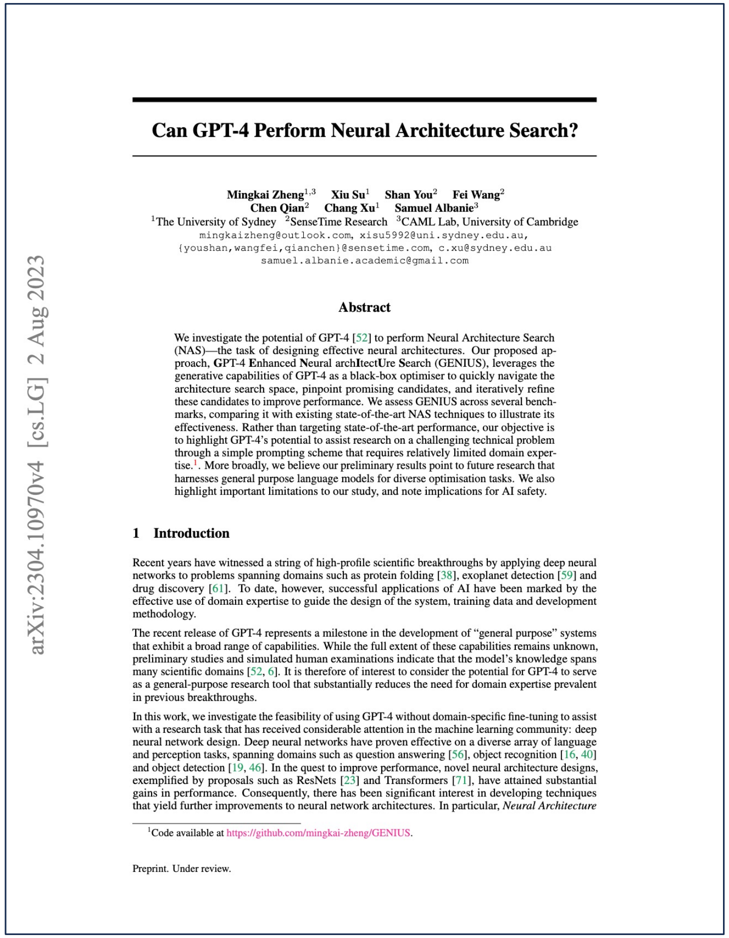
論文紹介

September 14th, 2023



Can GPT-4 Perform Neural Architecture Search?

Mingkai Zheng[1,3] **Xiu Su**[1] **Shan You**[2] **Fei Wang**[2]
Chen Qian[2] **Chang Xu**[1] **Samuel Albanie**[3]
[1]The University of Sydney [2]SenseTime Research [3]CAML Lab, University of Cambridge
mingkaizheng@outlook.com, xisu5992@uni.sydney.edu.au,
{youshan,wangfei,qianchen}@sensetime.com, c.xu@sydney.edu.au
samuel.albanie.academic@gmail.com

Abstract

We investigate the potential of GPT-4 [52] to perform Neural Architecture Search (NAS)—the task of designing effective neural architectures. Our proposed approach, **G**PT-4 **E**nhanced **N**eural arch**I**tect**U**re **S**earch (GENIUS), leverages the generative capabilities of GPT-4 as a black-box optimiser to quickly navigate the architecture search space, pinpoint promising candidates, and iteratively refine these candidates to improve performance. We assess GENIUS across several benchmarks, comparing it with existing state-of-the-art NAS techniques to illustrate its effectiveness. Rather than targeting state-of-the-art performance, our objective is to highlight GPT-4's potential to assist research on a challenging technical problem through a simple prompting scheme that requires relatively limited domain expertise.[1]. More broadly, we believe our preliminary results point to future research that harnesses general purpose language models for diverse optimisation tasks. We also highlight important limitations to our study, and note implications for AI safety.

1 Introduction

Recent years have witnessed a string of high-profile scientific breakthroughs by applying deep neural networks to problems spanning domains such as protein folding [38], exoplanet detection [59] and drug discovery [61]. To date, however, successful applications of AI have been marked by the effective use of domain expertise to guide the design of the system, training data and development methodology.

