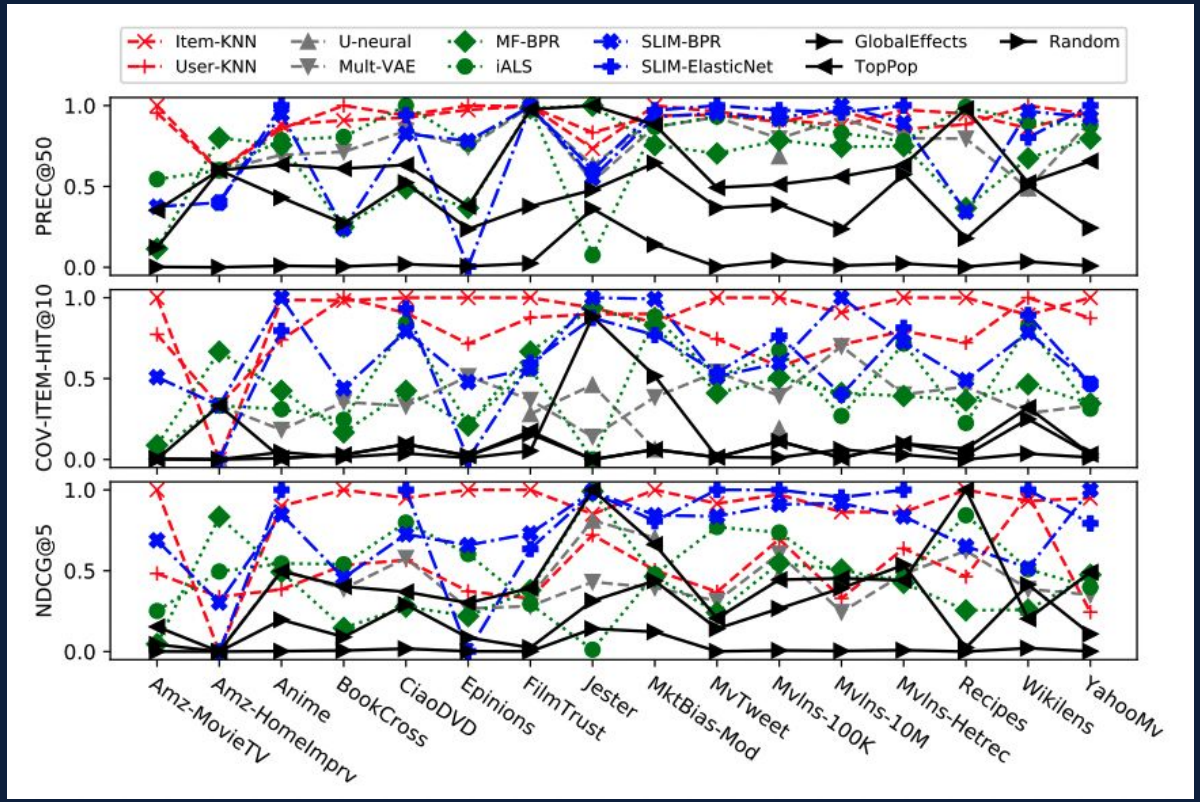
Meta-Learning for Recommender Systems

- 24 Algorithms, up to 100 hyperparameters, 85 datasets, 315 metrics

Rank	Item-KNN	P3alpha	SLIM-BPR	EASE-R	RP3beta	SVD	SLIM-ElasticNet	iALS	NMF	User-KNN	MF-Funk	TopPop	MF-Asy	MF-BPR	Mult-VAE	U-neural	GlobalEffects	CoClustering	Random	SlopeOne
Min.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	9	7
Max.	14	18	14	18	17	16	17	19	14	17	18	19	16	17	20	20	20	19	20	20
Mean	2.3	4.2	4.7	5.3	6	6	7	7	7.1	7.6	9.4	10.4	10.7	11.2	11.7	12.3	13.3	14.9	16.2	16.7

