ABACUS.AI

ABACUS.AI

BANANAS:

Bayesian Optimization with Neural Architectures for Neural Architecture Search



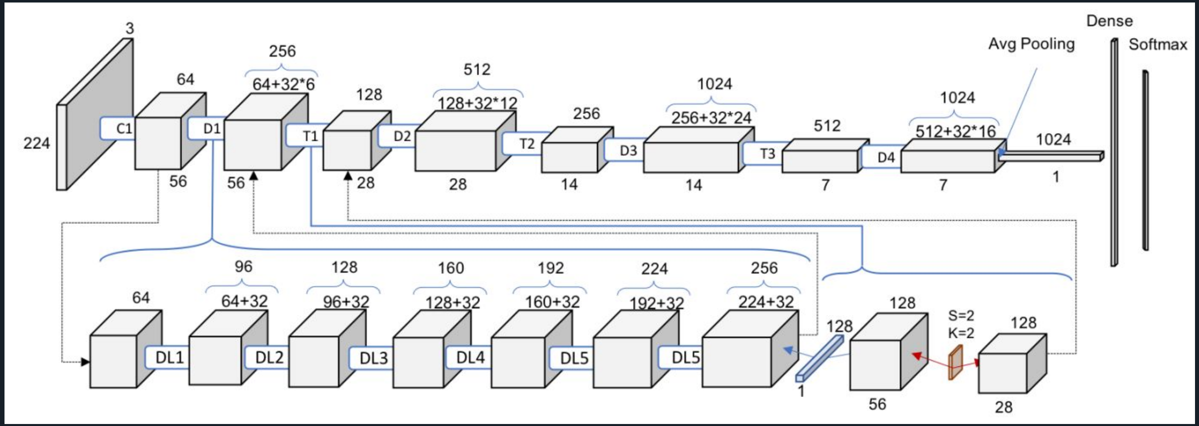
Colin White
Abacus.AI

Willie Neiswanger
Stanford University
and Petuum, Inc.

Yash Savani
Abacus.AI

Neural architecture search

Neural architectures are getting increasingly more specialized and complex



Source: https://towardsdatascience.com/understanding-and-visualizing-densenets-7f688092391a

ABACUS.AI

Roadmap

- Background
- “BayesOpt + neural predictor” framework
  - Encodings
  - Predictor / Uncertainty Calibration
  - Acquisition function
  - Acquisition function optimization

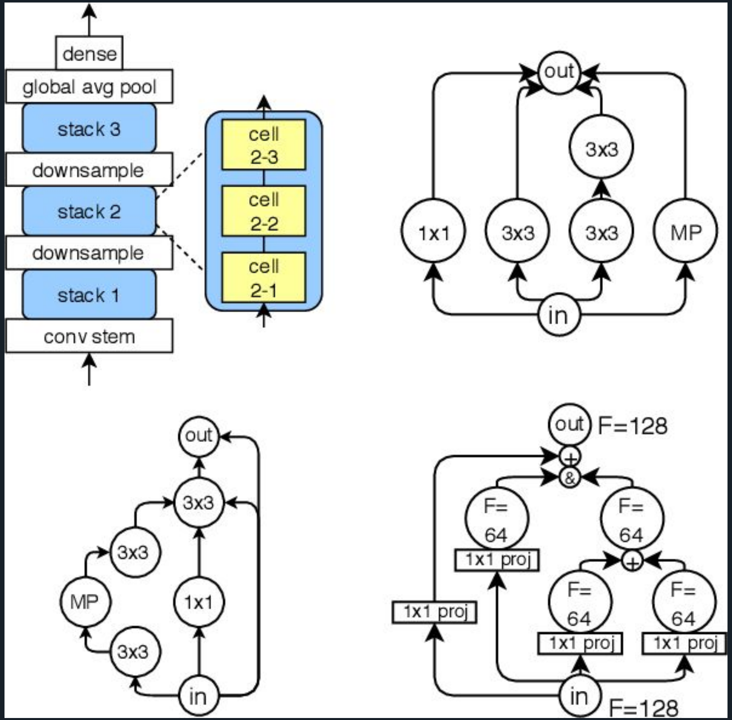
⇒ **BANANAS**

Cell-based search spaces

Search over a small labeled DAG

Stack the DAG on itself multiple times

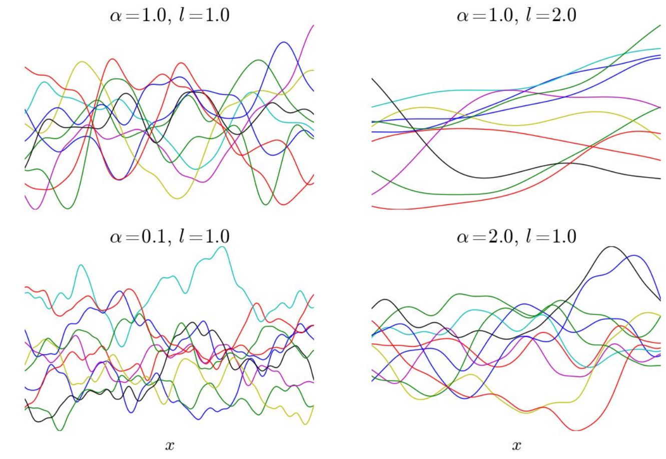
- NAS-Bench-101 [Ying et al. 2019]
- NAS-Bench-201 [Dong & Yang 2019]
- DARTS [Liu et al. 2018]

