NSGANetV2: Evolutionary Multi-objective Surrogate-Assisted Neural Architecture Search

Zhichao Lu(✉), Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti

Michigan State University, East Lansing, MI 48824, USA
{luzhicha,kdeb,goodman,banzhafw,vishnu}@msu.edu

Abstract. In this paper, we propose an efficient NAS algorithm for generating task-specific models that are competitive under multiple competing objectives. It comprises of two surrogates, one at the architecture level to improve sample efficiency and one at the weights level, through a supernet, to improve gradient descent training efficiency. On standard benchmark datasets (C10, C100, ImageNet), the resulting models, dubbed NSGANetV2, either match or outperform models from existing approaches with the search being orders of magnitude more sample efficient. Furthermore, we demonstrate the effectiveness and versatility of the proposed method on six diverse non-standard datasets, e.g. STL-10, Flowers102, Oxford Pets, FGVC Aircrafts etc. In all cases, NSGANetV2s improve the state-of-the-art (under mobile setting), suggesting that NAS can be a viable alternative to conventional transfer learning approaches in handling diverse scenarios such as small-scale or fine-grained datasets. Code is available at https://github.com/mikelzc1990/nsganetv2.

Keywords: NAS · Evolutionary algorithms · Surrogate-assisted search

1 Introduction

Neural networks have achieved remarkable performance on large scale supervised learning tasks in computer vision. A majority of this progress was achieved by architectures designed manually by skilled practitioners. Neural Architecture Search (NAS) [38] attempts to automate this process to find good architectures for a given dataset. This promise has led to tremendous improvements in convolutional neural network architectures, in terms of predictive performance, computational complexity and model size on standard large-scale image classification benchmarks such as ImageNet [27], CIFAR-10 [15], CIFAR-100 [15] etc. However, the utility of these developments, has so far eluded more widespread and

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-58452-8_3) contains supplementary material, which is available to authorized users.

A. Vedaldi et al. (Eds.): ECCV 2020, LNCS 12346, pp. 35–51, 2020.
https://doi.org/10.1007/978-3-030-58452-8_3

practical applications. These are cases where one wishes to use NAS to obtain high-performance models on custom non-standard datasets, optimizing possibly multiple competing objectives, and to do so without the steep computation burden of existing NAS methods.

The goal of NAS is to obtain both the optimal architecture and its associated optimal weights. The key barrier to realizing the full potential of NAS is the nature of its formulation. NAS is typically treated as a bi-level optimization problem, where an inner optimization loops over the weights of the network for a given architecture, while the outer optimization loops over the network architecture itself. The computational challenge of solving this problem stems from both the upper and lower level optimization. Learning the optimal weights of the network in the lower level necessitates costly iterations of stochastic gradient descent. Similarly, exhaustively searching the optimal architecture is prohibitive due to the discrete nature of the architecture description, size of search space and our desire to optimize multiple, possibly competing, objectives. Mitigating both of these challenges explicitly and simultaneously is the goal of this paper.

Many approaches have been proposed to improve the efficiency of NAS algorithms, both in terms of the upper level and the lower level. A majority of them focuses on the lower level, including weight sharing [1,18,25], proxy models [26,38], coarse training [30], etc. But these approaches still have to sample, explicitly or implicitly, a large number of architectures to evaluate in the upper level. In contrast, there is relatively little focus on improving the sample efficiency of the upper level optimization. A few recent approaches [9,17] adopt surrogates that predict the lower level performance with the goal of navigating the upper level search space efficiently. However, these surrogate predictive models are still very sample inefficient since they are learned in an offline stage by first sampling a large number of architectures that require full lower level optimization.

In this paper, we propose a practically efficient NAS algorithm, by adopting explicit surrogate models simultaneously at both the upper and the lower level. Our lower level surrogate adopts a fine-tuning approach, where the initial weights for fine-tuning are obtained by a supernet model, such as [1,4,5]. Our upper level surrogate adopts an online learning algorithm, that focuses on architectures in the search space that are close to the current trade-off front, as opposed to a random/uniform set of architectures used in the offline surrogate approaches [9, 12,17]. Our online surrogate significantly improves the sample efficiency of the upper level optimization problem in comparison to the offline surrogates. For instance, OnceForAll [5] and PNAS [17] sample 16,000 and 1,160[1] architectures, respectively, to learn the upper level surrogate. In contrast, we only have to sample 350 architectures to obtain a model with similar performance.

An overview of our approach is shown in Fig. 1. We refer to the proposed NAS algorithm as MSuNAS and the resulting architectures as NSGANetV2. Our method is designed to provide a set of high-performance models on a custom dataset (large or small scale, multi-class or fine-grained) while optimizing possibly multiple objectives of interest. Our key contributions are:

[1] Estimate from # of models evaluated by PNAS, actual sample size is not reported.