McElfresh, Khandagale, Valverde, Dickerson, White. NeurIPS 2022

Meta-Learning for Recommender Systems

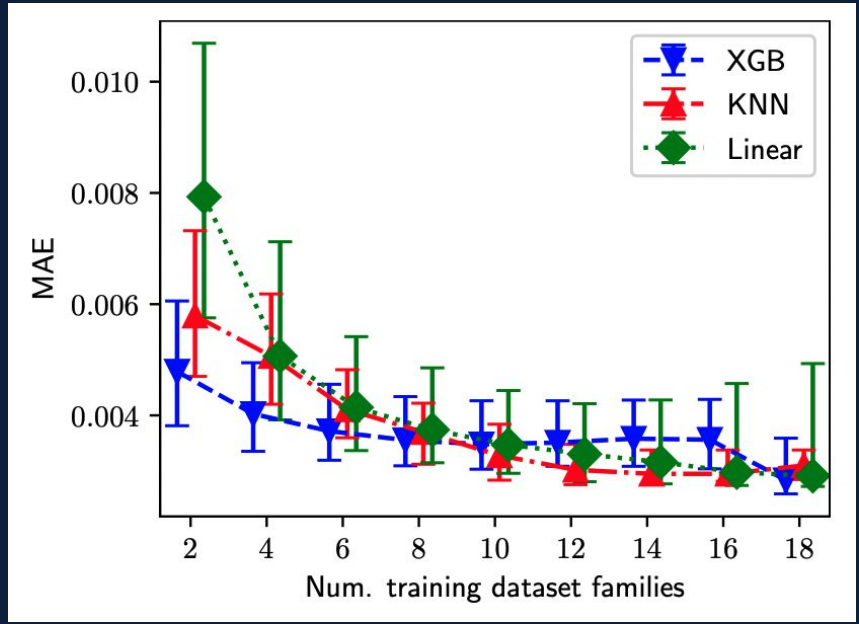


McElfresh, Khandagale, Valverde, Dickerson, White. NeurIPS 2022

RECZILLA

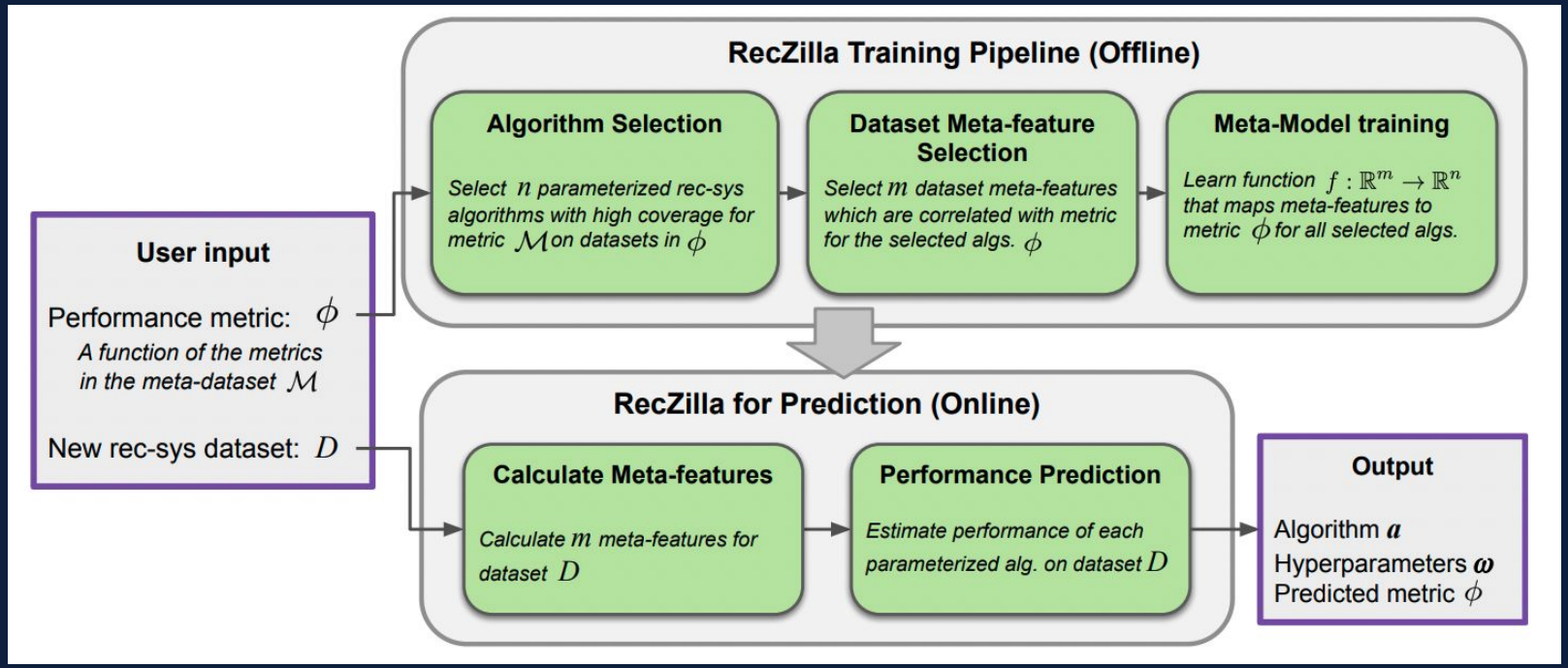
Dataset meta-features

- User distribution
- Item distribution
- Interaction distribution
- Landmarkers



McElfresh, Khandagale, Valverde, Dickerson, White. NeurIPS 2022

RECZILLA



McElfresh, Khandagale, Valverde, Dickerson, White. Preprint 2022

Roadmap

- Motivation and Introduction
- Performance Prediction
  - BANANAS
  - Learning curve extrapolation
  - Zero-cost proxies
- Recommender Systems
- De-biasing models

Face recognition

1. **For one-to-one matching, the team saw higher rates of false positives for Asian and African American faces relative to images of Caucasians.