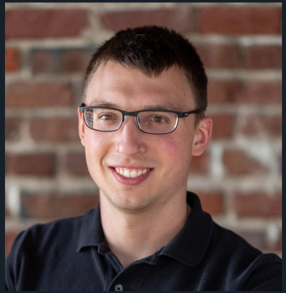
NAS-Bench-x11
and the Power of Learning Curves

Shen Yan
Michigan State University

Colin White
Abacus.AI

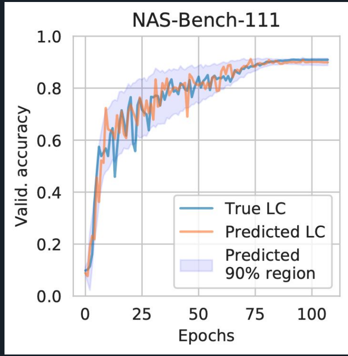
Yash Savani
Carnegie Mellon University

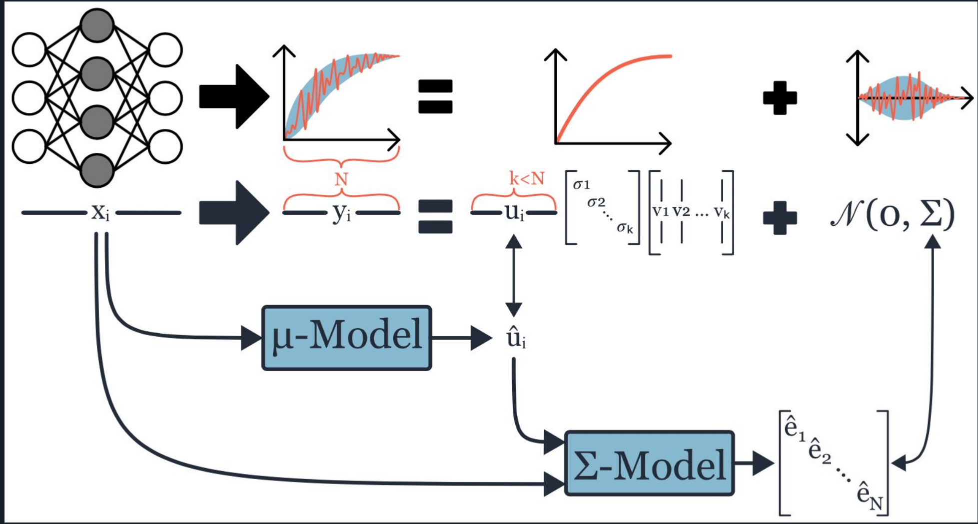
Frank Hutter
University of Freiburg
Bosch Center for AI

One-slide summary:

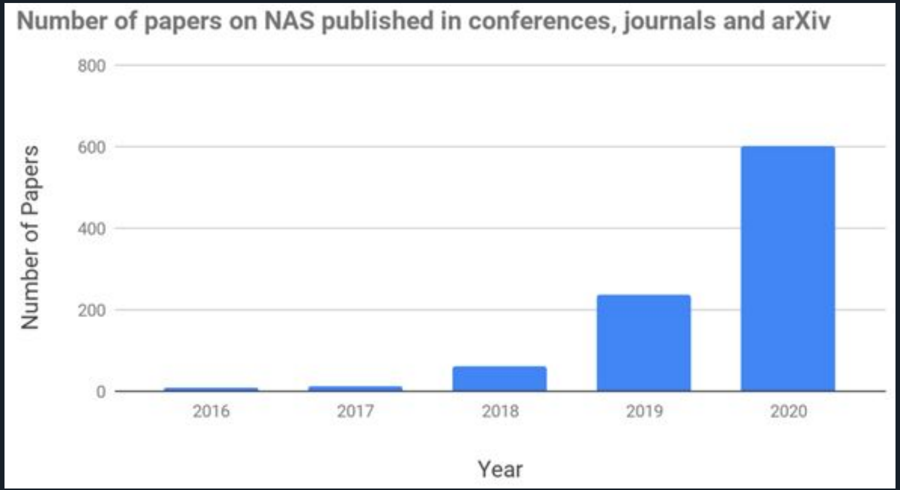
- We give a new technique to create surrogate NAS benchmarks **with realistic learning curves**
- We create NAS-Bench-111, NAS-Bench-311, and NAS-Bench-NLP11
- We use these to show popular NAS algorithms can be further improved by adding learning curve extrapolation





Neural architecture search

- Notoriously challenging to give fair comparisons [Li & Talkwalkar 2019], [Hutter & Lindauer 2020]
  - Computationally intensive
  - No common search spaces

