TRANSNAS-BENCH-101: IMPROVING TRANSFERRABILITY AND GENERALIZABILITY OF CROSS-TASK NEURAL ARCHITECTURE SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent breakthroughs of Neural Architecture Search (NAS) are extending the field's research scope towards a broader range of vision tasks and more diversified search spaces. While existing NAS methods mostly design architectures on one single task, algorithms that look beyond single-task search are surging to pursue a more efficient and universal solution across various tasks. Many of them leverage transfer learning and seek to preserve, reuse, and refine network design knowledge to achieve higher efficiency in future tasks. However, the huge computational cost and experiment complexity of cross-task NAS are imposing barriers for valuable research in this direction. Existing transferrable NAS algorithms are also based on different settings, e.g., datasets and search spaces, which raises concerns on performance comparability. Although existing NAS benchmarks provided some solutions, they all focus on one single type of vision task, i.e., classification. In this work, we propose TransNAS-Bench-101, a benchmark containing network performance across 7 tasks, covering classification, regression, pixel-level prediction, and self-supervised tasks. This diversity provides opportunities to transfer NAS methods among the tasks, and allows for more complex transfer schemes to evolve. We explore two fundamentally different types of search spaces: cell-level search space and macro-level search space. With 7,352 backbones evaluated on 7 tasks, 51,464 trained models with detailed training information are provided. Generating this benchmark takes about 193,760 GPU hours, which is equivalent to 22.12 years of computation on a single Nvidia V100 GPU. Analysis of 4 benchmark transfer schemes highlights that: (1) Direct deployment of both architectures and policies can easily lead to negative transfer unless guided by carefully designed mechanisms. (2) Evolutionary methods' role in transferrable NAS might be overlooked in the past. (3) It is a valid challenge for NAS algorithms to perform well across tasks and search spaces consistently. We also provide our suggestions for future research along with the analysis. With TransNAS-Bench-101, we hope to encourage the advent of exceptional NAS algorithms that raise cross-task search efficiency and generalizability to the next level.

1 INTRODUCTION

In recent years, networks found by Neural Architecture Search (NAS) methods are surpassing human-designed ones, setting state-of-the-art performance on various tasks (Tan & Le, 2019; Real et al., 2019). Ultimately, NAS is calling for algorithmic solutions that can discover near-optimal models for any task within any search spaces under moderate computational budgets. To pursue this goal, recent research in NAS quickly expanded its scope into broader vision domains such as object detection (Xu et al., 2019) and semantic segmentation (Chen et al., 2018). Many works have also started to investigate more general network design, such as choosing an optimal macro skeleton of a network (Yao et al., 2019; Xu et al., 2019).

Although many algorithms have successfully shortened NAS's search time from months to hours (Liu et al., 2018; Dong & Yang, 2019b), some research have shown their reliance on specific search spaces and datasets (Yang et al., 2019). There are also questions on these algorithms' efficiency when dealing with a large number of tasks (Chen et al., 2020). To look for solutions that work well

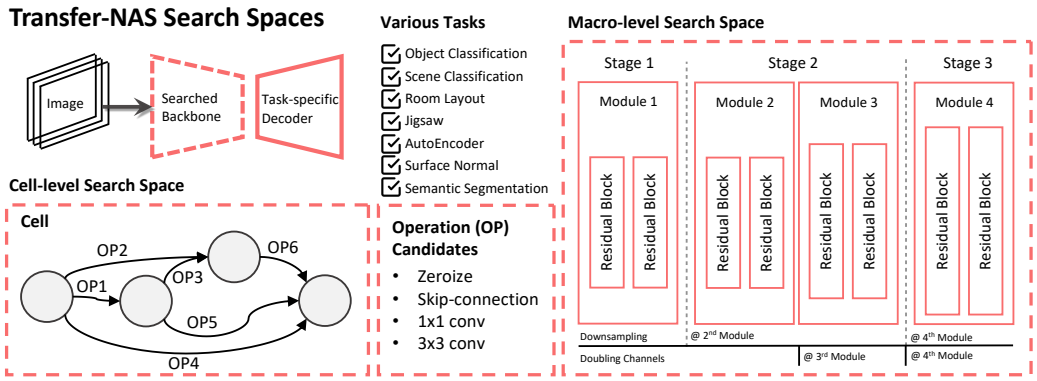


Figure 1: Our cell-level and macro-level search space in TransNAS-Bench-101. We design: a) A cell-level search space that treats each cell as a DAG; and b) A macro-level search space that allows flexible macro skeleton network design.

