

# Neural Architecture Search

REDS

Charles VIN, Mathis KOROGLU

Sorbonne Université

February 11, 2024

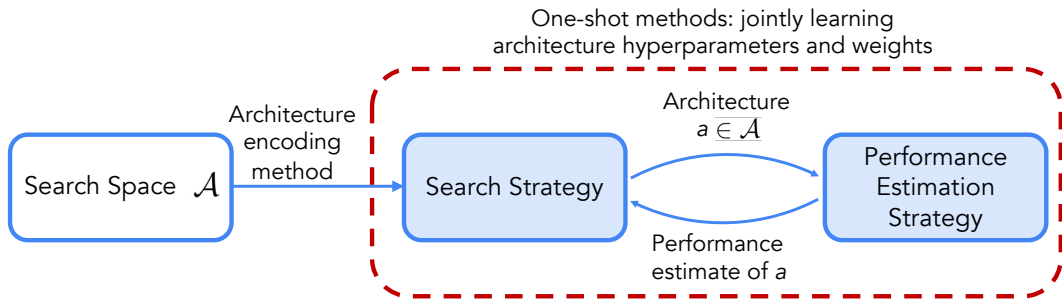
# Overview

---

## 1. Neural architecture search

- 1.1 Search space
- 1.2 Search strategy
- 1.3 Performance estimation strategy
- 1.4 Evaluation

## 2. Contribution



**Figure:** Overview of NAS.

A search strategy iteratively selects architectures (typically by using an architecture encoding method) from a predened search space  $\mathcal{A}$ .

The architectures are passed to a performance estimation strategy, which returns the performance estimate to the search strategy.

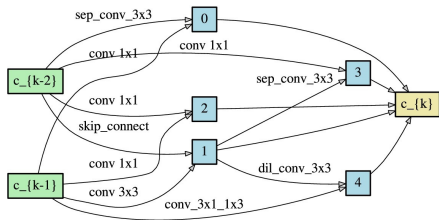
For one-shot methods, the search strategy and performance estimation strategy are inherently coupled.

# Search space

## Definition

The set of all architectures that the NAS algorithm is allowed to select.

- Size: from a few thousand to over  $10^{20}$ .
  - Reduction: adding domain knowledge.
- Introduce human bias →  $\times$  reduce the chance of finding truly novel architecture.



**Figure:** Architecture directed acyclic graph (DAG) with operation on nodes

# Search strategy

---

## Definition

The optimization technique used to find a high-performing architecture in the search space.

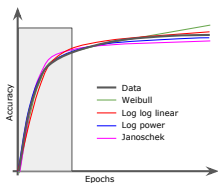
- Black-box optimization techniques : RL, Bayesian optimization, evolutionary search.
- One-shot techniques: supernet-hypernet based methods.

# Performance estimation strategy

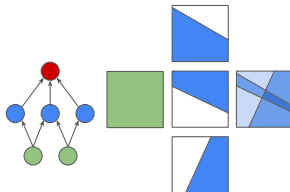
## Definition

Any method used to quickly predict the performance of neural architectures in order to avoid fully training the architecture.

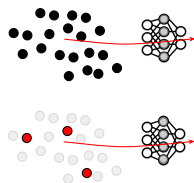
- Full training & evaluation.
- Performance estimation strategy.



Learning Curve  
Extrapolation



Zero-Cost Proxies



Subset Selection

# Benchmarks

## Definition

A NAS benchmark is defined as a dataset with a fixed train-test split, a search space, and a fixed evaluation pipeline for training the architectures.

- Tabular benchmarks: precomputed evaluations for all possible architecture in the search space.
- Allow to **simulate** hundreds of trials !
  - No more training.
  - Statistically significant comparisons (simulation with multiple seeds).
- Specialization: precomputed evaluation for NLP tasks, protein folding, astronomy imaging, for vision dataset (CIFAR-10(0), ImageNet, ...)
- Most popular NAS benchmarks
  - NAS-Bench-101/202
  - Surr-NAS-Bench-DARTS
  - TransNAS-Bench-101
  - NAS-Bench-Suite

# Evaluation

Table 2: **Comparison with Transferable NAS on NB201 Serach Space.** We present the accuracy achieved on four unseen datasets. Additionally, we provide the number of neural architectures (**Trained Archs**) that are actually trained to achieve accuracy. The accuracies are reported with 95% confidence intervals over 3 runs.