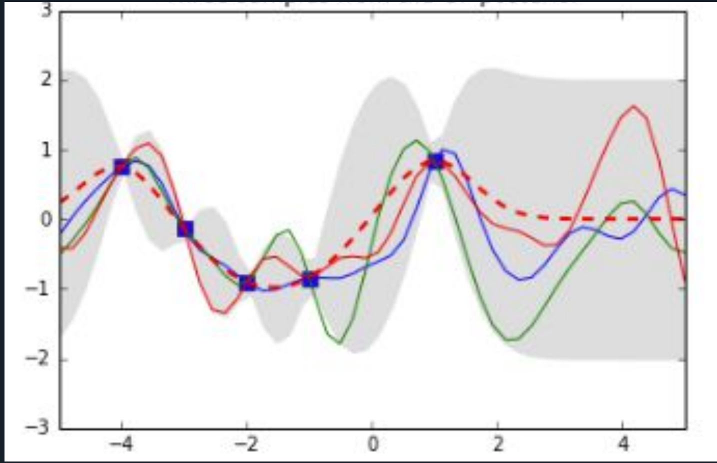


[Ying et al. 2019]

Bayesian optimization

- NASBOT [Kandasamy et al. ‘18], Auto-Keras [Jin et al. ‘18]
- Popular method in HPO, but not straightforward for NAS
  - Gaussian process - scalability
  - Hand-designed *distance function*

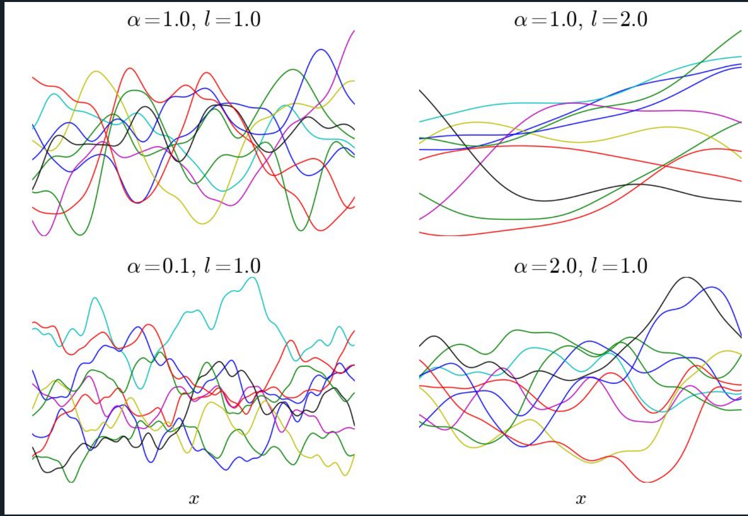


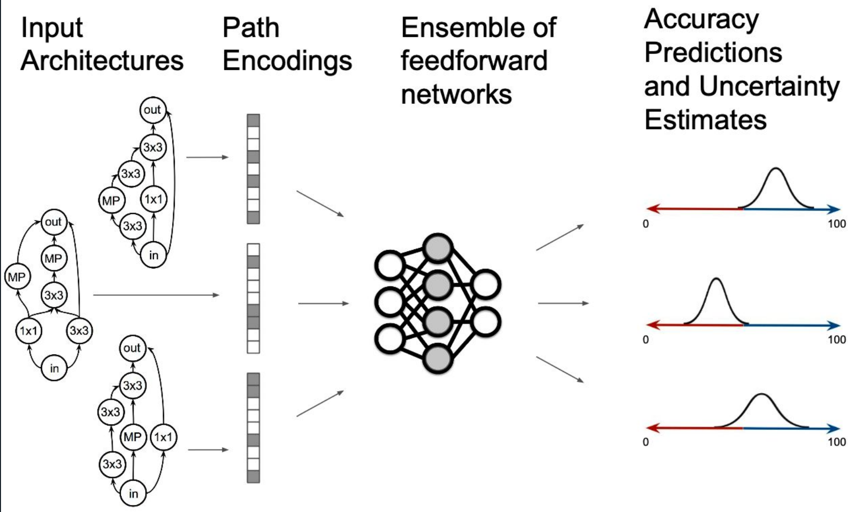


Sources: http://keyonvafa.com/gp-tutorial/, https://katbailey.github.io/post/gaussian-processes-for-dummies/

"BO + Neural Predictor" Framework

- NASGBO [Ma et al. '19], BONAS [Shi et al. '19], BANANAS





Gaussian process

Neural predictor

Source: http://keyonvafa.com/gp-tutorial/

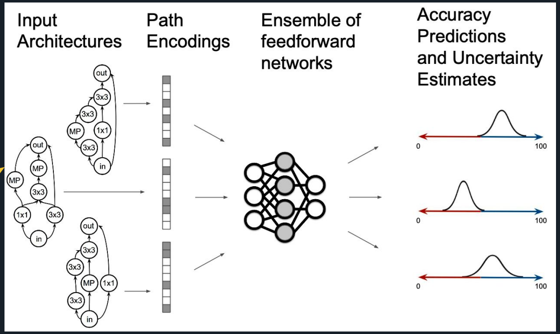
"BO + Neural Predictor" Framework

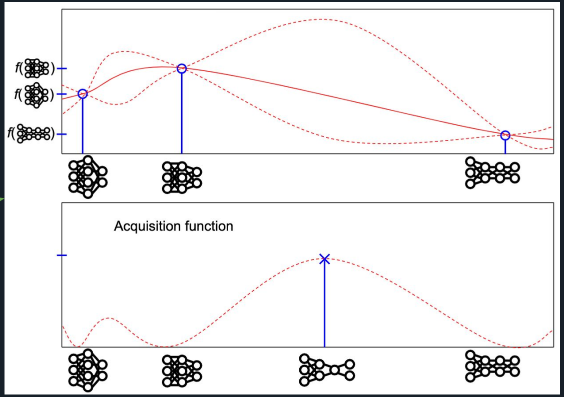
Algorithm 1 BANANAS

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 arches 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 $f(a_{t+1})$.