across multiple tasks, Transferrable NAS is a rising research direction. It is surging with exceptional algorithms leveraging tools in the transfer/meta learning paradigm (Pasunuru & Bansal, 2019; Wong et al., 2018; Shaw et al., 2019; Lian et al., 2020; Guo et al.). Chen et al. (2020) explores meta learning to transfer network design knowledge from small tasks to larger tasks, surpassing many efficient solutions based on parameter sharing. Cai et al. (2020) proposes a highly memory-efficient and effective transfer solution that does not require back-propagation for adaptation. While these algorithms each show compelling results, they are evaluated under different settings raises concerns on performance comparability. To avoid confusions, throughout this paper, we use the word *task* to distinguish NAS problems where the dataset or the vision domain is different (e.g., detection and segmentation).

In the meantime, NAS is prohibitively costly. The computational intensity of single-task NAS can be discouraging for many researchers, not to mention cross-task NAS experiments. To solve the computation and the aforementioned comparability problem, NAS-Bench-101 (Ying et al., 2019), and NAS-Bench-201(Dong & Yang, 2019a) were proposed. These benchmarks have offered great values for the NAS community, but we believe the research scope of NAS can be further enlarged beyond classification problems and cell-based search space.

The goal of finding universal solutions across tasks and search spaces, the comparability problem, and the computation barriers of transferrable NAS research lead to our proposal of TransNAS-Bench-101, which studies networks over 7 distinctive vision tasks: object classification, scene classification, semantic segmentation, autoencoding, room layout, surface normal, and jigsaw. Two types of search spaces are provided: One is the macro skeleton search space based on residual blocks, where the network depth and block operations (e.g., when to raise the channel or downsample the resolution) are decided. Another one is the widely-studied cell-based search space, where each cell can be treated as a directed acyclic graph (DAG). The macro-level and cell-level search space contains 3256 and 4096 networks, respectively. The 7,352 backbones are evaluated on all 7 tasks, and we provide detailed diagnostic information for all models. We also evaluated 4 transfer schemes compatible with both search spaces to serve as benchmarks for future research.

Our key contribution is a benchmark with networks fully evaluated on 7 tasks across two search spaces. Generating the whole benchmark takes 193,760 GPU hours, i.e., 22.12 years of computation on one NVIDIA V100 GPU. Still, it significantly lowers the cost of further research into cross-task neural architecture search. We also highlight problems and provide suggestions for future NAS research: (1) To extend NAS into different vision domains, it is important to look beyond cell-based search spaces, as we found that network macro structures have a larger impact on performance than cell structures on certain tasks. (2) The extent to which an algorithm can surpass random search in the same search space can be a crucial performance indicator. (3) Investigations of evolutionary-based transfer strategies, along with effective mechanisms to tweak transferred architectures and policies, are two promising directions for future transferrable NAS research. With the diversified settings in TransNAS-Bench-101, we hope to encourage the emergence of exceptional NAS algorithms that not only prevail in a few selected datasets, but also across a wide range of tasks and search spaces.

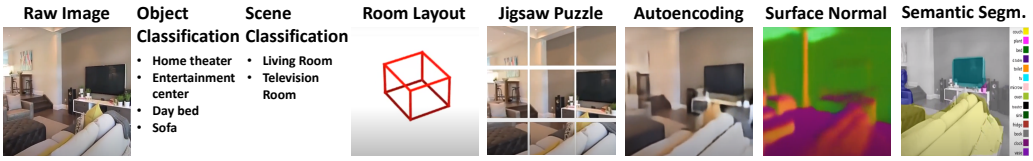


Figure 2: Vision tasks considered in our benchmarks. We carefully select those 7 tasks above to ensure both diversity and similarity across tasks from Taskonomy (Zamir et al., 2018).

2 THE TRANSNAS-BENCH-101 DATASET

2.1 SEARCH SPACES AND ARCHITECTURES

To plug in different networks for various tasks, our search space is focused on evolving the backbone, i.e. the common component of all the tasks considered. We provide two search spaces: a) A macro-level search space that designs the skeleton of a network, which was previously studied towards NAS in object detection and semantic segmentation; b) A cell-level search space following the widely studied cell-based search space, which applies to most weight-sharing NAS methods.