The recent release of GPT-4 represents a milestone in the development of "general purpose" systems that exhibit a broad range of capabilities. While the full extent of these capabilities remains unknown, preliminary studies and simulated human examinations indicate that the model's knowledge spans many scientific domains [52, 6]. It is therefore of interest to consider the potential for GPT-4 to serve as a general-purpose research tool that substantially reduces the need for domain expertise prevalent in previous breakthroughs.

In this work, we investigate the feasibility of using GPT-4 without domain-specific fine-tuning to assist with a research task that has received considerable attention in the machine learning community: deep neural network design. Deep neural networks have proven effective on a diverse array of language and perception tasks, spanning domains such as question answering [56], object recognition [16, 40] and object detection [19, 46]. In the quest to improve performance, novel neural architecture designs, exemplified by proposals such as ResNets [23] and Transformers [71], have attained substantial gains in performance. Consequently, there has been significant interest in developing techniques that yield further improvements to neural network architectures. In particular, *Neural Architecture*

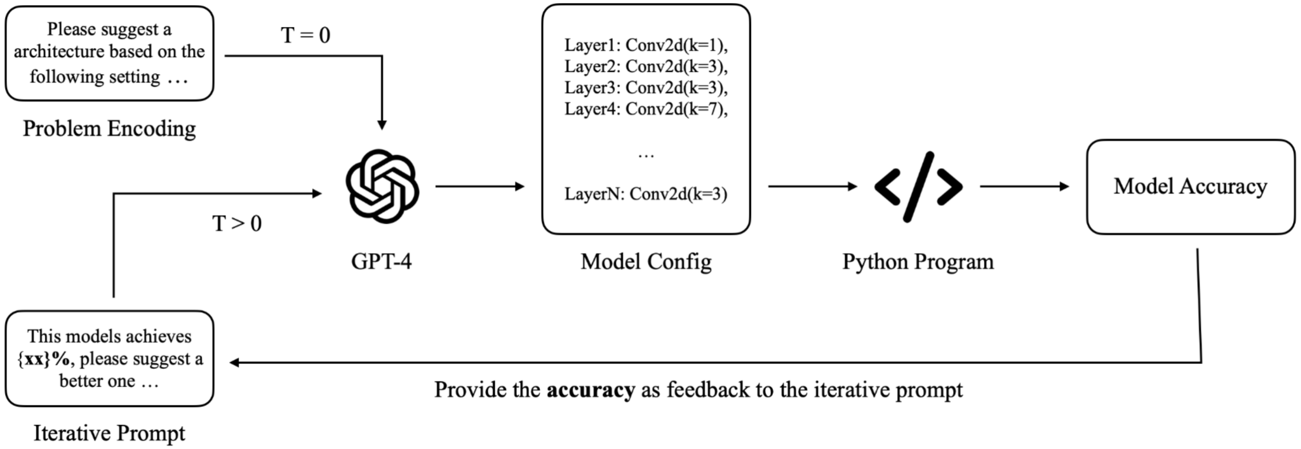
[1] Code available at https://github.com/mingkai-zheng/GENIUS.

Preprint. Under review.

In a nutshell

- What is the best neural network architecture for a given task?
- Usually an expensive search
  - Grid search, evolutionary strategy, Bayesian optimization, Hyperband
  - Differentiable Architecture Search (DARTS) arXiv:1806.09055
  - EfficientNAS arXiv:1807.06906
- This paper uses GPT-4 as a black-box optimizer in a query-feedback loop

Optimization loop



Please suggest a architecture based on the following setting …

Problem Encoding

T = 0

T > 0

GPT-4

Layer1: Conv2d(k=1),
Layer2: Conv2d(k=3),
Layer3: Conv2d(k=3),
Layer4: Conv2d(k=7),
…
LayerN: Conv2d(k=3)

Model Config

Python Program

Model Accuracy

This models achieves {**xx**}**%**, please suggest a better one …

Iterative Prompt

Provide the **accuracy** as feedback to the iterative prompt

“Lego” benchmarks

- NAS-Bench-Macro
  - Contains the performance of various networks on CIFAR-10
  - Each network has the same overall structure but some blocks vary
  - Search space: $3^8 = 6561$ possibilities
- Channel-Bench-Macro
  - Same idea, $4^7 = 16348$ possibilities

