EMT-NAS:Transferring architectural knowledge between tasks from different datasets

Peng Liao[1], Yaochu Jin[1,2]*, Wenli Du[1]*
[1] Key Laboratory of Smart Manufacturing in Energy Chemical Process, Ministry of Education, ECUST, China
[2] Faculty of Technology, Bielefeld University, Germany
pengliao@mail.ecust.edu.cn, yaochu.jin@uni-bielefeld.de, wldu@ecust.edu.cn

Abstract

The success of multi-task learning (MTL) can largely be attributed to the shared representation of related tasks, allowing the models to better generalise. In deep learning, this is usually achieved by sharing a common neural network architecture and jointly training the weights. However, the joint training of weighting parameters on multiple related tasks may lead to performance degradation, known as negative transfer. To address this issue, this work proposes an evolutionary multi-tasking neural architecture search (EMT-NAS) algorithm to accelerate the search process by transferring architectural knowledge across multiple related tasks. In EMT-NAS, unlike the traditional MTL, the model for each task has a personalised network architecture and its own weights, thus offering the capability of effectively alleviating negative transfer. A fitness re-evaluation method is suggested to alleviate fluctuations in performance evaluations resulting from parameter sharing and the mini-batch gradient descent training method, thereby avoiding losing promising solutions during the search process. To rigorously verify the performance of EMT-NAS, the classification tasks used in the empirical assessments are derived from different datasets, including the CIFAR-10 and CIFAR-100, and four MedMNIST datasets. Extensive comparative experiments on different numbers of tasks demonstrate that EMT-NAS takes 8% and up to 40% on CIFAR and MedMNIST, respectively, less time to find competitive neural architectures than its single-task counterparts.

1. Introduction

Many neural architecture search (NAS) algorithms [8, 16, 23, 56] have shown better performance on a specific task than manually designed deep neural networks [11, 14, 35, 51]. However, when the task (or dataset) changes, the NAS algorithm needs to be run again to find a new optimal network architecture for the new task, which is typical for single-task learning (STL).

By contrast, it has been shown that simultaneously learning multiple related tasks, known as multi-task learning (MTL) is beneficial [3, 52]. One class of MTL focuses on designing or training a single network to jointly solve multiple tasks formed by scene understanding (the same

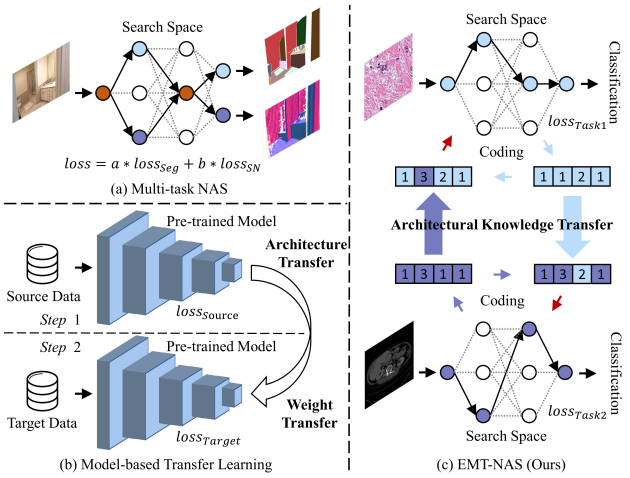

Figure 1. Conceptual differences between our work and two main existing methodologies for two-task learning. (a) Existing multi-task NAS typically handles multiple tasks such as semantic segmentation and surface normal prediction on the same dataset, where the loss function contains losses for multiple tasks, resulting in a network architecture that shares a common set of weight parameters. (b) Model-based transfer learning trains a network architecture on a source dataset and then transfers it to a target dataset by fine-tuning the weights using a separate loss function. (c) EMT-NAS simultaneously optimises multiple classification tasks from different datasets using a separate loss function for each task, resulting in an individual network architecture and corresponding weights for each task. EMT-NAS aims to optimise each task separately while sharing the knowledge of good network architectures to facilitate the optimisation of each task. Best viewed in colour.