Macro-level Search Space. Most NAS methods follow a fixed macro skeleton with a searched cell. However, the macro-level structure of the backbone can be crucial for final performance. Early-stage feature maps in a backbone have larger sizes as they capture texture details, whereas feature maps at later stages are smaller, and usually are more discriminative (Li et al., 2018). The allocation of computations over different stages is important for a backbone (Liang et al., 2020). Therefore, this search space contains networks with different depth (the total number of blocks), locations to down-sample the feature maps, and locations to raise the channels. As is illustrated in Figure 1, we group two residual blocks (He et al., 2016) into a module, and the networks are stacked with 4 to 6 modules. The module positions can be chosen to downsample the feature map for 1 to 4 times, and each time the spatial size will shrink by a factor of 2. The network can double its channels for 1 to 3 times at chosen locations. This search space thus consists of 3265 unique architectures.

Cell-level Search Space. We follow NAS-Bench-201(Dong & Yang, 2019a) to design our cell-level search space. It consists of 4096 densely connected DAGs, which enables the evaluation of some weight-sharing NAS methods such as DARTS (Liu et al., 2018) and ProxylessNAS (Cai et al., 2019). As is shown in Figure 1, our cell-level search space is obtained by assigning different operations (as edges) transforming the feature map from the source node to the target node. The predefined operation set has L = 4 representative operations: zeroize, skip connection, 1x1 convolution, and 3x3 convolution. The *convolution* in our setting represents an operation sequence of ReLU, convolution and batch normalization. Each DAG consists of 4 nodes and 6 edges, including basic residual block-like cell designs. The macro-level skeleton is fixed, which contains 5 modules with doubling channel and down-sampling feature map operations lying at the 1st, 3rd, 5th modules.

Adding up the 3265 and 4096 networks from the macro-level search space and the cell-level search space, we have 7352 unique architectures in total. All the architectures in both search spaces are carefully trained and evaluated across all the selected tasks. Thus a wide range of NAS algorithms can be examined and compared not only on a single task, but also across multiple tasks.

2.2 DATASETS

Unlike most NAS benchmarks that focus on classification tasks only, TransNAS-Bench-101 encourages the evaluation of algorithms across different tasks. This makes the selection of proper datasets challenging, since ideally, the datasets should have some commonalities while providing annotations for different tasks. Thanks to the great previous work Taskonomy (Zamir et al., 2018), which carefully studies the relationship between different visual tasks, we can follow their definitions of tasks and the released dataset. The original dataset consists of 4.5M images of indoor scenes from about 600 buildings. To control the computational budget, we randomly select 24 buildings containing 120K images from the original dataset and split the subset into 80K train / 20K val / 20K test set.

To ensure both diversity and similarity of the tasks, we carefully selected 7 tasks. As is shown in Figure 2, the selected tasks include a) image classification tasks: Object Classification (75 classes selected) and Scene Classification (47 classes selected); b) pixel-level prediction tasks: Surface

Table 1: Training hyperparameters and details of each tasks considered in this benchmark. All the architectures in the search space have been fully trained. We provide multiple metrics for evaluation on the train/val/test set. Each task requires a backbone-decoder network structure with task-specific decoder and loss function. GAP denotes global average pooling. CE denotes the cross entropy loss.

Tasks	Epochs	Decoder	LR	Optimizer	Output dim.	Loss	Eval. Metrics
Object Class.	25	GAP + Linear	0.1	SGD	75	Softmax+CE	Loss, Acc
Scene Class.	25	GAP + Linear	0.1	SGD	47	Softmax+CE	Loss, Acc
Room Layout	25	GAP + Linear	0.1	SGD	9	MSE loss	Loss
Jigsaw	10	GAP + Linear	0.1	SGD	1000	Softmax+CE	Loss, Acc
Autoencoding	30	14 Conv & Deconv	0.0005	Adam	256x256	GAN loss + L1	Loss, SSIM
Surface Normal	30	14 Conv & Deconv	0.0001	Adam	256x256	GAN loss + L1	Loss, L1, SSIM
Sem. Segment.	30	8 Conv & Deconv	0.1	SGD	256x256	Softmax+CE	Loss, Acc, mIoU

Normal and Semantic Segmentation; c) self-supervised task: Jigsaw Puzzle and Autoencoding; d) Regression task: Room Layout.

2.3 Training Details

In TransNAS-Bench-101, the seven different tasks require different network structures and loss functions. To train the networks on a given task, we define a default backbone-decoder network structure first, then iterate through the search space and change its backbone architecture. For pixel-level prediction tasks and autoencoding, the decoders' input channels and resolutions will change flexibly but minimally according to different representation shapes generated by the backbone. Since the original paper's implementation is based on an outdated version of Tensorflow, we reimplemented both the training and testing script with PyTorch for reproducibility. We mostly follow the Taskonomy paper to set up the hyper-parameters and training strategies, which is shown in Table 1. For all the tasks, the batch size is 128, and the input resolution is resized to 256×256. For all the architectures, we record multiple evaluation metrics for each epoch, as is listed in Table 1. The average training+evaluation time for one architecture varies from 1 GPU hours to 7 GPU hours for different tasks. Since we train every architecture in our search space for all the 7 tasks (i.e. $7352 \times 7 \approx 5 \times 10^4$ arch), the total computation cost is 193,760 GPU hours on V100 to generate the whole TransNAS-Bench-101. Users can use our API to conveniently query each architecture's information across tasks without additional computation costs. In this way, researchers could significantly speed up their research and focus solely on improving the algorithms.

