TransNAS-Bench-101: Improving transferability and Generalizability of Cross-Task Neural Architecture Search

Yawan Duan¹*, Xin Chen¹*, Hang Xu², Zewei Chen², Xiaodan Liang³, Tong Zhang⁴, Zhenguo Li²
¹ The University of Hong Kong, ² Huawei Noah’s Ark Lab, ³ Sun Yat-sen University, ⁴ The Hong Kong University of Science and Technology

Abstract

Recent breakthroughs of Neural Architecture Search (NAS) extend 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 a 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 enormous computational cost and experiment complexity of cross-task NAS are imposing barriers for valuable research in this direction. Existing NAS benchmarks all focus on one type of vision task, i.e., classification. In this work, we propose TransNAS-Bench-101, a benchmark dataset containing network performance across seven tasks, covering classification, regression, pixel-level prediction, and self-supervised tasks. This diversity provides opportunities to transfer NAS methods among tasks and allows for more complex transfer schemes to evolve. We explore two fundamentally different types of search space: cell-level search space and macro-level search space. With 7,352 backbones evaluated on seven tasks, 51,464 trained models with detailed training information are provided. 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. Our dataset and code will be available at Mindspore¹ and VEGA².

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 [25, 21]. Ultimately, NAS calls for algorithmic solutions that can discover near-optimal models for any task within any search space under moderate computational budgets. To pursue this goal, recent research in NAS quickly expanded its scope into broader vision domains such as object detection [27] and semantic segmentation [5]. [24] seeks to bridge the gap between sample-based and one-shot approaches. Besides searching for an optimal cell design in earlier works [18], many recent works also investigate the macro skeleton search of a network [29, 27].

Although many algorithms successfully shortened the search time of NAS from months to hours [18, 9], some research has shown their reliance on specific search spaces and datasets [28]. There are also questions on these algorithms’ efficiency when dealing with a large number of tasks [6]. A rising direction of NAS research is thus looking for solutions that acquire transferable knowledge and generalize well across multiple tasks and search spaces [20, 26, 23, 14, 10]. [6] explores meta-learning to transfer network design knowledge from small tasks to larger tasks, surpassing many efficient solutions based on parameter sharing. [3] proposes a highly memory-efficient and effective transfer solution that does not require backpropagation for adaptation. Central to these works’ investigations are two key considerations: the transferability of an algorithm, namely how much information an algorithm can effectively reuse, and generalizability, which evaluates whether a solution can be applied to different settings (e.g., search spaces) and still performs well.

Meanwhile, NAS is prohibitively costly. The computational intensity of single-task NAS can be discouraging, not to mention cross-task NAS experiments. To solve this computation limitation, NAS-Bench-101 [30] and NAS-Bench-201 [8] 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 spaces. To avoid confusion, throughout this paper we use dataset to refer to a set of original images, and task for pointing to certain vision domain. Hence, there can be multiple tasks for one
dataset (e.g., COCO detection and segmentation).

The goal of finding universal solutions across tasks and search spaces, the comparability problem, and the computational barriers of transferable NAS research lead to our proposal of TransNAS-Bench-101, which studies networks over seven distinct 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., where 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 3,256 and 4,096 networks, respectively. The 7,352 backbones are evaluated on all seven tasks, and we provide detailed diagnostic information such as task performance, inference time, and FLOPs for all models. We also evaluated four transfer schemes compatible with both search spaces to serve as benchmarks for future research.

Our key contribution is a benchmark dataset with networks fully evaluated on seven tasks across two search spaces. Generating the benchmark takes 193,760 GPU hours, i.e., 22.12 years of computation on one NVIDIA V100 GPU, but 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 can have bigger impact on performance on certain tasks. (2) The extent to which an algorithm surpasses random search is 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 research. With diversified settings in TransNAS-Bench-101, we hope to encourage the emergence of exceptional transferable NAS algorithms that generalize well under various settings.