*Corresponding author.
Code: https://github.com/PengLiao12/EMT-NAS

dataset) [38, 46], such as the NYU v2 dataset [37], the CityScapes dataset [4], and the Taskonomy dataset [48]. These tasks consist of semantic segmentation, surface normal prediction, depth prediction, keypoint detection and edge detection, which are known as multiple tasks that arise naturally [3]. Another typical scenario of MTL is inspired from human learning, where humans often transfer knowledge from one task to another related task, for example, the skills of playing squash and tennis could help each other to improve [52]. In this case, tasks may come from different datasets. More often than not, a pair of tasks from different datasets have a lower relatedness score than those from the same dataset [18]. This can be clearly shown if we analyse the task relatedness [1] between classification tasks from four medical datasets using the representation similarity analysis (RSA) [7]. The results of our analysis are presented in Fig. 2, from which we can see that the relatedness score becomes especially low when the datasets are from different data modalities, implying that MTL on tasks from different datasets is more challenging.

MTL is able to improve the performance of learning multiple related tasks by means of sharing common knowledge between tasks, including the network architecture and the weights of the model. Existing MTL designs one architecture for multiple tasks, although only a subset of that network architecture is practically used for each task. In other words, there are shares and differences between these subsets of model architectures [33]. In sharing the weights, if one or more tasks have a dominating influence in training the network weights, the performance of MTL may deteriorate on some of the dominated tasks, which is called negative transfer [40]. Therefore, this work aims to improve the performance of each task by finding a personalised network architecture for each task, thereby alleviating negative transfer resulting from the jointly trained weights.

Inspired by the successful research on multi-factorial evolutionary algorithms (MFEAs) [9], a recently proposed approach to knowledge transfer in evolutionary optimisation, this work designs an algorithm to separately search for a personalised network architecture and corresponding weight parameters for each task in the same search space. This way, knowledge transfer across different tasks can be achieved through crossover between these architectures to accelerate the search process on each task, provided that there are similarities between the multiple learning tasks.

Contributions: First, we propose an evolutionary multi-tasking NAS algorithm (EMT-NAS) for finding personalised network architectures for different tasks (datasets) in the same search space. Architectural knowledge transfer is achieved by optimising the supernet parameters of each task separately and recommending neural network architectures with good performance on their own task. Second, block-based crossover and bit-based mutation operators are designed to accommodate the transformation from a continuous space to a discrete space. Finally, a fitness re-evaluation method is introduced to alleviate fitness fluctuations over the generations resulting from parameter sharing and the mini-batch gradient descent based training to keep promising solutions. We present thorough experimental evaluations on different datasets, including medical classification benchmarks (PathMNIST, OrganMNIST_{Axial,Coronal,Sagittal}) and popular image classification benchmarks (CIFAR-10, CIFAR-100, ImageNet), demonstrating the efficacy and the value of each component of EMT-NAS. Our results demonstrate that, unlike traditional MTL, it is feasible to search for personalised network architectures for multiple tasks through architectural knowledge transfer, obtaining better performance and requiring less search time compared with single-task learning.

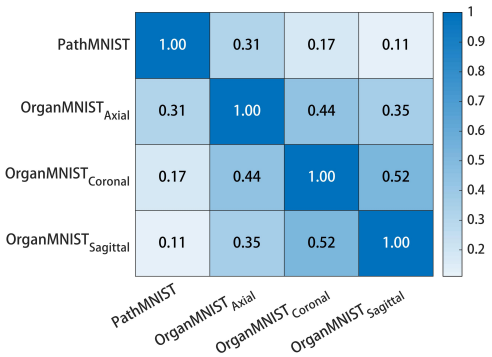

Figure 2. Task similarity matrix. The relatedness scores between four medical tasks are computed using ResNet-50 as a feature extractor. PathMNIST [44] is the tissue classification based on colorectal cancer histology slides, while OrganMNIST_{Axial,Coronal,Sagittal} [44] are organ classifications based on 3 types of 2D images formed by 3D computed tomography (CT) images. We observe that the relatedness between PathMNIST and OrganMNIST_{Axial,Coronal,Sagittal}, respectively, are all smaller than that between OrganMNIST_{Axial,Coronal,Sagittal}.