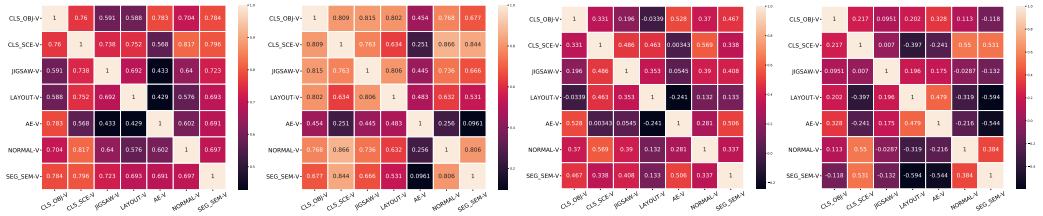
3 Related Work

To foster reproducibility and fair comparisons among algorithms, there are several existing NAS benchmarks. NAS-Bench-101 is the earliest work, which contains 423k unique architectures evaluated on the CIFAR-10 dataset. The networks are designed with a cell-based structure, where each cell is treated as a DAG. However, since NAS-Bench-101 only provides a portion of the networks within its search space, it is only applicable to a few selected algorithms, such as those based on parameter sharing (Liu et al., 2018; Pham et al., 2018; Han et al., 2020).

As an extension of NAS-Bench-101, NAS-Bench-201 was proposed to accommodate the growing needs. It provides training information of 15k networks, forming a complete search space. Similar to NAS-Bench-101, the networks are designed under a cell-based structure, but it could support many more algorithms with detailed diagnostic information. With training results over three datasets

Table 2: Comparisons of TransNAS-Bench-101 with previous benchmarks. Although TransNAS-Bench-101 has a smaller search space, it contains more datasets, domains, and search space types.

	# Datasets	# Task Domains	# Search Space Size	Search Space Type
NAS-Bench-101	1	1	510M	Cell
NAS-Bench-201	3	1	15.6K	Cell
TransNAS-Bench-101	7	4	7.3K	Cell & Macro



(a) Macro (100%) (b) Cell (100%) (c) Macro (50%) (d) Cell (50%)

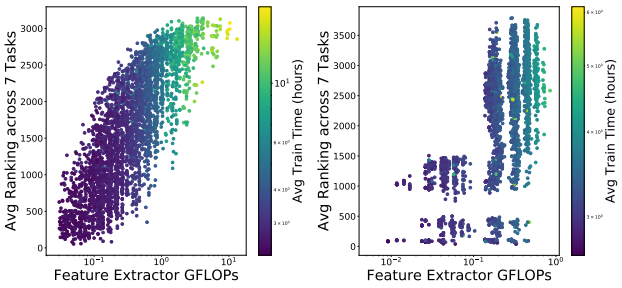
Figure 4: The Spearman rank correlations among tasks. Networks in the cell-level search space has higher correlations than the macro-level search space. The correlations shrinks quickly if we sample top 50% of the networks.

provided, it first enabled transfer learning across tasks. Ten benchmark algorithms are evaluated with extensive experiments in addition to network information.

TransNAS-Bench-101's commonalities with previous benchmarks mainly lie in: (1) It offers detailed network training information with all the networks in a complete search space. (2) It also adopts the cell-based search space, treating each cell as a DAG. However, TransNAS-Bench-101 evaluates its networks across a much more diversified set of tasks. It is also the first benchmark that provides a thorough analysis of the macro skeleton search space.

4 Analysis of TransNAS-Bench-101

Overview of architectures. The architectures' performance ranking, FLOPs, and training time are presented in Figure 3. We obtain rakings of each architecture's validation performance on all 7 tasks first, then plot the average ranking. A higher number means better performance ranking. We also provide each architecture's FLOPs and training time as a reference.

