Evolutionary Neural Cascade Search across Supernetworks

Alexander Chebykin
Centrum Wiskunde & Informatica
Amsterdam, The Netherlands
a.chebykin@cwi.nl

Tanja Alderliesten
Leiden University Medical Center
Leiden, the Netherlands
t.alderliesten@lumc.nl

Peter A. N. Bosman
Centrum Wiskunde & Informatica
Amsterdam, the Netherlands
TU Delft
Delft, The Netherlands
peter.bosman@cwi.nl

ABSTRACT

To achieve excellent performance with modern neural networks, having the right network architecture is important. Neural Architecture Search (NAS) concerns the automatic discovery of task-specific network architectures. Modern NAS approaches leverage supernetworks whose subnetworks encode candidate neural network architectures. These subnetworks can be trained simultaneously, removing the need to train each network from scratch, thereby increasing the efficiency of NAS.

A recent method called Neural Architecture Transfer (NAT) further improves the efficiency of NAS for computer vision tasks by using a multi-objective evolutionary algorithm to find high-quality subnetworks of a supernetwork pretrained on ImageNet. Building upon NAT, we introduce ENCAS — Evolutionary Neural Cascade Search. ENCAS can be used to search over multiple pretrained supernetworks to achieve a trade-off front of cascades of different neural network architectures, maximizing accuracy while minimizing FLOPs count.

We test ENCAS on common computer vision benchmarks (CIFAR-10, CIFAR-100, ImageNet) and achieve Pareto dominance over previous state-of-the-art NAS models up to 1.5 GFLOPs. Additionally, applying ENCAS to a pool of 518 publicly available ImageNet classifiers leads to Pareto dominance in all computation regimes and to increasing the maximum accuracy from 88.6% to 89.0%, accompanied by an 18% decrease in computation effort from 362 to 296 GFLOPs.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Genetic algorithms**; **Ensemble methods**.

KEYWORDS

Neural Architecture Search, Deep Learning, Computer Vision, AutoML, Evolutionary Computation

ACM Reference Format:
Alexander Chebykin, Tanja Alderliesten, and Peter A. N. Bosman. 2022. Evolutionary Neural Cascade Search across Supernetworks. In *Genetic and Evolutionary Computation Conference (GECCO '22), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3512290.3528749

GECCO '22, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9237-2/22/07.
https://doi.org/10.1145/3512290.3528749

1 INTRODUTION

In recent years, deep neural networks have been successfully applied in domains ranging from text summarization [7] to medical image segmentation [20]. Much of this success has been enabled by task-specific neural network architectures that are designed manually while making use of expert knowledge. The research direction of Neural Architecture Search (NAS) [58] has the goal of making architecture design automatic and data-driven. Tremendous progress has been made since the first approaches: performance of the found models improved [25, 39], their size decreased [15, 32] (smaller models usually work faster and require less storage space), and the search process itself became much more efficient (with required GPU-hours decreasing from tens of thousands [33] to single-digit numbers [42]).

These search efficiency gains can mostly be attributed to the idea of weight sharing via a supernetwork [32] (with performance prediction [3, 50] also playing a role). Instead of training the weights of each candidate architecture from scratch, a supernetwork is constructed such that each architecture in the search space is a subset of the supernetwork (see Fig. 1). To evaluate the quality of an architecture, the weights of the relevant part of the supernetwork are copied. With various architectures potentially sharing the same operations (e.g. convolution [24], attention [2]), the amount of training needed decreases drastically.

However, by requiring that each architecture is a path within a supernetwork, the supernetwork approach inherently limits the diversity of architectures that can be produced. Thus, the manual choice of which supernetwork to use for the automated NAS procedure plays a large role, as it restricts the search space before the search algorithm even starts. With the growing number of available supernetworks, the issue of choosing the supernetwork is becoming increasingly important, and yet, to the best of our knowledge, there exists no method taking it into account.

There are many ways to improve neural network performance that are different from NAS. Ensembling [36] is one such technique that involves passing the same input through several different models and combining their predictions to get a better final prediction. It has been shown to work well if the models' mistakes are independent [18], which is helped by the models being different from each other [51]. NAS has been used [11, 56] to produce models that together make a good ensemble. Modern supernetwork-based approaches seem very fitting to this purpose because they do not incur additional training costs for ensembles of arbitrarily large size (once a supernetwork is trained, weights for trained subnetworks can be extracted from it and used without additional retraining[1]).

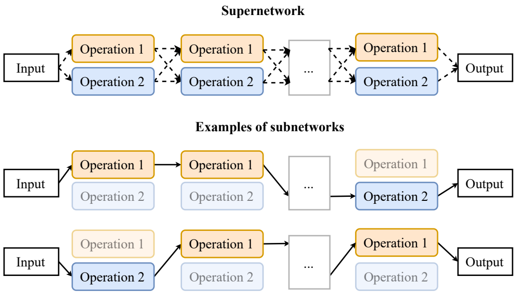


Figure 1: A schematic of a generic supernetwork.

Cascading [47] is a particular case of ensembling with the focus on efficiency: whereas an ensemble requires that every input is processed by every model, a cascade proceeds in a sequential manner, invoking a larger model only if the predictions of smaller models are not confident enough (judged by the confidence function, see Section 2.3 for details). Thus, easy inputs consume less computational resources, while harder inputs can still be predicted well by utilizing every model in the cascade. Despite the potential of cascades to produce efficient and effective models, they remain underexplored in the context of deep learning [49], and in particular no work has yet been done on combining NAS with cascade search.

To perform any kind of NAS, efficient search algorithms are indispensable. Evolutionary Algorithms (EAs) are particularly fit to the task, as they do not require the search space to be continuous, are known to solve real-world problems efficiently [12], and excel in solving multi-objective and dynamic problems [5, 41, 46].

Driven by the observations above, in this paper we present an algorithm called Evolutionary Neural CAscade Search (ENCAS). ENCAS is supernetwork-based and designed to take advantage of various pretrained supernetworks. ENCAS can search over a user-specified set of arbitrary supernetworks that may have e.g. different operations or numbers of layers (the only restriction being that subnetworks extracted from a supernetwork require no retraining). Our algorithm is multi-objective with the goal of finding models on the optimal effectivity-efficiency trade-off front.

The main contributions of our paper are threefold:

- This work is the first to research the combination of NAS and cascades.
- This work is the first to investigate the feasibility of using several different supernetworks in NAS. We explore whether the additional diversity they provide is helpful for creating cascades.
- The ENCAS algorithm is introduced to search for efficient and effective cascades.