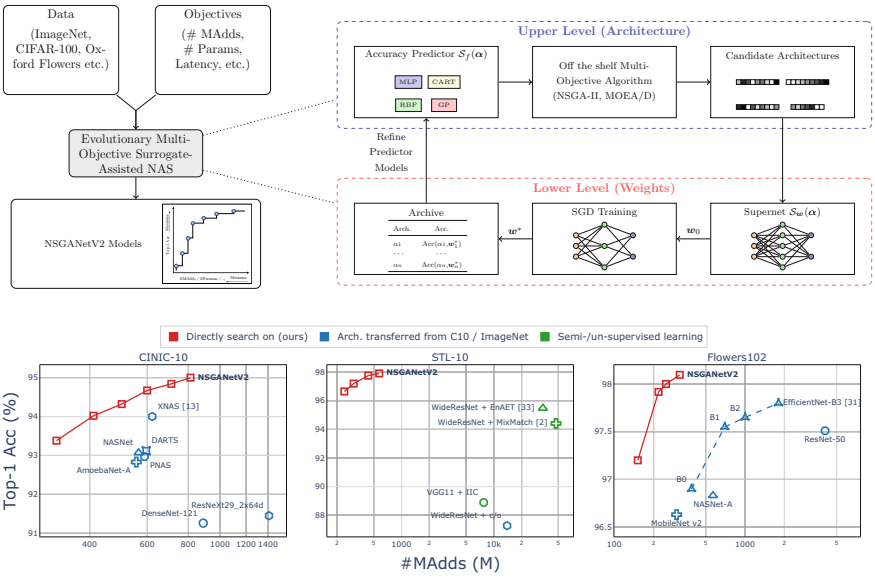


Fig. 1. (Top) **Overview:** Given a dataset and objectives, MSuNAS obtains a task-specific set of models that are competitive in all objectives with high search efficiency. It comprises of two surrogates, one at the upper level to improve sample efficiency and one at the lower level, through a supernet, to improve weight learning efficiency. (Bottom) Performance of the set of task-specific models, i.e. NSGANetV2s, on three different types of non-standard datasets, compared to SOTA from transfer learning [23,31] and semi-/un-supervised learning [2,33]

- An alternative approach to solve the bi-level NAS problem, i.e., simultaneously optimizing the architecture and learn the optimal model weights. However, instead of gradient based relaxations (e.g., DARTS), we advocate for surrogate models. Overall, given a dataset and a set of objectives to optimize, MSuNAS can design custom neural network architectures as efficiently as DARTS but with higher performance and extends to multiple, possibly competing objectives.
- A simple, yet highly effective, online surrogate model for the upper level optimization in NAS, resulting in a significant increase in sampling efficiency over other surrogate-based approaches.
- Scalability and practicality of MSuNAS on many datasets corresponding to different scenarios. These include standard datasets like ImageNet, CIFAR-10 and CIFAR-100, and six non-standard datasets like CINIC-10 [10] (multi-class), STL-10 [8] (small scale mutli-class), Oxford Flowers102 [24] (small scale fine-grained) etc. Under mobile settings ($\leq$ 600M MAdds), MSuNAS leads to SOTA performance.

Table 1. Comparison of existing NAS methods

Methods	Search method	Performance Prediction	Weight sharing	Multiple objective	Dataset searched
NASNet [38]	RL				C10
ENAS [25]	RL		✓		C10
PNAS [17]	SBMO	✓			C10
DPP-Net [12]	SBMO	✓		✓	C10
DARTS [18]	Gradient		✓		C10
LEMONADE [13]	EA		✓	✓	C10, C100
ProxylessNAS [6]	RL+gradient		✓	✓	C10, ImageNet
MnasNet [30]	RL			✓	ImageNet
ChamNet [9]	EA	✓		✓	ImageNet
MobileNetV3 [14]	RL+expert			✓	ImageNet
MSuNAS (ours)	EA	✓	✓	✓	C10, C100, ImageNet, Pets, STL-10, Aircraft, DTD, CINIC-10, Flowers102

2 Related Work

Lower Level Surrogate: Existing approaches [4,18,21,25] primarily focus on mitigating the computational overhead induced by SGD-based weight optimization in the lower level, as this process needs to be repeated for every architecture sampled by a NAS method in the upper level. A common theme among these methods involves training a supernet which contains all searchable architectures as its sub-networks. During search, accuracy using the weights inherited from the supernet becomes the metric to select architectures. However, completely relying on supernet as a substitute of actual weight optimization for evaluating candidate architectures is unreliable. Numerous studies [16,35,36] reported a weak correlation between the performance of the searched architectures (predicted by weight sharing) and the ones trained from scratch (using SGD) during the evaluation phase. MSuNAS instead uses the weights inherited from the supernet only as an initialization to the lower level optimization. Such a fine-tuning process affords the computation benefit of the supernet, while at the same time improving the correlation in the performance of the weights initialized from the supernet and those trained from scratch (Table 1).

Upper Level Surrogate: MetaQNN [1] uses surrogate models to predict the final accuracy of candidate architectures (as a time-series prediction) from the first 25% of the learning curve from SGD training. PNAS [17] uses a surrogate model to predict the top-1 accuracy of architectures with an additional branch added to the cell structure that are repeatedly stacked together. Fundamentally, both of these approaches seek to extrapolate rather than interpolate the

performance of the architecture using the surrogates. Consequently, as we show later in the paper, the rank-order between the predicted accuracy and the true accuracy is very low[2] (0.476). OnceForAll [5] also uses a surrogate model to predict accuracy from architecture encoding. However, the surrogate model is trained offline for the entire search space, thereby needing a large number of samples for learning (16K samples -> 2 GPU-days -> 2x search cost of DARTS for just constructing the surrogate model). Instead of using uniformly sampled architectures and their validation accuracy to train the surrogate model to approximate the entire landscape, ChamNet [9] trains many architectures through full lower level optimization and selects only 300 samples with high accuracy with diverse efficiency (FLOPs, Latency, Energy) to train a surrogate model offline. In contrast, MSuNAS learns a surrogate model in an online fashion only on the samples that are close to the current trade-off front as we explore the search space. The online learning approach significantly improves the sample efficiency of our search, since we only need lower level optimization (full or surrogate assisted) for the samples near the current Pareto front.

Multi-objective NAS: Approaches that consider more than one objective to optimize the architecture can be categorized into two groups: (i) scalarization, and (ii) population based approaches. The former include, ProxylessNAS [6], MnasNet [30], FBNet [34], and MobileNetV3 [14] which use a scalarized objective that encourages high accuracy and penalizes compute inefficiency at the same time, e.g., maximize $Acc * (Latency/Target)^{-0.07}$. These methods require a pre-defined preference weighting of the importance of different objectives before the search, which typically requires a numbers of trials. Methods in the latter category include [7,12,13,19,20] and aim to approximate the entire Pareto-efficient frontier simultaneously. These approaches rely on heuristics (e.g., EA) to efficiently navigate the search space, which allows practitioners to visualize the trade-off between the objectives and to choose a suitable network *a posteriori* to the search. MSuNAS falls in the latter category using surrogate models to mitigate the computational overhead.