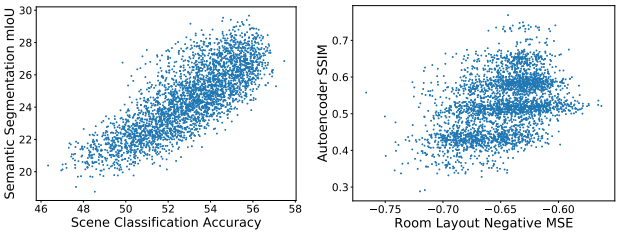


(a) Macro-level (b) Cell-level

Figure 3: The Architecture performance ranking, FLOPs, and training time in both search spaces. The longest single network training time is 18h on one Nvidia V100 GPU.

The pattern shows that a network with more FLOPs and longer training time tends to perform better on the given tasks within a reasonable range, but it does not include some of the search space's largest networks. Figure 3 also reveals some distinctive characteristics of both search spaces. The macro-level search space has its networks more evenly spread out in terms of FLOPs, whereas networks in the cell-level search space are more concentrated at certain numbers. The macro-level search space's network FLOPs vary across a wider range from 10^8 to 10^{11}, while the cell-level search space's architectures range across 10^8 to 10^{10}, which is a magnitude smaller. It can take up to 18 hours to train a network in the macro-level search space, which is three times the GPU hour needed to train the most computationally demanding network on the cell-level search space.

Correlations among tasks. We analyze cross-task correlations by calculating the Spearman Rank Correlation Scores among the tasks, and present the results in Figure 4. Although object classification and scene classification are both classification tasks, scene classification has higher correlations with surface normal and semantic segmentation tasks on both search spaces than object classification. This phenomenon shows that tasks within the same domain, even though they are based on essentially the same images, might not necessarily be closer in terms of architectural performance. We visualize the network performance of tasks with the highest (0.817) and lowest (0.429) correlations on the macro-level search space in Figure 5.



(a) Highest correlation (b) Lowest correlation

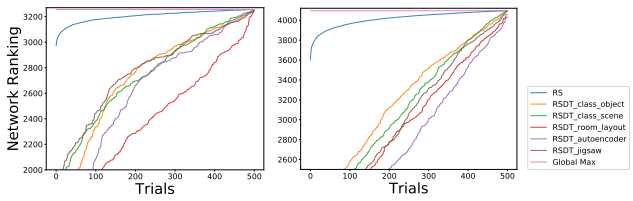
Figure 5: Tasks with the highest and lowest correlations on the macro-level search space.

The Autoencoding task has very distinctive behaviors under two search spaces. Similar to semantic segmentation and surface normal, it is an image translation task that outputs 256x256 images. With networks in the macro-level search space, the autoencoding task has moderate correlations with semantic segmentation (0.691) and surface normal (0.602). However, under the cell-level search space, it has almost no correlations with semantic segmentation (0.0961), and very weak correlation with surface normal (0.256). This huge discrepancy shows that the selection of search space can significantly impact certain tasks. Some search spaces can dramatically lower the difficulty of NAS transfer, and some might have inherent disadvantages. It again highlights the importance of validating an algorithm's performance on multiple search spaces to obtain unbiased evaluations.

The correlations among tasks shrink quickly if we plot the graphs with only the top 50% networks' performance information. Some tasks still have relatively strong correlations, but others rapidly drop to below zero. This shows that the direct transfer strategy of architectures might not always yield good results, and robustly transferrable algorithms should be wary of it to avoid negative transfer.

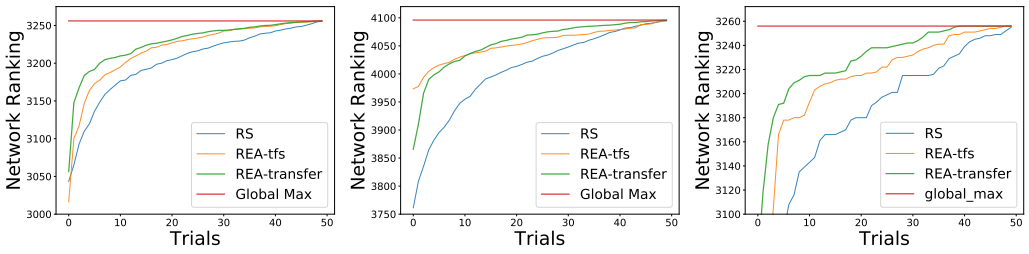
5 Benchmark Algorithms

In this section, we evaluate 4 transfer schemes of different types: (1) Random Search (RS) (Bergstra & Bengio, 2012); (2) Direct transfer of top architectures proposed by random search (RSDT); (3) Direct policy transfer of reinforcement learning-based algorithm, e.g., Proximal Policy Optimization (PPO) (Schulman et al., 2017); (4) Meta-learning based algorithm, e.g., CATCH (Chen et al., 2020); (5) Evolution algorithms with transferred population initialization, e.g., Regularized Evolution Algorithm (REA) (Real et al., 2019). Each selected algorithm represents a distinctive type of transfer scheme, and they are compatible with both search spaces. Although many current weight-sharing algorithms have shown promising results, most of them operate on cell-based search spaces only, and it would take non-trivial modifications to the algorithms for them to search on the macro-level search space. Therefore, we do not include them in our initial benchmark.