2 RELATED WORK

2.1 Neural Architecture Search

The first methods to learn neural network architectures trained each candidate architecture from scratch, taking tens of thousands of GPU hours [58]. ENAS [32] introduced the ideas of weight sharing and supernetworks, which drastically decreased search costs. Multiple algorithms followed, most famously DARTS [25] that reformulated the problem of operation selection as a continuous one, achieving great efficiency.

However, supernetwork weights were discarded after NAS, with the final model being retrained from scratch (because it led to better results). This becomes costly when more than one model is required, e.g. considering both server-based and smartphone-based deployment. OnceForAll (OFA) [8] introduced an algorithm for training supernetwork weights such that they could be used in extracted subnetworks without any further training . AttentiveNAS [48] and AlphaNet [39] introduced training techniques that lead to even better performance (albeit using a different search space).

Neural Architecture Transfer [27] (NAT) is an approach for fine-tuning a pretrained supernetwork for smaller datasets. The key difference from the previous approaches is that the architectures are adapted together with the weights in a multi-objective search procedure, trading off size and accuracy. The main idea is training only subnetworks close to the currently known trade-off front. To this end, a population of networks is evolved by Non-dominated Sorting Genetic Algorithm III (NSGA-III) [13], a prominent many-objective EA. In this work, we aim to reproduce and to build upon NAT. It should be noted that NAT uses two sets of supernetwork weights that share the same search space[2]; in contrast, in this work we are primarily interested in supernetworks that represent different search spaces.

2.2 Search for architectures of neural ensembles

Neural Ensemble Search [56] (NES) is the first approach to search architectures of neural ensembles. It trained multiple networks separately, without weight sharing. The follow-up work, Multi-headed Neural Ensemble Search [30] (MH-NES), utilizes weight sharing by having different ensemble members share first layers of the network. MH-NES achieves gains in robustness, search efficiency and model efficiency. Neural Ensemble Architecture Search (NEAS) [11] is a similar approach with the key idea of gradual removal of the least promising operations from the supernetwork.

Neural Ensemble Search via Sampling (NESS) [40] is supernet-based but does not require the ensemble models to have the same first layers. For this, a supernetwork is first trained, then subnetworks are sampled via novel sampling algorithms.

Our algorithm, ENCAS, substantially differs from all the ones discussed above. Firstly, ENCAS searches for architectures of cascades, for which no prior work exists (to the best of our knowledge). Secondly, ENCAS is truly multi-objective, with a single run producing multiple networks on a trade-off front of model size and performance, whereas NES, MH-NES, and NESS do not directly optimize model efficiency; NEAS uses model size as a constraint in single-objective optimization, which thus needs to be run once for each target model size. Thirdly, none of the existing algorithms take advantage of pretrained supernetworks, while we purposefully design our algorithm to rely on them, bearing in mind that

[1] This holds for modern state-of-the-art approaches [8, 39, 48] but does not hold for all, especially older, approaches [25, 42].

[2] This was not clear to us from [27], but it was explained to us in private communication with the authors.

pretraining plays a huge role in the success of deep learning approaches [14]. Finally, all existing algorithms require the user to specify the ensemble size in advance. Our algorithm has the *maximum* cascade size as a hyperparameter, meaning that it can output cascades with fewer networks if adding more networks does not improve the performance.

2.3 Cascades of neural networks

As mentioned, cascading is a way of efficiently combining multiple available models. It requires the user to choose a confidence function and confidence thresholds. The confidence function estimates how confident a model is in its prediction for a specific input. An example of that could be the maximum predicted probability or the gap between the largest and the second-largest logit values [43]. The confidence thresholds are used to decide when to stop evaluation and to return the current output. A cascade operates sequentially in the following way: the current model makes a prediction and a confidence value for it is determined by the confidence function; if this value is above the confidence threshold for the current model, the cascade is terminated, otherwise the process is repeated for the next model. Note that the output of a cascade can be either the output of the last used model [43] (i.e. the outputs of unconfident models are discarded), or the averaged outputs of all the used models [49]. We follow [49] in using the averaged outputs.

Cascades are popular in machine learning [23, 54], but in deep learning there are only a few works utilizing them [1, 10]. Recently, [49] pointed out that cascades can dominate single models in terms of performance, efficiency, and training time. Cascades in [49] were constructed via an exhaustive search of a small search space of predefined networks.

GreedyCascade [43] achieved good results by designing an efficient greedy algorithm for cascade selection from a somewhat larger number of networks. GreedyCascade has a fundamental limitation: by construction, it cannot produce a cascade that would perform better than the best model in the model pool. Our algorithm does not have this limitation in its design. In addition, GreedyCascade scales quadratically in the number of networks.

Our algorithm searches in a search space that is substantially larger than ever used before for cascade search (it contains hundreds of networks instead of dozens). Also, ENCAS is multi-objective and requires only a single run to create the trade-off front, unlike the exhaustive search procedure of [49].

2.4 Evolutionary algorithms

An EA is a population-based optimization algorithm that relies on the ideas of *(1)* fitness-based selection and *(2)* variation (most often mutation and crossover, i.e. information transfer between solutions in the population). EAs achieve SOTA results on a variety of benchmark and real-world problems [4, 12, 31].

In this paper, we use NSGA-III [13] (as part of NAT) and the Multi-objective Gene-pool Optimal Mixing Evolutionary Algorithm (MO-GOMEA) [29]. NSGA-III relies on non-dominated sorting and pre-supplied reference points to keep the population spread-out in the objective space. The two key ideas behind MO-GOMEA are linkage learning (which leads to dependent variables being exchanged between solutions as a single group) and Gene-pool Optimal Mixing (which ensures that crossover always leads to a fitness improvement). Since we use the algorithm without any modifications, we refer the interested reader to [29, 45] for details. We choose to use this algorithm because it performs well in many problems [28, 35] and because it does not require setting any hyperparameters (such as population size, crossover type, or mutation rate).

3 METHODS

3.1 Searching for cascades of dynamic size

Let us assume that a supernetwork has been trained via the NAT procedure. In addition to the supernetwork weights, the procedure generates a trade-off front of network architectures. The architectures from this front will be used in ENCAS.