Type	Method	CIFAR-10		CIFAR-100		Aircraft		Oxford-IIIT Pets	
		Accuracy (%)	Trained Archs	Accuracy (%)	Trained Archs	Accuracy (%)	Trained Archs	Accuracy (%)	Trained Archs
One-shot NAS*	ResNet (He et al., 2016)	93.97 $\pm$ 0.00	N/A	70.86 $\pm$ 0.00	N/A	47.01 $\pm$ 1.16	N/A	25.58 $\pm$ 3.43	N/A
	RS (Bergstra & Bengio, 2012)	93.70 $\pm$ 0.36	> 500	71.04 $\pm$ 1.07	> 500	-	-	-	-
	REA (Real et al., 2019)	93.92 $\pm$ 0.30	> 500	71.84 $\pm$ 0.99	> 500	-	-	-	-
	REINFORCE (Williams, 1992)	93.85 $\pm$ 0.37	> 500	71.71 $\pm$ 1.09	> 500	-	-	-	-
	RSPTS (Li & Talwalkar, 2019)	84.07 $\pm$ 3.61	N/A	52.31 $\pm$ 5.77	N/A	42.19 $\pm$ 3.88	N/A	22.91 $\pm$ 1.65	N/A
	SETN (Dong & Yang, 2019a)	87.64 $\pm$ 0.00	N/A	59.09 $\pm$ 0.24	N/A	44.84 $\pm$ 3.96	N/A	25.17 $\pm$ 1.68	N/A
	GDAS (Dong & Yang, 2019b)	93.61 $\pm$ 0.09	N/A	70.70 $\pm$ 0.30	N/A	53.52 $\pm$ 0.48	N/A	24.02 $\pm$ 2.75	N/A
	PC-DARTS (Xu et al., 2020)	93.66 $\pm$ 0.17	N/A	66.64 $\pm$ 2.34	N/A	26.33 $\pm$ 3.40	N/A	25.31 $\pm$ 1.38	N/A
	DrNAS (Chen et al., 2021)	94.36 $\pm$ 0.00	N/A	<b>73.51<math>\pm</math>0.00</b>	N/A	46.08 $\pm$ 7.00	N/A	26.73 $\pm$ 2.61	N/A
	BOHB (Falkner et al., 2018)	93.61 $\pm$ 0.52	> 500	70.85 $\pm$ 1.28	> 500	-	-	-	-
BO-based NAS	GP-UCB	<b>94.37<math>\pm</math>0.00</b>	58	73.14 $\pm$ 0.00	100	41.72 $\pm$ 0.00	40	40.60 $\pm$ 1.10	11
	BANANAS (White et al., 2021a)	<b>94.37<math>\pm</math>0.00</b>	46	<b>73.51<math>\pm</math>0.00</b>	88	41.72 $\pm$ 0.00	40	40.15 $\pm$ 1.59	17
	NASBOWL (Ru et al., 2021)	94.34 $\pm$ 0.00	100	<b>73.51<math>\pm</math>0.00</b>	87	53.73 $\pm$ 0.83	40	41.29 $\pm$ 1.10	17
	HEBO (Cowen-Rivers et al., 2022)	94.34 $\pm$ 0.00	100	72.62 $\pm$ 0.20	100	49.32 $\pm$ 6.10	40	40.55 $\pm$ 1.15	18
Transferable NAS	TNAS (Shala et al., 2023)	<b>94.37<math>\pm</math>0.00</b>	29	<b>73.51<math>\pm</math>0.00</b>	59	59.15 $\pm$ 0.58	26	40.00 $\pm$ 0.00	6
	MetaD2A (Lee et al., 2021a)	<b>94.37<math>\pm</math>0.00</b>	100	73.34 $\pm$ 0.04	100	57.71 $\pm$ 0.20	40	39.04 $\pm$ 0.20	40
	DiffusionNAG (Ours)	<b>94.37<math>\pm</math>0.00</b>	<b>5</b>	<b>73.51<math>\pm</math>0.00</b>	<b>5</b>	<b>59.63<math>\pm</math>0.92</b>	<b>2</b>	<b>41.32<math>\pm</math>0.84</b>	<b>2</b>

Figure: Result table example



# Evaluation

**Table 8**

Comparison of the evaluation results on CIFAR-10 and CIFAR-100.

Architecture	Test Err. (%)		Params (.M)	Search Cost (GPU-days)	Search Method
	C-10	C-100			
NASNet-A [8]	2.65	-	3.3	1800	RL
AmoebaNet-A [20]	3.34	-	3.2	3150	Evolution
PNAS [40]	3.41	-	3.2	225	SMBO
RelativeNAS [41]	2.34	15.86	3.93	0.4	Evolution
DARTS (first order) [10]	3	17.76	3.3	1.5	Gradient
SNAS + mild constraint [42]	2.98	-	2.9	1.5	Gradient
ProxylessNAS [29]	2.08	-	5.7	4	Gradient
P-DARTS <sup>†</sup> (C-10) [18]	2.5	16.55	3.4	0.3	Gradient
P-DARTS <sup>†</sup> (C-100) [18]	2.62	15.92	3.6	0.3	Gradient
P-DARTS <sup>†</sup> (C-10-Large) [18]	2.25	15.27	10.5	0.3	Gradient
P-DARTS <sup>†</sup> (C-100-Large) [18]	2.43	14.64	11	0.3	Gradient
<b>Ours<sup>†</sup></b> (C-10)	2.47 ± 0.03	-	2.04	1.3*	RL
<b>Ours<sup>†</sup></b> (C-100)	2.58 ± 0.05	-	3.43	1.3*	RL
<b>Ours<sup>†</sup></b> (C-10-Large)	-	15.3 ± 0.04	9.57	1.3*	RL
<b>Ours<sup>†</sup></b> (C-100-Large)	-	14.6 ± 0.03	10.5	1.3*	RL