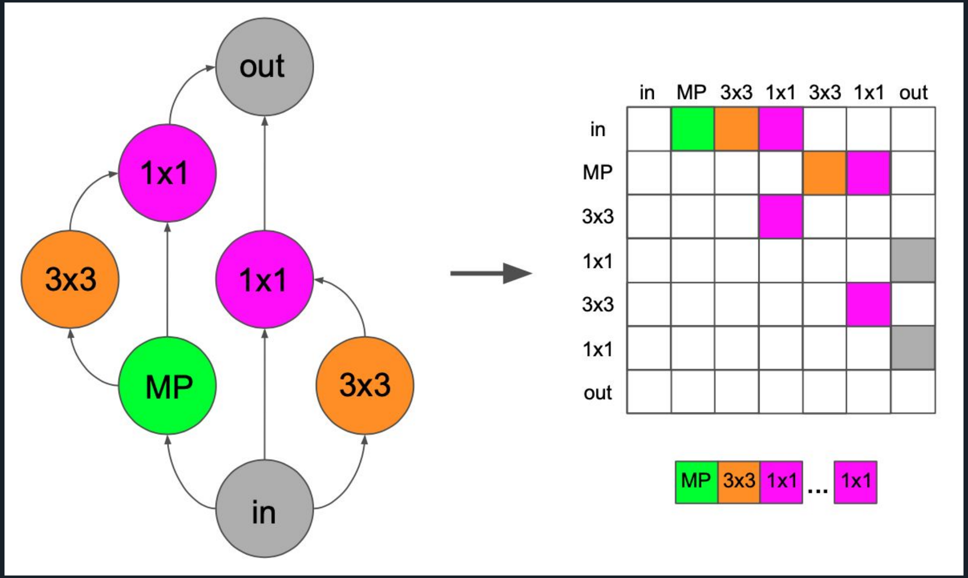
Output: $a^* = \text{argmin}_{t=0,\ldots,T} f(a_t)$.

