NASTransfer: Analyzing Architecture Transferability in Large Scale Neural Architecture Search

**Rameswar Panda[1,2], Michele Merler[1], Mayoore S Jaiswal[1], Hui Wu[1,2],
Kandan Ramakrishnan[4], Ulrich Finkler[1], Chun-Fu Richard Chen[1,2], Minsik Cho[1],
Rogerio Feris[1,2], David Kung[1], Bishwaranjan Bhattacharjee[1]**

[1]IBM Research
[2]MIT-IBM Watson AI Lab

Abstract

Neural Architecture Search (NAS) is an open and challenging problem in machine learning. While NAS offers great promise, the prohibitive computational demand of most of the existing NAS methods makes it difficult to directly search the architectures on large-scale tasks. The typical way of conducting large scale NAS is to search for an architectural building block on a small dataset (either using a proxy set from the large dataset or a completely different small scale dataset) and then transfer the block to a larger dataset. Despite a number of recent results that show the promise of transfer from proxy datasets, a comprehensive evaluation of different NAS methods studying the impact of different source datasets has not yet been addressed. In this work, we propose to analyze the architecture transferability of different NAS methods by performing a series of experiments on large scale benchmarks such as ImageNet1K and ImageNet22K. We find that: (i) The size and domain of the proxy set does not seem to influence architecture performance on the target dataset. On average, transfer performance of architectures searched using completely different small datasets (e.g., CIFAR10) perform similarly to the architectures searched directly on proxy target datasets. However, design of proxy sets has considerable impact on rankings of different NAS methods. (ii) While different NAS methods show similar performance on a source dataset (e.g., CIFAR10), they significantly differ on the transfer performance to a large dataset (e.g., ImageNet1K). (iii) Even on large datasets, random sampling baseline is very competitive, but the choice of the appropriate combination of proxy set and search strategy can provide significant improvement over it. We believe that our extensive empirical analysis will prove useful for future design of NAS algorithms.

Introduction

Neural Architecture Search (NAS) is a very active area of research (Elsken, Metzen, and Hutter 2019), aiming at automatic design of deep learning networks for various applications spanning from image classification (Liu et al. 2018a; Tan et al. 2019; Wu et al. 2019; Xie et al. 2019) to NLP (Liu, Simonyan, and Yang 2018; Pham et al. 2018; Zoph and Le 2017), from object detection (Chen et al. 2019; Ghiasi, Lin, and Le 2019; Wang et al. 2019) to semantic segmentation (Liu et al. 2019). A number of NAS strategies have been proposed, including evolutionary methods (Baker et al. 2018; Liu et al. 2018b; Real et al. 2019; Zhou et al. 2020), reinforcement learning (Liu et al. 2018a; Pham et al. 2018; Zhong et al. 2018; Zoph and Le 2017), and gradient-based methods (Chang et al. 2019; Liu, Simonyan, and Yang 2018; Nayman et al. 2019; Xie et al. 2019; Zela et al. 2020). Despite impressive results on common benchmark datasets, the prohibitive computational demand of existing NAS methods makes it difficult to directly search the architectures on large-scale datasets (e.g., ImageNet). Motivated by this, many methods have been proposed to improve the efficiency of NAS by shifting the training and evaluation of candidate architectures from the entire target set to proxy tasks, which could mean learning with only a few blocks, training for a few epochs (Kyriakides and Margaritis 2020) or using proxy sets (Liu, Simonyan, and Yang 2018; Pham et al. 2018; Zhou et al. 2020). Proxy sets could either be smaller versions of the target dataset obtained through sampling, or datasets with similar distribution to the target, but with reduced number of classes or number of examples per class.

Despite a number of recent work showing promising transfer results, comparison between different NAS methods in terms of architecture transfer remains a novel and rarely addressed problem. Specifically, it is not clear to what extent the architectures depend on the proxy dataset on which the search is conducted and how does the performance on the target dataset depend on the NAS method that is used to search the architectures. Moreover, a thorough study on applicability of proxy datasets to large scale contexts such as ImageNet22K is still missing. In fact, even direct training and evaluation of standard human-designed architectures has been relatively limited for ImageNet22K, given not only its sheer scale ($\sim$14M images) but also its large imbalance across classes (Cho et al. 2017; Chilimbi et al. 2014; Codreanu, Podareanu, and Saletore 2017; Zhang et al. 2015).

Motivated by this, in this paper, instead of focusing on beating the latest benchmark numbers on small scale datasets like CIFAR10 (Krizhevsky 2009), we take a step back and aim at filling the above gap with an extensive empirical study on architecture transferability of different NAS methods that explains and suggests best practices for proxy sets design and successful transfer at large scale. We compare four representative NAS methods such as ENAS (Pham et al. 2018), NSGANet (Lu et al. 2019), NAO (Luo et al. 2018) and DARTS (Liu, Simonyan, and Yang 2018) using the commonly used DARTS search space on six diverse datasets, MIT67 (Quattoni and Torralba 2009), FLOWERS102 (Nilsback and Zisserman 2008), CIFAR10 (Krizhevsky 2009), CIFAR100 (Krizhevsky 2009), ImageNet1K (Deng et al. 2009) and ImageNet22K (Russakovsky et al. 2015), to analyze their transfer performance under different settings. Our findings suggest that transfer performance of architectures searched using completely different small datasets perform similarly to the architectures searched directly on proxy target datasets (e.g., CIFAR10 proves to be a valuable dataset for transferring architectures to ImageNet22K). Reliably good search for large scale datasets can be performed on proxy sets even smaller than two orders of magnitude with respect to the target ones.

Furthermore, we show that (a) While different NAS methods show similar performance on a source dataset, they significantly differ on the transfer performance to a large dataset. (b) Even on large datasets, random sampling remains a strong baseline to surpass, but the choice of the appropriate combination of proxy set and search strategy can provide significant improvement over it.