3 Proposed Approach

The neural architecture search problem for a target dataset $\mathcal{D} = \{\mathcal{D}_{trn}, \mathcal{D}_{vld}, \mathcal{D}_{tst}\}$ can be formulated as the following bilevel optimization problem [3],

$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\boldsymbol{\alpha}) = \big(f_1(\boldsymbol{\alpha}; \boldsymbol{w}^*(\boldsymbol{\alpha})), \ldots, f_k(\boldsymbol{\alpha}; \boldsymbol{w}^*(\boldsymbol{\alpha})), f_{k+1}(\boldsymbol{\alpha}), \ldots, f_m(\boldsymbol{\alpha})\big)^T, \\ \text{subject to} \quad & \boldsymbol{w}^*(\boldsymbol{\alpha}) \in \text{argmin} \ \mathcal{L}(\boldsymbol{w}; \boldsymbol{\alpha}), \\ & \boldsymbol{\alpha} \in \boldsymbol{\Omega}_\alpha, \quad \boldsymbol{w} \in \boldsymbol{\Omega}_w, \end{aligned} \tag{1}$$

[2] In the supplementary material we show that better rank-order correlation at the search stage ultimately leads to finding better performing architectures.

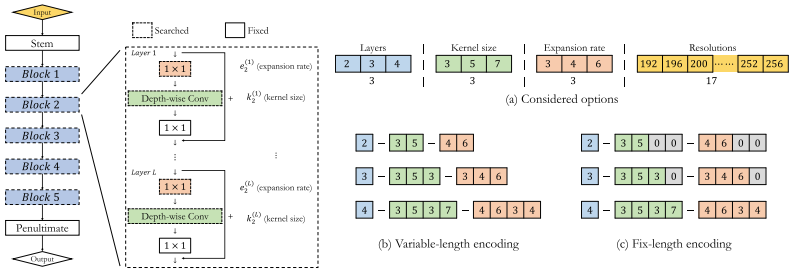


Fig. 2. Search Space: A candidate architecture comprises five computational blocks. Parameters we search for include image resolution, number of layers (L) in each block and the expansion rate (e) and the kernel size (k) in each layer.

where the upper level variable $\boldsymbol{\alpha}$ defines a candidate CNN architecture, and the lower level variable $\boldsymbol{w}(\boldsymbol{\alpha})$ defines the associated weights. $\mathcal{L}(\boldsymbol{w}; \boldsymbol{\alpha})$ denotes the cross-entropy loss on the training data $\mathcal{D}_{trn}$ for a given architecture $\boldsymbol{\alpha}$. $\mathbf{F} : \boldsymbol{\Omega} \rightarrow \mathbb{R}^m$ constitutes m desired objectives. These objectives can be further divided into two groups, where the first group (f_1 to f_k) consists of objectives that depend on both the architecture and the weights—e.g., predictive performance on validation data $\mathcal{D}_{vld}$, robustness to adversarial attack, etc. The other group (f_{k+1} to f_m) consists of objectives that only depend on the architecture—e.g., number of parameters, floating point operations, latency etc.

3.1 Search Space

MSuNAS searches over four important dimensions of convolutional neural networks (CNNs), including depth (# of layers), width (# of channels), kernel size and input resolution. Following previous works [5,14,30], we decompose a CNN architecture into five sequentially connected blocks, with gradually reduced feature map size and increased number of channels. In each block, we search over the number of layers, where only the first layer uses stride 2 if the feature map size decreases, and we allow each block to have minimum of two and maximum of four layers. Every layer adopts the inverted bottleneck structure [28] and we search over the expansion rate in the first 1×1 convolution and the kernel size of the depth-wise separable convolution. Additionally, we allow the input image size to range from 192 to 256. We use an integer string to encode these architectural choices, and we pad zeros to the strings of architectures that have fewer layers so that we have a fixed-length encoding. A pictorial overview of this search space and encoding is shown in Fig. 2.

3.2 Overall Algorithm Description

The problem in Eq. 1 poses two main computational bottlenecks for conventional bi-level optimization methods. First, the lower level problem of learning the optimal weights $\boldsymbol{w}^*(\boldsymbol{\alpha})$ for a given architecture $\boldsymbol{\alpha}$ involves a prolonged training

Algorithm 1: MSuNAS

Input : $\mathcal{SS}$ (search space),
$\mathcal{S}_w$ (supernet),
$\mathcal{C}$ (complexity obj),
N (initial samples),
K (max. iterations).

1 $\mathcal{A} \leftarrow \emptyset$;
2 **while** $i < N$ **do**
3 $\quad \alpha \leftarrow$ sample($\mathcal{SS}$)
4 $\quad w_o \leftarrow \mathcal{S}_w(\alpha)$
5 $\quad acc \leftarrow \text{SGD}(\alpha, w_o)$
6 $\quad \mathcal{A} \leftarrow \mathcal{A} \cup (\alpha, acc)$
7 **end**
8 **while** $j < K$ **do**
9 $\quad \mathcal{S}_f \leftarrow$ construct from $\mathcal{A}$ // (MLP / CART / RBF / GP)
10 $\quad \tilde{\boldsymbol{\alpha}} \leftarrow \text{NSGA-II}(\mathcal{S}_f, \mathcal{C})$
11 $\quad \boldsymbol{\alpha} \leftarrow$ subset from $\tilde{\boldsymbol{\alpha}}$
12 $\quad$ **for** α *in* $\boldsymbol{\alpha}$ **do**
13 $\quad\quad w_o \leftarrow \mathcal{S}_w(\alpha)$
14 $\quad\quad acc \leftarrow \text{SGD}(\alpha, w_o)$
15 $\quad\quad \mathcal{A} \leftarrow \mathcal{A} \cup (\alpha, acc)$
16 $\quad$ **end**
17 **end**
18 **Return** NDsort($\mathcal{A}$).