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

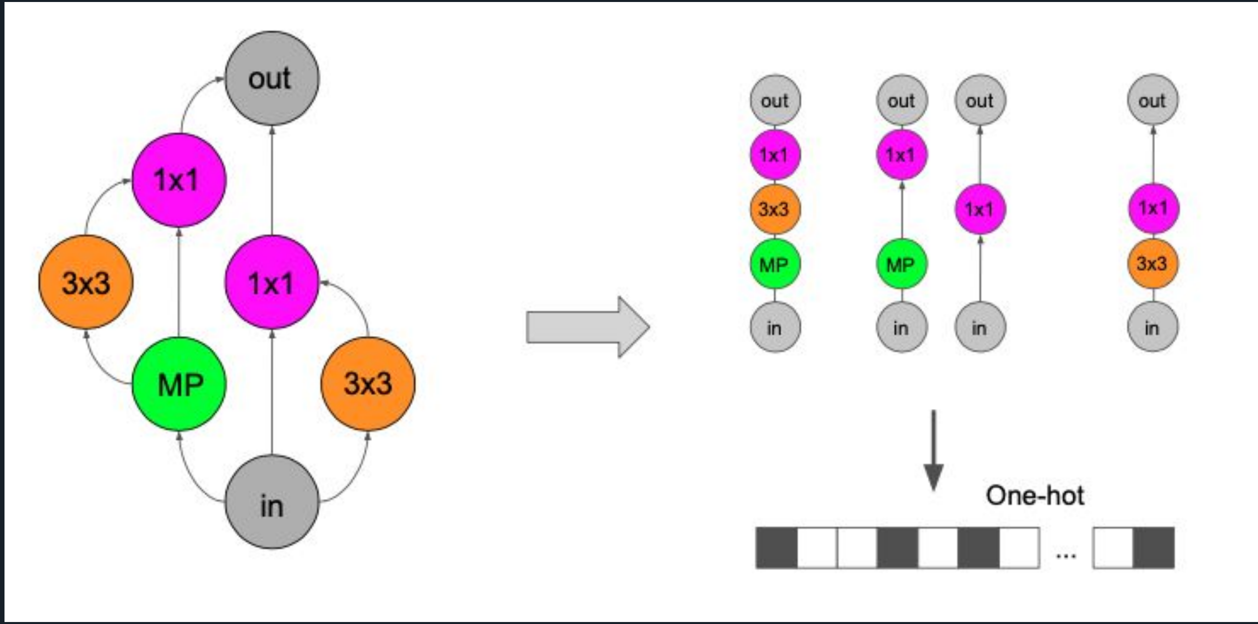
Adjacency Matrix Encoding

Most NAS algorithms use the adjacency matrix encoding

Features are highly dependent on one another



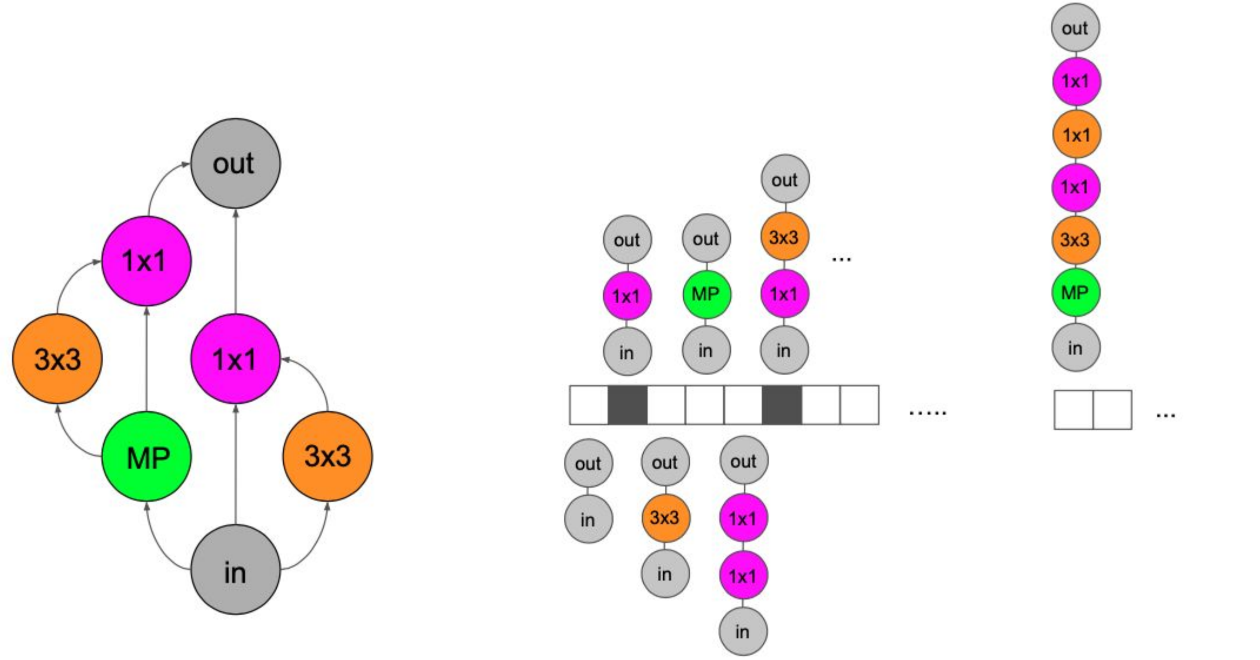
Path Encoding



Each path from input to output is a feature

Much more direct correlation with accuracy

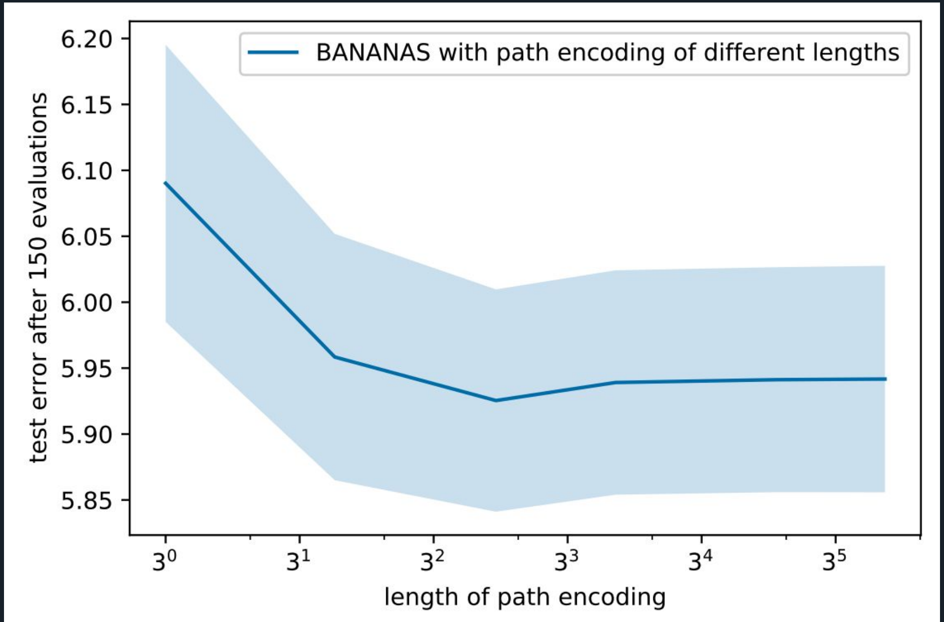
Truncated Path Encoding

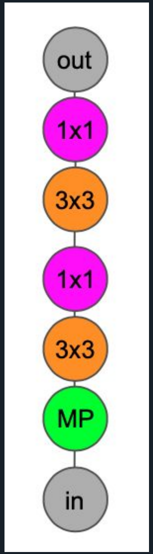


$$3^0 + 3^1 + 3^2 + 3^3 + 3^4 + 3^5 = 364$$