2. Related Work

Below we summarise the similarities and differences with our work in the following themes.

Multi-Task Learning is designed to learn a network with shared representation from multiple tasks in deep learning. A major challenge in MTL is to balance the joint learning of all tasks to avoid negative transfer [40], motivated by [17] to balance the gradient magnitudes by setting the task-specific weights in the loss. The back-propagation loss function consists of a linear weighting of the losses for each task [18], as shown in Fig. 1a, which may additionally include purpose-specific losses, e.g., the sparsity and sharing of the architecture [38], or loss of generators [32]. Both the coefficients in the loss function and the complexity of the losses affect the final performance of the MTL. In [36],

MTL is considered as multi-objective optimisation, searching for the Pareto optimal solutions to avoid a linear weighting of different losses.

However, all above mentioned algorithms deal with tasks from the same dataset. In this work, updating of the weight parameters of the supernet for each task is independent, i.e., each task has an independent loss function. Rather than solving multiple tasks that arise naturally, the tasks in our MTL setting come from different datasets, or even have from different data modalities.

Transfer Learning establishes a mapping from the source domain to the target domain [15]. Model-based transfer learning is shown in Fig. 1b, where the first step pretrains the model on the source dataset and the second step tunes the parameters of the pre-trained model on the target dataset. During the transfer, the model structure is also adjusted to adapt it to the target task [22]. Note that in the proposed framework, the architectural, rather than the weights parameters, is shared.

Neural Architecture Search has been shown to be effective in the search for network architectures for specific tasks. Three main search strategies have been adopted in NAS, namely reinforcement learning (RL) [6, 53, 57], evolutionary algorithms (EAs) [20, 24, 26, 30, 39], and gradient descent (GD) [25, 41, 43, 45]. With the help of parameter sharing [28], the performance of new neural architectures can be evaluated without training from scratch [31]. It should be pointed out that while parameter sharing can greatly reduce the computation time, it may lead to inaccurate performance assessments of candidate submodels [42] and so-called multi-model forgetting, when multiple models are subsequently trained [2]. Although parameter sharing combined with the mini-batch gradient descent method can further reduce the computation time, promising candidate neural architectures may get lost in evolutionary NAS due to strong fluctuations in performance caused by the mini-batch gradient descent method [50]. Therefore, in our work, a fitness re-evaluation method is proposed to alleviate fitness fluctuations in population iterations.

Evolutionary Multi-Tasking is proposed for solving numerical optimisation problems. When the function landscapes or optimal solutions of two tasks have a certain degree of similarity, promising candidate solutions belonging to one task may be helpful for the other task. Evolutionary multi-tasking has been extended to multi-objective [27], constrained multi-objective [29], and surrogate-assisted for expensive problems [21], and has been show to be effective for practical problems such as hyperspectral band selection [10] and multi-scene scheduling [49]. In this work, we use evolutionary multi-tasking for the first time to transfer architectural knowledge across multiple tasks from different datasets, and block-based crossover and bit-based mutation are designed to accommodate the transformation from a continuous space to a discrete space.