** The differentials often ranged from a factor of 10 to 100 times, depending on the individual algorithm. False positives might present a security concern to the system owner, as they may allow access to impostors.
2. **Among U.S.-developed algorithms, there were similar high rates of false positives in one-to-one matching for Asians, African Americans and native groups** (which include Native American, American Indian, Alaskan Indian and Pacific Islanders). The American Indian demographic had the highest rates of false positives.

'The Computer Got It Wrong': How Facial Recognition Led To False Arrest Of Black Man

June 24, 2020 · 8:00 AM ET

BOBBY ALLYN

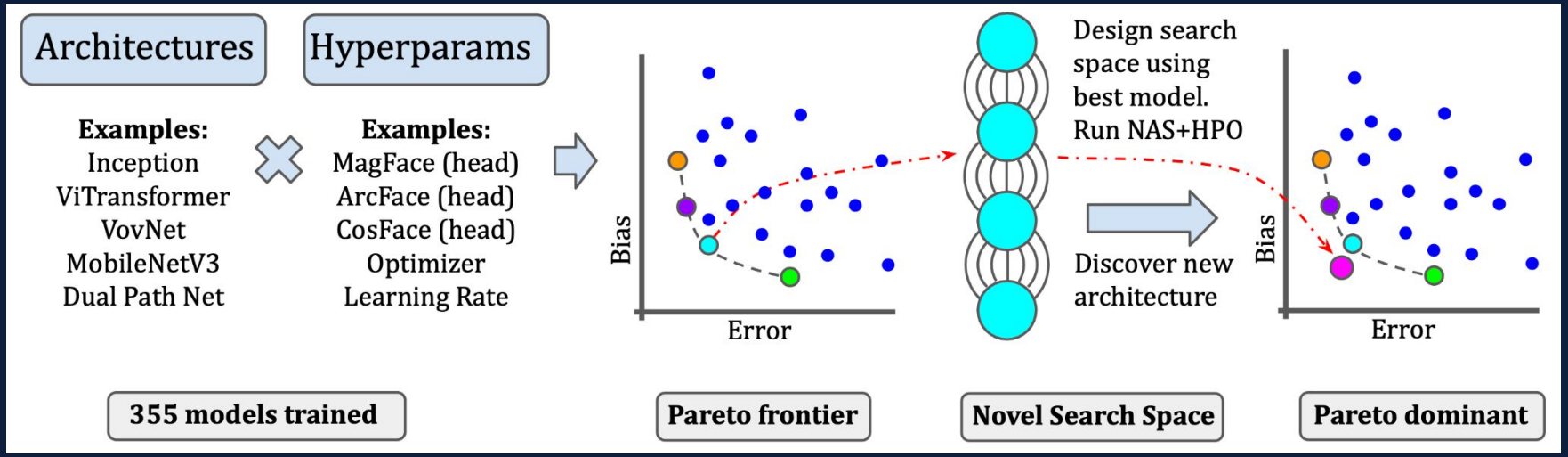
A US government study confirms most face recognition systems are racist

by Karen Hao December 20, 2019

A U.S. Customs and Border Protection officer helps a passenger navigate a facial recognition kiosk at the airport.
DAVID J. PHILLIP/AP