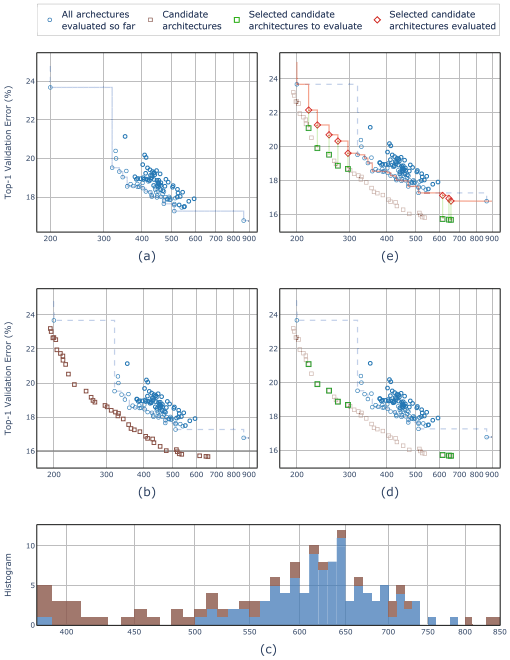


Fig. 3. A sample run of MSuNAS on ImageNet: In each iteration, accuracy-prediction surrogate models $\mathcal{S}_f$ are constructed from an archive of previously evaluated architectures (a). New candidate architectures (brown boxes in (b)) are obtained by solving the auxiliary single-level multi-objective problem $\tilde{F} = \{\mathcal{S}_f, \mathcal{C}\}$ (line 10 in Algorithm 1). A subset of the candidate architectures is chosen to diversify the Pareto front (c)–(d). The selected candidate architectures are then evaluated and added to the archive (e). At the conclusion of search, we report the non-dominated architectures from the archive. The x-axis in all sub-figures is #MAdds. (Color figure online)

process—e.g., one complete SGD training on ImageNet dataset takes two days on an 8-GPU server. Second, even though there exist techniques like weight-sharing to bypass the gradient-descent-based weight learning process, extensively sampling architectures at the upper level can still render the overall process computationally prohibitive, e.g., 10,000 evaluations on ImageNet take 24 GPU hours, and for methods like NASNet, AmoebaNet that require more than 20,000 samples, it still requires days to complete the search even with weight-sharing.

Algorithm 1 and Fig. 3 show the pseudocode and corresponding steps from a sample run of MSuNAS on ImageNet, respectively. To overcome the aforementioned bottlenecks, we use surrogate models at both upper and lower levels to make our NAS algorithm practically useful for a variety of datasets and objectives. At the upper level, we construct a surrogate model that predicts the top-1 accuracy from integer strings that encode architectures. Previous approaches [5,9,29] that also used surrogate-modeling of the accuracy follow an offline

approach, where the accuracy predictor is built from samples collected separately prior to the architecture search and not refined during the search. We argue that such a process makes the search outcome highly dependent on the initial training samples. As an alternative, we propose to model and refine the accuracy predictor iteratively in an online manner during the search. In particular, we start with an accuracy predictor constructed from only a limited number of architectures sampled randomly from the search space. We then use a standard multi-objective algorithm (NSGA-II [11], in our case) to search using the constructed accuracy predictor along with other objectives that are also of interest to the user. We then evaluate the outcome architectures from NSGA-II and refine the accuracy predictor model with these architectures as new training samples. We repeat this process for a pre-specified number of iterations and output the non-dominated solutions from the pool of evaluated architectures.

3.3 Speeding Up Upper Level Optimization

Recall that the nested nature of the bi-level problem makes the upper level optimization computationally very expensive, as every upper level function evaluation requires another optimization at the lower level. Hence, to improve the efficiency of our approach at the upper level, we focus on reducing the number of architectures that we send to the lower level for learning optimal weights. To achieve this goal, we need a surrogate model to predict the accuracy of an architecture before we actually train it. There are two desired properties of such a predictor: (1) high rank-order correlation between the predicted and the true performance; and (2) sample efficient such that the required number of architectures to be trained through SGD are minimized for constructing the predictor.

We first collected four different surrogate models for accuracy prediction from the literature, namely, Multi Layer Perceptron (MLP) [17], Classification And Regression Trees (CART) [29], Radial Basis Function (RBF) [1] and Gaussian Process (GP) [9]. From our ablation study, we observed that no one surrogate model is consistently better than others in terms of the above two criteria on all datasets (see Sect. 4.1). Hence, we propose a selection mechanism, dubbed Adaptive Switching (AS), which constructs all four types of surrogate models at every iteration and adaptively selects the best model via cross-validation.

With the accuracy predictor selected by AS, we apply the NSGA-II algorithm to simultaneously optimize for both accuracy (predicted) and other objectives of interest to the user (line 10 in Algorithm 1). For the purpose of illustration, we assume that the user is interested in optimizing #MAdds as the second objective. At the conclusion of the NSGA-II search, a set of non-dominated architectures is output, see Fig. 3(b). Often times, we cannot afford to train all architectures in the set. To select a subset, we first select the architecture with highest predicted accuracy. Then we project all other architecture candidates to the #MAdds axis, and pick the remaining architectures from the sparse regions that help in extending the Pareto frontier to diverse #MAdds regimes, see Fig. 3(c)–(d). The architectures from the chosen subset are then sent to the lower level for SGD

training. We finally add these architectures to the training samples to refine our accuracy predictor models and proceed to next iteration, see Fig. 3(e).

3.4 Speeding Up Lower Level Optimization

To further improve the search efficiency of the proposed algorithm, we adopt the widely-used weight-sharing technique [4,21,22]. First, we need a supernet such that all searchable architectures are sub-networks of it. We construct such a supernet by taking the searched architectural hyperparameters at their maximum values, i.e., with four layers in each of the five blocks, with expansion ratio set to 6 and kernel size set to 7 in each layer (See Fig. 2). Then we follow the progressive shrinking algorithm [5] to train the supernet. This process is executed once before the architecture search. The weights inherited from the trained supernet are used as a warm-start for the gradient descent algorithm during architecture search.