Related Work

Neural Architecture Search. Neural Architecture Search has attracted intense attention in recent years. Typically, a NAS algorithm first defines a search space and then employs a search strategy within that space. During the search phase, some evaluation criteria are chosen to rank the relative performance of possible architecture candidates (Elsken, Metzen, and Hutter 2019; Yu et al. 2020). Recent studies (Dong and Yang 2020; Li and Talwalkar 2019; Yang, Esperança, and Carlucci 2020; Ying et al. 2019) have shown that performance is highly dependent on the elaborately designed search space, within which the difference between different search strategies results less significant than initially thought, especially when compared to random search (Li and Talwalkar 2019; Yang, Esperança, and Carlucci 2020). On the other hand, the search phase for candidate architectures within the search space highly influences the efficiency of a NAS algorithm. The original reinforcement learning based method (Zoph and Le 2017), for example, required hundreds of GPUs in order to evaluate and rank each proposed architecture. Different methods have been proposed to reduce the search and evaluation costs, including micro-search of primary building cells (Zhong et al. 2018; Zoph et al. 2018), prediction of candidate architectures performance based on learning curves (Baker et al. 2018; Finkler et al. 2020) or surrogate models (Liu, Simonyan, and Yang 2018), and parameter sharing between child models (Bender et al. 2018; Brock et al. 2018; Liu, Simonyan, and Yang 2018; Pham et al. 2018; Zela, Siems, and Hutter 2020).

Architecture Transferability. Most NAS approaches usually perform well when searching an architecture for a specific dataset and/or task, but have a hard time generalizing. In order to overcome the computational burden of running NAS searches for every new target domain, methods have been developed for joint training and efficient transfer of prior knowledge between multiple search spaces and tasks (Borsos, Khorlin, and Gesmundo 2019; Wistuba 2019; Lu et al. 2020). Some methods obtain transferability based on meta-learning (Lian et al. 2020) or learning general supernets from which specialized subnets can be sampled without any additional training (Lu et al. 2020). Other approaches search for the best cell on a small proxy dataset and then trasnfer to the large target dataset by stacking together more copies of this cell, each with their own parameters to design a convolutional architecture (Zoph et al. 2018). In this work, we focus on the latter type of approaches and investigate their applicability at large scale.

NAS Proxies. Although recent NAS methods (Liu et al. 2018a; Liu, Simonyan, and Yang 2018; Pham et al. 2018) improve the search efficiency to some extent, the search process is still time-consuming and requires vast computation overhead when searching in a large search space since all network candidates need to be trained and evaluated. Differentiable approaches such as DARTS (Liu, Simonyan, and Yang 2018) require high GPU memory consumption, which still makes direct search on large dataset prohibitive. A widely used approach to address efficiency in NAS methods is to search for an architectural building block on a small dataset (either using a proxy set from the large dataset or a completely different small scale dataset) and then transfer the block to the larger dataset by replicating and stacking it multiple times in order to increase network capacity according to the scale of the dataset. While proxy sets have been largely used to expand search results from small (CIFAR10, CIFAR100) to mid-size (ImageNet1K) datasets, and some works have been able to perform search on mid size datasets (Cai, Zhu, and Han 2019), a study of their applicability to large scale datasets is still missing. In this work, we offer a detailed and extensive study of the effects of proxy sets on network transferability to large scale targets. We hope this will contribute to an established protocol of reproducibility when studying NAS algorithms going from small-to-medium proxies to large scale target datasets.

NASTransfer Benchmark

In this section, we discuss the details about our proposed NASTransfer benchmark, in terms of datasets, methods and evaluation metric used to compare different methods.

Objective. Our goal is to provide diagnostic information on the architecture transfer performance of different NAS methods and proxy sets for large scale NAS, which can be potentially used for better designs of future NAS algorithms. We adopt a common search space and training protocol to avoid the effect of the manually engineered tricks and search space widely used in different NAS methods.

Datasets and Proxy Sets. We select six diverse and challenging computer vision datasets in image classification, namely MIT67 (Quattoni and Torralba 2009), FLOWERS102 (Nilsback and Zisserman 2008), CIFAR10 and CIFAR100 (Krizhevsky 2009), ImageNet1K (Deng et al. 2009) and ImageNet22K (Russakovsky et al. 2015) to evaluate the performance of different methods[1]. While most of the existing analysis on NAS (Yang, Esperança, and Carlucci 2020;

[1] ImageNet is used only for research purposes to allow bench-

Zela, Siems, and Hutter 2020; Ying et al. 2019; Dong and Yang 2020) focus on small scale datasets such as CIFAR10, we show large-scale experiments on the ImageNet22K (Russakovsky et al. 2015) dataset that contains over 14 million labeled high-resolution images belonging to around 22K different categories. The ImageNet22K dataset skew is reflective of real world tasks and provides a natural testbed for our method when comparing training sets of different sizes.

As proxy sets for the larger datasets, we employed not only small-scale datasets such as CIFAR10, but also sampled subsets of ImageNet1K and ImageNet22K directly. Namely, we investigated the proxies listed in Table 1, which are of two types: randomly selected and uniformly selected. For random selection, we picked a list of N classes and used all of their images. In uniform selection we were interested in maintaining the overall distribution of examples for all classes in the dataset, therefore we sorted classes by their number of images and then uniformly sampled in order to obtain the desired number of classes N in the subset. In order to maintain the order of magnitude consistent across multiple proxies, we then took the same fraction of images from every class, ensuring that the total would meet the requirement and at the same time maintain the overall distribution intact. This is particularly important when designing a proxy set for a non-uniform, imbalanced distribution such as the one of ImageNet22K. For example ImageNet22K Proxy 2 was designed to have the same overall distribution of the full dataset, but the same number of images of ImageNet22K Proxy 1. In order to do so, we sampled 0.97% of images from each class in the dataset, and eliminated classes for which only one image remained for training or validation, thus keeping only approximately 13k classes out the the 22k total. For ImageNet22K Proxy 3 instead we uniformly picked 100 classes whose total number of images would be the same as ImageNet22K Proxy 1. We split each of those datasets into a training, validation and testing subsets with proportions 40/40/20 and use standard data pre-processing and augmentation techniques.