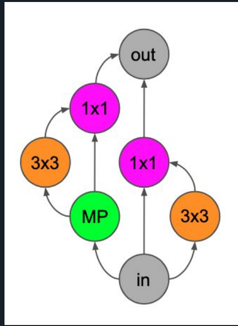
Exponential in the number of nodes

Truncated Path Encoding





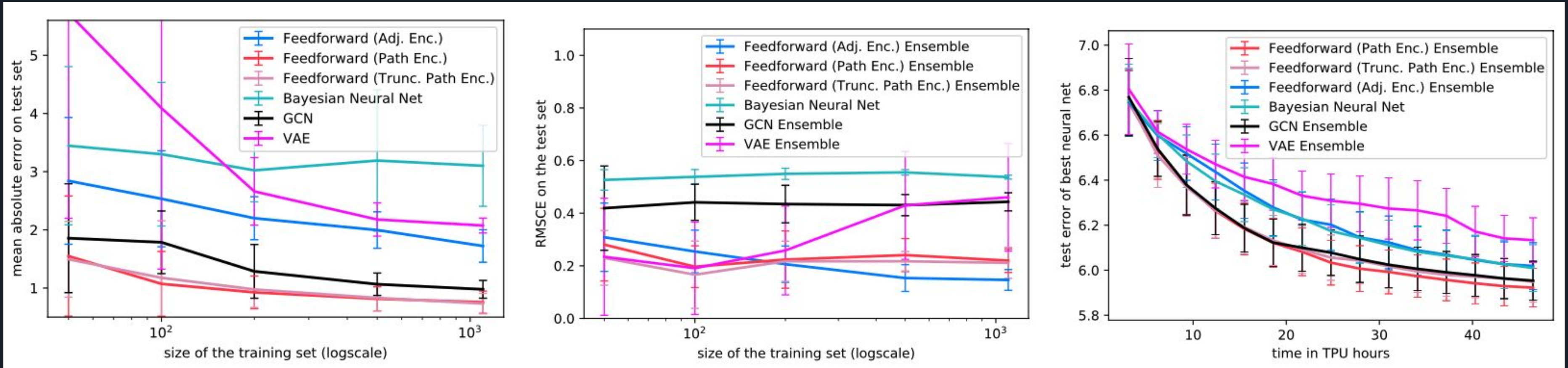
Rare topology



Common topology

Theorem 4.1 (informal). *Given integers $r, c > 0$, there exists an N such that $\forall\, n > N$, there exists a set of n paths $\mathcal{P}'$ such that the probability that $G_{n,n+c,r}$ contains a path not in $\mathcal{P}'$ is less than $\frac{1}{n^2}$.*

Uncertainty prediction + architecture



Standalone (MAE)

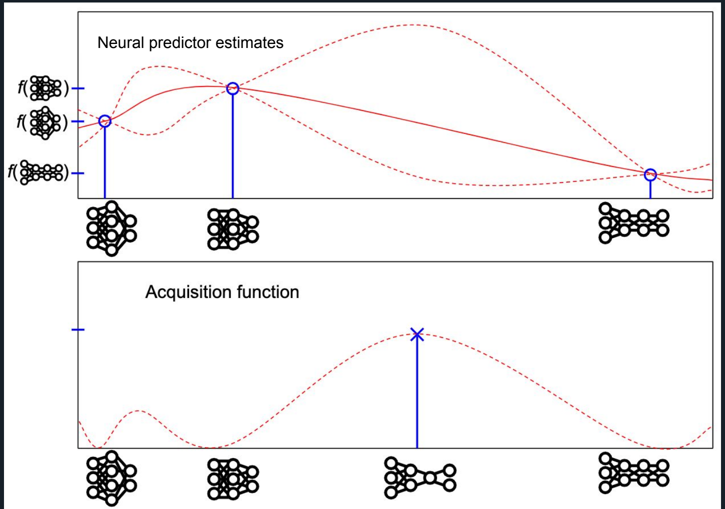
Uncertainty (RMSCE)