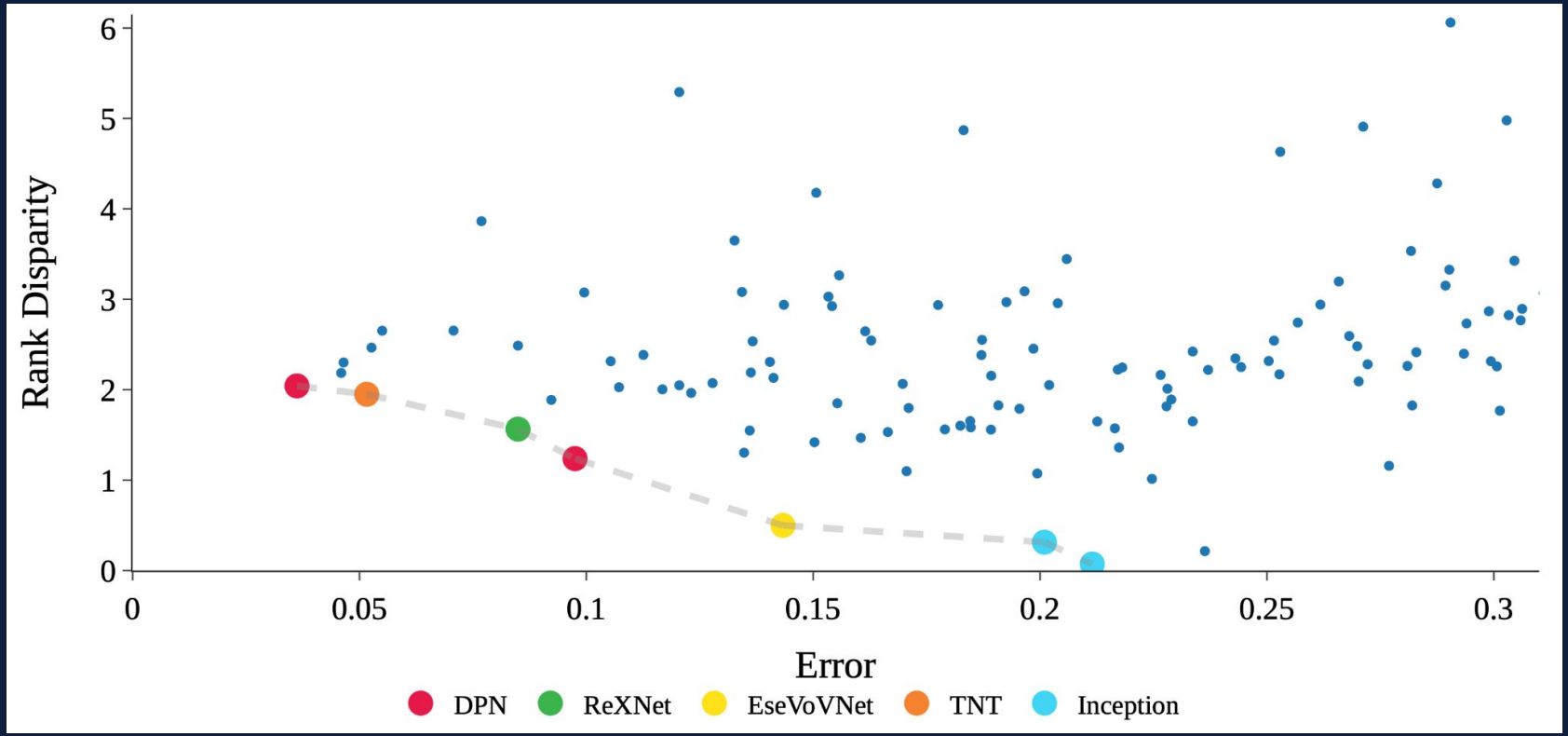
https://www.nist.gov/news-events/news/2019/12/nist-study-evaluates-effects-race-age-sex-face-recognition-software