Creating a cascade of a specific size out of a predefined model pool comes down to selecting the appropriate sequence of models, and their confidence thresholds. Let us focus on the models first. Each out of N models can be encoded as a categorical value $1..N$. To consider cascades with fewer models (or even a single model), we add the value 0 to encode a "no operation" model — a model that does nothing. Then, a solution to the problem of searching for a cascade of maximum size k can be encoded as a list of k values, each ranging from 0 to N. In the interest of robustness we do not search for the weights of the models in the cascade, and take a simple average of their outputs instead.

As to the confidence threshold values, they are encoded as $k-1$ additional categorical variables. In our experiments we use 51 possible values from 0.0 to 1.0 with step size 0.02. If all thresholds are equal to 1, the cascade becomes an ensemble, since the confidence of any model will always be smaller than 1, and thus all the models will be used.

A visual representation of a solution can be seen in Figure 2.

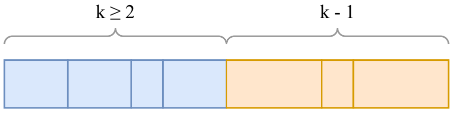


Figure 2: Representation of a solution for ENCAS.

To evaluate a solution, every subsequent model is used to only update probabilities of inputs for which previous models were not confident (once the confidence for an input is above the current threshold, cascading stops). The final probabilities are used as cascade predictions to evaluate its performance. The FLOPs of a cascade are computed as a weighted sum of the FLOPs of all the models in the cascade, where each weight is the fraction of the total number of inputs that this model was used on. Since the models in the model pool are known in advance, their outputs on the validation set can be precomputed, which leads to fast search times of under 1 GPU-hour even on a large dataset (e.g. ImageNet) and with hundreds of base models to choose from.

Note that this approach is trivial to extend to multiple supernetworks by adding the models from the trade-off fronts of all the supernetworks to the model pool. Since the algorithm relies on network outputs, the problem of different supernetworks having different operations is side-stepped.

Any multi-objective search algorithm can be run to maximize validation accuracy and minimize FLOPs. We use MO-GOMEA [29]. Pseudocode of ENCAS is listed in Algorithm 1.

Algorithm 1: ENCAS

```
Input   : Supernetworks $\{S_i\}$, possible thresholds $\{t_i\}$,
          maximum cascade size $k$
model_pool = []
for $S_i$ in $\{S_i\}$ do
    trade_off_front$_i$ = NAT($S_i$)
    model_pool = model_pool ∪ trade_off_front$_i$
fitness_func = make_fitness_func(model_pool, $\{t_i\}$, $k$)
/* make_fitness_func defines the procedure for decoding and
   encoding the values (see Fig. 2), and for evaluating a
   solution, i.e. a cascade (see Section 2.3).              */
cascades = MO-GOMEA(fitness_func)
return cascades // the trade-off front
```

Empirically, we observed that ENCAS finds hundreds of cascades. To reduce that number and to combat overfitting, the trade-off front found by ENCAS is filtered: we traverse the non-dominated front from least accurate to most accurate cascades, and a cascade is kept if its accuracy on the validation set after rounding to the first significant digit is higher than the accuracy of the previous cascade.

3.2 Joint training and cascade search

In ENCAS, only the models from the trade-off front of each supernetwork are considered. This means that ENCAS works with models that are very good on their own, but it also means that it cannot create cascades of models that might be subpar individually but extremely good together. Models with weights that complement each other in this way may not even exist in separately-trained supernetworks, so ideally the training of a supernetwork and the cascade search should happen simultaneously. To investigate whether this idea is feasible, we construct a version of ENCAS called ENCAS-joint (indicating that training and search are performed jointly).

ENCAS-joint extends NAT to training several different supernetworks simultaneously. Whereas in NAT a solution represents an architecture of a single model, in ENCAS-joint a solution represents architectures of all the models in a cascade, their target positions, and the threshold values. Confidence thresholds are restricted to 10 possible values from 0.1 to 1.0 with step size 0.1 to decrease the search space size; the confidence threshold of the last network is not used. These per-supernetwork representations are concatenated to encode the whole cascade. Figure 3 visualizes the encoding.

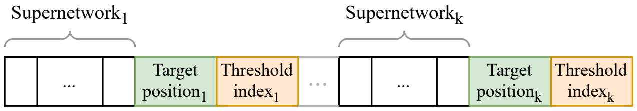


Figure 3: Representation of a solution in ENCAS-joint.

Note that our encoding of the order of the networks is generally inefficient (i.e. a permutation would be more efficient), but since we use a small number of supernetworks (5), the number of possible orderings is quite small, and our Cartesian encoding should suffice.

Before evaluating a cascade, the networks are ordered (with ties broken arbitrarily), after which the cascade is evaluated as usual. NAT requires defining a surrogate that predicts the fitness of an architecture. To extend this to multiple supernetworks in a simple way, we create such a surrogate for each supernetwork used, and an additional surrogate that combines outputs of supernetwork-wise surrogates for a prediction for the whole cascade. Yet another surrogate is used to estimate the FLOPs count for a cascade from FLOPs of individuals models, target positions, and thresholds (this is necessary because changing thresholds or the order of networks impacts not only its performance but also the FLOPs count). Each of the surrogates is a Radial Basis Function (RBF) [6] ensemble, the same as in NAT.

Each supernetwork is trained separately based on which subnetworks from it are present in the population. In order not to disadvantage supernetworks that might require more training to achieve good accuracy, we train all the supernetworks for an equal number of steps. To avoid tuning hyperparameters of NSGA-III to the new scenario, we exchange it for MO-GOMEA, for which no hyperparameters are tuned.

Note that this approach has a limitation of not allowing a cascade to contain several models from the same supernetwork. Therefore, it is possible to further improve results by running ENCAS on the supernetworks trained by ENCAS-joint (we refer to this combination as ENCAS-joint+). In the next section we compare all versions of our algorithm.

4 EXPERIMENTS

We conduct experiments on established computer vision benchmark datasets: ImageNet (ILSVRC2012) [37], CIFAR-10 [22], and CIFAR-100 [22]. In our experiments, we consider the bi-objective problem of maximizing top-1 accuracy while minimizing the FLOPs count. Note that hyperparameter selection and all search procedures were performed on the validation subsets, while the experimental results are reported on the test sets. This means that a trade-off front that is monotonous when evaluated on the validation set often becomes non-monotonous when evaluated on the test set. Since one should not use the test set to select models, we show all the models, even if they are dominated once the test accuracy is considered.