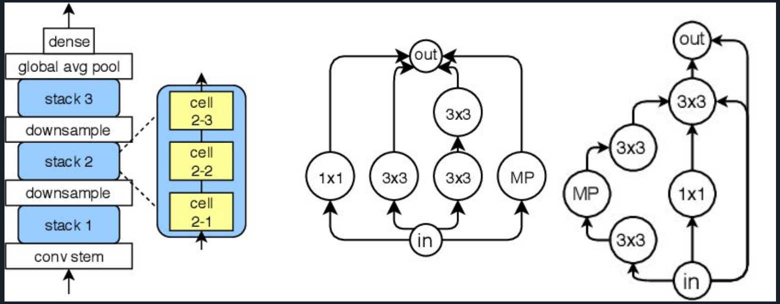


Tabular NAS Benchmarks

Train all architectures in a search space

Used to **simulate** NAS experiments

- NAS-Bench-101 [Ying et al. 2019]
  - Size 423k
- NAS-Bench-201 [Dong & Yang 2019]
  - Size 15k

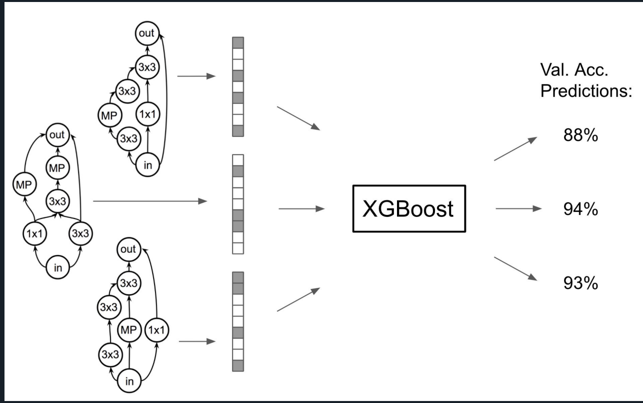



```python
# Load the data from file (this will take some time)
nasbench = api.NASBench('/path/to/nasbench.tfrecord')

# Create an Inception-like module (5x5 convolution replaced with two 3x3
# convolutions).
model_spec = api.ModelSpec(
    # Adjacency matrix of the module
    matrix=[[0, 1, 1, 1, 0, 1, 0],    # input layer
            [0, 0, 0, 0, 0, 0, 1],    # 1x1 conv
            [0, 0, 0, 0, 0, 0, 1],    # 3x3 conv
            [0, 0, 0, 0, 1, 0, 0],    # 5x5 conv (replaced by two 3x3's)
            [0, 0, 0, 0, 0, 0, 1],    # 5x5 conv (replaced by two 3x3's)
            [0, 0, 0, 0, 0, 0, 1],    # 3x3 max-pool
            [0, 0, 0, 0, 0, 0, 0]],   # output layer
    # Operations at the vertices of the module, matches order of matrix
    ops=[INPUT, CONV1X1, CONV3X3, CONV3X3, CONV3X3, MAXPOOL3X3, OUTPUT])

# Query this model from dataset, returns a dictionary containing the metrics
# associated with this model.
data = nasbench.query(model_spec)
```

NAS-Bench-101

Surrogate NAS Benchmarks

- NAS-Bench-301 [Siems et al. 2020]
  - Based on DARTS search space
  - Size 10^{18}

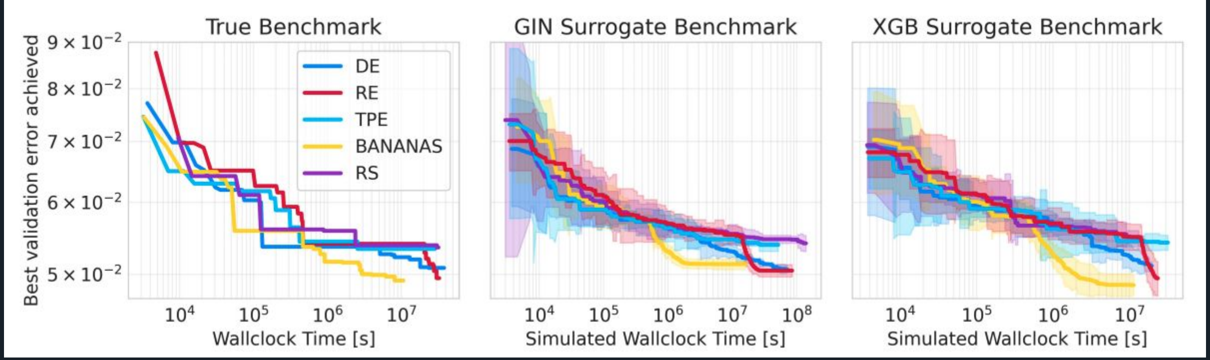


Enables much larger NAS Benchmarks

	NAS methods	# eval
	RS (Bergstra & Bengio, 2012)	23746
Evolution	DE (Awad et al., 2020)	7275
	RE (Real et al., 2019)	4639
BO	TPE (Bergstra et al., 2011)	6741
	BANANAS (White et al., 2019)	2243
	COMBO (Oh et al., 2019)	745
One-Shot	DARTS (Liu et al., 2019b)	2053
	PC-DARTS (Xu et al., 2020)	1588
	DrNAS (Chen et al., 2020)	947
	GDAS (Dong & Yang, 2019)	234

Table 2: NAS methods used to cover the search space.