2. The TransNAS-Bench-101 dataset

2.1. Search Spaces and Architectures

To plug in different networks for various tasks, our search space focuses on evolving the backbone, i.e., the mutual component of all the tasks considered. We provide two search spaces: a) A macro-level search space that designs the macro 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 the 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 [13]. The allocation of computations over different stages is also vital for a backbone [15]. Therefore, our search space contains networks with different depth (the total number of blocks), locations to down-sample feature maps, and locations to raise the channels. As is illustrated in Figure 1, we group two residual blocks [11] into a module, and the networks are stacked with 4 to 6 modules. The module positions can be chosen to downsample the feature map 1 to 4 times, and each time the spatial size will shrink by a factor of 2. The network can double its channels 1 to
Transferable NAS research is still in its infancy, we hope to provide as many common grounds for transfer as possible. This makes selecting proper datasets challenging since, ideally, the datasets should share some commonalities while covering a diversity of tasks. Thanks to the great previous work Taskonomy [31], which provides sufficient images with labels on different tasks (see Figure 2), we can study algorithms without worrying that the datasets are by nature too distinct for transfer. 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 a total of 120K images from the original dataset and split the subset into 80K train / 20K val / 20K test set.

2.3. Vision Tasks

We carefully selected seven tasks that lie on the intersection of (1) covering all major task categories in Taskonomy’s task similarity tree [31], and (2) align with the community’s general research interests (e.g., classification tasks, or common and cheap pretrain tasks). As is shown in Figure 2, the selected tasks include a) image classification tasks: Object Classification and Scene Classification; b) pixel-level prediction tasks: Surface Normal and Semantic Segmentation; c) self-supervised task: Jigsaw Puzzle and Autoencoding; d) point regression task: Room Layout.

Object Classification. Object classification is a 75-way classification problem that recognize objects. Labels pro-

Table 1: Training hyperparameters and details of each task considered in this benchmark. All the architectures in the search space have been fully trained. We provide multiple metrics for evaluation on the train/valid/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. All the tasks except Autoencoding and Surface Normal use a cosine annealing with a linear warmup learning rate scheduler.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Decoder</th>
<th>LR</th>
<th>Optimizer</th>
<th>Output dim.</th>
<th>Loss</th>
<th>Eval. Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Class.</td>
<td>GAP + Linear</td>
<td>0.1</td>
<td>SGD</td>
<td>75</td>
<td>Softmax+CE</td>
<td>Loss, Acc</td>
</tr>
<tr>
<td>Scene Class.</td>
<td>GAP + Linear</td>
<td>0.1</td>
<td>SGD</td>
<td>47</td>
<td>Softmax+CE</td>
<td>Loss, Acc</td>
</tr>
<tr>
<td>Room Layout</td>
<td>GAP + Linear</td>
<td>0.1</td>
<td>SGD</td>
<td>9</td>
<td>MSE loss</td>
<td>Loss</td>
</tr>
<tr>
<td>Jigsaw</td>
<td>GAP + Linear</td>
<td>0.1</td>
<td>SGD</td>
<td>1000</td>
<td>Softmax+CE</td>
<td>Loss, Acc</td>
</tr>
<tr>
<td>Autoencoding</td>
<td>14 Conv &amp; Deconv</td>
<td>0.0005</td>
<td>Adam</td>
<td>256x256</td>
<td>GAN loss+L1</td>
<td>Loss, SSIM</td>
</tr>
<tr>
<td>Surface Normal</td>
<td>14 Conv &amp; Deconv</td>
<td>0.0001</td>
<td>Adam</td>
<td>256x256</td>
<td>GAN loss+L1</td>
<td>Loss, L1, SSIM</td>
</tr>
<tr>
<td>Sem. Segment.</td>
<td>8 Conv &amp; Deconv</td>
<td>0.1</td>
<td>SGD</td>
<td>256x256</td>
<td>Softmax+CE</td>
<td>Loss, Acc, mIoU</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th># Datasets</th>
<th># Task Domains</th>
<th># Search Space Size</th>
<th>Search Space Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NAS-Bench-101</td>
<td>1</td>
<td>1</td>
<td>510M</td>
</tr>
<tr>
<td>NAS-Bench-201</td>
<td>3</td>
<td>1</td>
<td>15.6K</td>
</tr>
<tr>
<td>TransNAS-Bench-101</td>
<td>1</td>
<td>7</td>
<td>7.3K</td>
</tr>
</tbody>
</table>

The labels provided by the Taskonomy dataset are generated through a network pre-trained on the MSCOCO dataset. Our selected subset contains 17 semantic classes.