The validation sets for CIFAR-10 and CIFAR-100 consist of 10,000 images randomly split off from the training set (that contain 50,000 images; test sets contain 10,000 images). For ImageNet we rely on the pretrained networks that use the whole training set and report the results on the ILSVRC2012 validation set, as is established practice, since the true test set is not publicly available. Images seen during training cannot be used during the search because their activations have different statistics [43], and for comparability our results should be reported on the ILSVRC2012 validation set (which is treated as the test set). As the actual validation set, we therefore used 20,683 images from ImageNetV2 [34], which is a dataset designed to match the ImageNet collection procedure as closely as possible[3]. We would prefer to avoid using this additional

[3] ImageNetV2 contains three sets of 10,000 images that were collected slightly differently, we use images from all three sets: removing duplicates gives 20,683 images.

data, but cannot; as the number of these images is only 1.6% of the ImageNet training set size, we assume that the unfair advantage we gain by using it is negligible.

We use the normalized hypervolume indicator [57] as a metric of the quality of a trade-off front (see Appendix F for details). Every experiment is run 10 times, mean and standard deviation are reported. We plot the median run (in terms of hypervolume), along with a shaded area delimited by the worst and best fronts achieved over all the runs. Appendix A contains our hyperparameters. Search time is measured on a single Nvidia 2080TI GPU. For statistical testing we use the Wilcoxon signed-rank test [53] with Bonferroni correction [17] (target p-value=0.01, 20 tests, corrected p=0.0005, mentions of statistical significance in the text imply smaller p, all p-values are reported in Appendix E).

Our code is public[4]. We have worked with our own implementation of NAT because we did not have access to the authors' code of NAT that was used for the NAT article.

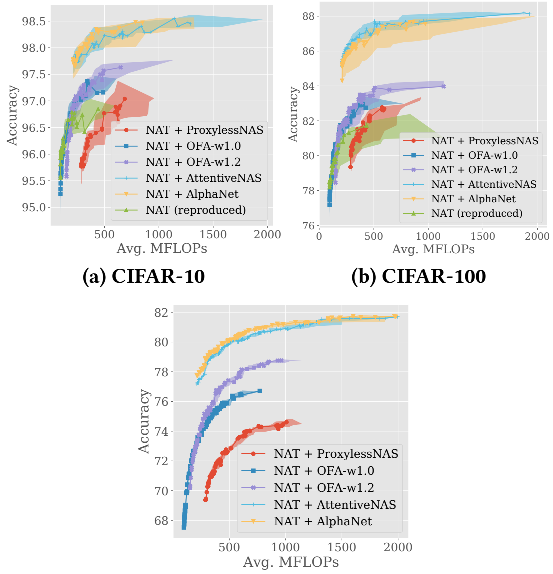
4.1 Baseline supernetworks

We are interested in utilizing pretrained supernetworks, as training one from scratch takes on the order of thousands of GPU-hours [8]. Unfortunately, many papers do not release either code or pretrained weights. As more supernetworks become available in the future, the value of searching across supernetworks should only increase.

In our experiments we rely on five different supernetworks pretrained on ImageNet: AttentiveNAS [48], AlphaNet [39], ProxylessNAS [9], OFA-w1.0 [8], OFA-w1.2 [8]. All of them are built from inverted residual blocks [38]. Moreover, ProxylessNAS, OFA-w1.0, OFA-w1.2 have the same search space, with only width multipliers being different (to get the actual number of neurons in a layer, the base number of neurons is multiplied by the width multiplier). AttentiveNAS and AlphaNet are from the same search space, but the weights were trained via different approaches.

To adapt a supernetwork to CIFAR-10 and CIFAR-100, we apply the NAT procedure for each supernetwork separately. This produces trade-off fronts of models, which in the following sections will be used for cascade construction. For CIFAR-10 and CIFAR-100 we also reproduce NAT with its original hyperparameters using OFA-w1.0 and OFA-w1.2 (due to computational constraints, we do not reproduce NAT for ImageNet). For ImageNet, there is no need to further update the weights, however the trade-off front still needs to be found. For this reason, we run a version of NAT with no training and no reevaluation of the already evaluated networks.

Results of using NAT with each supernetwork are presented in Fig. 4. It can be seen that the choice of the supernetwork impacts the resulting trade-off front significantly, with supernetworks that perform better on ImageNet also performing better on CIFAR-10 and CIFAR-100, as expected [21]. Additionally, our reproduced NAT achieves results inferior to those reported in [27], even after we introduced changes that were not in the article but suggested by the authors in private communication (see Appendix A). This prompted us to look for better hyperparameters, which are used in all our experiments (including those in Fig. 4). With these hyperparameters, search time is 30 GPU-hours for OFA-w1.0, OFA-w1.2, or ProxylessNAS, and 45 GPU-hours for AlphaNet or AttentiveNAS.



(a) CIFAR-10

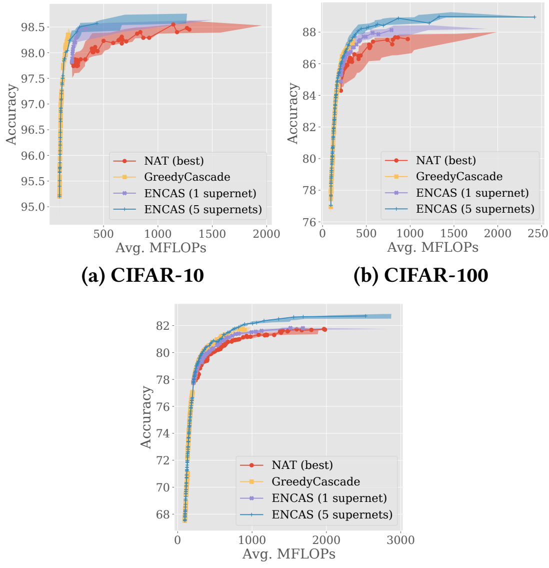
(b) CIFAR-100

(c) ImageNet

Figure 4: Results of running NAT [27] with different supernetworks on CIFAR-10, CIFAR-100, ImageNet.

4.2 Cascading best NAT results

Figure 5 and Tables 1, 2, 3 show that using ENCAS on the NAT results with the single best supernetwork leads to the models on the fronts becoming more efficient across all datasets, with hypervolumes increasing (statistically significant; difference in maximum accuracy is not statistically significant). Visually, the effect is small on ImageNet, and noticeable on CIFAR-10 and CIFAR-100.



(a) CIFAR-10

(b) CIFAR-100

(c) ImageNet

Figure 5: Comparing ENCAS to the baselines.