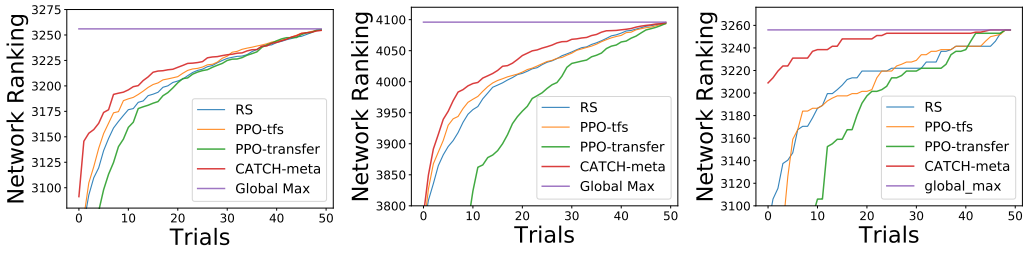
(a) Macro-level (b) Cell-level

Figure 6: Comparison of random search and direct architecture transfer. RSDT stands for Random Search from Direct Architecture Transfer from a specific source task.



(a) Macro-level (b) Cell-level (c) REA on Room Layout.

Figure 7: Comparison of the transfer and train-from-scratch (tfs) results of REA. REA-transfer has slight but stable improvements across all tasks.



(a) Macro-level (b) Cell-level (c) CATCH on Object Class.

Figure 8: Comparison of PPO and CATCH. CATCH largely improves PPO-transfer's performance, and it works exceedingly well on object classification.

Evaluation of direct architecture transfer. Many NAS algorithms have attempted to use architectures as the central component for transfer (Zoph et al., 2018; Pasunuru & Bansal, 2019), which inspired us to look into the efficiency of direct transfer of top architectures found by RS. In Figure 6, we present the search result of such a transfer scheme. For each trial, we first run RS on a task for 50 epochs, then take the top performer and transfer it to a target task. We then record the ranking of the searched network on the target task. This process is repeated for 500 times for each possible combination of the source task and target task. We sort the transferred architectures' rankings for each target task and plot them into curves distinguished by different source tasks. After running 500×7 trials, each target task has 6 curves, representing the transfer result from a source task. We then take the average of each curve across all 7 tasks. We also run 500 trials of RS on each task, and plot the average curves on the ranking of architectures found by it within 50 epochs. Figure 6 shows that direct transfer of architectures found by RS almost always yields worse results than RS by the same budget. It shows that even if any NAS algorithm can be guaranteed to find the optimal architecture on a source task given enough time, its direct transfer performance on the target task can still suffer.

Evaluation of policy transfer. We use a multi-layer perceptron (MLP) as the policy network for PPO, and pretrain the policy for 50 epochs on a less time-costly source task. The pretrained policy is then adapted to the target task to search for another 50 epochs. CATCH adopts the same budget for pretrain and transfer, and repeat each trial for 50 times. Similar to the experiments above, we plot the search results on each target task first, then take the average of the curves across all tasks. The result is shown in Figure 8.

Direct transfer of PPO policies shows worse results than its non-transfer version, a phenomenon commonly referred to as negative transfer. We hypothesize this is due to PPO's overfitting to the source task during the pretrain phase. CATCH mediates it with two added components: An encoder that provides task information that guides its policy, and an evaluator that filters inferior candidates. From our experiments, these added components do improve the transfer results. As Table 3 indicates, it shows exceedingly good performance under certain settings such as object classification on the macro-level search space, but it also struggles on some other tasks, such as room layout on cell-level search space.

Evaluation of evolution algorithms with transferred population initialization. We reproduced REA in our benchmark, and the result is presented in Figure 7. During the pretrain phase, we randomly initialize a population, then set the pretrain budget, i.e., the total number of architecture to search during the pretrain phase, to be 50. After training on the given budget on a source task, we take the top 10 architectures in the pretrain history as the initialization of population on the target task, and search for 50 epochs. Although it does not have significant boosts on certain tasks like CATCH does, it maintains a relatively stable performance improvement across all examined tasks, which results in the slight surpass from its train-from-scratch version when the curves are averaged.