Training set



NAS Benchmarks

Benchmark	Size	Queryable	Based on	Full train info
NAS-Bench-101	423k	✓		✗
NAS-Bench-201	6k	✓		✓
NAS-Bench-NLP	10^{53}	✗		✗
NAS-Bench-301	10^{18}	✓	DARTS	✗
NAS-Bench-ASR	8k	✓		✓

No learning curves - can only simulate black-box algorithms!

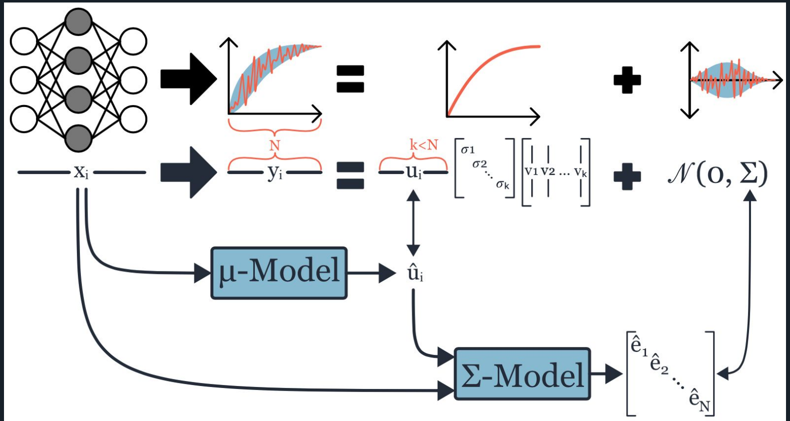
NAS Benchmarks

Benchmark	Size	Queryable	Based on	Full train info
NAS-Bench-101	423k	✓		✗
NAS-Bench-201	6k	✓		✓
NAS-Bench-NLP	10^{53}	✗		✗
NAS-Bench-301	10^{18}	✓	DARTS	✗
NAS-Bench-ASR	8k	✓		✓
NAS-Bench-111	423k	✓	NAS-Bench-101	✓
NAS-Bench-311	10^{18}	✓	DARTS	✓
NAS-Bench-NLP11	10^{53}	✓	NAS-Bench-NLP	✓

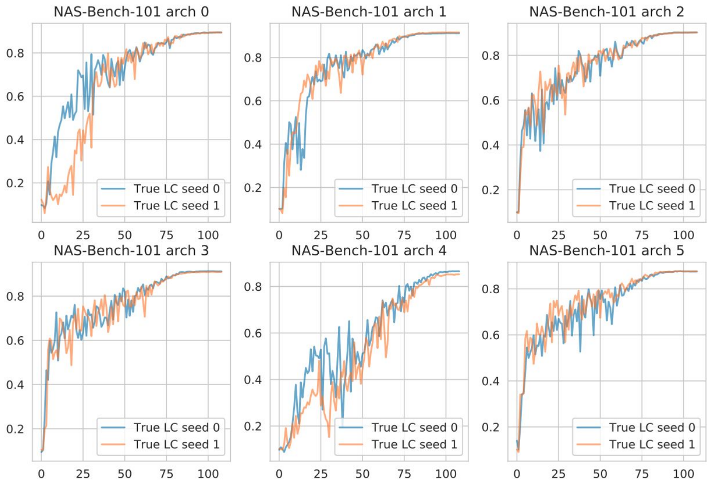
No learning curves - can only simulate black-box algorithms!

Roadmap

- Motivation
- Generating Learning Curves
- Evaluation
- The Power of Learning Curves
- Conclusion



Generating Learning Curves



Generating realistic noise is critical

We can't just use a surrogate model to predict the entire learning curve

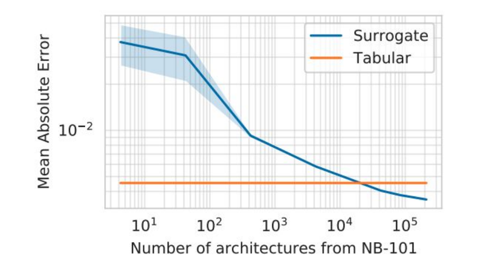
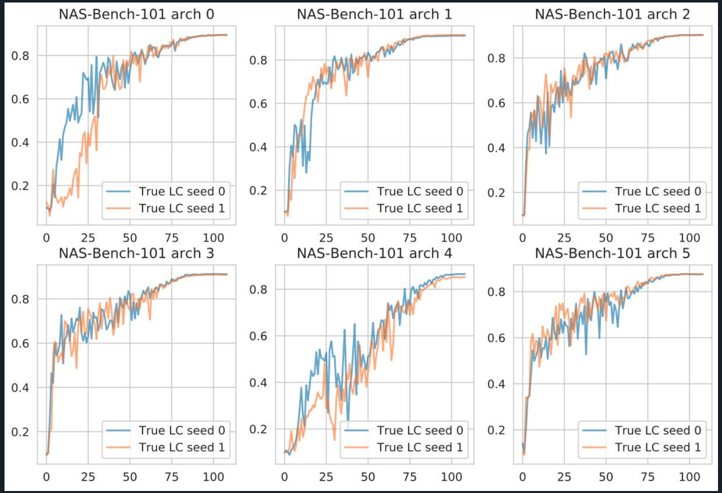


Figure 1: Number of architectures used for training the GIN surrogate model vs MAE on the NAS-Bench-101 dataset.