Face recognition

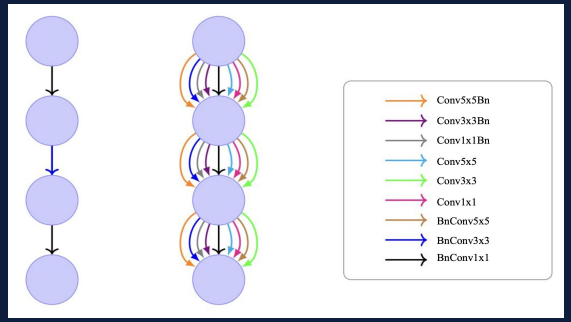


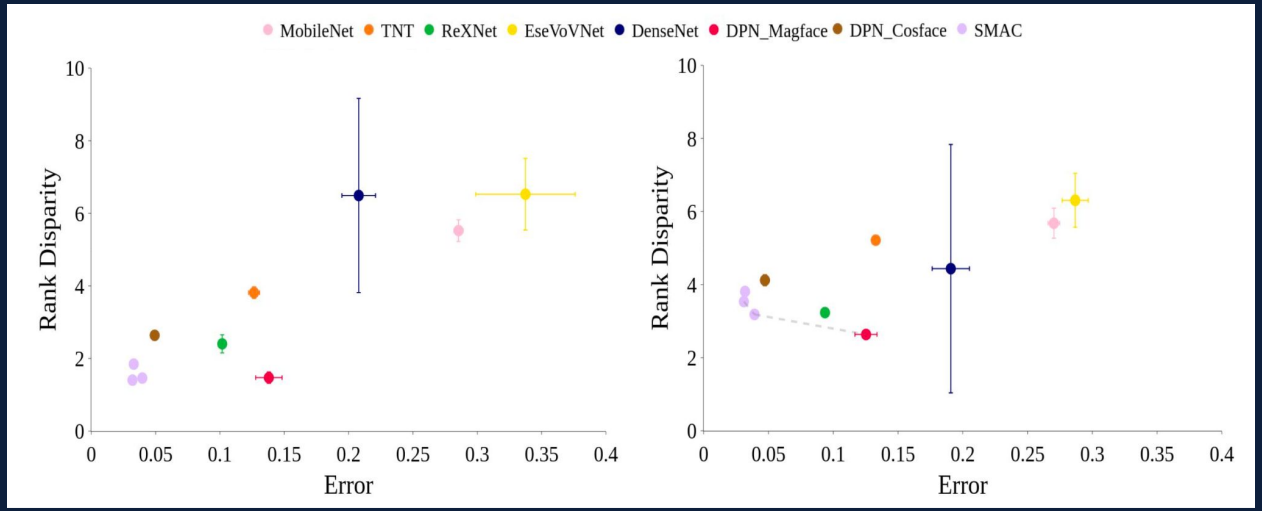
Sukthanker et al., 2022

Face recognition

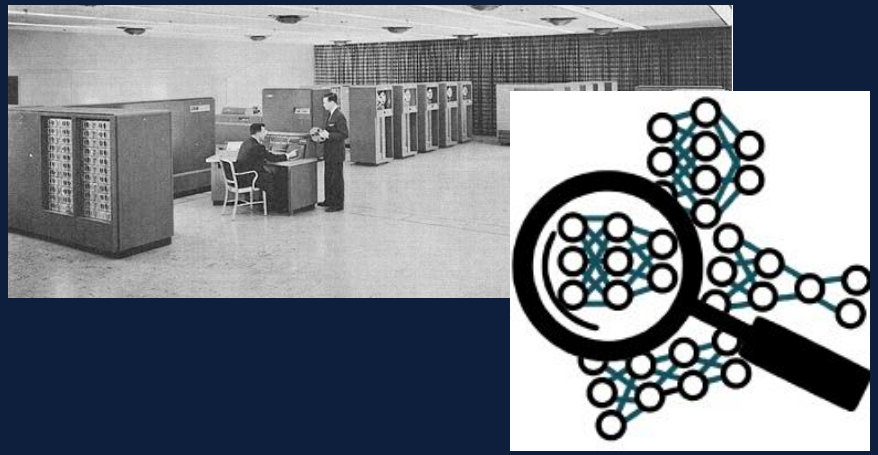


Face recognition





Thanks! Questions?



Colin White colin@abacus.ai

https://abacus.ai/publications