Methods and Search Space. We compare four representative NAS methods: DARTS (Liu, Simonyan, and Yang 2018), ENAS (Pham et al. 2018), NSGANet (Lu et al. 2019) and NAO (Luo et al. 2018), including the random sampling (Yang, Esperança, and Carlucci 2020) baseline. We choose these methods as they have a reasonable search time, specifically under 4 GPU-days on CIFAR10 dataset. We perform micro-search at cell level within the DARTS search (Liu, Simonyan, and Yang 2018): 3×3 and 5×5 separable convolutions, 3×3 and 5×5 dilated separable convolutions, 3×3 max pooling, 3×3 average pooling, identity, and zero. All operations are of stride one (if applicable) and the convolved feature maps are padded to preserve their spatial resolution. ReLU-Conv-BN order are used for convolutional operations, and each separable convolution is always applied twice. Note that our NASTransfer benchmark has a fixed search space and hence provides a unified benchmark for analyzing transferability of different NAS algorithms.

Set Name	Selection Method	# of Classes	# Train Images	# Val Images
MIT67	All	67	5K	1.4K
FLOWERS102	All	102	6.5K	1.7K
CIFAR10	All	10	50K	10K
CIFAR100	All	100	50K	10K
ImNet1K P1	Random	100	128K	5K
ImNet1K P2	Random	200	258K	10K
ImNet1K P3	Random	300	384K	15K
ImNet1K P4	Random	200	128K	5K
ImNet1K P5	Uniform	1,000	128K	5K
ImNet22K P1	Random	100	35K	35K
ImNet22K P2	Uniform	13,377	35K	35K
ImNet22K P3	Uniform	100	35K	35K
ImNet1K	All	1,000	1.2M	50K
ImNet22K	All	21,841	7.5M	7M

Table 1: Proxy sets used in our experiments. ImNet1K P1 refers to the Proxy 1 selected from ImageNet1K. In Uniform selection the distribution of examples is reflective, although in smaller scale, of the overall distribution of the entire dataset. This is of particular relevance for ImageNet22K, where images are not uniformly distributed over classes.

Training and Evaluation Protocol. NAS algorithms traditionally work in two phases: first *search*, in which the best architecture is determined based on the search algorithm employed, and second *augmentation*, which consists in training from scratch the model selected during the search phase. We choose the search hyperparameters as close as possible to the ones reported in the original papers. Experiments on all datasets use the same hyperparameters except the number of training epochs. For augmentations, we use cross entropy loss, SGD optimizer with learning rate 0.025, momentum 0.9, seed 2, initial number of channels 36, and gradient clipping set at 5. The impact of the seed was investigated in one ablation study for all methods in Table 3. While different augmentation strategies haven shown to be effective in improving the results, we did not use any such data augmentation strategy or other learning tricks to make a fair comparison among different NAS methods. We provide the effect of different augmentation strategies such as Drop path (Larsson, Maire, and Shakhnarovich 2017), Auxiliary towers (Xie et al. 2019) and Cutout (DeVries and Taylor 2017) in the supplementary material which shows that these widely used augmentation strategies have a larger impact on small datasets, but fails to provide consistent improvements on large datasets. The number of cells was fixed to 20 for all experiments and the number of training epochs per dataset was set to 600, 600, 600, 600, 120 and 60 for augment runs on MIT67, FLOWERS102, CIFAR10, CIFAR100, ImageNet1K and ImageNet22K, respectively. All searches were performed on a single GPU, while augment runs were done on single GPU for CIFAR10 and CIFAR100. We use a minimum of 8 to a maximum of 96 GPUs for ImageNet1K and ImageNet22K augmentation experiments.

Metrics. Following (Yang, Esperança, and Carlucci 2020), we compute both Top-1 classification accuracy of augmentation runs on target datasets as a metric of performance

marking against prior results. The trained models in this work are not used for commercial purposes.

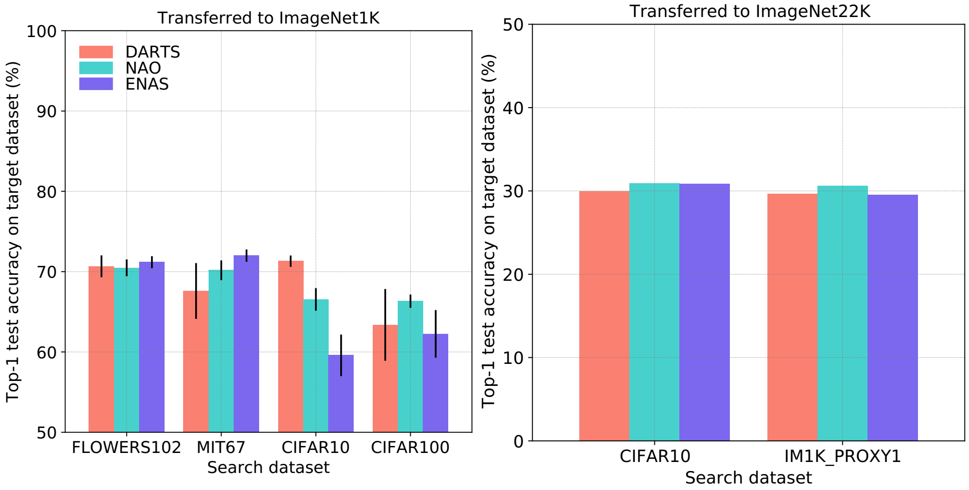

Figure 1: Architecture transfer performance on large datasets. The the overall size of the proxy set is not an important factor for trasferability. CIFAR10 proves to be a valuable proxy set for both ImageNet1K and ImageNet22K. Best viewed in color.

for each method, and relative improvement (RI) over a random sampling baseline, which is computed as $RI = 100 \times \frac{Acc_m - Acc_r}{Acc_r}$, where Acc_m and Acc_r represent Top-1 accuracy of the search method and random sampling, respectively. RI provides a measure of the quality of each search strategy alone, since both searched and randomly sampled architectures share the same search space and training protocol. A good general-purpose NAS method is expected to yield $RI > 0$ consistently over different searches and across different sub tasks. Note that this comparison is not against random search, but rather against random sampling, i.e., the average architecture of the search space. In our experiments, we compute Acc_r as the average of augmentation runs over N randomly sampled architectures.