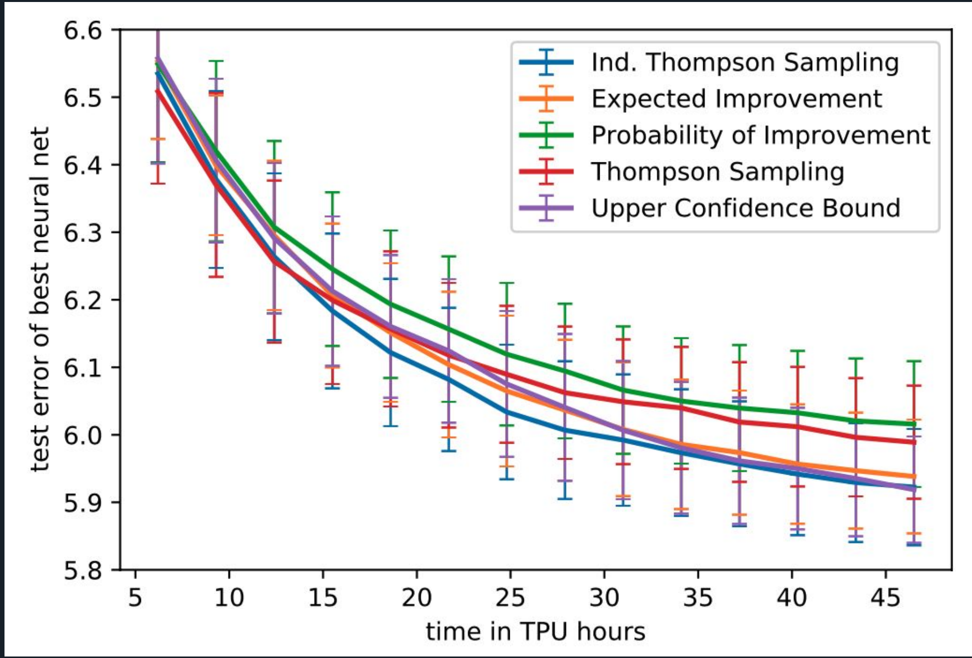
Perf. in BO framework

GraphNN and path-encoding perform best

Acquisition Function

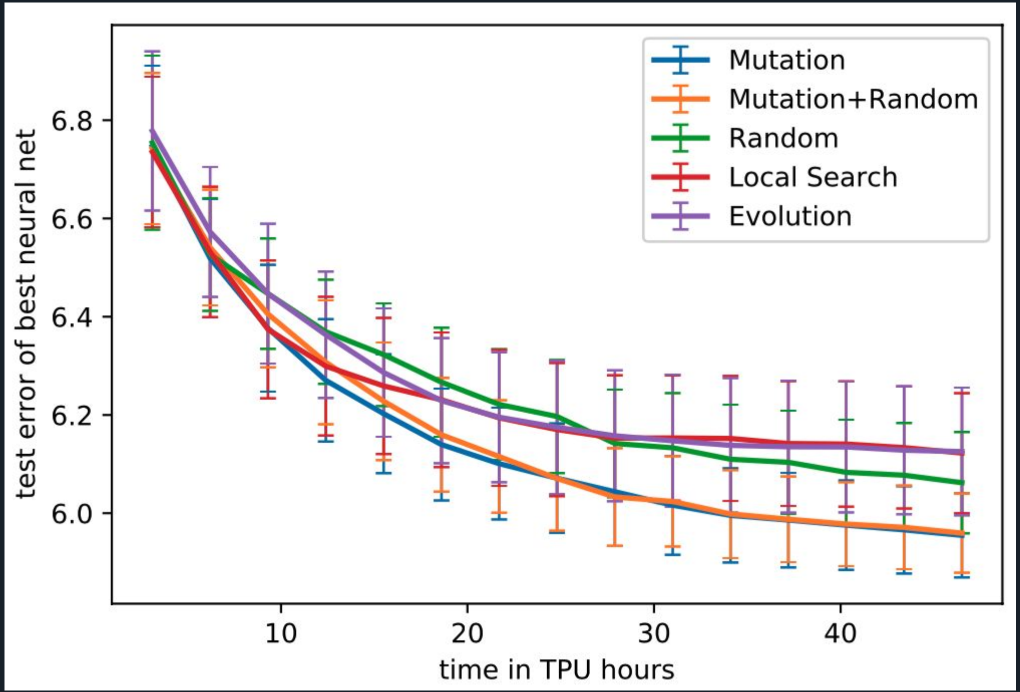
- Exploration vs. exploitation





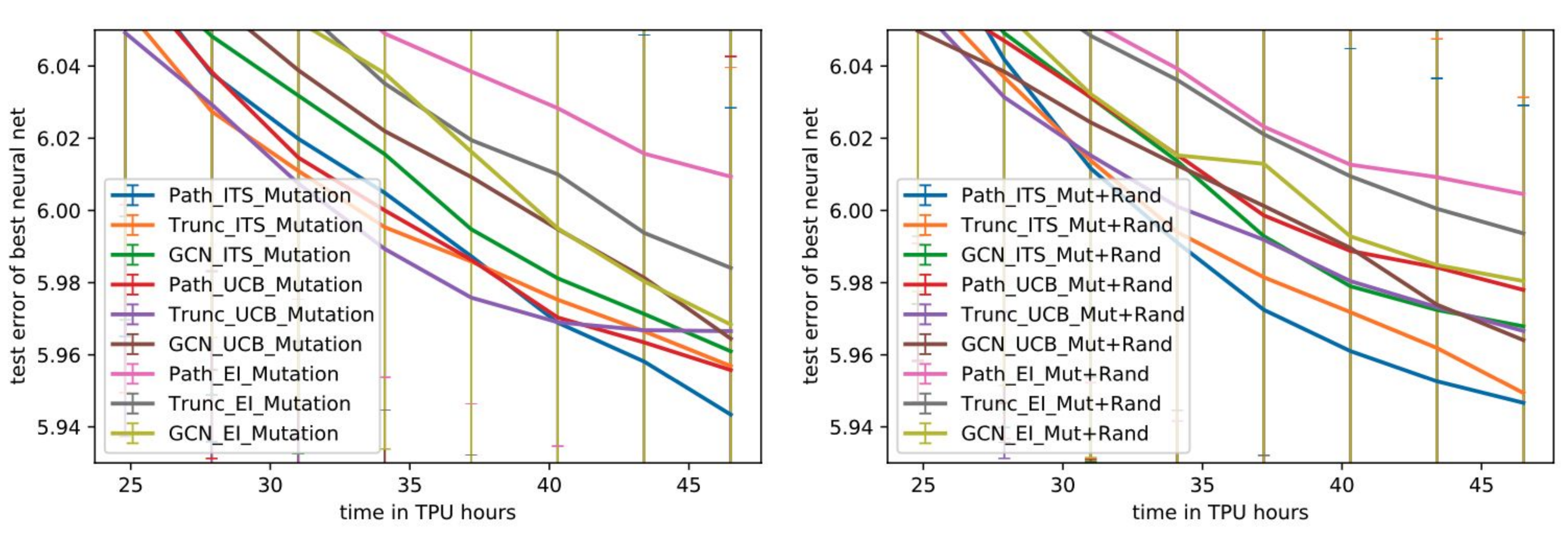
Acquisition Function Optimization

- *Small mutations* of the best architectures is best
- Predictions are most accurate when close to training data