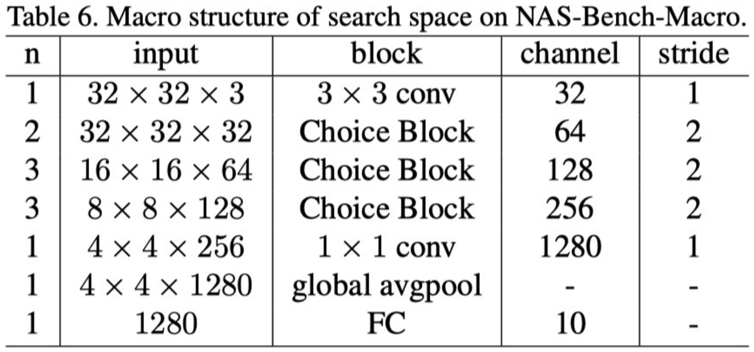
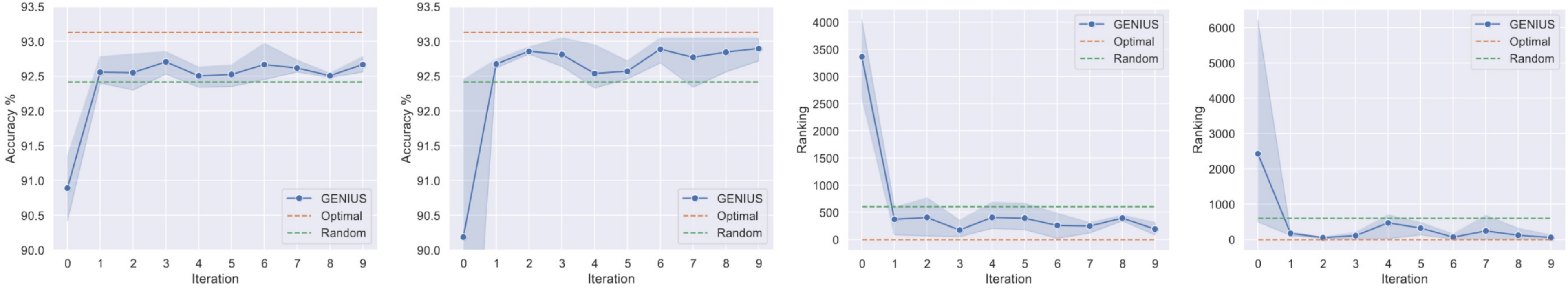


Table 6. Macro structure of search space on NAS-Bench-Macro.

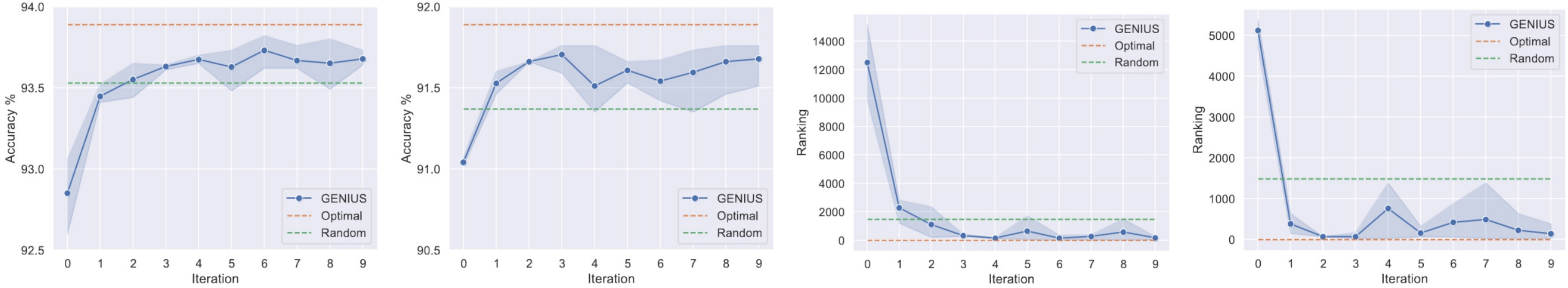
n	input	block	channel	stride
1	$32 \times 32 \times 3$	3×3 conv	32	1
2	$32 \times 32 \times 32$	Choice Block	64	2
3	$16 \times 16 \times 64$	Choice Block	128	2
3	$8 \times 8 \times 128$	Choice Block	256	2
1	$4 \times 4 \times 256$	1×1 conv	1280	1
1	$4 \times 4 \times 1280$	global avgpool	-	-
1	1280	FC	10	-