4 Experiments and Results

In this section, we evaluate the surrogate predictor, the search efficiency and the obtained architectures on CIFAR-10 [15], CIFAR-100 [15], and ImageNet [27].

4.1 Performance of the Surrogate Predictors

To evaluate the effectiveness of the considered surrogate models, we uniformly sample 2,000 architectures from our search space, and train them using SGD for 150 epochs on each of the three datasets and record their accuracy on 5,000 held-out images from the training set. We then fit surrogate models with different number of samples randomly selected from the 2,000 collected. We repeat the process for 10 trials to compare the mean and standard deviation of the rank-order correlation between the predicted and true accuracy, see Fig. 4. In general, we observe that no single surrogate model consistently outperforms the others on all three datasets. Hence, at every iteration, we adopt an Adaptive Switching (AS) routine that compares the four surrogate models and chooses the best based on 10-fold cross-validation. It is evident from Fig. 4 that AS works better than any one of the four surrogate models alone on all three datasets. The construction time of the AS is negligible (relatively to the search cost).

4.2 Search Efficiency

In this section, we first compare the search efficiency of MSuNAS to other single-objective methods on both CIFAR-10 and ImageNet. To quantify the speedup, we compare the two governing factors, namely, the total number of architectures evaluated by each method to reach the reported accuracy and the number of epochs undertaken to train each sampled architecture during search. The results are provided in Table 2. We observe that MSuNAS is **20x faster** than methods

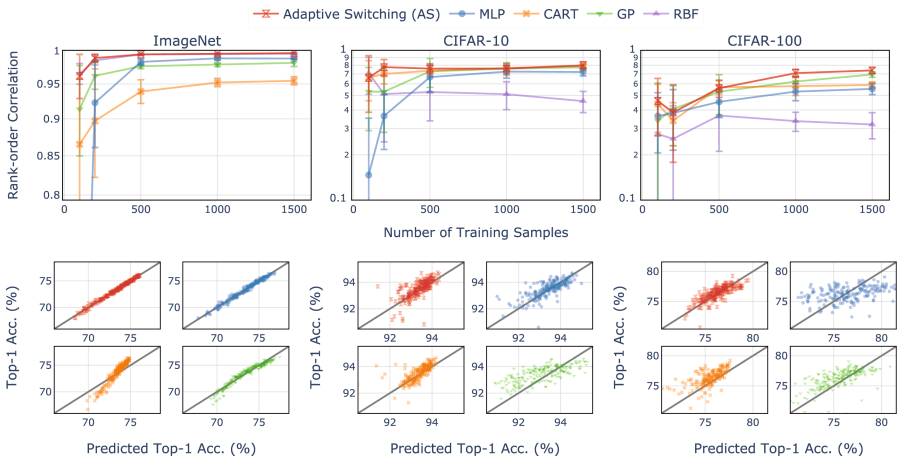


Fig. 4. Comparing the relative prediction performance of the proposed Adaptive Switching (AS) method to the existing four surrogate models. Top row compares Spearman rank-order correlation coefficient as number of training samples increases. Bottom row visualizes the true vs. predicted accuracy under 500 training samples (RBF method is omitted to conserve space).

that use RL or EA. When compared to PNAS [17], which also utilizes an accuracy predictor, MSuNAS is still at least **3x faster**.

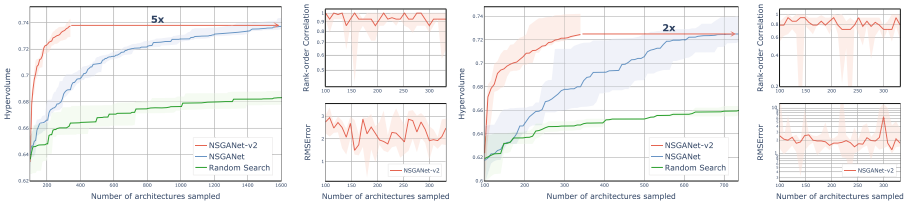
We then compare the search efficiency of MSuNAS to NSGANet [20] and random search under a bi-objective setup: Top-1 accuracy and #MAdds. To perform the comparison, we run MSuNAS for 30 iterations, leading to 350 architectures evaluated in total. We record the cumulative hypervolume [37] achieved against the number of architectures evaluated. We repeat this process five times on both ImageNet and CIFAR-10 datasets to capture the variance in performance due to randomness in the search initialization. For a fair comparison to NSGANet, we apply the search code to our search space and record the number of architectures evaluated by NSGANet to reach a similar hypervolume than that achieved by MSuNAS. The random search baseline is performed by uniformly sampling from our search space. We plot the mean and the standard deviation of the hypervolume values achieved by each method in Fig. 5. Based on the incremental rate of hypervolume metric, we observe that MSuNAS is **2–5x faster**, on average, in achieving a better Pareto frontier in terms of number of architectures evaluated.

4.3 Results on Standard Datasets

Prior to the search, we train the supernet following the training hyperparameters setting from [5]. For each dataset, we start MSuNAS with 100 randomly sampled architectures and run for 30 iterations. In each iteration, we evaluate 8 architectures selected from the candidates recommended by NSGA-II according to the accuracy predictor. For searching on CIFAR-10 and CIFAR-100, we fine

Table 2. Comparing the relative search efficiency of MSuNAS to other single-objective methods: "#Model" is the total number of architectures evaluated during search, "#Epochs" is the number of epochs used to train each architecture during search. † and ‡ denote training epochs with and without a supernet to warm-start the weights, respectively.