3. Proposed Work

3.1. Evolutionary Multi-Tasking Optimization

In an MFEA, each individual in the population belongs to only one of the tasks to be optimized, and knowledge transfer between tasks is realized with crossover at a probability between individuals belonging to different tasks. A multi-tasking optimization problem consisting of K single-objective minimization tasks can be formulated as follows:

$$\{\mathbf{x}_1^*, \mathbf{x}_2^*, \ldots, \mathbf{x}_K^*\} = \{\operatorname{argmin} f_1(\mathbf{x}_1), \operatorname{argmin} f_2(\mathbf{x}_2), \ldots, \operatorname{argmin} f_K(\mathbf{x}_K)\} \quad (1)$$

where $f_k(\mathbf{x}_k), k = 1, 2, \ldots, K$ is the k-th task. $\mathbf{x}_k^* = (x_{k,1}^*, x_{k,2}^*, \ldots, x_{k,D_M}^*), k = 1, 2, \ldots, K$ is the optimal solution of the k-th task, D_M is the dimension of the k-th task.

3.2. Search Space and Architecture Encoding

To facilitate the transfer of architectural knowledge between multiple tasks, the same search space is used for multiple tasks, i.e., the candidate architectures are consistent for each task. Similar to [57], the search space is based on normal and reduced cells. And the optional operations in each block include depthwise-separable convolution, dilated convolution, max pooling, average pooling and identity. The chromosome of each individual encodes the structure of the block, i.e., four integer bits (the first two encoding the inputs, and the last two the operation) represent a block. The coding of the five blocks constitutes a cell coding. Since two types of cells are used, 40 integers in total are used to represent each neural architecture. More details can be found in Supplementary Material A.

3.3. EMT-NAS

The overall framework of the proposed EMT-NAS is presented in Fig. 3, where an evolutionary algorithm searches for optimal neural architectures for two different tasks at the same time. The main components of EMT-NAS are in the upper panel of Fig. 3, which is composed of randomly generated initial population, offspring generation by means of crossover and mutation according to the parents' skill factor τ (see Section 3.4), and a top K selection for each task as the environmental selection. The full pseudo code and a detailed description of EMT-NAS can be found in Supplementary Material B.

Each task has a set of supernet weights that do not affect each other. Sampled training is a method for continuous updating of the supernet weights on the training set based on weight sharing. For each mini-batch of the training data of each task at one epoch, one individual (representing one candidate architecture) belonging to this task will