Comparison across Transfer NAS Algorithms The average performance of each algorithm is presented in Table 3. REA-transfer is the top performer among all evaluated algorithms, achieving 46.38 and 51.46 average performance on the cell-level and macro-level search spaces. CATCH has very close performance on the cell-level search space (0.14), but this difference is enlarged when the search is conducted on the macro-level search space (0.74). The experiments highlight that: (1) Direct transfer of architectures performs significantly worse than random search; (2) Direct

Table 3: Performance comparison of different transferrable NAS methods. Room layout's L2 loss is multiplied by a factor of 100 for better readability. The transferred versions of REA and PPO are pretrained on the least time-consuming task, Jigsaw. The average scores of all algorithms are calculated with the remaining 6 tasks.

Tasks		Cls. Object	Cls. Scene	Autoencoding	Surf. Normal	Sem. Segment.	Room Layout	Jigsaw	avg.
Metric		*Acc.*	*Acc.*	*SSIM*	*SSIM*	*mIoU*	*L2 loss*	*Acc.*	
Cell level	RS	45.16±0.4	54.41±0.3	55.94±0.8	56.85±0.6	25.21±0.4	61.48±0.8	94.47±0.3	46.01
	RSDT	43.95±0.5	52.83±0.8	51.74±1.7	55.33±0.5	22.99±0.4	64.57±1.0	92.78±0.9	43.71
	REA-tfs	45.39±0.2	54.62±0.2	56.96±0.1	57.22±0.3	25.52±0.3	61.75±0.8	94.62±0.3	46.33
	REA-transfer	45.51±0.3	54.61±0.2	56.52±0.6	57.20±0.7	25.46±0.4	61.04±1.0	-	**46.38**
	PPO-tfs	45.19±0.3	54.37±0.2	55.83±0.7	56.90±0.6	25.24±0.3	61.38±0.7	94.46±0.3	46.02
	PPO-transfer	44.81±0.6	54.15±0.5	55.7.0±1.5	56.60±0.7	24.89±0.5	62.01±1.0	-	45.69
	CATCH	45.27±0.5	54.38±0.2	56.13±0.7	56.99±0.6	25.38±0.4	60.70±0.7	-	46.24
	Global Best	46.32	54.94	57.72	59.62	26.27	59.38	95.37	47.58
Macro level	RS	46.85±0.3	56.5±0.4	70.06±3.1	60.70±0.9	28.37±0.5	59.35±1.0	96.78±0.2	50.52
	RSDT	45.70±0.4	55.06±0.6	59.52±2.4	58.96±0.7	26.69±0.5	62.24±0.7	95.80±0.5	47.28
	REA-tfs	47.09±0.4	56.57±0.4	69.98±3.6	60.88±1.0	28.87±0.4	58.73±1.1	96.88±0.2	50.78
	REA-transfer	46.98±0.4	56.60±0.3	73.41±2.9	61.02±0.8	28.90±0.5	58.18±1.3	-	**51.46**
	PPO-tfs	46.84±0.4	56.48±0.3	70.92±3.2	60.82±0.8	28.31±0.5	58.84±1.1	96.76±0.2	50.75
	PPO-transfer	46.76±0.5	56.47±0.4	70.54±2.9	60.80±1.3	28.31±0.6	59.17±0.8	-	50.62
	CATCH	47.29±0.3	56.49±0.3	70.36±3.0	60.85±0.7	28.71±0.4	59.37±0.6	-	50.72
	Global Best	47.96	57.48	76.88	64.35	29.66	56.28	97.02	53.34

policy transfer works better than direct architecture transfer, whereas it often results in negative transfer; (3) it is possible to improve the policy transfer's robustness with added mechanisms, such as CATCH's encoder and evaluator; (4) Maintaining consistent performance across tasks and search spaces remains a valid challenge for many NAS algorithms.

6 Discussions and Conclusion

Major challenges of transferrable NAS research. After working closely with the benchmark, we realize that (1) the top networks can be very different across tasks. Therefore, the transfer schemes should be able to respond quickly if the task nature has significantly changed. However, effectively detecting and responding to such changes can be difficult. (2) Transfer learning methods usually do not assume they have knowledge about future tasks, but if the policy is specifically designed for NAS, it is possible to incorporate certain NAS features to speed up learning. The major challenges lie in what proper NAS information we can provide and how.