	Method	Type	Top1 Acc	#MAdds	#Model	Speedup	#Epochs	Speedup
CIFAR-10	NASNet-A [38]	RL	97.4%	569M	20,000	57x	20	up to 4x
	AmoebaNet-B [26]	EA	97.5%	555M	27,000	77x	25	up to 5x
	PNASNet-5 [17]	SMBO	96.6%	588M	1,160	3.3x	20	up to 4x
	MSuNAS (ours)	EA	98.4%	468M	350	1x	$5^{\dagger}/20^{\ddagger}$	1x
ImageNet	MnasNet-A [30]	RL	75.2%	312M	8,000	23x	5	up to 5x
	OnceForAll [5]	EA	76.0%	230M	16,000	46x	0	-
	MSuNAS (ours)	EA	75.9%	225M	350	1x	$0^{\dagger}/5^{\ddagger}$	1x



(a) ImageNet (b) CIFAR-10

Fig. 5. Comparing the relative search efficiency of MSuNAS to other methods under bi-objective setup on ImageNet (a) and CIFAR-10 (b). The left plots in each subfigure compares the hypervolume metric [37], where a larger value indicates a better Pareto front achieved. The right plots in each subfigure show the Spearman rank-order correlation (top) and the root mean square error (bottom) of MSuNAS. All results are averaged over five runs with standard deviation shown in shaded regions.

tune the weights inherited from the supernet for five epochs then evaluate on 5K held-out validation images from the original training set. For searching on ImageNet, we re-calibrate the running statistics of the BN layers after inheriting the weights from the supernet, and evaluate on 10K held-out validation images from the original training set. At the conclusion of the search, we pick the four architectures from the achieved Pareto front, and further fine-tune for additional 150–300 epochs on the entire training sets. For reference purpose, we name the obtained architectures as NSGANetV2-s/m/l/xl in ascending #MAdds order. Architectural details can be found in the supplementary materials.

Table 3 shows the performance of our models on the ImageNet 2012 benchmark [27]. We compare models in terms of predictive performance on the validation set, model efficiency (measured by #MAdds and latencies on different hardware), and associated search cost. Overall, NSGANetV2 consistently either matches or outperforms other models across different accuracy levels with highly competitive search costs. In particular, NSGANetV2-s is **2.2% more accurate** than MobileNetV3 [14] while being equivalent in #MAdds and latencies;

Table 3. ImageNet Classification [27]: comparing NSGANetV2 with manual and automated design of efficient networks. Models are grouped into sections for better visualization. Our results are underlined and best result in each section is in bold. CPU latency (batchsize = 1) is measured on Intel i7-8700K and GPU latency (batchsize = 64) is measured on 1080Ti. † The search cost excludes the supernet training cost. ‡ Estimated based on the claim that PNAS is 8x faster than NASNet from [17].

Model	Type	Search Cost (GPU days)	#Params	#MAdds	CPU Lat. (ms)	GPU Lat. (ms)	Top-1 Acc. (%)	Top-5 Acc. (%)
NSGANetV2-s	auto	1†	6.1M	225M	9.1	30	**77.4**	**93.5**
MobileNetV2 [28]	manual	0	3.4M	300M	**8.3**	**23**	72.0	91.0
FBNet-C [34]	auto	9	5.5M	375M	9.1	31	74.9	-
ProxylessNAS [6]	auto	8.3	7.1M	465M	8.5	27	75.1	92.5
MobileNetV3 [14]	combined	-	5.4M	219M	10.0	33	75.2	-
OnceForAll [5]	auto	2†	6.1M	230M	9.5	31	76.9	-
NSGANetV2-m	auto	1†	7.7M	312M	**11.4**	**37**	**78.3**	**94.1**
EfficientNet-B0 [31]	auto	-	5.3M	390M	14.4	46	76.3	93.2
MixNet-M [32]	auto	-	5.0M	360M	24.3	79	77.0	93.3
AtomNAS-C+ [22]	auto	1†	5.5M	329M	-	-	77.2	93.5
NSGANetV2-l	auto	1†	8.0M	400M	**12.9**	**52**	**79.1**	**94.5**
PNASNet-5 [17]	auto	250‡	5.1M	588M	35.6	82	74.2	91.9
NSGANetV2-xl	auto	1†	8.7M	593M	**16.7**	**73**	**80.4**	**95.2**
EfficientNet-B1 [31]	auto	-	7.8M	700M	21.5	78	78.8	94.4
MixNet-L [32]	auto	-	7.3M	565M	29.4	105	78.9	94.2

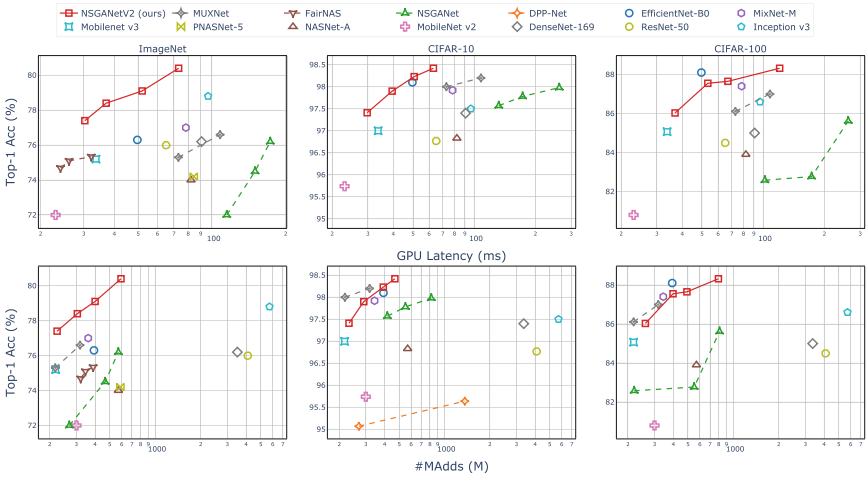


Fig. 6. Accuracy vs **Efficiency**: Top row compares predictive accuracy vs. GPU latency on a batch of 64 images. Bottom row compares predictive accuracy vs. number of multi-adds in millions. Models from multi-objective approaches are joined with lines. Our models are obtained by directly searching on the respective datasets. In most problems, MSuNAS finds more accurate solutions with fewer parameters.