“Lego” benchmarks

NAS-Bench-Macro

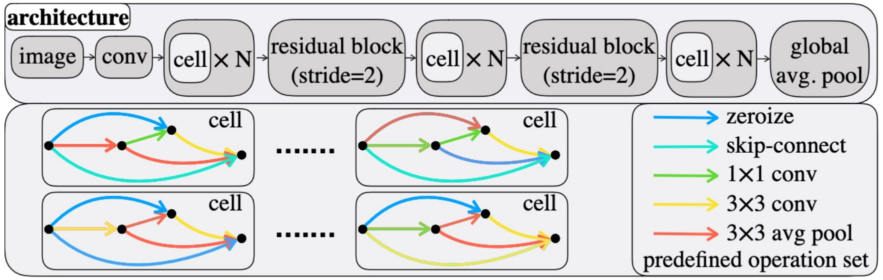


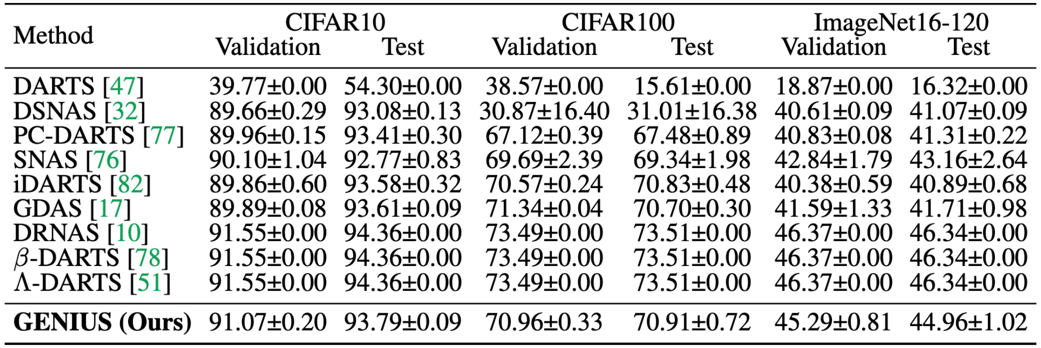
Channel-Bench-Macro



NAS-Bench-201

- Focus on designing (rather than choosing) cells via their computation graph
- Search space: $5^6 = 15625$ possibilities
- Good results but beaten by state-of-the-art (DARTS)