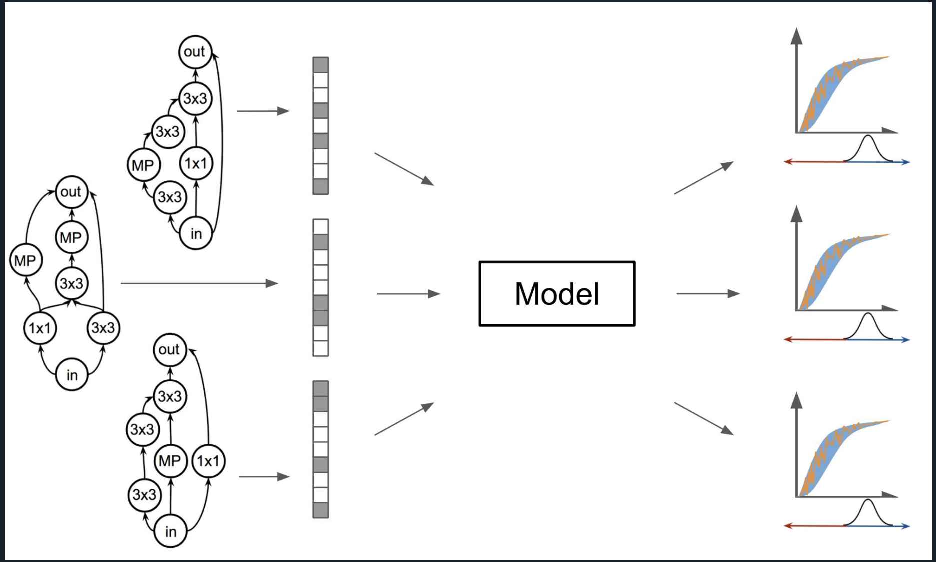
NAS-Bench-301

This would lead to de-noised learning curves

Generating Learning Curves

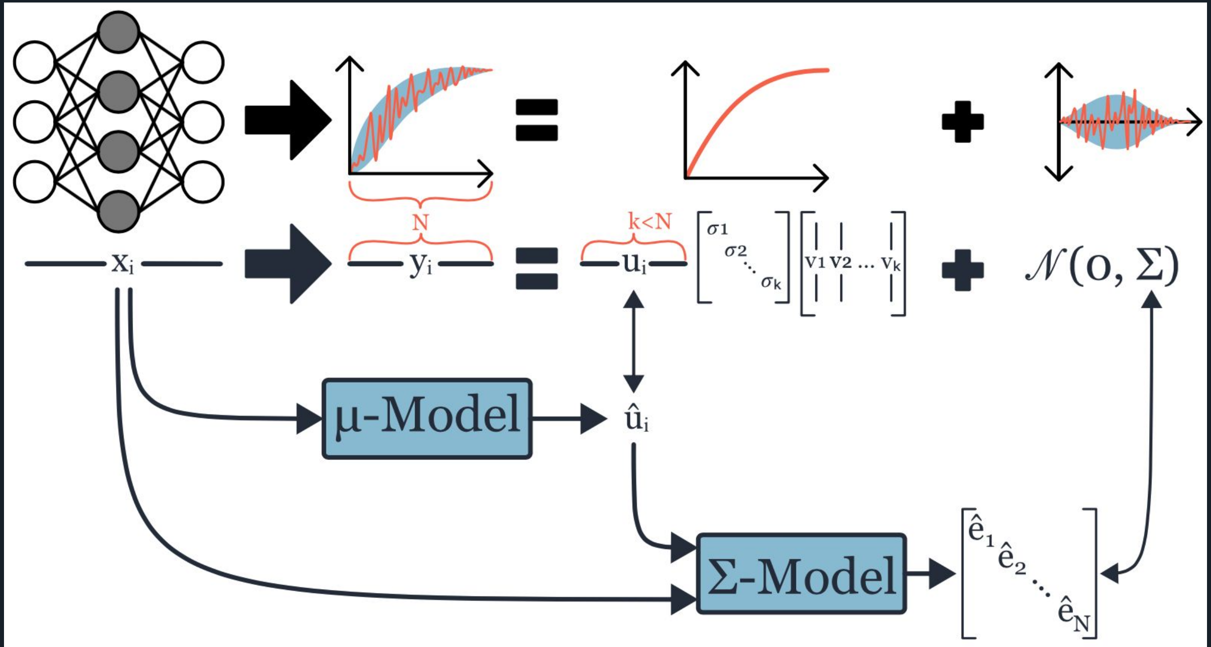


Generating realistic noise is critical



Goal: given architecture encoding, predict a distribution

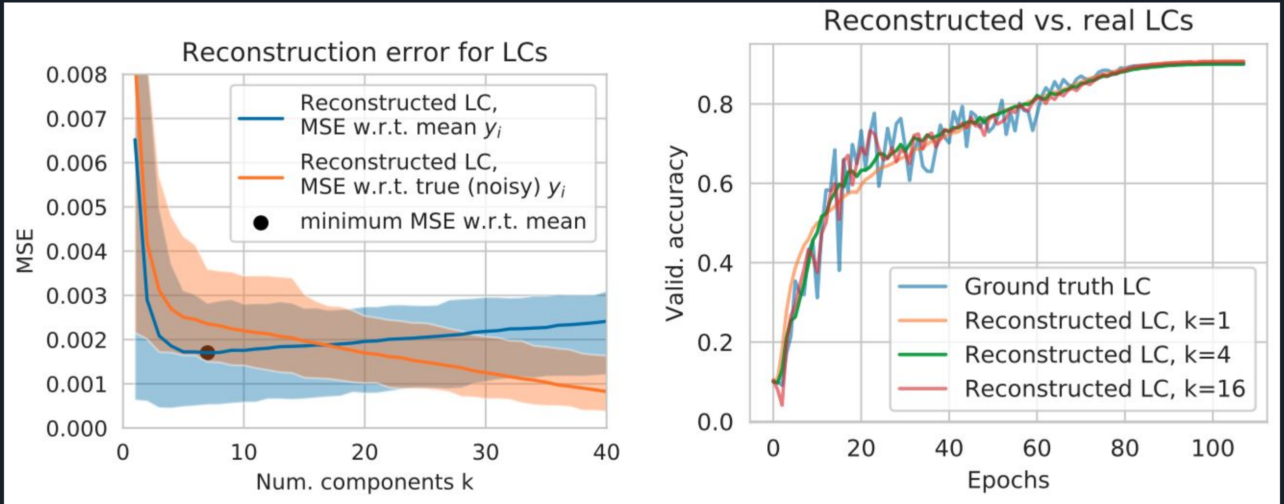