Results & Analysis

In this section, we provide detailed analysis on the architecture transfer performance of various NAS methods under different proxy sets, comparison with the random sampling baseline and effect of hyperparameters on the performance of different NAS methods.

Transferring Architectures. Despite recent efforts to significantly improve the speed of search algorithms, performing direct search on large scale target datasets remains prohibitive, unless extremely powerful resources are utilized. For example for NSGANet, the search times on CIFAR10 and CIFAR100 with single-GPU is 96 GPU-hours on average, while single-GPU direct search on ImageNet1K Proxy 1 would take almost two months (which we estimated based on the progress of a four days long run on CIFAR), over one year and half on the full ImageNet1K and approximately 19 years on ImageNet22K. Even the fastest search method we analyzed, ENAS (0.375 days for CIFAR10), would require over one year on ImageNet22K. Therefore the need for effective, small-scale proxy sets that could provide a ranking of searched architectures which remains consistent when transferring to the target large scale datasets becomes crucial. But, how to properly select a proxy-set? From Figure 1, we observe that the size of the proxy set does not influence the effectiveness of an architecture searched on it and then transferred to a vastly larger set. In fact, even architectures searched on the very small FLOWERS102 and MIT67 sets, with less than 10K images, yield actually better results than searches on CIFAR100 for all search strategies. Only DARTS performs better with search on CIFAR10 than the smaller datasets. It is interesting to see that for most search methods transferring to ImageNet1K from CIFAR10 is more effective than transferring from CIFAR100. The overall number of images in CIFAR10 and CIFAR100 is the same, but the number of examples per class is $5,000$ for the former and only 500 for the latter (half of the ImageNet1K distribution). When comparing the two CIFAR datasets, ENAS seems to privilege a smaller number of examples per class, while DARTS shows an opposite trend.

We also notice how the domain of the proxy set does not seem to influence architecture performance on the target dataset. Intuitively, one would think that a proxy set built from a subset of the target dataset (ImageNet1K Proxy 1 in the Figure, on the right) will yield better results than a proxy set coming from a different dataset (CIFAR10). That appears not to be the case for the target dataset ImageNet22K, as shown in Figure 1 on the right. While both CIFAR10 and ImageNet contain images of natural scenes, there is still a significant difference in terms of image subjects and even resolution. Nonetheless, CIFAR10 proves to be a valuable proxy set for large-scale datasets like ImageNet22K.

Direct Search using Proxy Sets. In order to determine the benefit of employing proxy sets directly sampled from the target datasets for architecture search, we compared the performance of the searched architectures versus randomly sampled ones for each of the target datasets. For each method, including random sampling, search was conducted five times, and the resulting mean and standard deviations of the five runs are reported in Figure 2. We observe that for medium scale, uniformly distributed datasets such as ImageNet1K, the rankings of search strategies remains unaffected by proxy set design as DARTS is the best perform-