2.4. 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 feature maps generated by the backbone. Since the original paper’s implementation is based on an early 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 shown in Table 1. For all the tasks, the batch size is 128, and the input resolution is resized to 256 × 256. We record multiple evaluation metrics in each epoch for all the architectures, as is listed in Table 1. Since we train every architecture in our search space for all the 7 tasks (i.e., 7352 × 7 ≈ 5 × 10⁵ 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

Table 3: We show the basic statistics of the two search spaces, including average performance metric scores, average model FLOPs, the average number of model parameters, and average training time. Note that here model denotes the whole backbone-decoder model structure.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Metric</td>
<td>FLOPs (×10⁶)</td>
<td>Model Params (×10⁶)</td>
<td>Train Time (hr)</td>
<td>FLOPs (×10⁶)</td>
<td>Model Params (×10⁶)</td>
<td>Train Time (hr)</td>
<td></td>
</tr>
<tr>
<td>Cell</td>
<td>Performance</td>
<td>39.71 ± 5.92</td>
<td>45.23 ± 12.15</td>
<td>0.46 ± 0.10</td>
<td>0.52 ± 0.06</td>
<td>19.95 ± 5.52</td>
<td>-0.73 ± 0.12</td>
</tr>
<tr>
<td>level</td>
<td>FLOPs (×10⁶)</td>
<td>2.44 ± 1.44</td>
<td>2.44 ± 1.44</td>
<td>4.90 ± 1.44</td>
<td>4.90 ± 1.44</td>
<td>5.19 ± 1.44</td>
<td>2.44 ± 1.44</td>
</tr>
<tr>
<td></td>
<td>Model Params (×10⁶)</td>
<td>1.17 ± 0.71</td>
<td>1.17 ± 0.71</td>
<td>3.97 ± 0.71</td>
<td>3.97 ± 0.71</td>
<td>2.27 ± 0.71</td>
<td>1.17 ± 0.71</td>
</tr>
<tr>
<td></td>
<td>Train Time (hr)</td>
<td>2.08 ± 0.23</td>
<td>2.13 ± 0.23</td>
<td>5.03 ± 0.54</td>
<td>4.90 ± 0.55</td>
<td>5.98 ± 1.69</td>
<td>2.01 ± 0.31</td>
</tr>
<tr>
<td>Macro</td>
<td>Performance</td>
<td>44.24 ± 1.38</td>
<td>52.92 ± 2.08</td>
<td>0.52 ± 0.08</td>
<td>0.57 ± 0.02</td>
<td>24.47 ± 2.07</td>
<td>-0.65 ± 0.03</td>
</tr>
<tr>
<td></td>
<td>Model Params (×10⁶)</td>
<td>1.18 ± 0.91</td>
<td>1.18 ± 0.91</td>
<td>3.83 ± 1.02</td>
<td>3.83 ± 1.02</td>
<td>2.13 ± 1.02</td>
<td>1.17 ± 0.91</td>
</tr>
<tr>
<td></td>
<td>Train Time (hr)</td>
<td>2.57 ± 2.35</td>
<td>2.56 ± 2.25</td>
<td>7.07 ± 3.91</td>
<td>7.10 ± 4.02</td>
<td>7.04 ± 2.99</td>
<td>2.59 ± 2.25</td>
</tr>
</tbody>
</table>

The labels provided by Taskonomy dataset [31] are activations generated by a ResNet-152 model [11] pre-trained on ImageNet [7].