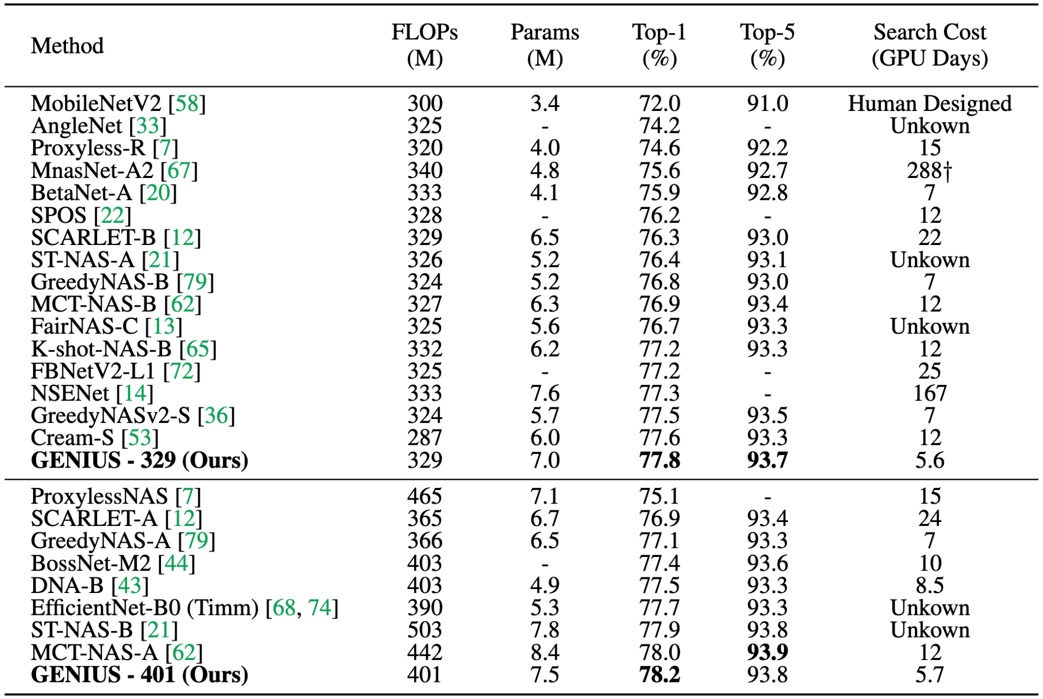




Method	CIFAR10		CIFAR100		ImageNet16-120	
	Validation	Test	Validation	Test	Validation	Test
DARTS [47]	39.77±0.00	54.30±0.00	38.57±0.00	15.61±0.00	18.87±0.00	16.32±0.00
DSNAS [32]	89.66±0.29	93.08±0.13	30.87±16.40	31.01±16.38	40.61±0.09	41.07±0.09
PC-DARTS [77]	89.96±0.15	93.41±0.30	67.12±0.39	67.48±0.89	40.83±0.08	41.31±0.22
SNAS [76]	90.10±1.04	92.77±0.83	69.69±2.39	69.34±1.98	42.84±1.79	43.16±2.64
iDARTS [82]	89.86±0.60	93.58±0.32	70.57±0.24	70.83±0.48	40.38±0.59	40.89±0.68
GDAS [17]	89.89±0.08	93.61±0.09	71.34±0.04	70.70±0.30	41.59±1.33	41.71±0.98
DRNAS [10]	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
β-DARTS [78]	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
Λ-DARTS [51]	91.55±0.00	94.36±0.00	73.49±0.00	73.51±0.00	46.37±0.00	46.34±0.00
GENIUS (Ours)	91.07±0.20	93.79±0.09	70.96±0.33	70.91±0.72	45.29±0.81	44.96±1.02

Large-scale experiments

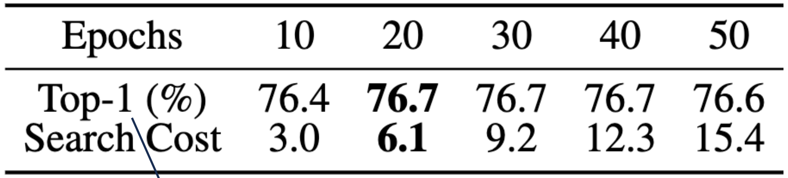
- Lego-type benchmark on ImageNet with more layer options
- Search space: $7^{30} \approx 2.2 \times 10^{25}$ possibilities
- FLOPs constraints
- Marginally better results, but significantly faster



Method	FLOPs (M)	Params (M)	Top-1 (%)	Top-5 (%)	Search Cost (GPU Days)
MobileNetV2 [58]	300	3.4	72.0	91.0	Human Designed
AngleNet [33]	325	-	74.2	-	Unkown
Proxyless-R [7]	320	4.0	74.6	92.2	15
MnasNet-A2 [67]	340	4.8	75.6	92.7	288†
BetaNet-A [20]	333	4.1	75.9	92.8	7
SPOS [22]	328	-	76.2	-	12
SCARLET-B [12]	329	6.5	76.3	93.0	22
ST-NAS-A [21]	326	5.2	76.4	93.1	Unkown
GreedyNAS-B [79]	324	5.2	76.8	93.0	7
MCT-NAS-B [62]	327	6.3	76.9	93.4	12
FairNAS-C [13]	325	5.6	76.7	93.3	Unkown
K-shot-NAS-B [65]	332	6.2	77.2	93.3	12
FBNetV2-L1 [72]	325	-	77.2	-	25
NSENet [14]	333	7.6	77.3	-	167
GreedyNASv2-S [36]	324	5.7	77.5	93.5	7
Cream-S [53]	287	6.0	77.6	93.3	12
GENIUS - 329 (Ours)	329	7.0	**77.8**	**93.7**	5.6
ProxylessNAS [7]	465	7.1	75.1	-	15
SCARLET-A [12]	365	6.7	76.9	93.4	24
GreedyNAS-A [79]	366	6.5	77.1	93.3	7
BossNet-M2 [44]	403	-	77.4	93.6	10
DNA-B [43]	403	4.9	77.5	93.3	8.5
EfficientNet-B0 (Timm) [68, 74]	390	5.3	77.7	93.3	Unkown
ST-NAS-B [21]	503	7.8	77.9	93.8	Unkown
MCT-NAS-A [62]	442	8.4	78.0	**93.9**	12
GENIUS - 401 (Ours)	401	7.5	**78.2**	93.8	5.7