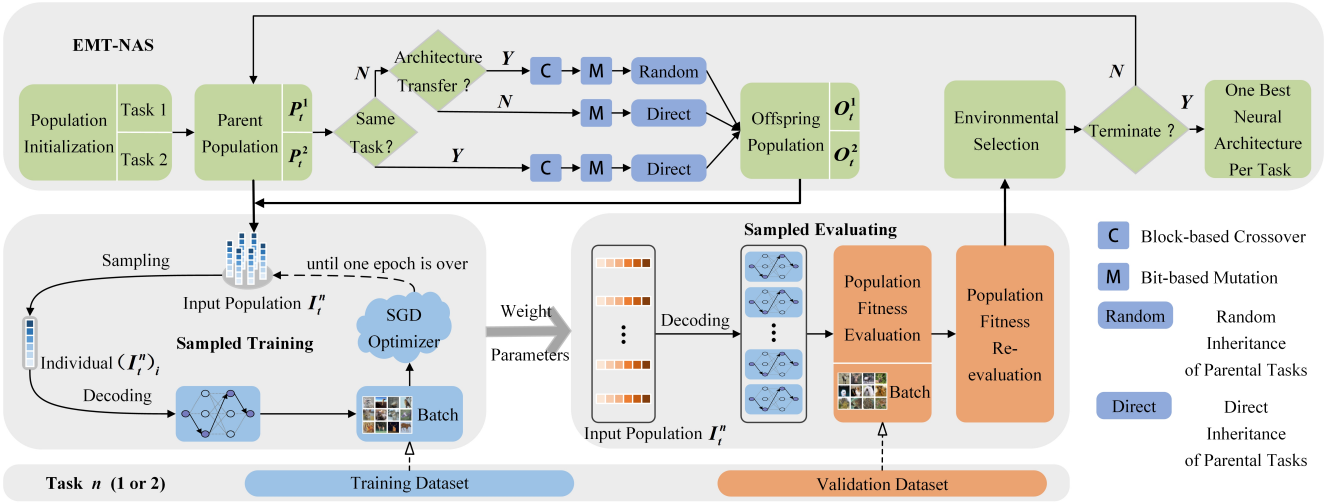

Figure 3. The flow chart of EMT-NAS, where two tasks are used as an example. The difference with the single-tasking evolutionary algorithm is that each individual in the population may belong to the same or different tasks. In addition, each task has a separate set of weights for the supernet. The lower panel shows the sampled training based on a weight sharing method on the training set and the sampled evaluation based on a fitness re-evaluation method on the validation set. Best viewed in colour.

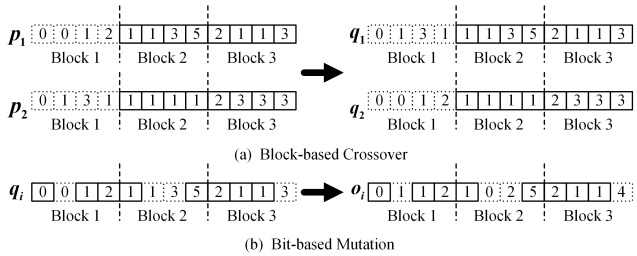

Figure 4. Block-based crossover and bit-based mutation.

be randomly sampled and trained on the mini-batch using the gradient based method. After training, sampled evaluating computes the fitness of the individual by calculating the classification accuracy on the validation data of the same task. Each individual is evaluated once on a random mini-batch of the corresponding task validation dataset. Hence, the computational cost of training the weighting parameters and evaluating the fitness can be significantly reduced.

3.4. Offspring Generation

Transfer of useful knowledge between tasks is the key reason for EMT-NAS to enhance its performance. As in MFEAs, knowledge transfer in EMT-NAS is realised through simultaneously evolving neural architectures for multiple related tasks in one population. More specifically, knowledge exchange across the tasks is achieved by performing crossover on individuals optimizing architectures on different tasks.

In reproduction, two parents will be selected each time at a probability proportional to the individuals' fitness value. That is, the greater the fitness of an individual, the larger chance it will have to produce offspring. If the two selected parents belong to the same task, then crossover will be carried out to generate two offspring, both inheriting the same skill factor τ of their parents. In addition, mutation will also be applied to the two offspring to promote exploration.

When the pair of the selected parent individuals belong to different tasks, two different procedures will be carried out according to a predefined probability, called random mating probability (RMP). If a randomly generated number between 0 and 1 is less than RMP, a crossover operator will be applied to the two parents to generate two offspring, which are then mutated. The two tasks of the parents will be randomly assigned to one of the offspring at an equal probability. At a probability of 1-RMP, only mutations will be performed on each of the two parents to generate two offspring. Consequently, the skill factor τ of each offspring is the same as that of its parent. The probability threshold RMP is a parameter to be specified by the user and a sensitivity analysis of RMP will be provided in the experimental Section 4.6.

Illustrative examples of the crossover and mutation are given in Fig. 4. For clarity, we only show a segment of the parent chromosome consisting of three blocks. There is a strong correlation between the input source and the operation of the connection, which has a great impact on the performance of the network structure [50]. In block-based crossover, the parent chromosomes p_1 and p_2, in blocks, are swapped at less than the crossover probability CP to generate offspring q_1 and q_2. In bit-based mutation, the parent chromosome q_i, in bits, is randomly replaced from the candidate space at less than the mutation probability MP, generating offspring o_i. This is done by changing the input source or operation to improve the performance of the network architecture.