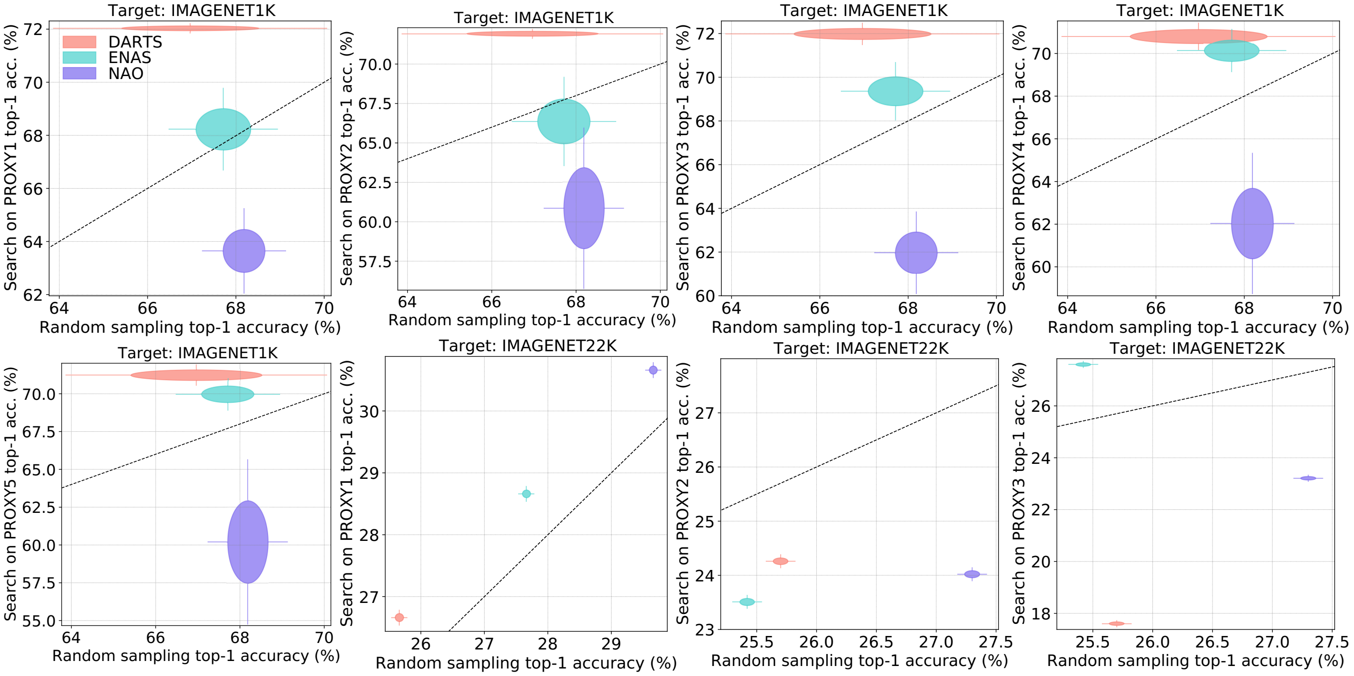

Figure 2: Direct search using proxy sets. Comparison of different NAS methods using sampled proxy sets from same target dataset. Methods lying in diagonal perform the same as randomly sampled architecture, while methods above the diagonal outperform it. We use total five proxy sets for ImageNet1K and three proxy sets for ImageNet22K. Best viewed in color.

ing method, followed by ENAS and NAO. The number of classes and examples in the proxy set does not bring noticeable improvements, as long as a minimum is guaranteed, as shown by the comparable results using Proxy 1, 2 and 3. Overall, random sampling of a reduced number classes when building the proxy set seems to provide better performance than keeping all the classes in the target dataset and reducing the number of examples per class (Proxy 5).

While analyzing the search for very large scale datasets with a skewed distribution (ImageNet22K), the design of proxy set has a large impact not only on the overall improvement, but also on the rankings of different NAS methods. ImageNet22K Proxy 1 results are significantly superior to all other proxies sampled from ImageNet22K for DARTS, ENAS and NAO, and better than random sampling. When searching on ImageNet22K proxies 2 and 3, the random sampling baseline becomes difficult to beat for all methods, and DARTS goes from being the top ranked to the bottom one. This underlines the importance of carefully selecting and designing the proxy set for reliable architecture search. Random sampling of a subset of classes while maintaining the number of images per class proves to be more beneficial, than trying to keep all classes represented in the dataset and eliminating a large portion of examples per class to maintain the search time practically feasible.

Architecture Transfer vs Proxy-based Direct Search. From the results reported in Figure 3, we can see that on average, transfer performance of architectures searched using completely different small datasets (e.g., MIT67, CIFAR10) can perform similarly or even better than architectures searched directly on proxy target datasets. However, design of proxy sets has considerable impact on rankings of different NAS methods. From the Figure, we can see that for NAO, using a small, unrelated small dataset as proxy provides consistently better results than the best possible proxy sampled from the target dataset, both for ImageNet1K and ImageNet22K. It is surprising to observe how well CIFAR10 works as proxy for ImageNet22k across all methods. Using CIFAR10 produces better results not only than proxy sets from ImageNet1K, but also better than proxy sets directly sampled from ImageNet22K. One would assume that using a subset of the target dataset for search would be beneficial, especially when the distribution across classes is significantly skewed as it is for ImageNet22K. The results of our experiments suggest that a small proxy set, different from the target dataset, albeit in the same general field (natural images classification) can lead the search process to find valuable architectures for datasets even at the scale of ImageNet22K: 14 million images. For reference, the state-of-the-art published result on ImageNet22K is 36.9 Top-1 accuracy using a Wide Residual Network WRN-50-4-2 (Codreanu, Podareanu, and Saletore 2017), project Adam's network (Chilimbi et al. 2014) achieved 29.8%, whereas the architecture searched with NAO using CIFAR10 as a proxy yields 30.91 Top-1 accuracy.

Comparison with Random Sampling. We compare with randomly sampled architectures within the same search space to verify the effectiveness of each method for every augment runs. We want to emphasize that this comparison is not against random search, but rather against random sampling, i.e., the average architecture of the search space. We sample 5 architectures randomly from the search space and