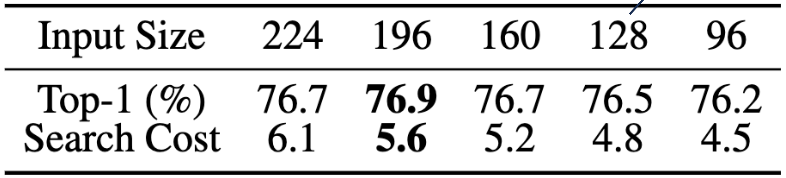
Ablation (training tuning)

- During neural architecture search, candidates are trained quickly; during validation, candidate architecture are trained in a more standard way
- This study explores two tradeoffs for quick training



Epochs	10	20	30	40	50
Top-1 (%)	76.4	**76.7**	76.7	76.7	76.6
Search Cost	3.0	**6.1**	9.2	12.3	15.4

Performance of the final (retrained) model

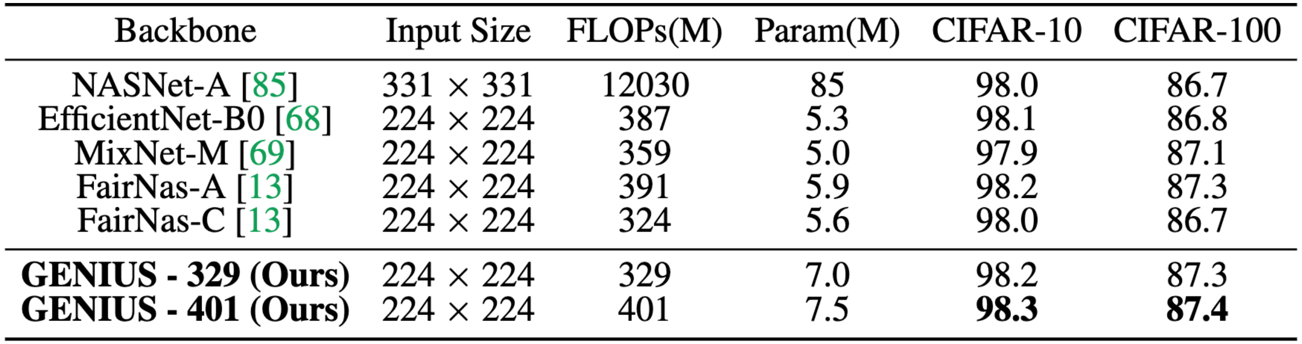


i.e. the image is resized to 128x128

Input Size	224	196	160	128	96
Top-1 (%)	76.7	**76.9**	76.7	76.5	76.2
Search Cost	6.1	**5.6**	5.2	4.8	4.5

Transfer learning

- Take a model pretrained on ImageNet and fine-tune it on CIFAR10 and CIFAR100
- Achieved performance on-par with state-of-the-art
- Also studied transfer learning for object recognition tasks



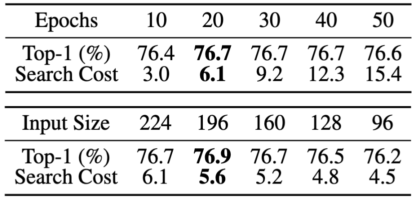
Backbone	Input Size	FLOPs(M)	Param(M)	CIFAR-10	CIFAR-100
NASNet-A [85]	331 × 331	12030	85	98.0	86.7
EfficientNet-B0 [68]	224 × 224	387	5.3	98.1	86.8
MixNet-M [69]	224 × 224	359	5.0	97.9	87.1
FairNas-A [13]	224 × 224	391	5.9	98.2	87.3
FairNas-C [13]	224 × 224	324	5.6	98.0	86.7
GENIUS - 329 (Ours)	224 × 224	329	7.0	98.2	87.3
GENIUS - 401 (Ours)	224 × 224	401	7.5	**98.3**	**87.4**

Remarks

- From all the performant designs, we can see how GPT-4 tends to design neural networks:
  - In initial stages, employ simpler operations fo capture low-level information
  - Increase complexity progressively
- GPT-4 is a black-box optimizer which poses a few problems:
  - A black-box is black, we don’t know *why or how* an architecture was generated; would changing the prompt change or even invalidate the results?
  - Lack of reproducability
  - (Potential) Benchmark contamination
  - *Enfeeblement*: desining networks is a crucial knowledge that we may loose by excessively delegating to AI