[4] https://github.com/AwesomeLemon/ENCAS

Table 1: ImageNet performance, "acc." is top-1 accuracy. The method producing the highest accuracy is in bold.

Method	Hyper-volume	Max acc.	Max MFLOPs
EfficientNet B0-B3 [44]	0.464	81.6	1800
EfficientNet C0-C3 [49]	0.482	82.2	1800
MobileNetV3 [19]	0.396	76.6	350
OFA (#75) [8]	0.451	80.0	595
NEAS [11]	0.458	80.0	574
NSGANetV2 [26]	0.477	80.4	593
AlphaNet [39]	0.490	80.8	709
BigNAS [55]	0.480	80.9	1040
NAT (best) [27]	0.506$_{\pm 0.001}$	81.69$_{\pm 0.02}$	1962$_{\pm 33}$
GreedyCascade [43]	0.520$_{\pm 0.002}$	81.72$_{\pm 0.11}$	927$_{\pm 36}$
ENCAS (1 supernet)	0.509$_{\pm 0.001}$	81.78$_{\pm 0.05}$	1929$_{\pm 424}$
ENCAS (5 supernets)	**0.537**$_{\pm 0.001}$	**82.72**$_{\pm 0.10}$	2616$_{\pm 221}$

Table 2: CIFAR-100 performance, "acc." is top-1 accuracy. The method producing the highest accuracy is in bold.

Method	Hyper-volume	Max acc.	Max MFLOPs
EfficientNet B0-B3 [44]	0.664	89.9	1800
NSGANetV2 [26]	0.658	88.3	796
GDAS [16]	—	81.87	519
SETN [15]	—	82.75	722
NAT (reproduced) [27]	0.523$_{\pm 0.012}$	81.53$_{\pm 0.54}$	682$_{\pm 298}$
NAT (best) [27]	0.659$_{\pm 0.004}$	87.94$_{\pm 0.23}$	1277$_{\pm 287}$
GreedyCascade [43]	0.665$_{\pm 0.005}$	87.25$_{\pm 0.25}$	341$_{\pm 30}$
ENCAS (1 supernet)	0.663$_{\pm 0.005}$	88.09$_{\pm 0.21}$	1051$_{\pm 353}$
ENCAS (5 supernets)	0.699$_{\pm 0.003}$	88.96$_{\pm 0.17}$	1401$_{\pm 420}$
ENCAS-joint	0.681$_{\pm 0.005}$	88.55$_{\pm 0.19}$	6678$_{\pm 1735}$
ENCAS-joint+	**0.701**$_{\pm 0.005}$	**89.10**$_{\pm 0.22}$	1750$_{\pm 418}$

4.3 Cascading all NAT results

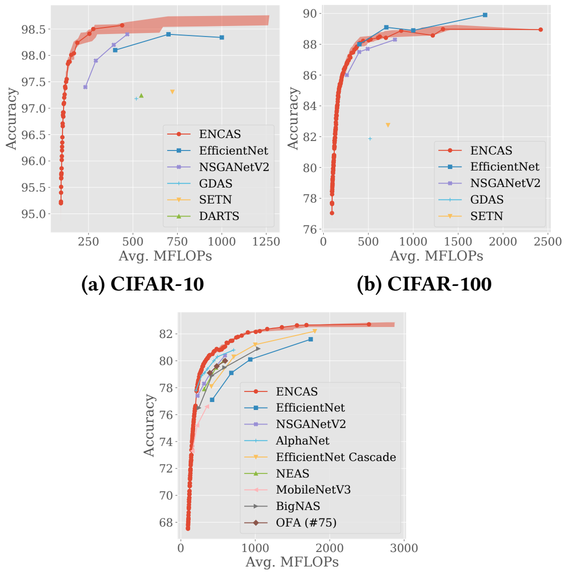
The next question is whether using supernetworks other than the best one will improve the results of ENCAS. As shown in Figure 5, the results are strongly improved, with the differences in hypervolume and maximum accuracy to the best NAT baseline (and to ENCAS with 1 supernetwork) being statistically significant. We hypothesize that inclusion of better and more diverse supernetworks would make the gap even larger. Search time of ENCAS is 1 GPU-hour (with approximately 300 base models). Since we report all the cascades found by ENCAS (several dozen), we do not name them, but for the ease of reference we name a subset (see Appendix D).

4.4 Comparison to SOTA

We compare ENCAS with the SOTA cascade search algorithm GreedyCascade [43] by applying it to the same model pool. As can be seen in Figure 5, ENCAS matches the performance of GreedyCascade for smaller FLOPs on all datasets, and can find cascades with better accuracy than the best baseline model, which GreedyCascade cannot do. Note that the runtime of both algorithms (under 1 hour) is negligible in comparison to the supernetwork adaptation time (tens of hours). Differences in hypervolume and maximum accuracy between ENCAS and GreedyCascade are statistically significant.

Our results are also compared with those of previous efficient NAS algorithms (see Tables 1, 2, 3). Fig. 6 shows that the trade-off fronts produced by ENCAS dominate other NAS approaches under 1.5 GFLOPs across the datasets. But it can also be seen that for CIFAR-100 while ENCAS is on par with EfficientNet-B0 to B2, it is outperformed by B3, even though the supernetworks we use outperform EfficientNet B0 to B3 on ImageNet. This may occur because training an individual network is much easier than training a supernetwork (in our experience, training a supernetwork is hard due to subnetworks having to share both weights and hyperparameters).

Note that we do not compare search times of different algorithms because the corresponding publications often report times that are not comparable due to e.g. using different hardware, not accounting for supernetwork training or final network retraining time. A fair comparison would require us to run all the algorithms, for which we lack compute. For future reference, the runtime associated with each part of our pipeline is mentioned in the section describing it.



(a) CIFAR-10 **(b) CIFAR-100**

(c) ImageNet

Figure 6: Comparing ENCAS with SOTA NAS algorithms.