Exhaustive experiment

Path encoding; ITS; Mutation



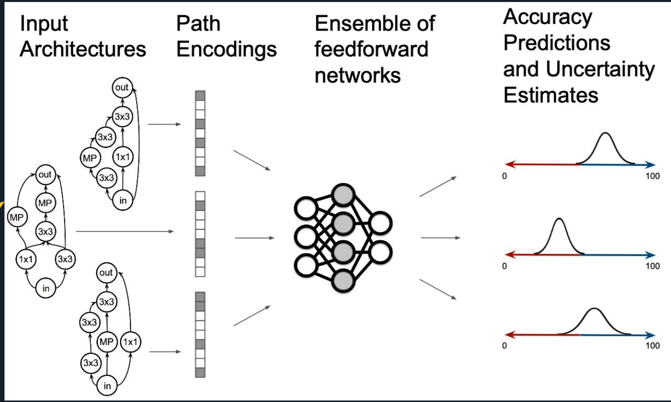
BANANAS

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

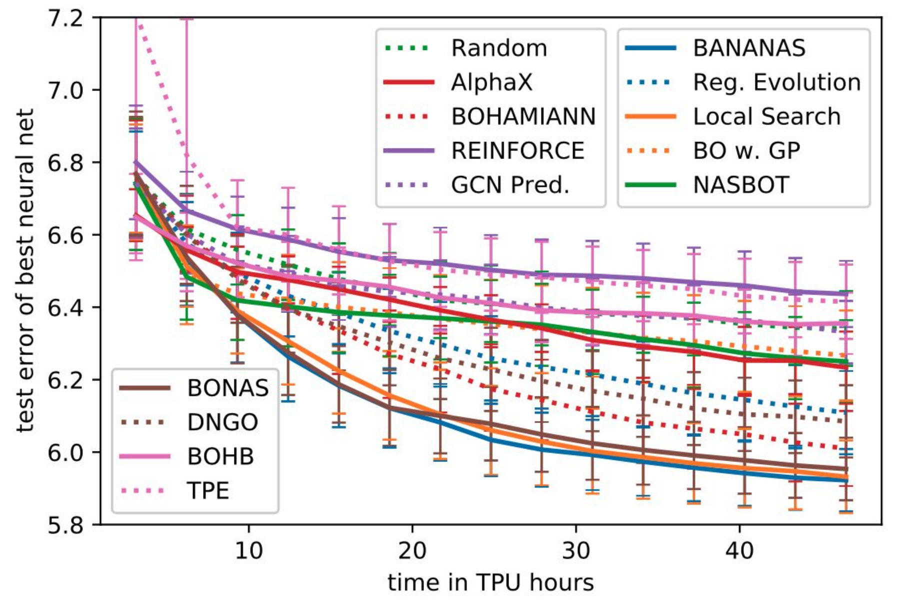


Path encoding, ensemble

Small mutations

Independent Thompson Sampling

NASBench-101 and DARTS Results



NASBench-101

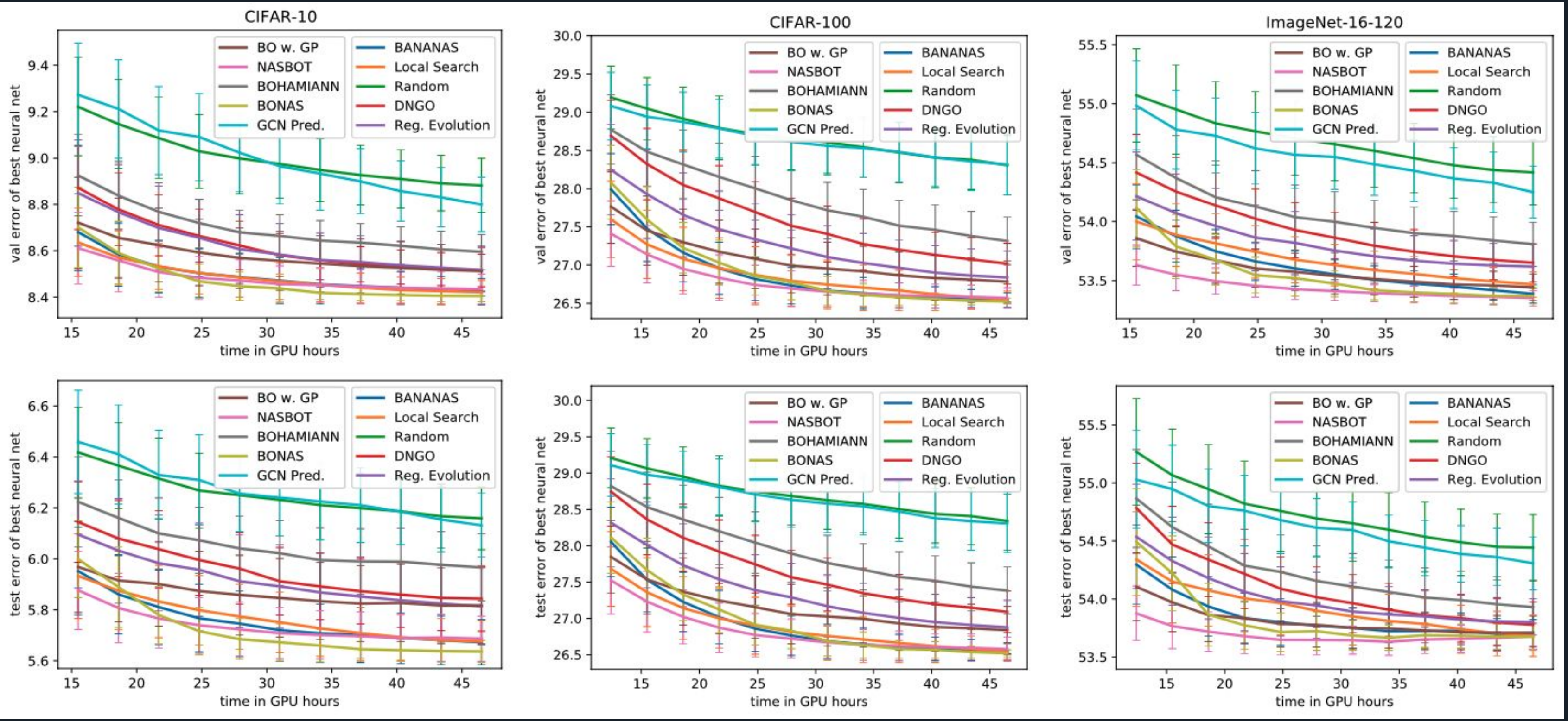
Table 1: Comparison of NAS algorithms on the DARTS search space. The runtime unit is total GPU-days on a Tesla V100.