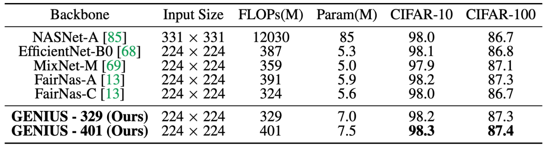
Opinions & questions

- [Ablation study] the accuracy does not change much. This show that GPT-4 does not need precise metrics, only a sense of direction
- [Transfer learning] not a surprise. The metric that should have been reported is fine-tuning time



Epochs	10	20	30	40	50
Top-1 (%)	76.4	**76.7**	76.7	76.7	76.6
Search Cost	3.0	**6.1**	9.2	12.3	15.4

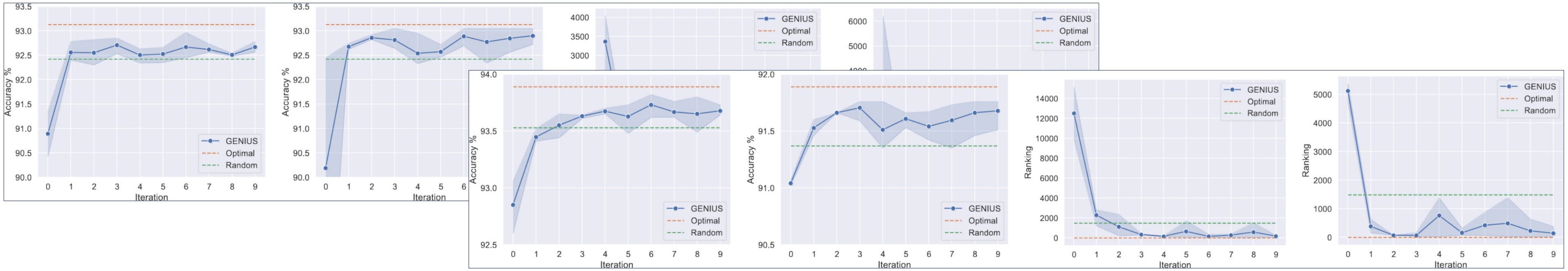
Input Size	224	196	160	128	96
Top-1 (%)	76.7	**76.9**	76.7	76.5	76.2
Search Cost	6.1	**5.6**	5.2	4.8	4.5



Backbone	Input Size	FLOPs(M)	Param(M)	CIFAR-10	CIFAR-100
NASNet-A [85]	331 × 331	12030	85	98.0	86.7
EfficientNet-B0 [68]	224 × 224	387	5.3	98.1	86.8
MixNet-M [69]	224 × 224	359	5.0	97.9	87.1
FairNas-A [13]	224 × 224	391	5.9	98.2	87.3
FairNas-C [13]	224 × 224	324	5.6	98.0	86.7
GENIUS - 329 (Ours)	224 × 224	329	7.0	98.2	87.3
GENIUS - 401 (Ours)	224 × 224	401	7.5	**98.3**	**87.4**

Opinions & questions

- [“Lego” benchmarks]
  - The first network (iteration 0) is designed without any prior feedback
  - The second network (iteration 1) is designed with the performance of the first in mind
  - The jump in performance is really big
  - Why not “skip” the 0-th iteration by starting with a random architecture and a nominal (fake) performance?



Opinions & questions

- [Design process design] what’s the next step?