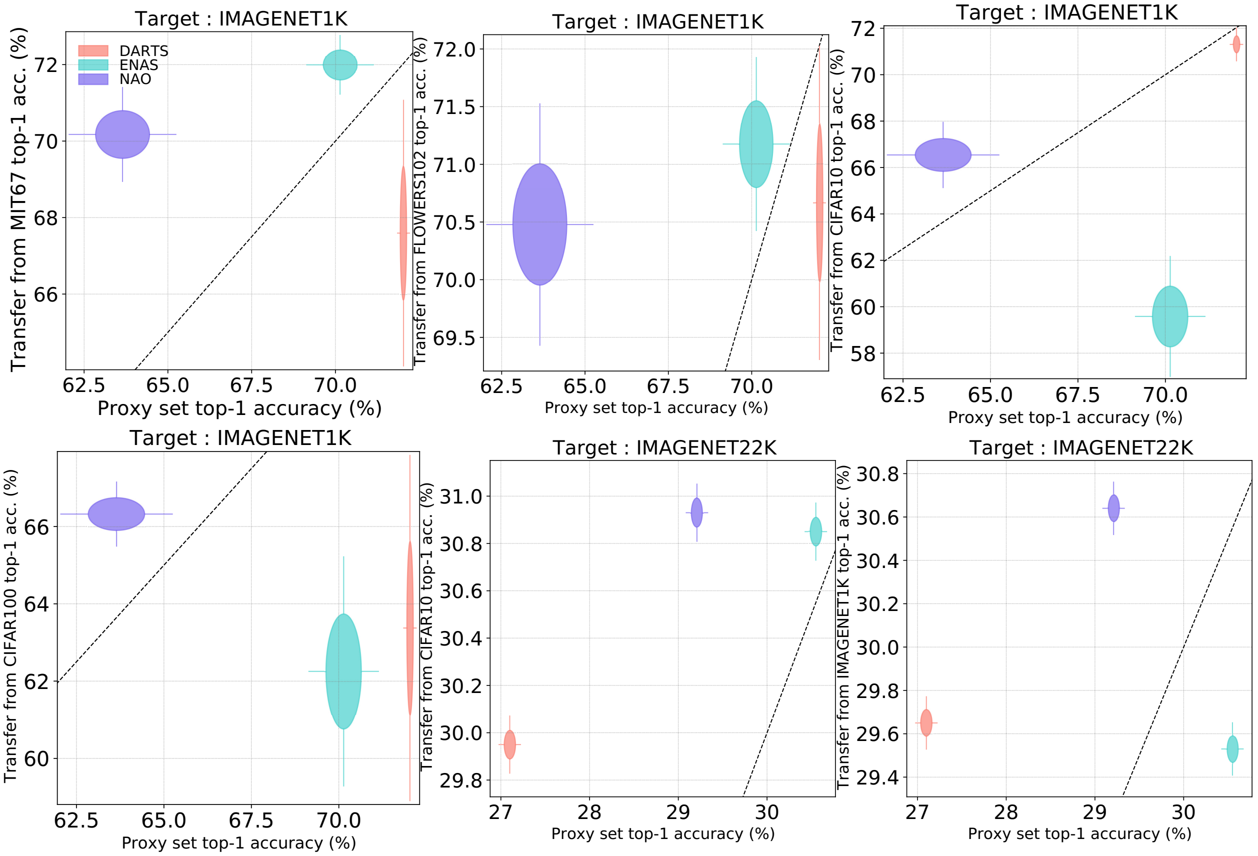

Figure 3: Architecture transfer vs Proxy-based direct search. Comparison of transfer performance with the best proxy-based direct search on ImageNet1K and ImageNet22K datasets. Methods lying in the diagonal indicate that transfer performance is similar to the direct proxy-based search, while methods above the diagonal outperform it. Best viewed in color.

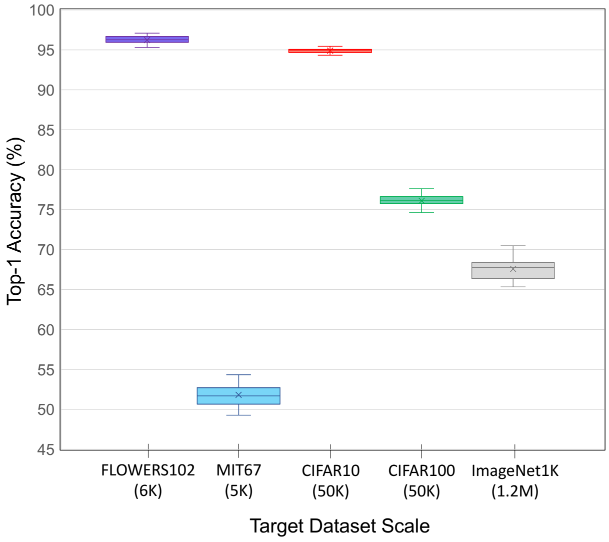

Figure 4: Random Sampling. Standard deviation of top-1 accuracy over 30 runs increases with scale and diversity of datasets. Best viewed in color.

compare with the same number of architectures searched by each method. From the results in Figure 2 and Table 2 the random sampling strategy proves to be a very strong baseline, confirming that the effect of a search strategy is less influential for final performance of a given architecture compared to accurately designing the search space. This effect becomes particularly evident when trying to transfer from a small proxy set to a larger one, especially when the number of examples per class varies significantly between proxy and target sets. This is the case for the transfer experiment between CIFAR100 and ImageNet1K, where the architectures learned by all search methods on CIFAR100, with the exception of NSGANet, perform significantly worse than the direct application of randomly sampled ones. Interestingly, CIFAR10 seems instead like a good proxy for transfer to all other target datasets, including ImageNet22K. To further analyze the effect of number of random sampled architectures, we sample 25 more randomly sampled architectures (total 30) on FLOWERS102, MIT67, CIFAR10, CIFAR100 and ImageNet1K. The influence of search space design and the strength of the random sampling baseline becomes less important as the scale and complexity of the target dataset increases. The standard deviation around the average performance of 30 randomly sampled architectures expands as the scale of the target dataset increases and as

	S: CIFAR10 T: CIFAR100	S: FLOWERS102 T: ImageNet1K	S: MIT67 T: ImageNet1K	S: CIFAR10 T: ImageNet1K	S: CIFAR100 T: ImageNet1K	S: CIFAR10 T: ImageNet22K	S: ImageNet1K T: ImageNet22K
NSGANet	1.38	-9.16	-7.02	3.39	0.92	-0.72	-
ENAS	1.42	6.12	4.84	-16.26	-11.75	6.16	1.62
DARTS	1.01	6.57	0.89	5.93	-18.71	3.06	2.03
NAO	-0.89	2.56	1.90	0.99	-2.68	6.44	5.44

Table 2: Relative improvement metric RI for various transfer experiments. S and T indicate the source set and target set respectively. Given its much longer search time, we did not perform NSGANet search on ImageNet1K proxies.

Method	CIFAR10 – ImageNet1K Transfer		ImageNet1K Proxy 1 Direct	
	AS = 2	AS = 3	AS = 2	AS = 3
NSGANet	69.86	69.27	-	-
ENAS	56.58	72.69	69.40	64.70
DARTS	71.58	72.01	71.89	71.68
NAO	68.24	64.52	64.98	61.85
Random Sampling	-	-	70.45	71.35

Table 3: Ablation studies on augmentation seed. Results show performance on two different augmentation seeds. The default value of augmentation seed (AS) is set to 2.