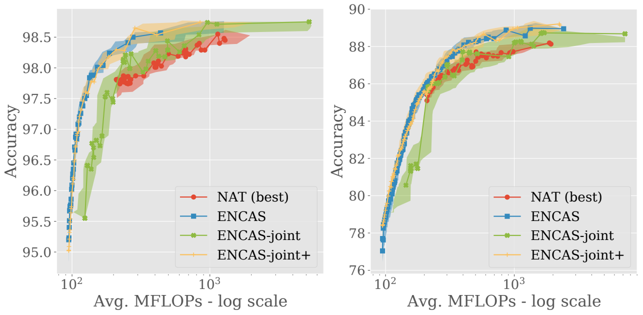
4.5 Joint training and cascade search

Is joint weight training and search of cascade architectures beneficial? In Fig. 7 we can see that ENCAS-joint finds a trade-off front that is worse than the one found by ENCAS. This likely happens due to the increased size of the search space. However, the trade-off front found by ENCAS-joint is better than the best NAT one.

We further see that ENCAS-joint+ (running ENCAS on the supernetworks trained by ENCAS-joint) improves upon ENCAS-joint on both CIFAR-10 and CIFAR-100. But is it better than running ENCAS on separately trained supernetworks? Although ENCAS-joint+

Table 3: CIFAR-10 performance, "acc." is top-1 accuracy. The method producing the highest accuracy is in bold.

Method	Hyper-volume	Max acc.	Max MFLOPs
EfficientNet B0-B2 [44]	0.863	98.4	1000
NSGANetV2 [27]	0.904	98.4	468
GDAS [16]	—	97.18	519
SETN [15]	—	97.31	722
DARTS [25]	—	97.24	547
NAT (reproduced) [27]	$0.899_{\pm 0.003}$	$96.80_{\pm 0.13}$	$361_{\pm 119}$
NAT (best) [27]	$0.911_{\pm 0.002}$	$98.46_{\pm 0.08}$	$1390_{\pm 274}$
GreedyCascade [43]	$0.935_{\pm 0.002}$	$98.31_{\pm 0.05}$	$191_{\pm 16}$
ENCAS (1 supernet)	$0.911_{\pm 0.002}$	$98.45_{\pm 0.09}$	$698_{\pm 321}$
ENCAS (5 supernets)	$0.941_{\pm 0.002}$	$98.60_{\pm 0.09}$	$749_{\pm 298}$
ENCAS-joint	$0.935_{\pm 0.002}$	$98.68_{\pm 0.04}$	$4858_{\pm 1126}$
ENCAS-joint+	$\mathbf{0.943}_{\pm 0.002}$	$\mathbf{98.68}_{\pm 0.08}$	$1060_{\pm 444}$



(a) CIFAR-10

(b) CIFAR-100

Figure 7: Investigating benefits of joint training and search.

appears to outperform ENCAS in terms of hypervolume and maximum accuracy (see Tables 2, 3), these results are not statistically significant.

Note that these experiments are not performed on ImageNet due to limitations in computational power. Training supernetworks jointly is computationally intensive in general (search time of ENCAS-joint is 240 GPU-hours) while also lacking flexibility, as adding or removing a network means restarting the whole process from scratch. Given inconclusiveness of improvements brought by ENCAS-joint+, we recommend using ENCAS and separate training of supernetworks (see Section 6 for further discussion).

4.6 Applying ENCAS to SOTA ImageNet models

ENCAS relies on the architectures discovered via supernetwork-based NAS. However, using a supernetwork means that these architectures are necessarily not very large. Because of this, NAS results are typically evaluated in the context of a mobile phone setting, which is usually taken to mean ≤ 600 MFLOPs.

However, nowadays there are hundreds of large well-performing ImageNet-pretrained models available online. Can our good results be extended from the mobile phone setting to dominating the complete trade-off front? To answer this question, we take 518 ImageNet models from the Pytorch Image Models (`timm`) [52] library, and run our search procedure on them.

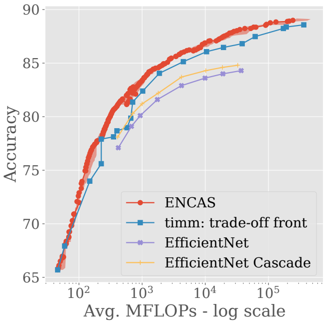


Figure 8: ENCAS discovers a dominating trade-off front on ImageNet by searching for cascades of 518 `timm` models (from which only the models on the trade-off front are shown).

As shown in Figure 8, this indeed leads to a dominating trade-off front. The increase of 0.4 percentage points in the maximum ImageNet performance leads to our largest cascade achieving the highest ImageNet accuracy of publicly available models ($89.01_{\pm 0.10}$), while simultaneously decreasing FLOPs by 18% (from 362 GFLOPs to $296_{\pm 77}$ GFLOPs). Our cascades outperform those in [49], in large part thanks to the ability of our algorithm to use a search space containing hundreds of models, which is not feasible for the exhaustive search approach used in [49].

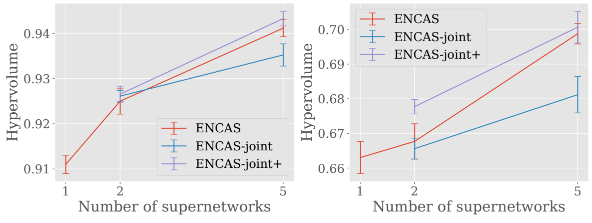
5 ADDITIONAL EXPERIMENTS

In this section we further investigate the impact of using more than one supernetwork. Due to space constraints, a comparison to ensembles is provided in Appendix B and a comparison to random search is provided in Appendix C.

5.1 Impact of increasing the number of supernetworks

Figure 9 shows the impact of increasing the number of supernetworks used in ENCAS from 1 to 2 to 5. A trend of increasing hypervolume can be clearly observed.

For joint training (ENCAS-joint), the hypervolume also increases, but not as much. This can be explained by the increase in the search space that every additional supernetwork brings. Interestingly, the hypervolume obtained with ENCAS-joint+ grows about as fast as with ENCAS, which we interpret to mean that the joint training and search over an increasing number of supernetworks is beneficial for weights while harmful for the simultaneous search (given the same search budget). The larger search space may require a larger search budget to achieve better results, and therefore ENCAS-joint may have higher potential to benefit from more compute. Running ENCAS using the weights trained by ENCAS-joint (i.e. ENCAS-joint+) realizes the benefit of better weights and ameliorates the downside of a larger search space.



(a) CIFAR-10 (b) CIFAR-100

Figure 9: Impact of increasing the number of supernetworks in ENCAS, ENCAS-joint, ENCAS-joint+.

5.2 Is using different supernetworks better than using the best one trained several times?

Experiments in section 4.3 demonstrated that using several supernetworks is better than using just the best one. But is the source of the effect the diversities of architectures found, or just the increased quantity of networks with different weights? To answer this question, we train (via NAT) the best supernetwork 5 times with different seeds on CIFAR-10 and CIFAR-100 and apply ENCAS to the resulting trade-off fronts.