Scene Classification. Like object classification, scene classification is a 47-way classification problem that predicts the room type in the image. Its labels come from a ResNet-152 model pre-trained on MIT Places dataset [32]. Our selected dataset contains 47 classes out of the original 365 classes.

Room Layout. This task is to estimate and align a 3D bounding box defining the room layout. It requires a network to semantic information, such as “what constitutes a room,” and includes scene geometry.

Jigsaw. We follow [19] to design the self-supervised task Jigsaw. The input image is divided into nine patches and shuffled according to one of 1,000 preset permutations. The objective is to classify which permutation is used. This task’s inclusion is inspired by a recent work [17] that explores neural architecture search without using labels.

Autoencoding. Autoencoding is a pixel-level prediction task that encodes images into low-dimensional latent representations then reconstructs the raw image. The training settings follow Conditional Adversarial Nets in Pix2Pix [12].

Surface Normal. Like autoencoding, surface normal is a pixel-wise prediction task that predicts surface normal statistics. The network structure and training procedure is the same as autoencoding.

Semantic Segmentation. Semantic Segmentation conducts pixel-level prediction on the class of its components.
this way, researchers can significantly speed up their research and focus solely on improving the algorithms without tediously implementing and tuning different tasks.

2.5. Network Information in TransNAS-Bench API

We provide the train / validation / test performance information of each network at every epoch. The benchmark also contains each network’s inference time, FLOPs, the total number of parameters, and time elapsed during each training epoch. Each model’s inference time is measured on one Tesla V100 with one image of shape (3, 720, 1080). FLOPs are computed with one image of shape (3, 224, 224).

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.

As an extension of NAS-Bench-101, NAS-Bench-201 was proposed to accommodate the growing needs. It provides training information about 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 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 an entire 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

4.1. Overview of architectures

The architectures’ performance ranking, FLOPs, parameters, and training time are presented in Figure 3. We obtain rankings of each architecture’s validation performance on all 7 tasks first, then plot the average ranking. A higher number means a better performance ranking.
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. The FLOPs and parameter patterns are similar in the cell-level search space, but the pattern significantly changed when we investigate the macro-level search space. It can take up to 12 hours to train a network in the macro-level search space, which is twice the GPU hour needed to train the most computationally demanding network on the cell-level search space.

4.2. Correlations among tasks

We analyze cross-task correlations by calculating the Spearman’s Rank Correlation coefficient 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.

The Autoencoding task has very distinctive behaviors under the 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 considerable discrepancy shows that the selection of search space can significantly impact specific 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 transferable algorithms should be wary of it to avoid negative transfer.

5. Benchmark Algorithms

Because the field of transferable NAS is new and nascent, there are only limited number of transferable architecture search algorithms. We implemented four baseline algorithms using TransNAS-Bench-101 and provide implementation details and results in this section.

We evaluated 5 transfer schemes of different types: (1) Random Search (RS) [2]; (2) Direct transfer of top architectures (DT); (3) Direct policy transfer of reinforcement learning-based algorithm, e.g., Proximal Policy Optimization (PPO) [22]; (4) Meta-learning based algorithm, e.g., CATCH [6]; (5) Evolutionary algorithms with transferred population initialization, e.g., Regularized Evolutionary Algorithm (REA) [21]. Each selected algorithm represents a distinctive type of transfer scheme, and they are compatible with both search spaces. The algorithms are tested using the MindSpore tools [1].

Evaluation of Random Search and Direct Transfer. Many NAS algorithms have attempted to use architectures as the central component for transfer [33, 20], which inspired us to look into the efficiency of direct transfer of top architectures found by RS. For both search spaces, we randomly search 50 architectures in each trial. The architecture with the highest validation accuracy will be selected as the final searched result of this particular trial. In Table 4 we include the results for direct transfer (DT). For each task, we take the best architecture from the remaining 6 tasks, and average their performance on the selected task. The results are significantly worse than all the search algorithms. It shows that even if some NAS algorithm can be guaranteed to find the optimal architecture on a source task, its direct transfer performance on the target task can still suffer.