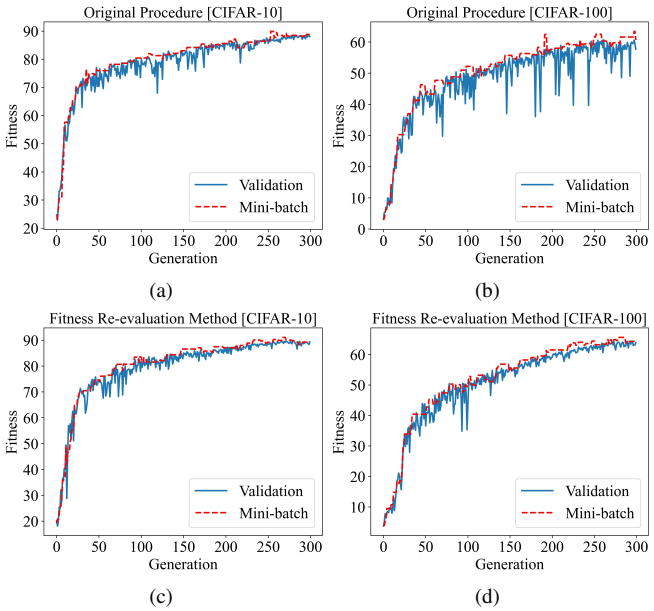

Figure 5. Empirical evaluations of the proposed fitness re-evaluation method on CIFAR-10 and CIFAR-100. In the figures, the red dashed lines denote the estimated fitness based on the sampled evaluation without re-evaluation ((a) and (b)) and with the proposed re-evaluation ((c) and (d)), while the blue solid lines denote the true fitness based on an accurate evaluation using all validation data. From these results, we can see that strong fluctuations in the true fitness of the network have been reduced when the proposed fitness re-evaluation is adopted.

3.5. Fitness Re-evaluation

There are two important reasons why the fitness re-evaluation method was proposed. First, when using the mini-batch gradient descent training, the sign of the gradient resulting from each mini-batch data may be inconsistent. Consequently, the performance of a network estimated using the sampled evaluating may fluctuate strongly over the generations. Second, such fluctuations are made even more serious because of parameter sharing adopted in training.

To examine this issue more closely, we plot the fitness (the validation accuracy) of the best individual from a typical run of evolutionary NAS on CIFAR-10 and CIFAR-100 simultaneously in Figs. 5a and 5b. We can observe that there is a big difference between the estimated and true validation accuracies and the true performance fluctuates strongly over the generations. As a result, promising individual may get lost because of the strong fitness fluctuations and inaccurate sampled evaluations.

To alleviate strong fluctuations in the fitness, we propose to use the fitness of the corresponding parent individual to modify the fitness of the offspring so that the increase or decrease will become less dramatic. Thus, the fitness of an offspring is modified as follows:

$$F = F_p + (F_o - F_p) \times \log_T t \tag{2}$$

where, F_o, F_p, F are the fitness value of the offspring, the fitness of the parent that generates the offspring, and the modified fitness of the offspring; $t = 2, \cdots, T$ is the current generation, and T is the maximum number of generations. We call the modification of the offspring fitness according to Equation (2) fitness re-evaluation.

The logarithmic function in Eq. 2 is based on T, which increases to 1.0 as t increases. Thus, $F_o - F_p$ will be penalised by the logarithm function. Now we re-run the evolutionary NAS on the two tasks (CIFAR-10 and CIFAR-100) by applying the fitness re-evaluation method, and the true fitness as well as the re-evaluated fitness using the sampled validation of the best individual are plotted in Figs. 5c and 5d. From these results, we can see that the individual's true fitness value improves more steadily when the proposed fitness re-evaluation method is adopted, indicating that the modified fitness evaluated based on the sampled evaluating can better reflect the true fitness of the individuals.