Two-part technique



(1) Predict mean LC

(2) Predict noise

Predicting the mean learning curve

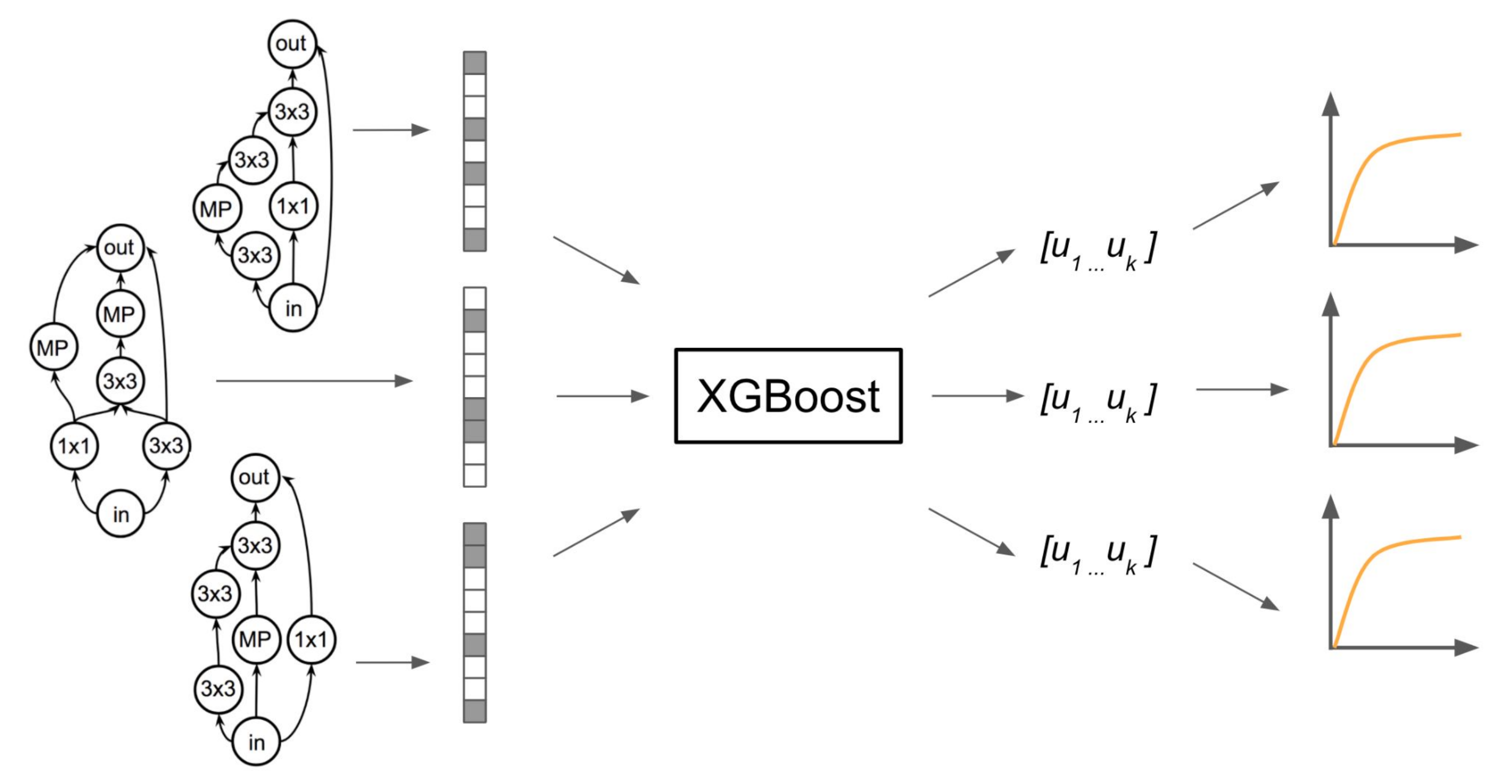


Compress the learning curves from the training set

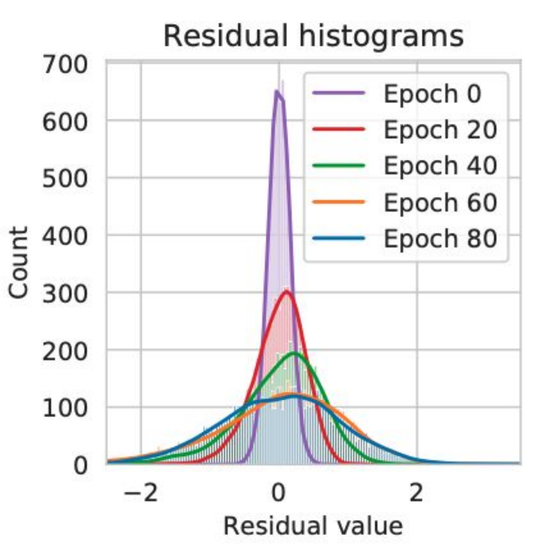
Predict only the top 5 principal components