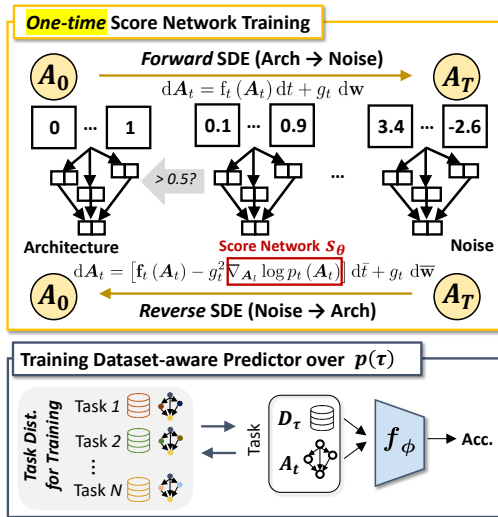
Figure: Result table example

# Contribution

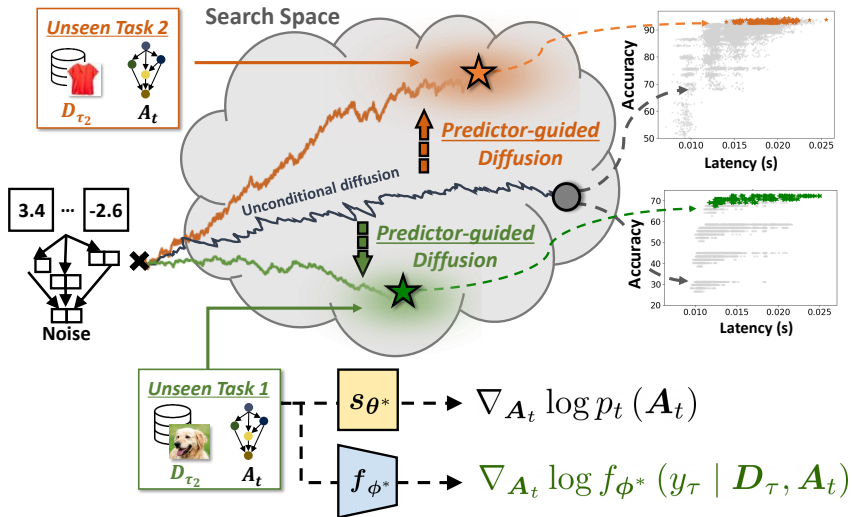
---

- Problem: Classical NAS need to train thousand of architecture.
- Switch from searching architecture to directly generate architecture using a graph diffusion model.
- diffusionNAG

# Contribution



# Contribution



# Contribution

---

## Limits

- Need to retrain  $f_\phi$  for every metrics
  - Accuracy
  - Inference time
  - Memory usage
  - Adversarial attack resistance
  - ...
- Only one metric is optimized: no compromise possible
- Linear combination of metrics: not ideal to find a good compromise

# Our contrib title

- Train a multi-objective predictor using the same task aware dataset, but enhanced with metrics of interest
- Guide the diffusion with the multi-objective predictor:
  - Generate a bigger super-network with disableable blocks
  - ie: equal to an encoding of points near the pareto front (best possible compromise between metrics)
- Constraints encoding within the same score network
  - Layer compatibility in a block
  - Compatibility between blocks inputs and outputs