4. Experimental Results

4.1. Experimental Settings

To validate the effectiveness of our approach, we set single-tasking (evolutionary single-tasking based NAS) as the baseline. Most of the experiments are conducted on MedMNIST [44], which is a collection of 10 pre-processed medical datasets. We selected the PathMNIST dataset of colon pathology, OrganMNIST_Axial, OragnMNIST_Coronal, and OrganMNIST_Sagittal through different processing methods in abdominal CT along three axes. We use these four tasks to construct multitasks with task numbers of 2, 3 and 4, respectively. We also conduct experiments on CIFAR-10 and CIFAR-100 [19], and then transfer the searched best network architecture to ImageNet [34] to explore its effectiveness. For the sake of simplicity, we denote PathMNIST, OrganMNIST_Axial, OragnMNIST_Coronal, OrganMNIST_Sagittal, CIFAR-10, and CIFAR-100 as Path, Organ_A, Organ_C, Organ_S, C-10, and C-100. Similar to [57], the cross-entropy loss function is used for the six tasks and the parameter settings are presented in Table 1 for the search phase. For fair comparisons, the same parameters as in EMT-NAS are maintained for single tasks during the search phase and retraining phase. All experiments are statistical results of five independent runs with an exception to the results on ImageNet, where all experiments were run once only on 1 Nvidia GeForce RTX 2080Ti due to limited computational resources. More details are given in Supplementary Material C.

4.2. Medical Multi-Tasking

Here, we evaluate the network architectures searched by Single-Tasking and EMT-NAS on the medical multitasking. The results when the number of tasks varies from two to

Table 1. A summary of hyper-parameter settings.

Categories	Parameters	Settings (MedMNIST/CIFAR)
Search space	Initial channels	48/20
	Repetitions of cells	1
Gradient descent	Batch size for training	128
	Batch size for validation	1000
	Weight decay	1.00E-04
	Dropout	0.1
	Momentum	0.9
	Initial learning rate	0.1
	Learning rate schedule	Cosine Annealing
Search strategy	Population size	20
	Generations	100/300
	Random mating probability	0.3
	Crossover probability	1.00
	Mutation probability	0.02

Table 2. Comparison on MedMNIST.

Index	# Tasks	Model	Path (%)	Organ_A (%)	Organ_C (%)	Organ_S (%)	GPU Days (%) ↓
	1	ResNet-18 [11]	84.4	92.1	88.9	76.2	
	1	ResNet-50 [11]	86.4	91.6	89.3	74.6	
	1	Auto-skearn [8]	18.6	56.3	67.6	60.1	
	1	AutoKeras [16]	86.4	92.9	91.5	80.3	
	1	AutoML [56]	81.1	81.8	86.1	70.6	
	1	SI-EvoNAS [50]	90.5±0.7	93.0±0.3	91.8±0.4	80.1±0.3	
	1	Single-Tasking (Baseline)	87.4±2.9	94.0±0.7	91.5±0.8	80.7±0.5	+00.0
①	2	EMT-NAS (Ours)	91.6±0.5	95.1±0.2			-11.0
②	2	EMT-NAS (Ours)	91.2±0.7		92.3±0.3		-18.4
③	2	EMT-NAS (Ours)	90.5±0.9			81.5±0.2	-12.8
④	2	EMT-NAS (Ours)		94.8±0.5	92.2±0.5		-28.4
⑤	2	EMT-NAS (Ours)		95.0±0.3		81.3±0.5	-23.2
⑥	2	EMT-NAS (Ours)			92.2±0.3	81.5±0.2	**-40.1**
⑦	3	EMT-NAS (Ours)		**95.3±0.3**	92.3±0.2	81.7±0.4	-36.0
⑧	3	EMT-NAS (Ours)	91.4±0.3		92.3±0.3	**81.8±0.3**	-29.9
⑨	3	EMT-NAS (Ours)	**91.9±0.2**	95.0±0.2		81.4±0.3	-23.3
⑩	3	EMT-NAS (Ours)	91.7±0.6	95.0±0.4	**92.5±0.2**		-25.1
⑪	4	EMT-NAS (Ours)	91.2±0.6	94.9±0.3	92.2±0.3	81.4±0.5	-31.4

four in comparison to the corresponding single tasks are given in Table 2. In the top rows of the table, the results of the first five results on the four medical datasets taken from [44] are listed. SI-EvoNAS [50] is a state-of-the-art evolutionary NAS algorithm. Single-Tasking is an evolutionary single-tasking algorithm that we have implemented as the baseline. In column *GPU Days (%)*, the reduction in the runtime of EMT-NAS relative to the corresponding Single-Tasking are listed as a percentage.