NSGANetV2-xl achieves **80.4% Top-1 accuracy** under 600M MAdds, which is **1.5% more accurate** and **1.2x more efficient** than EfficientNet-B1 [31]. Additional comparisons to models from multi-objective approaches are provided in Fig. 6.

For CIFAR datasets, Fig. 6 compares our models with other approaches in terms of both predictive performance and computational efficiency. On CIFAR-10, we observe that NSGANetV2 dominates all previous models including (1) NASNet-A [38], PNASNet-5 [17] and NSGANet [20] that search on CIFAR-10 directly, and (2) EfficientNet [31], MobileNetV3 [14] and MixNet [32] that fine-tune from ImageNet.

5 Scalability of MSuNAS

5.1 Types of Datasets

Existing NAS approaches are rarely evaluated for their search ability beyond standard benchmark datasets, i.e., ImageNet, CIFAR-10, and CIFAR-100. Instead, they follow a conventional transfer learning setup, in which the architectures found by searching on standard benchmark datasets are transferred, with weights fine-tuned, to new datasets. We argue that such a process is conceptually contradictory to the goal of NAS, and the architectures identified under such a process are sub-optimal. In this section we demonstrate the scalability of MSuNAS to six[3] additional datasets with various forms of difficulties, in terms of diversity in classification classes (multi-classes vs. fine-grained) and size of training set (see Table 4). We adopt the settings of the CIFAR datasets as outlined in Sect. 3. For each dataset, one search takes less than one day on 8 GPU cards.

Table 4. Non-standard Datasets for MSuNAS

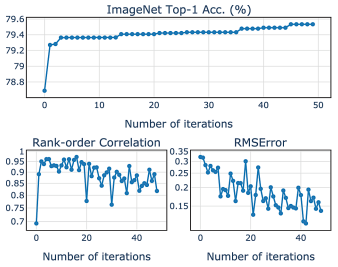
Datasets	Type	#Classes	#Train	#Test
CINIC-10 [10]	Multi-class	10	90,000	90,000
STL-10 [8]	Multi-class	10	5,000	8,000
Flowers102 [24]	Fine-grained	102	2,040	6,149

Figure 1 (Bottom) compares the performance of NSGANetV2 obtained by searching directly on the respective datasets to models from other approaches that transfer architectures learned from either CIFAR-10 or ImageNet. Overall, we observe that NSGANetV2 significantly outperforms other models on all three datasets. In particular, NSGANetV2 achieves a better performance than the currently known state-of-the-art on CINIC-10 [23] and STL-10 [2]. Furthermore, on Oxford Flowers102, NSGANetV2 achieves better accuracy to that of EfficientNet-B3 [31] while using **1.4B fewer** MAdds.

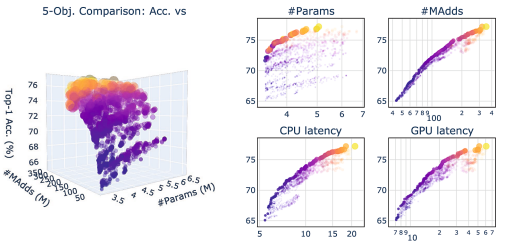
5.2 Number of Objectives

Single-Objective Formulation: Adding a hardware efficiency target as a penalty term to the objective of maximizing predictive performance is a common workaround to handle multiple objectives in the NAS literature [6,30,34].

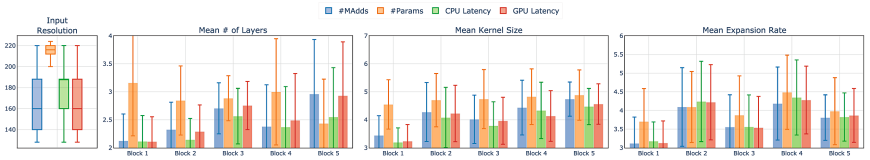
[3] Due to space constraints, we report results from three datasets in the main paper and three more in the supplementary material.



(a) Maximize Top-1 Acc.



(b) 5-Objective scenario.



(c) Non-dominated architectures under different efficiency objectives.

Fig. 7. Scalability of MSuNAS to different numbers and types of objectives: optimizing (a) a scalarized single-objective on ImageNet; (b) five objectives including accuracy, Params, MAdds, CPU and GPU latency, simultaneously. (c) Post-optimal analysis on the architectures that are non-dominated according to different efficiency objectives.

We demonstrate that our proposed algorithm can also effectively handle such a scalarized single-objective search. Following the scalarization method in [30], we apply MSuNAS to maximize validation accuracy on ImageNet with 600M MAdds as the targeted efficiency. The accumulative top-1 accuracy achieved and the performance of the accuracy predictor are provided in Fig. 7a. Without further fine-tuning, the obtained architecture yields 79.56% accuracy with 596M MAdds on the ImageNet validation set, which is more accurate and **100M fewer** MAdds than EfficientNet-B1 [31].

Many-Objective Formulation: Practical deployment of learned models are rarely driven by a single objective, and most often, seek to trade-off many different, possibly competing, objectives. As an example of one such scenario, we use MSuNAS to simultaneously optimize five objectives—namely, the accuracy on ImageNet, #Params, #MAdds, CPU and GPU latency. We follow the same search setup as in the main experiments and increase the budget to ensure a thorough search on the expanded objective space. We show the obtained Pareto-optimal (to five objectives) architectures in Fig. 7b. We use color and marker size to indicate CPU and GPU latency, respectively. We observe that a Pareto surface emerges, shown in the left 3D scatter plot, suggesting that trade-offs exist between objectives, i.e., #Params and #MAdds are not fully correlated. We then project all architectures to 2D, visualizing accuracy vs. each one of the four considered efficiency measurements, and highlight the architectures that are non-dominated in the corresponding two-objective cases. We observe that

many architectures that are non-dominated in the five-objective case are now dominated when only considering two objectives. Empirically, we observe that accuracy is highly correlated with #MAdds, CPU and GPU latency, but not with #Params, to some extent.

