Supplementary Materials for TransNAS-Bench-101: Improving Transferability and Generalizability of Cross-Task Neural Architecture Search

Yawen Duan[1,*], Xin Chen[1,*], Hang Xu[2], Zewei Chen[2], Xiaodan Liang[3,†], Tong Zhang[4], Zhenguo Li[2]
[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

1. Detailed Information of TransNAS-Bench-101 Benchmark Dataset

We provide the train/validation/test performance information of each network at each epoch. One can also find each network's inference time, FLOPs, the total number of parameters, and time elapsed during each training epoch from the dataset. Each network'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).

2. Training Details of Each Task

Object Classification. The labels provided by the Taskonomy dataset [14] are activations generated by a ResNet-152 model [3] pre-trained on ImageNet [2]. For object classification, we train networks with the provided activations. Since we use a subset of the Taskonomy dataset, we identified 75 classes of objects that appear in our selected subset for network training. The data augmentations applied for this task are random flip, color jittering, and normalization. For each network, the decoder part contains a Global Average Pooling (GAP) layer and a linear layer. Referring to the settings of Taskonomy, each network is trained for 25 epochs. Throughout the learning process, we use a cosine annealing scheduler to gradually reduce the learning rate from 0.1 to 0 for fast convergence. The optimizer for parameters is SGD with the momentum factor 0.9, 0.0005 weight decay, and Nesterov momentum is enabled.

Scene Classification. Similar to Object Classification, the Taskonomy dataset's labels for scene classification comes from an ImageNet pre-trained ResNet-152 model. Our selected dataset contains 47 classes out of the original 365 classes. Referring to the settings of Taskonomy, each network is trained for 25 epochs. The data augmentation, decoder, optimizer, and learning rate scheduler settings are the same as Object Classification tasks.

Room Layout. The goal of this task is to estimate and align a 3D bounding box. In the Taskonomy dataset, such a bounding box is defined by a 9-dimension vector. The network is updated through computing the Mean Square Error (MSE) loss with the provided labels. The data augmentation methods used are color jittering and normalization. Following the settings of Taskonomy, each network is trained for 25 epochs. The decoder, optimizer, and learning rate scheduler settings are the same as Object Classification and Scene Classification.

Jigsaw Content Prediction. Jigsaw's inclusion is inspired by a recent work [7] that explores the potential of self-supervised tasks in architecture search. We follow [10] to design the self-supervised task Jigsaw. The input image is divided into 9 patches and shuffled according to one of 1000 preset permutations. The goal of this task is to classify which permutation is used. We use a Siamese network to extract the feature map of each of the 9 image tiles and concatenate them. We apply random flip, color jitter, and random grayscale with a probability of 0.3 for data augmentation. Referring to the settings of Taskonomy, each network is trained for 10 epochs since Jigsaw tasks converge very quickly. The decoder, optimizer, and learning rate scheduler settings are the same as above.

Semantic Segmentation. The labels provided by the Taskonomy dataset on semantic segmentation are generated through a network pre-trained on the MSCOCO [6] dataset. Our selected subset contains 17 semantic classes. We apply random flip, color jitter, and normalization for data augmentation. For this task, we use the SGD optimizer with a learning rate of 0.1, along with a cosine annealing scheduler. Referring to the settings in Taskonomy, each network is trained for 30 epochs. The decoder, optimizer, and learning rate scheduler settings are the same as above.

Autoencoding. The generator networks in the Autoencoding task follow an encoder-decoder structure in Pix2Pix [4], where the encoders are the searched backbones and the decoders contain 14 layers of convolution and deconvolution. We train the generator network using conditional GAN [8] with a discriminator containing 7 convolution lay-

*Equal contribution. ({kmdaniel, cyn0531}@connect.hku.hk)
†Corresponding author. (xdliang328@gmail.com)

ers. We apply spectral normalization [9] to stabilize the discriminator. The generator is trained with the L1 loss with weight 0.99 and GAN loss with weight 0.01. We use structural similarity index measure (SSIM) [13] as the metric for network performance evaluation. The data augmentations applied for the generator are random flip and color jittering. Both the generator and discriminator use Adam [5] optimizer to stabilize the training with an initial learning rate of 0.0005. Referring to the settings in Taskonomy, each network is trained for 30 epochs.

Surface Normal. We use the same generator, discriminator, evaluation metric, and loss for surface normal and autoencoding tasks. For surface normal, the optimizers for both generator and discriminator are Adam with a learning rate of 0.0001. Referring to the settings in Taskonomy, each network is trained for 30 epochs.

3. Cross-task Training Results

We plot the network performance relations for all tasks in Figure 1. Networks in the cell-level search space have a much greater performance gap than networks in the macro-level search space because certain cell designs can easily lead to poor network performance (e.g., choosing skip-connection for all operations). The network performance in most tasks is positively correlated.