Experiment	NAS Method	Number of Cells		
		20	40	60
CIFAR10 – Imagenet1K Transfer	NSGANet	68.38	71.62	62.16
	ENAS	56.58	72.50	72.55
	DARTS	71.58	63.75	53.98
	NAO	68.24	63.87	58.54
ImageNet1K Proxy 1	ENAS	69.40	69.61	60.31
	DARTS	71.89	70.04	60.60
	NAO	64.98	67.32	52.45
ImageNet1K	Random Sampling	70.45	65.52	52.82

Table 4: Effect of number of cells on ImageNet1K. Results show performance with three different number of cells on ImageNet1K dataset. Increase in number of cells does lead to increase in performance.

the average accuracy decreases (see Figure 4). This trend signifies a larger opportunity for impact of the search strategy within the space for datasets that present a hard classification task, like MIT67, and/or large-scale datasets like ImageNet1K. As direct search on large scale datasets is basically intractable in practice, it also shows the importance of finding good proxy sets where search is feasible and performance gains transfer to the target dataset.

Effect of Hyperparameters. We conduct extensive ablation studies over the augmentation training hyperparameters in order to precisely determine the merits of search methods and choice of proxy set versus training protocols. Namely, we investigated augment results by varying seed (Table 3) and number of cells (Table 4). In general we observe that that the parameters of the training protocol have a larger impact on small datasets, but it fails to provide consistent improvements on large datasets, whereas the choice of an appropriate proxy set and search strategy are more relevant. From Table 3, it appears that ENAS is particularly sensitive to choice of augmentation seed, especially when transferring from CIFAR10 to ImageNet1K datset.

Conclusions and Best Practices

We have presented the first extensive study on the design and transfer value of proxy sets for NAS at large scale across different search methods. We compared four standard search strategies on proxy datasets ranging from small (5K images) to large (384K images) and their transferability to very large scale target sets including for the first time ImageNet22K (14M images). The results of our experiments and ablation studies suggest the following set of best practices when choosing proxy sets. (i) The the overall size of the proxy set is not an important factor for trasferability. Reliably good search for large scale datasets can be performed on proxy sets even smaller than two order of magnitude with respect to the target ones. (ii) The domain of the proxy set does not seem to influence architecture performance on the target dataset. Transfer performance of architectures searched using different small datasets (e.g., MIT67, CIFAR10) can perform similarly or even better than architectures searched directly on proxies of target datasets (ImageNet1K and ImageNet22K). (iii) For proxy sets directly sampled from the target set, the random sampling of a subset of classes maintaining the same number of images per class is more beneficial than trying to keep all the classes from the target dataset and the reducing the number of examples per class. (iv) As the scale of target dataset increases, the choice of proxy set and search strategy matters more on the final augmentation performance on the target dataset than the training protocol and hyper-parameters setting. NAS methods showing similar performance on a source dataset (e.g., CIFAR10), produce largely different transfer performances to a large dataset (e.g. ImagenNet22K). (v) Random sampling remains a strong baseline to surpass, but the choice of the appropriate combination of proxy set and search strategy can provide significant improvement over it. In future, we plan to further study the transferablity from proxy sets of significantly different domains. We also believe that including the appropriate selection of proxy set in combination with a search strategy within a unified NAS framework can lead to not only efficient but also more effective NAS at large scale.

Acknowledgments

This research used resources of the Oak Ridge Leadership Computing Facility (ORNL), which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. It also used resources of the IBM T.J. Watson Research Center Scaling Cluster (WSC).

References

Baker, B.; Gupta, O.; Raskar, R.; and Naik, N. 2018. Accelerating Neural Architecture Search using Performance Prediction. In *ICLR Workshops*.

Bender, G.; Kindermans, P.-J.; Zoph, B.; Vasudevan, V.; and Le, Q. 2018. Understanding and Simplifying One-Shot Architecture Search. In *ICML*.

Borsos, Z.; Khorlin, A.; and Gesmundo, A. 2019. Transfer NAS: Knowledge Transfer between Search Spaces with Transformer Agents. *arXiv preprint 1906.08102* .

Brock, A.; Lim, T.; Ritchie, J.; and Weston, N. 2018. SMASH: One-Shot Model Architecture Search through HyperNetworks. In *ICLR*.

Cai, H.; Zhu, L.; and Han, S. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *ICLR*.

Chang, J.; zhang, x.; Guo, Y.; MENG, G.; XIANG, S.; and Pan, C. 2019. DATA: Differentiable ArchiTecture Approximation. In *Advances in Neural Information Processing Systems*.

Chen, Y.; Yang, T.; Zhang, X.; MENG, G.; Xiao, X.; and Sun, J. 2019. DetNAS: Backbone Search for Object Detection. In *Advances in Neural Information Processing Systems*.

Chilimbi, T.; Suzue, Y.; Apacible, J.; and Kalyanaraman, K. 2014. Project Adam: Building an Efficient and Scalable Deep Learning Training System. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 571–582.

Cho, M.; Finkler, U.; Kumar, S.; Kung, D.; Saxena, V.; and Sreedhar, D. 2017. PowerAI DDL. In *arXiv preprint 1708.02188*.

Codreanu, V.; Podareanu, D.; and Saletore, V. 2017. https://communities.surf.nl/artikel/achieving-deep-learning-training-in-less-than-40-minutes-on-imagenet-1k-best-accuracy-and (Last accessed: March 01, 2021).

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

DeVries, T.; and Taylor, G. W. 2017. Improved Regularization of Convolutional Neural Networks with Cutout. *arXiv preprint 1708.04552* .

Dong, X.; and Yang, Y. 2020. NAS-Bench-201: Extending the Scope of Reproducible Neural Architecture Search. In *ICLR*.

Elsken, T.; Metzen, J. H.; and Hutter, F. 2019. Neural Architecture Search: A Survey. *JMLR* 20(55): 1–21.

Finkler, U.; Merler, M.; Panda, R.; Jaiswal, M. S.; Wu, H.; Ramakrishnan, K.; Chen, C.-F.; Cho, M.; Kung, D.; Feris, R.; et al. 2020. Large Scale Neural Architecture Search with Polyharmonic Splines. *arXiv preprint arXiv:2011.10608* .

Ghiasi, G.; Lin, T.; and Le, Q. V. 2019. NAS-FPN: Learning Scalable Feature Pyramid Architecture for Object Detection. In *CVPR*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. *Technical Report, University of Toronto* .