Evaluation of policy transfer. For both search spaces, the NAS problem can be formulated as a sequential decision problem, and the reinforcement learning algorithm PPO aims to select each attribute choice to form a network. We use a multi-layer perceptron (MLP) as the policy net-
and average the search results across all tasks. Figure 7(c) shows sorted search results on object classification. Each curve in 50 trials and each trial to search 50 networks.

Evaluation of evolutionary algorithms with transferred population initialization. We reproduced REA in our benchmark, and the result is presented in Figure 6. We randomly initialize a population during the pre-train phase, to be 5.

The direct transfer of PPO policies shows worse results than its non-transfer version, a phenomenon commonly referred to as negative transfer. We conjecture that this is due to PPO’s overfitting to the source task during the pre-train 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 under certain circumstances. As Table 4 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.

Comparison across Transfer NAS Algorithms. The
average performance of each algorithm is presented in Table 4. REA-transfer is the top performer among all evaluated algorithms, finding networks on the 99.02 and 99.11 percentile on the cell-level and macro-level search spaces. The experiments highlight that: (1) Direct transfer of architectures performs significantly worse than random search; (2) Direct 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 challenge for NAS algorithms.

6. Discussions and Conclusion

Major challenges of transferable 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 prior 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 effectively designing such a scheme to provide the most relevant information.

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 transferring policies and architectures, including some carefully designed mechanisms might help tweak the transferred components toward directions favorable by the target task. (3) Evolutionary methods are not typical strategies studied by the transfer learning community, but its performance on the benchmark suggests that it might worth further investigation. Similarly, there might be other strategies outside of the pool of conventional transfer methods that are promising for transferable NAS research.

In this paper, we present TransNAS-Bench-101, a benchmark for improving the transferability and generalizability of NAS algorithms. We evaluate 7,352 neural networks on 7 vision tasks, provide detailed analysis on the benchmark, then point out challenges and suggestions for future research. It is difficult for algorithms to robustly maintain its performance when the task nature has shifted, 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 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.

Table 4: Performance comparison of different transferable 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 rightmost column reports the percentile of searched networks (Perc.) in the benchmark, averaged across all target tasks (i.e. all tasks except Jigsaw).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell level</td>
<td>RS</td>
<td>45.16 ± 0.4</td>
<td>54.41 ± 0.3</td>
<td>55.94 ± 0.8</td>
<td>56.85 ± 0.6</td>
<td>25.21 ± 0.4</td>
<td>61.48 ± 0.8</td>
<td>94.47 ± 0.3</td>
<td>97.91</td>
</tr>
<tr>
<td>REA-transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>REA-tfs</td>
<td>45.39 ± 0.2</td>
<td>54.62 ± 0.2</td>
<td>56.96 ± 0.1</td>
<td>57.22 ± 0.3</td>
<td>25.52 ± 0.3</td>
<td>61.75 ± 0.8</td>
<td>94.62 ± 0.3</td>
<td>99.06</td>
</tr>
<tr>
<td>PPO-tfs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPO-transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CATCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macro level</td>
<td>RS</td>
<td>46.85 ± 0.3</td>
<td>56.5 ± 0.4</td>
<td>70.06 ± 3.1</td>
<td>60.70 ± 0.9</td>
<td>28.37 ± 0.5</td>
<td>59.35 ± 1.0</td>
<td>96.78 ± 0.2</td>
<td>98.18</td>
</tr>
<tr>
<td>REA-tfs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REA-transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPO-tfs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PPO-transfer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CATCH</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DT</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Global Best</td>
<td></td>
<td>47.96</td>
<td>57.48</td>
<td>76.88</td>
<td>64.35</td>
<td>29.66</td>
<td>56.28</td>
<td>97.02</td>
<td>100.</td>
</tr>
</tbody>
</table>
References

[10] 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. 1

5259