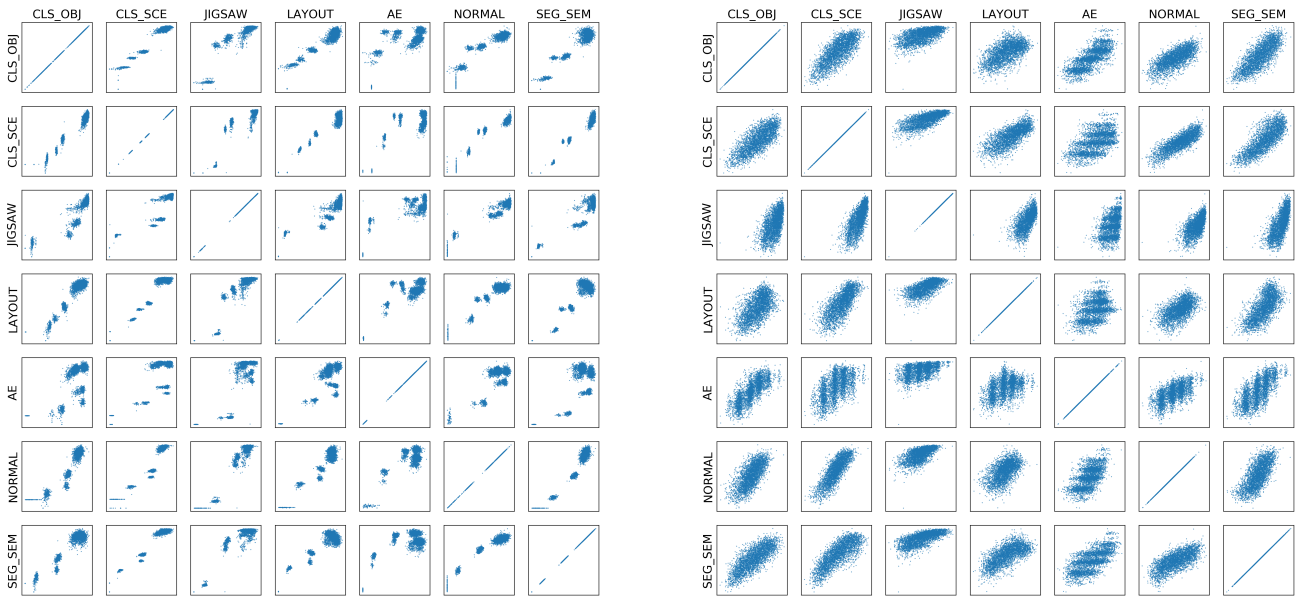
4. Convergence Analysis of Trained Networks

To show the extent to which the ranking of networks in our search space has stabilized, we plot the network ranking correlation between consecutive epochs in Figure 2. We query the network performance at each epoch, then calculate the network ranking correlation between epoch t and epoch $t-1$. A higher value at epoch t means that this additional training epoch does not significantly change the relative advantage of each network in the search space. We plot such correlation on all tasks in Figure 2. From the figures, we can see that the network rankings in most tasks tend to stabilize as they approach the end of the training. Despite the relatively short training budget for each task, the networks have displayed good convergence results.

5. Algorithm Training Details

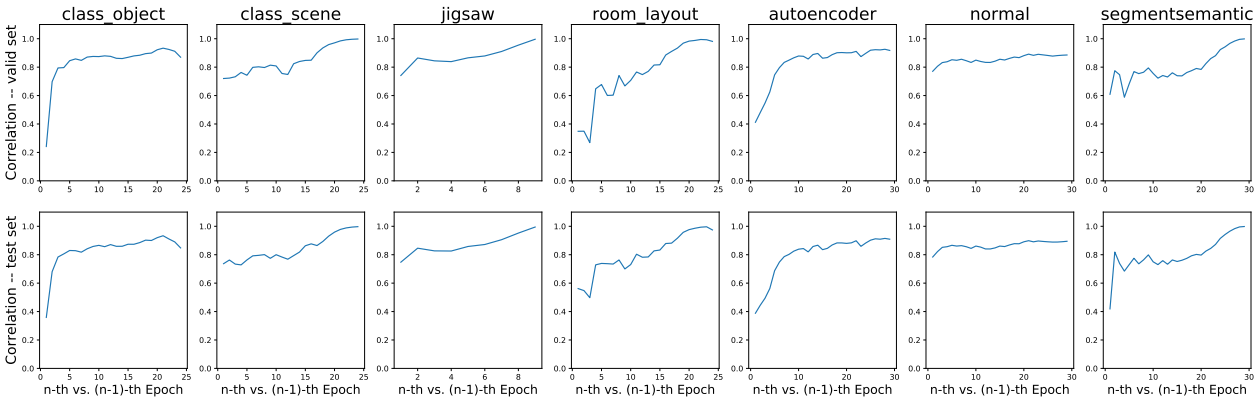
Because the field of transferrable NAS is new and nascent, a limited number of transferrable architecture search algorithms in the research community have been developed. We implemented four baseline algorithms using TransNAS-Bench-101 and provide implementation details of each algorithm below.