Suggestions for future NAS research: (1) It is important to study efficient NAS strategies that work beyond cell-level search space, as some network attributes, such as the macro skeleton, might have a larger impact on performance for some tasks. (2) When comparing algorithms, besides looking into their final search result, it is also essential to see how much improvements they make upon RS in the same search space. Table 3's REA-tfs on autoencoding has a much higher score in the macro-level search space than algorithms in the cell-level search space, but it performs even worse than RS. (3) When transferring policies and architectures, including some carefully designed mechanisms might help tweak the transferred components toward directions favorable by the target task. (4) Evolutionary methods are not typical strategies studied by the transfer learning literature, but its performance hints that further research can possibly be a promising direction. Also, there might be other strategies outside of the pool of conventional transfer methods that can generalize well.

In this paper, we present TransNAS-Bench-101, a benchmark for improving the transferrability and generalizability of NAS algorithms. We evaluate 7,352 neural networks on 7 image tasks, provide detailed analysis on the benchmark, then point out challenges and suggestions for future research. It is difficult for algorithms to maintain its performance when the task nature has shifted robustly, and experiments show that there is still large room for improvement in NAS methods' generalizability. With this work, we hope to make cross-task NAS research more accessible, and encourage more exceptional algorithms that are both efficient and flexible on multiple tasks and search spaces to evolve. In the future, we will try to (1) enlarge our search spaces, and (2) evaluate all networks with more different seeds. We welcome researchers to test their algorithms' generalizability on TransNAS-Bench-101, and we are happy to include their results in future versions of our benchmark.

REFERENCES

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, 13(1):281–305, 2012.

Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. In *ICLR*, 2019.

Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. Tiny transfer learning: Towards memory-efficient on-device learning. *arXiv preprint arXiv:2007.11622*, 2020.

Liang-Chieh Chen, Maxwell Collins, Yukun Zhu, George Papandreou, Barret Zoph, Florian Schroff, Hartwig Adam, and Jon Shlens. Searching for efficient multi-scale architectures for dense image prediction. In *Advances in neural information processing systems*, pp. 8699–8710, 2018.

Xin Chen, Yawen Duan, Zewei Chen, Hang Xu, Zihao Chen, Xiaodan Liang, Tong Zhang, and Zhenguo Li. Catch: Context-based meta reinforcement learning for transferrable architecture search. *arXiv preprint arXiv:2007.09380*, 2020.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2019a.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pp. 1761–1770, 2019b.

Yong Guo, Yaofo Chen, Yin Zheng, Peilin Zhao, Jian Chen, Junzhou Huang, and Mingkui Tan. Breaking the curse of space explosion: Towards efficient nas with curriculum search.

Shi Han, Pi Renjie, Xu Hang, Li Zhenguo, Kwok James Tin-Yau, and Zhang Tong. Bridging the gap between sample-based and one-shot neural architecture search with bonas. In *NeurIPS*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Detnet: A backbone network for object detection. In *ECCV*, 2018.

Dongze Lian, Yin Zheng, Yintao Xu, Yanxiong Lu, Leyu Lin, Peilin Zhao, Junzhou Huang, and Shenghua Gao. Towards fast adaptation of neural architectures with meta learning. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=r1eowANFvr`.

Feng Liang, Chen Lin, Ronghao Guo, Ming Sun, Wei Wu, Junjie Yan, and Wanli Ouyang. Computation reallocation for object detection. In *ICLR*. OpenReview.net, 2020.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *International Conference on Learning Representations*, 2018.

Ramakanth Pasunuru and Mohit Bansal. Continual and multi-task architecture search. *arXiv preprint arXiv:1906.05226*, 2019.

Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*, pp. 4095–4104, 2018.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, volume 33, pp. 4780–4789, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Albert Shaw, Wei Wei, Weiyang Liu, Le Song, and Bo Dai. Meta architecture search. In *Advances in Neural Information Processing Systems*, pp. 11227–11237, 2019.

Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pp. 6105–6114, 2019.

Catherine Wong, Neil Houlsby, Yifeng Lu, and Andrea Gesmundo. Transfer learning with neural automl. In *Advances in Neural Information Processing Systems*, pp. 8356–8365, 2018.

Hang Xu, Lewei Yao, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Auto-fpn: Automatic network architecture adaptation for object detection beyond classification. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 6649–6658, 2019.

Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. Nas evaluation is frustratingly hard. In *International Conference on Learning Representations*, 2019.

Lewei Yao, Hang Xu, Wei Zhang, Xiaodan Liang, and Zhenguo Li. Sm-nas: Structural-to-modular neural architecture search for object detection. *arXiv preprint arXiv:1911.09929*, 2019.

Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pp. 7105–7114, 2019.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, pp. 3712–3722, 2018.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8697–8710, 2018.