NAS Algorithm	Source	Avg. Test error	Runtime	Method
Random search	[35]	3.29	4	Random
Local search	[66]	3.49	11.8	Local search
DARTS	[35]	2.76	5	Gradient-based
ASHA	[30]	3.03	9	Successive halving
Random search WS	[30]	2.85	9.7	Random
DARTS	Ours	2.68	5	Gradient-based
ASHA	Ours	3.08	9	Successive halving
BANANAS	Ours	**2.64**	11.8	BO + neural predictor

DARTS

ABACUS.AI

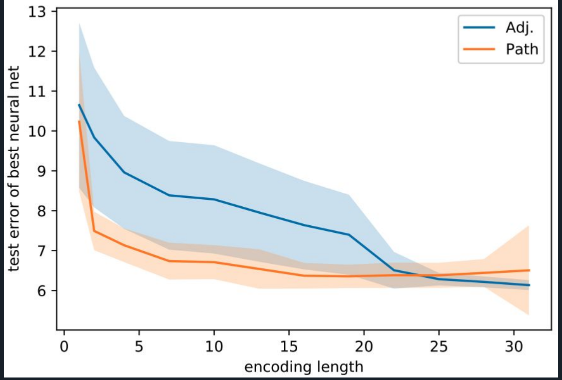
NASBench-201 Results



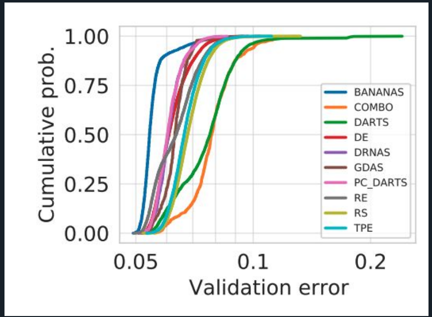
Subsequent Work

NAS Methods	#Queries	Test Accuracy (%)	Encoding	Search Method
Random Search [23]	1000	93.54	Discrete	Random
RL [23]	1000	93.58	Discrete	REINFORCE
BO [23]	1000	93.72	Discrete	Bayesian Optimization
RE [23]	1000	93.72	Discrete	Evolution
NAO [14]	1000	93.74	Supervised	Gradient Decent
BANANAS [49]	500	94.08	Supervised	Bayesian Optimization
RL (ours)	400	93.74	Supervised	REINFORCE
BO (ours)	400	93.79	Supervised	Bayesian Optimization
arch2vec-RL	**400**	**94.10**	Unsupervised	REINFORCE
arch2vec-BO	400	94.05	Unsupervised	Bayesian Optimization

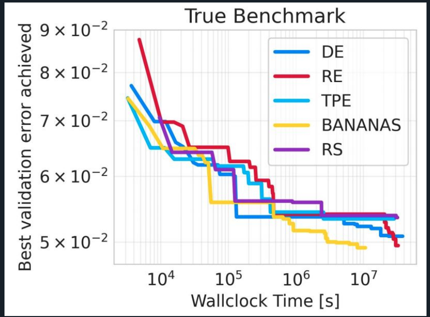
[Yan et al. ‘20]



[White et al. ‘20]



[Siems et al. ‘20]



[Siems et al. ‘20]

Algorithm	Test Error (in %)
TPE	6.43 +- 0.16
BOHB	6.40 +- 0.12
Random Search	6.36 +- 0.12
Alpha X	6.31 +- 0.13
NASBOT	6.35 +- 0.10
Reg Evolution	6.20 +- 0.13
ReMAADE	6.15 +- 0.13
BANANAS	5.77 +- 0.31

[Krishna et al. ‘20]

Conclusion

- “BO + Neural Predictor” is a powerful NAS framework
  - Encoding, surrogate model, acquisition function, acquisition function optimization
- BANANAS is a performant instantiation of the framework

https://github.com/naszilla/naszilla

Thanks!

ABACUS.AI

ABACUS.AI