Regularized Evolution for Image Classifier Architecture Search (REA). [11]. For both search spaces, the population size and sample size are set to 10. We set the number of cycles as 40. Hence, a total of 50 architectures would be selected. The best validation accuracy throughout the evaluation of a network would be chosen as the fitness. For the

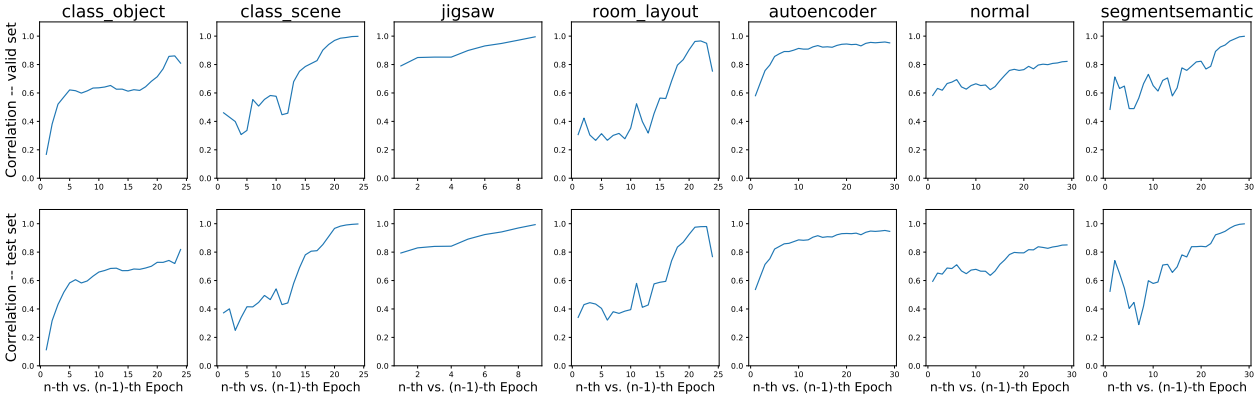


(a) Cell-level (b) Macro-level

Figure 1: Performance of cell-level and macro-level networks across all seven tasks.



(a) Cell-level search space



(b) Macro-level search space

Figure 2: Figure 2(a)-2(b) displays the network rank correlation between the n-th epoch and $(n-1)$-th epoch on the cell-level search space and the macro-level search space. The first row in both subfigures are correlations on the validation set, and the second rows are correlations on the test set.

macro-level search space, the mutation operation would be randomly adding, deleting, or changing a module type.

Proximal Policy Optimization (PPO). [12]. For both search spaces, we can formulate the NAS problem as a sequential decision problem, and the reinforcement learning algorithm PPO aims to select each attribute choice to form a network. We set the learning rate as 0.01, and it decays by 0.999 for every 15 steps. The optimizer is Adam. We set the clipping parameter ϵ as 0.2, memory size as 100, discount γ as 0.99, GAE parameter λ as 0.95, value function coefficient as 1, and entropy coefficient as 0.01. For PPO-transfer, we first pre-train the policy by applying the policy to search on the lowest cost tasks, jigsaw, then transfer to target tasks.

Context-based Meta Reinforcement Learning for Transferrable Architecture Search (CATCH). [1]. CATCH uses PPO as its controller to sample networks. It also uses a network evaluator to predict the network performance and uses a context encoder to learn a task-specific embedding to guide the search. CATCH incorporated meta reinforcement learning by first meta-training the policy on various tasks, such as jigsaw, classification tasks, and then adapt the meta-trained policy to a target task. We set the hyperparameters for the controller the same as those of PPO. For the context encoder, we set its learning rate as 0.0005, KL Divergence weight as 0.1. For the evaluator, we set its learning rate as 0.0005, initial epsilon ϵ as 1, and it decays by 0.025 for every 4 steps to encourage exploration. In the adaptation phase, the initial epsilon for the evaluator is set to 0.5 and decays by 0.025 for every 2 steps to encourage exploitation.

References

[1] 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. 3

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[3] 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*, pages 770–778, 2016. 1

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 1

[5] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[7] Chenxi Liu, Piotr Dollár, Kaiming He, Ross Girshick, Alan Yuille, and Saining Xie. Are labels necessary for neural architecture search? *arXiv preprint arXiv:2003.12056*, 2020. 1

[8] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 1

[9] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 2

[10] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision*, pages 69–84. Springer, 2016. 1

[11] 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, pages 4780–4789, 2019. 2

[12] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 3

[13] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004. 2

[14] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, pages 3712–3722, 2018. 1