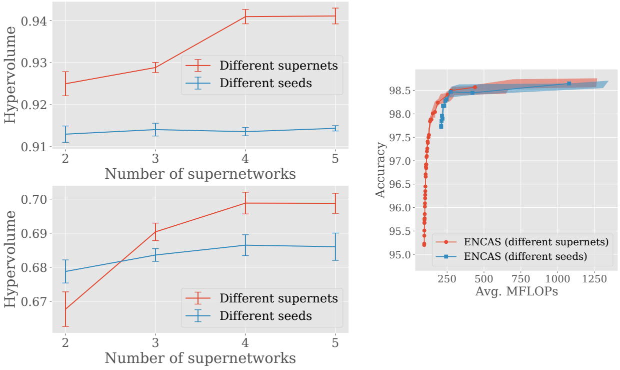


Figure 10: Comparison between hypervolume when using different supernetworks, or the best supernetwork trained with different seeds on *(top left)* CIFAR-10, *(bottom left)* CIFAR-100. *(right)* shows comparison between trade-off fronts of 5 supernetworks, or 5 seeds of the best supernetwork on CIFAR-10.

In Fig. 10 (left) we see that using different runs of the best supernetwork barely increases hypervolume, in contrast to using different supernetworks. If we inspect the trade-off fronts in Fig. 10 (right) for 5 supernetworks and for 5 seeds of the best supernetwork on CIFAR-10, we can see that the difference is in low-FLOPs models that are missing from the best supernetwork but are present in other supernetworks. Therefore, using diverse supernetworks is helpful for obtaining a larger trade-off front coverage. However, on the side of the most accurate networks, using multiple restarts of NAT with the best supernetwork is sufficient to get close to the best performance, unlike what could be expected, since the diversity in architectures and weights is arguably lower.

6 DISCUSSION

While our approach achieves good results with limited resource usage, it still suffers from limitations. Notably, it relies on pretrained supernetworks, which are currently not very diverse, architecture-wise. Additionally, these supernetworks need to allow extraction and usage of subnetworks without retraining in order for our approach to work, which limits their selection even further.

In our experiments, we find that performing search after the supernetwork weights have been adapted is not much worse than joint training and search. This can mean that there is not a lot of benefit to be gained by fine-tuning architecture choices of different cascade components to each other; alternatively, perhaps ENCAS-joint was simply not able to realize these benefits, for instance because it may require more computational resources than we used in our experiments.

This paper has demonstrated the benefits brought by the usage of cascades. This reinforces the main thesis of [49]: researchers should pay more attention to cascades. However, the warnings of [49] should also be repeated, as they apply to any cascade approach, including ours: the decrease in FLOPs brought by cascades can be realized either when processing images one-by-one, or when processing a large amount of images offline. The benefits are not realized in online batch processing: once a batch has been created, due to the parallel nature of GPU accelerators, processing a part of a batch takes approximately the same resources as processing the whole batch.

7 CONCLUSION

In this paper, we considered the automatic creation of cascades of deep neural networks. We developed an effective algorithm called ENCAS that builds upon the literature on efficient NAS by searching for cascades across pretrained supernetworks either simultaneously with weight training or after weight training. ENCAS is the first NAS algorithm that searches for cascade architectures. It does so by solving the multi-objective optimization problem of finding well-performing small cascades with the help of an EA (MO-GOMEA).

ENCAS was found to outperform SOTA efficient NAS approaches on several image classification datasets. Its search procedure can also be applied to an arbitrary model pool. By applying it to well-performing publicly available ImageNet models, we achieved a dominating trade-off front on ImageNet.

Finally, we find that searching for neural network architectures in more than one pretrained supernetwork is beneficial despite the limited diversity of the currently available supernetworks, which is expected to only increase with time.

ACKNOWLEDGMENTS

We thank Arkadiy Dushatskiy for insightful discussions, Marco Virgolin for introducing us to [49], Zhichao Lu for sharing his knowledge about NAT and parts of the NAT codebase.

This work is part of the research project DAEDALUS funded via the Open Technology Programme of the Netherlands Organization for Scientific Research (NWO), project number 18373; part of the funding is provided by Elekta and ORTEC LogiqCare. This work was carried out (in part) on the Dutch national e-infrastructure with the support of SURF Cooperative.

REFERENCES

[1] Anelia Angelova, Alex Krizhevsky, Vincent Vanhoucke, Abhijit Ogale, and Dave Ferguson. 2015. Real-Time Pedestrian Detection With Deep Network Cascades. In *Proceedings of BMVC 2015*.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2017. Accelerating neural architecture search using performance prediction. *arXiv preprint arXiv:1705.10823* (2017).

[4] Anton Bouter, Tanja Alderliesten, Bradley R Pieters, Arjan Bel, Yury Niatsetski, and Peter A N Bosman. 2019. GPU-accelerated bi-objective treatment planning for prostate high-dose-rate brachytherapy. *Medical Physics* 46, 9 (2019), 3776–3787.

[5] Jürgen Branke and Hartmut Schmeck. 2003. Designing evolutionary algorithms for dynamic optimization problems. In *Advances in Evolutionary Computing*. Springer, 239–262.

[6] David S Broomhead and David Lowe. 1988. *Radial basis functions, multi-variable functional interpolation and adaptive networks*. Technical Report. Royal Signals and Radar Establishment Malvern (United Kingdom).

[7] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901.

[8] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2020. Once for All: Train One Network and Specialize it for Efficient Deployment. In *International Conference on Learning Representations*.

[9] Han Cai, Ligeng Zhu, and Song Han. 2019. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. In *International Conference on Learning Representations*. https://openreview.net/forum?id=HylVB3AqYm

[10] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. 2015. Learning complexity-aware cascades for deep pedestrian detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 3361–3369.

[11] Minghao Chen, Jianlong Fu, and Haibin Ling. 2021. One-Shot Neural Ensemble Architecture Search by Diversity-Guided Search Space Shrinking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 16530–16539.

[12] Raymond Chiong, Thomas Weise, and Zbigniew Michalewicz. 2012. *Variants of evolutionary algorithms for real-world applications*. Springer.

[13] Kalyanmoy Deb and Himanshu Jain. 2013. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation* 18, 4 (2013), 577–601.