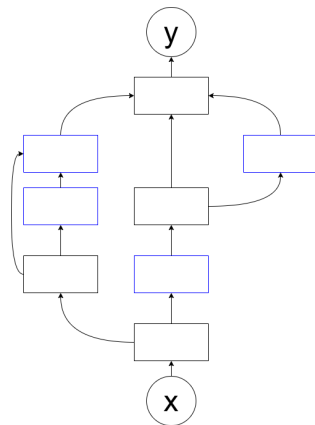
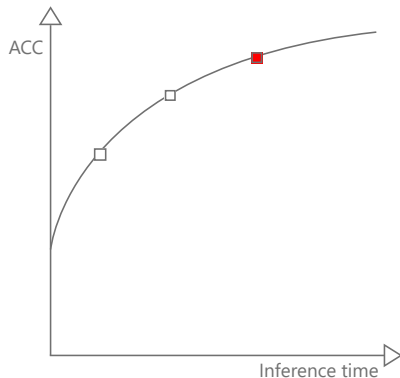
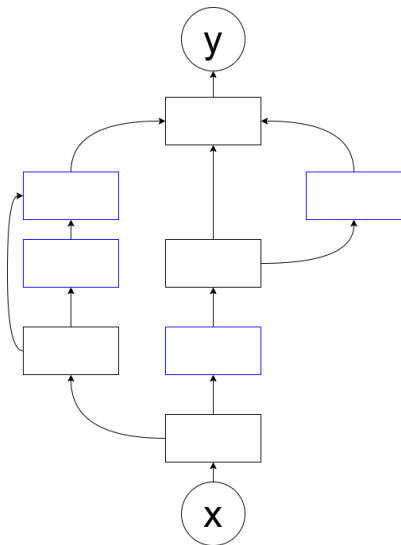
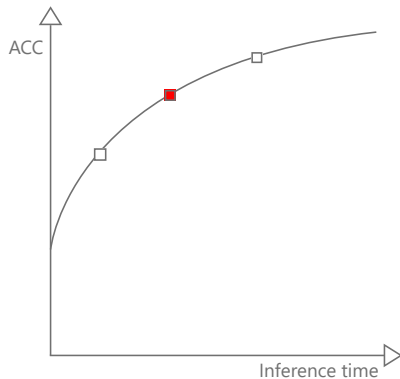
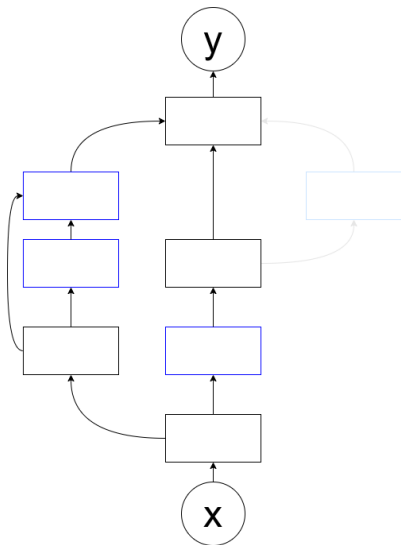


Figure: Example of disableable architecture

- Train the obtained super-network after diffusion inference
- Disable the blocks depending on the metrics of interest

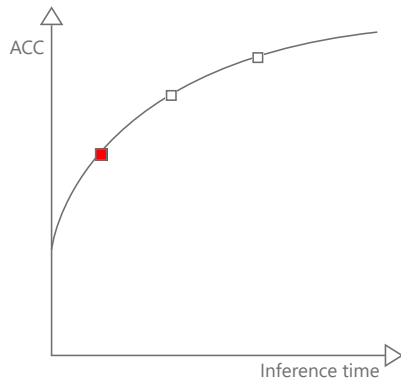
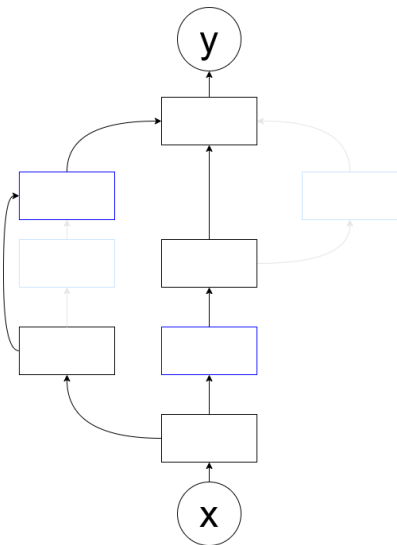


- Train the obtained super-network after diffusion inference
- Disable the blocks depending on the metrics of interest





- Train the obtained super-network after diffusion inference
- Disable the blocks depending on the metrics of interest



# Advantages and limits of this approach

---

- Flexibility for the user : fix the metrics threshold only after performing the NAS
- Dynamic architecture depending on external parameters
- But :
  - Predictor needs to be retrained if a new metric is added
  - Diffusion inference more costly
  - No guarantees if the pareto optimality is well covered by the disableable blocks

# Evaluation and results

---

## Two evaluations

- Architecture search
- generated architecture performances
- generation cheaper than previous full space search
- multi-objective benchmarks for the obtained architectures

Results for a classifier model on CIFAr-10			
Model	Ours	DiffusionNAG	(egrg) best RL methods
architecture search time (min)	23.6	544	2000
Inference costs of the generated architecture (MFLOPs)	250	434	670

# References

---



John Smith (2012)

Title of the publication

*Journal Name* 12(3), 45 – 678.

**Thank you**