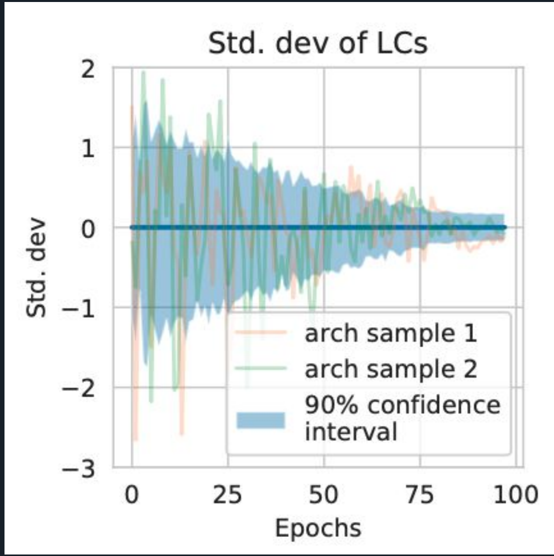
SVD helps to reduce the noise

Predicting the mean learning curves



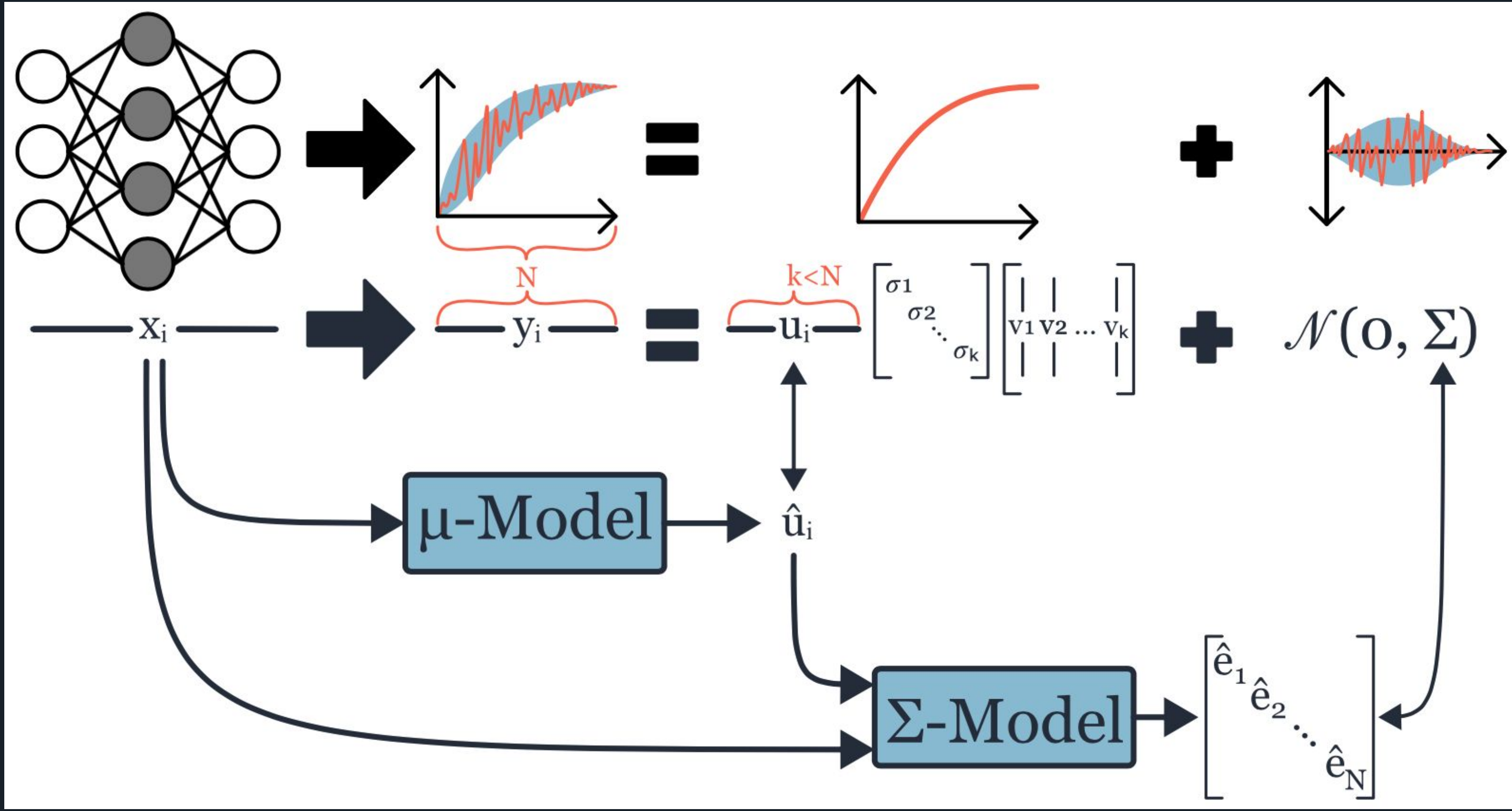
Noise modeling





Compute the residuals, then use a sliding window to approximate STDev’s

Full technique



NAS-Bench-x11

We create

- NAS-Bench-111
  - Created a new training set of size 1500
- NAS-Bench-311
  - Used the 60k architectures from NAS-Bench-301
- NAS-Bench-NLP11
  - Used the 14k architectures from NAS-Bench-NLP
  - Improved by adding acc's from first three epochs

API and surrogate benchmarks: https://github.com/automl/NAS-Bench-x11

Evaluation (mean learning curves)

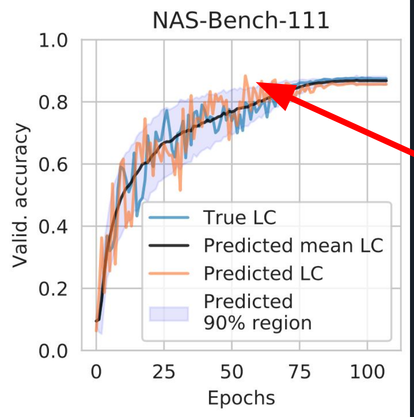
	Avg. R^2	Final R^2	Avg. KT	Final KT
Tabular (1 seed)	0.553	0.778	0.529	0.654
Tabular (2 seeds)	0.672	0.845	0.581	0.709
Tabular (3 seeds)	0.707	0.854	0.602	0.718
Tabular (4 seeds)	0.727	0.870	0.617	0.732
NAS-Bench-311	0.715	0.838	0.628	0.711

Similar rank correlation to a 3-seed tabular benchmark

Evaluation (noise model)

Benchmark	Avg. R^2	Final R^2	Avg. KT	Final KT	Avg. KL	Final KL
NAS-Bench-111	0.529	0.630	0.531	0.645	2.016	1.061
NAS-Bench-111 (w. accs)	0.630	0.853	0.611	0.794	1.710	0.926
NAS-Bench-311	0.779	0.800	0.728	0.788	0.905	0.600
NAS-Bench-NLP11	0.326	0.314	0.505	0.475	-	-
NAS-Bench-NLP11 (w. accs)	0.878	0.895	0.878	0.844	-	-