6 Conclusion

This paper introduced MSuNAS, an efficient neural architecture search algorithm for rapidly designing task-specific models under multiple competing objectives. The efficiency of our approach stems from (i) online surrogate-modeling at the level of the architecture to improve the sample efficiency of search, and (ii) a supernet based surrogate-model to improve the weights learning efficiency via fine-tuning. On standard datasets (CIFAR-10, CIFAR-100 and ImageNet), NSGANetV2 matches the state-of-the-art with a search cost of one day. The utility and versatility of MSuNAS are further demonstrated on non-standard datasets of various types of difficulties and on different number of objectives. Improvements beyond the state-on-the-art on STL-10 and Flowers102 (under mobile setting) suggest that NAS is a more effective alternative to conventional transfer learning approaches.

References

1. Baker, B., Gupta, O., Raskar, R., Naik, N.: Accelerating neural architecture search using performance prediction. arXiv preprint arXiv:1705.10823 (2017)
2. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.A.: Mixmatch: a holistic approach to semi-supervised learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
3. Bracken, J., McGill, J.T.: Mathematical programs with optimization problems in the constraints. Oper. Res. **21**(1), 37–44 (1973). http://www.jstor.org/stable/169087
4. Brock, A., Lim, T., Ritchie, J., Weston, N.: SMASH: one-shot model architecture search through hypernetworks. In: International Conference on Learning Representations (ICLR) (2018)
5. Cai, H., Gan, C., Wang, T., Zhang, Z., Han, S.: Once for all: train one network and specialize it for efficient deployment. In: International Conference on Learning Representations (ICLR) (2020)
6. Cai, H., Zhu, L., Han, S.: ProxylessNAS: direct neural architecture search on target task and hardware. In: International Conference on Learning Representations (ICLR) (2019)
7. Chu, X., Zhang, B., Xu, R., Li, J.: FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search. arXiv preprint arXiv:1907.01845 (2019)
8. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (2011)
9. Dai, X., et al.: ChamNet: towards efficient network design through platform-aware model adaptation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)

10. Darlow, L.N., Crowley, E.J., Antoniou, A., Storkey, A.J.: CINIC-10 is not ImageNet or CIFAR-10. arXiv preprint arXiv:1810.03505 (2018)
11. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
12. Dong, J.-D., Cheng, A.-C., Juan, D.-C., Wei, W., Sun, M.: DPP-Net: device-aware progressive search for pareto-optimal neural architectures. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11215, pp. 540–555. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01252-6_32
13. Elsken, T., Metzen, J.H., Hutter, F.: Efficient multi-objective neural architecture search via Lamarckian evolution. In: International Conference on Learning Representations (ICLR) (2019)
14. Howard, A., et al.: Searching for MobileNetV3. In: International Conference on Computer Vision (ICCV) (2019)
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical report. Citeseer (2009)
16. Li, L., Talwalkar, A.: Random search and reproducibility for neural architecture search. arXiv preprint arXiv:1902.07638 (2019)
17. Liu, C.: Progressive neural architecture search. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11205, pp. 19–35. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01246-5_2
18. Liu, H., Simonyan, K., Yang, Y.: DARTS: differentiable architecture search. In: International Conference on Learning Representations (ICLR) (2019)
19. Lu, Z., Deb, K., Boddeti, V.N.: MUXConv: information multiplexing in convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020)
20. Lu, Z., et al.: NSGA-Net: neural architecture search using multi-objective genetic algorithm. In: Genetic and Evolutionary Computation Conference (GECCO) (2019)
21. Luo, R., Tian, F., Qin, T., Chen, E., Liu, T.Y.: Neural architecture optimization. In: Advances in Neural Information Processing Systems (NeurIPS) (2018)
22. Mei, J., et al.: AtomNAS: fine-grained end-to-end neural architecture search. In: International Conference on Learning Representations (ICLR) (2020)
23. Nayman, N., Noy, A., Ridnik, T., Friedman, I., Jin, R., Zelnik, L.: XNAS: neural architecture search with expert advice. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
24. Nilsback, M., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 6th Indian Conference on Computer Vision, Graphics Image Processing (2008)
25. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameters sharing. In: International Conference on Machine Learning (ICML) (2018)
26. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: AAAI Conference on Artificial Intelligence Conference on Artificial Intelligence (2019)
27. Russakovsky, O., et al.: ImageNet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
28. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: inverted residuals and linear bottlenecks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)

29. Sun, Y., Wang, H., Xue, B., Jin, Y., Yen, G.G., Zhang, M.: Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. IEEE Trans. Evol. Comput. (2019). https://doi.org/10.1109/TEVC.2019.2924461
30. Tan, M., et al.: MnasNet: platform-aware neural architecture search for mobile. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
31. Tan, M., Le, Q.V.: EfficientNet: rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning (ICML) (2019)
32. Tan, M., Le, Q.V.: MixConv: mixed depthwise convolutional kernels. In: British Machine Vision Conference (BMVC) (2019)
33. Wang, X., Kihara, D., Luo, J., Qi, G.J.: EnAET: Self-trained ensemble autoencoding transformations for semi-supervised learning. arXiv preprint arXiv:1911.09265 (2019)
34. Wu, B., et al.: FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
35. Xie, S., Kirillov, A., Girshick, R., He, K.: Exploring randomly wired neural networks for image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
36. Yu, K., Sciuto, C., Jaggi, M., Musat, C., Salzmann, M.: Evaluating the search phase of neural architecture search. In: International Conference on Learning Representations (ICLR) (2020)
37. Zitzler, E., Thiele, L.: Multiobjective optimization using evolutionary algorithms - a comparative case study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.P. (eds.) PPSN 1998. LNCS, vol. 1498. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0056872
38. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2018)