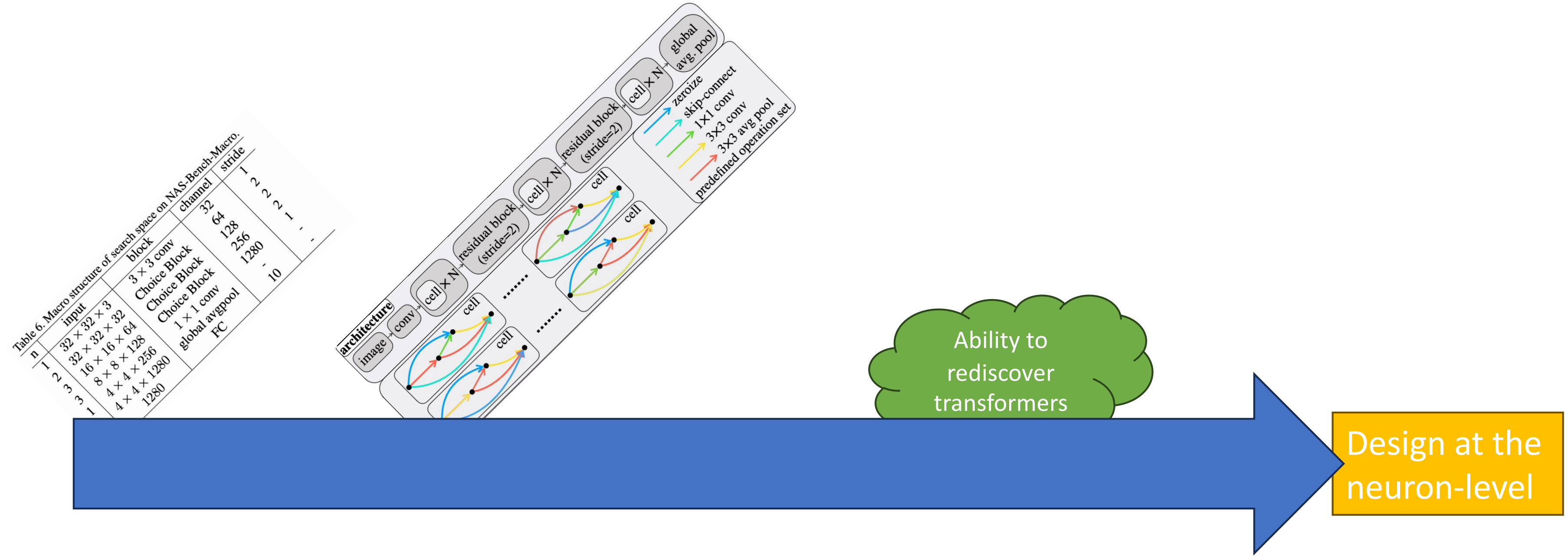


Table 6. Macro structure of search space on NAS-Bench-Macro.

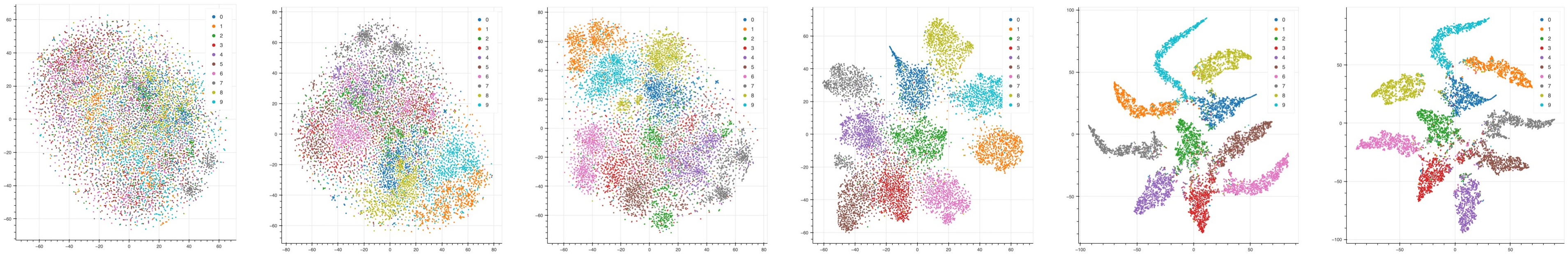
n	input	block	channel	stride
1	32 × 32 × 3	3 × 3 conv	32	1
2	32 × 32 × 32	Choice Block	64	2
3	16 × 16 × 64	Choice Block	128	2
3	8 × 8 × 128	Choice Block	256	2
1	4 × 4 × 256	1 × 1 conv	1280	1
	4 × 4 × 1280	global avgpool	-	-
	1280	FC	10	-

Ability to rediscover transformers

Design at the neuron-level

Opinions & questions

- [Design process design]
    - Start with a “standard” search
    - Then fine-tune the computation graph of each block starting from the start to optimize global performance (e.g. accuracy) and *local performance* (e.g. class entropy in latent space)


0
1
2
3
4
5
6
7
8
9