[14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. Decaf: A deep convolutional activation feature for generic visual recognition. In *International Conference on Machine Learning*. PMLR, 647–655.

[15] Xuanyi Dong and Yi Yang. 2019. One-shot neural architecture search via self-evaluated template network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 3681–3690.

[16] Xuanyi Dong and Yi Yang. 2019. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1761–1770.

[17] Olive Jean Dunn. 1961. Multiple comparisons among means. *J. Amer. Statist. Assoc.* 56, 293 (1961), 52–64.

[18] Roberto Esposito and Lorenza Saitta. 2003. Monte Carlo theory as an explanation of bagging and boosting. In *IJCAI*, Vol. 3. 499–504.

[19] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. 2019. Searching for MobileNetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1314–1324.

[20] Fabian Isensee, Paul F Jaeger, Simon AA Kohl, Jens Petersen, and Klaus H Maier-Hein. 2021. nnU-Net: a self-configuring method for deep learning-based biomedical image segmentation. *Nature methods* 18, 2 (2021), 203–211.

[21] Simon Kornblith, Jonathon Shlens, and Quoc V Le. 2019. Do better ImageNet models transfer better?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2661–2671.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. *Learning multiple layers of features from tiny images*. Technical Report.

[23] Ignas Kukenys and Brendan McCane. 2008. Classifier cascades for support vector machines. In *2008 23rd International Conference Image and Vision Computing New Zealand*. IEEE, 1–6.

[24] Yann LeCun, Yoshua Bengio, et al. 1995. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks* 3361, 10 (1995), 1995.

[25] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. DARTS: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[26] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. 2020. NSGANetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision*. Springer, 35–51.

[27] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. 2021. Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 9 (2021), 2971–2989.

[28] Ngoc Hoang Luong, Marinus OW Grond, Han La Poutré, and Peter A N Bosman. 2015. Scalable and practical multi-objective distribution network expansion planning. In *2015 IEEE Power & Energy Society General Meeting*. IEEE, 1–5.

[29] Ngoc Hoang Luong, Han La Poutré, and Peter A N Bosman. 2014. Multi-objective gene-pool optimal mixing evolutionary algorithms. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. 357–364.

[30] Ashwin Raaghav Narayanan, Arber Zela, Tonmoy Saikia, Thomas Brox, and Frank Hutter. 2021. Multi-headed Neural Ensemble Search. In *Workshop on Uncertainty and Robustness in Deep Learning (UDL@ICML'21)*.

[31] Sanyapong Petchrompo, Anupong Wannakrairot, and Ajith Kumar Parlikad. 2022. Pruning pareto optimal solutions for multi-objective portfolio asset management. *European Journal of Operational Research* 297, 1 (2022), 203–220.

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

[33] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4780–4789.

[34] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. 2019. Do ImageNet classifiers generalize to ImageNet?. In *International Conference on Machine Learning*. PMLR, 5389–5400.

[35] S Rodrigues, Pavol Bauer, and Peter A N Bosman. 2016. Multi-objective optimization of wind farm layouts–Complexity, constraint handling and scalability. *Renewable and Sustainable Energy Reviews* 65 (2016), 587–609.

[36] Lior Rokach. 2010. Ensemble-based classifiers. *Artificial Intelligence Review* 33, 1 (2010), 1–39.

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision* 115, 3 (2015), 211–252.

[38] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4510–4520.

[39] Rishab Sharma, Rahul Deora, and Anirudha Vishvakarma. 2020. AlphaNet: An Attention Guided Deep Network for Automatic Image Matting. In *2020 International Conference on Omni-layer Intelligent Systems*. IEEE, 1–8.

[40] Yao Shu, Yizhou Chen, Zhongxiang Dai, and Bryan Kian Hsiang Low. 2021. Going Beyond Neural Architecture Search with Sampling-based Neural Ensemble Search. *arXiv preprint arXiv:2109.02533* (2021).

[41] Gulshan Singh and Kalyanmoy Deb. 2006. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. 1305–1312.

[42] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. 2019. Single-path NAS: Designing hardware-efficient convnets in less than 4 hours. *arXiv preprint arXiv:1904.02877* (2019).

[43] Matthew Streeter. 2018. Approximation algorithms for cascading prediction models. In *International Conference on Machine Learning*. PMLR, 4752–4760.

[44] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6105–6114. https://proceedings.mlr.press/v97/tan19a.html

[45] Dirk Thierens and Peter A N Bosman. 2011. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. 617–624.

[46] David A Van Veldhuizen and Gary B Lamont. 1998. *Multiobjective evolutionary algorithm research: A history and analysis*. Technical Report. Citeseer.

[47] Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1. IEEE, I–I.

[48] Dilin Wang, Meng Li, Chengyue Gong, and Vikas Chandra. 2021. AttentiveNAS: Improving neural architecture search via attentive sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6418–6427.

[49] Xiaofang Wang, Dan Kondratyuk, Kris M Kitani, Yair Movshovitz-Attias, and Elad Eban. 2020. Multiple networks are more efficient than one: Fast and accurate models via ensembles and cascades. *arXiv preprint arXiv:2012.01988* (2020).

[50] Colin White, Arber Zela, Binxin Ru, Yang Liu, and Frank Hutter. 2021. How Powerful are Performance Predictors in Neural Architecture Search?. In *Advances in Neural Information Processing Systems*, Vol. 34. 28454–28469.

[51] Jörg Wichard, Christian Merkwirth, and Maciej Ogorzalek. 2003. Building ensembles with heterogeneous models. (2003).

[52] Ross Wightman. 2019. PyTorch Image Models. https://github.com/rwightman/pytorch-image-models. https://doi.org/10.5281/zenodo.4414861

[53] Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*. Springer, 196–202.

[54] Zhixiang Xu, Matt J Kusner, Kilian Q Weinberger, Minmin Chen, and Olivier Chapelle. 2014. Classifier cascades and trees for minimizing feature evaluation cost. *The Journal of Machine Learning Research* 15, 1 (2014), 2113–2144.

[55] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. 2020. BigNAS: Scaling up neural architecture search with big single-stage models. In *European Conference on Computer Vision*. Springer, 702–717.

[56] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Christopher C. Holmes, Frank Hutter, and Yee Whye Teh. 2021. Neural Ensemble Search for Uncertainty Estimation and Dataset Shift. In *Thirty-Fifth Conference on Neural Information Processing Systems*.

[57] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. 2003. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation* 7, 2 (2003), 117–132.

[58] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).