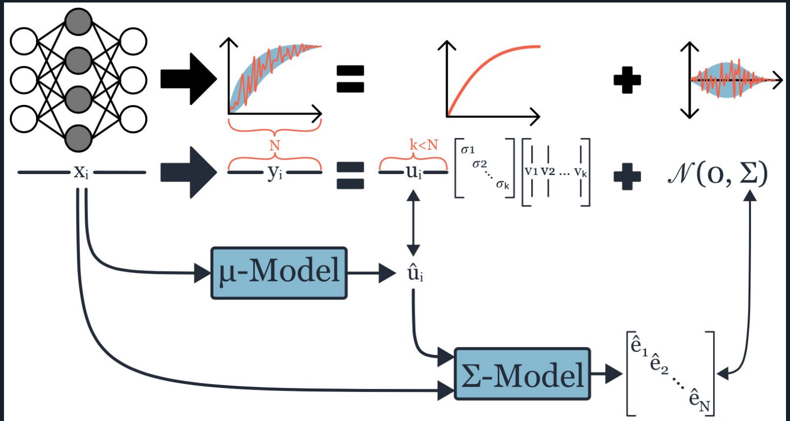
Spike anomalies



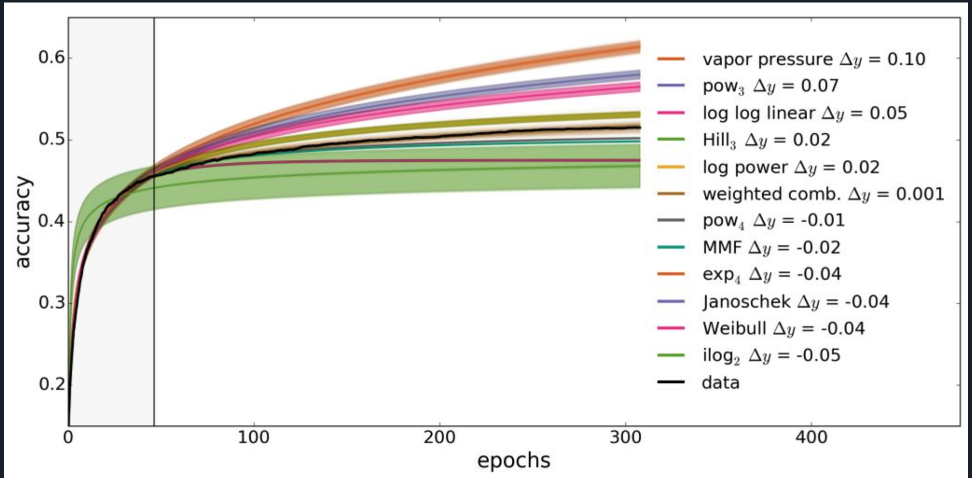
Compare probability of anomalies of surrogates vs. real data

Roadmap

- Motivation
- Generating Learning Curves
- Evaluation
- The Power of Learning Curves
- Conclusion



Learning Curve Extrapolation (LCE)



[Domhan et al. 2015], [Baker et al. 2017]

Used to speed up black-box NAS algorithms

- Reg. Evolution, BANANAS, local search, etc

Use LCE to stop training bad architectures early

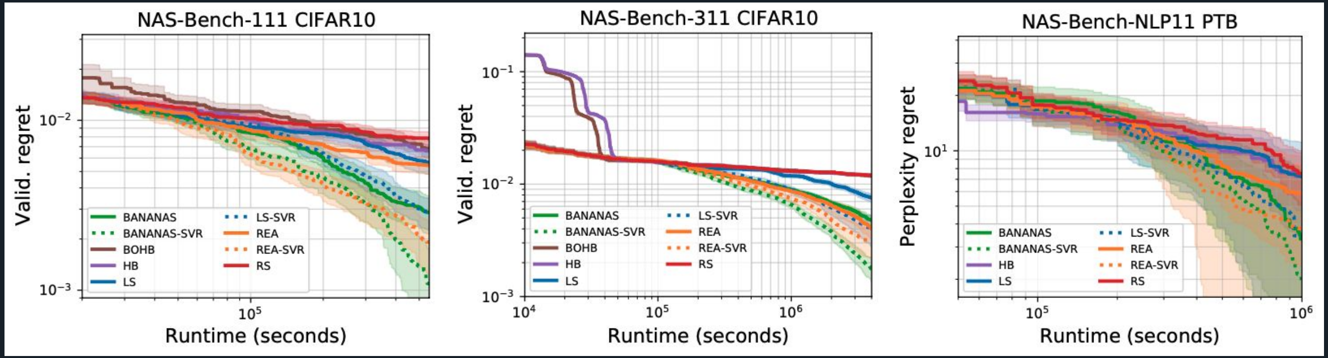
LCE Framework

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



Conclusions & Future Work

- New technique: surrogate benchmarks with full training information
  - Learning curves with realistic noise
- NAS-Bench-111, NAS-Bench-311, NAS-Bench-NLP11
- Framework to add LCE to black-box NAS algorithms

Code: https://github.com/automl/NAS-Bench-x11

Thanks!