Kyriakides, G.; and Margaritis, K. 2020. The effect of reduced training in neural architecture search. *Neural Computing and Applications* .

Larsson, G.; Maire, M.; and Shakhnarovich, G. 2017. FractalNet: Ultra-Deep Neural Networks without Residuals. In *ICLR*.

Li, L.; and Talwalkar, A. 2019. Random Search and Reproducibility for Neural Architecture Search. In *Conference on Uncertainty in Artificial Intelligence UAI*.

Lian, D.; Zheng, Y.; Xu, Y.; Lu, Y.; Lin, L.; Zhao, P.; Huang, J.; and Gao, S. 2020. Towards Fast Adaptation of Neural Architectures with Meta Learning. In *ICLR*.

Liu, C.; Chen, L.-C.; Schroff, F.; Adam, H.; Hua, W.; Yuille, A.; and Fei-Fei, L. 2019. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *CVPR*.

Liu, C.; Zoph, B.; Neumann, M.; Shlens, J.; Hua, W.; Li, L.-J.; Fei-Fei, L.; Yuille, A.; Huang, J.; and Murphy, K. 2018a. Progressive Neural Architecture Search. In *ECCV*.

Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; and Kavukcuoglu, K. 2018b. Hierarchical Representations for Efficient Architecture Search. In *ICLR*.

Liu, H.; Simonyan, K.; and Yang, Y. 2018. DARTS: Differentiable Architecture Search. In *ICLR*.

Lu, Z.; Sreekumar, G.; Goodman, E.; Banzhaf, W.; Deb, K.; and Boddeti, V. N. 2020. Neural Architecture Transfer. *arXiv preprint 2005.05859* .

Lu, Z.; Whalen, I.; Boddeti, V.; Dhebar, Y.; Deb, K.; Goodman, E.; and Banzhaf, W. 2019. NSGA-Net: Neural Architecture Search Using Multi-Objective Genetic Algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*.

Luo, R.; Tian, F.; Qin, T.; Chen, E.-H.; and Liu, T.-Y. 2018. Neural Architecture Optimization. In *Advances in neural information processing systems*.

Nayman, N.; Noy, A.; Ridnik, T.; Friedman, I.; Jin, R.; and Zelnik-Manor, L. 2019. XNAS: Neural Architecture Search with Expert Advice. In *Advances in Neural Information Processing Systems*.

Nilsback, M.-E.; and Zisserman, A. 2008. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*.

Pham, H.; Guan, M.; Zoph, B.; Le, Q.; and Dean, J. 2018. Efficient Neural Architecture Search via Parameters Sharing. In *ICML*.

Quattoni, A.; and Torralba, A. 2009. Recognizing indoor scenes. In *CVPR*.

Real, E.; Aggarwal, A.; Huang, Y.; and Le, Q. V. 2019. Regularized Evolution for Image Classifier Architecture Search. In *AAAI*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV* 115(3): 211–252.

Tan, M.; Chen, B.; Pang, R.; Vasudevan, V.; Sandler, M.; Howard, A.; and Le, Q. V. 2019. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In *CVPR*.

Wang, N.; Gao, Y.; Chen, H.; Wang, P.; Tian, Z.; and Shen, C. 2019. NAS-FCOS: Fast Neural Architecture Search for Object Detection. *arXiv preprint 1906.04423* .

Wistuba, M. 2019. XferNAS: Transfer Neural Architecture Search. *arXiv preprint 1907.08307* .

Wu, B.; Dai, X.; Zhang, P.; Wang, Y.; Sun, F.; Wu, Y.; Tian, Y.; Vajda, P.; Jia, Y.; and Keutzer, K. 2019. FB-Net: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. In *CVPR*.

Xie, S.; Kirillov, A.; Girshick, R.; and He, K. 2019. Exploring Randomly Wired Neural Networks for Image Recognition. In *ICCV*.

Xie, S.; Zheng, H.; Liu, C.; and Lin, L. 2019. SNAS: stochastic neural architecture search. In *ICLR*.

Yang, A.; Esperança, P. M.; and Carlucci, F. M. 2020. NAS evaluation is frustratingly hard. In *ICLR*.

Ying, C.; Klein, A.; Christiansen, E.; Real, E.; Murphy, K.; and Hutter, F. 2019. NAS-Bench-101: Towards Reproducible Neural Architecture Search. In *ICML*.

Yu, K.; Sciuto, C.; Jaggi, M.; Musat, C.; and Salzmann, M. 2020. Evaluating The Search Phase of Neural Architecture Search. In *ICLR*.

Zela, A.; Elsken, T.; Saikia, T.; Marrakchi, Y.; Brox, T.; and Hutter, F. 2020. Understanding and Robustifying Differentiable Architecture Search. In *ICLR*.

Zela, A.; Siems, J.; and Hutter, F. 2020. NAS-Bench-1Shot1: Benchmarking and Dissecting One-shot Neural Architecture Search. In *ICLR*.

Zhang, H.; Hu, Z.; Wei, J.; Xie, P.; Kim, G.; Ho, Q.; and Xing, E. 2015. Poseidon: A System Architecture for Efficient GPU-based Deep Learning on Multiple Machines. In *arXiv preprint 1512.06216*.

Zhong, Z.; Yan, J.; Wu, W.; Shao, J.; and Liu, C.-L. 2018. Practical Block-Wise Neural Network Architecture Generation. In *CVPR*.

Zhou, D.; Zhou, X.; Zhang, W.; Loy, C. C.; Yi, S.; Zhang, X.; and Ouyang, W. 2020. EcoNAS: Finding Proxies for Economical Neural Architecture Search. In *CVPR*.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *ICLR*.

Zoph, B.; Vasudevan, V.; Shlens, J.; and Le, Q. V. 2018. Learning Transferable Architectures for Scalable Image Recognition. In *CVPR*.