From Table 2, it is clear that the classification accuracy of the neural architectures found by EMT-NAS is higher than the networks found by all single-tasking NAS on the corresponding tasks, and the search time is reduced by 11%-40% compared to the baseline. This demonstrates that it is possible to improve the performance of each task through architectural knowledge transfer only. In addition, the best classification accuracy of the network architecture has been obtained when the number of tasks is three. This shows that which tasks and how many tasks are formulated as a multi-tasking problem will influence the performance of algorithm in solving each of the individual tasks.

Multi-tasking with the same data modality: EMT-NAS (numbered ④, ⑤, ⑥, ⑦) searched for network architectures with a higher classification accuracy than those found by Single-Tasking. As described in [44], crop 2D images from the center slices of the 3D CT bounding boxes in axial/coronal/sagittal views (planes) are obtained in datasets Organ_A, Organ_C, and Organ_S. Therefore, these datasets are correlated, and it is reasonable to make use of the common knowledge of these datasets to improve classification accuracy. In addition, ⑦ VS ④, ⑤ and ⑥, respectively, have the best performance on Organ_A, Organ_C, and Organ_S. This demonstrates that the three tasks enhance each other to a greater extent than two by two. Therefore, architectural knowledge transfer works well for different datasets of the same data modality.

Multi-tasking with different data modalities: The classification accuracy of EMT-NAS (numbered ①, ②, ③, ⑧, ⑨ and ⑩) are slightly higher than that of EMT-NAS (numbered ④, ⑤ and ⑥) on Organ_A, Organ_C, and Organ_S datasets, respectively. From [44], we know that Path is for predicting survival from colorectal cancer histology slides, which means that Organt_A, Organ_C, and Organ_S can benefit more from Path. By contrast, EMT-NAS has transferred different degrees of useful knowledge from Organ_A, Organ_C, and Organ_S, respectively, to Path. From the table, we can observe that the classification accuracy of EMT-NAS (numbered ①, ② and ③) decreases sequentially on PATH. Organ_A, Organ_C, and Organ_S are obtained by different processing methods in abdominal CT along three axes, so their data characteristics are different. Clearly, Path and Organ_A are most beneficial to each other. This also corroborates with the task relational analysis in Fig. 2. Therefore, although these datasets are from different data modalities and are weakly correlated, they can still improve their respective performance through architectural knowledge transfer.

4.3. CIFAR-10 and CIFAR-100

We again used RSA in Fig. 2 to analyse the task relatedness between C-10 and C-100, with a task relatedness score of 0.38. C-100 is made up of 20 classes (each containing five subclasses), while C-10 is 10 classes. The results show that ResNet-50, trained on C-100, extracts many different features from the same image compared to those trained on C-10. Therefore, C-10 and C-100 are differentiated.

The representative algorithms can be roughly divided into four categories, namely manually designed, RL-based, GD-based, and EAs-based. The comparative results on C-10 and C-100 are listed in Table 3. Note that all algorithms are meant for single-tasking search, i.e., they either search for an optimal network architecture on C-10 and C-100, respectively, or search on C-10 and then the found architectures are transferred to C-100. By contrast, EMT-NAS searches for optimal neural architectures simultaneously on C-10 and C-100, and therefore, the GPU days of EMT-NAS is the total time for searching for both architectures.

In Table 3, EMT-NAS stacks normal blocks twice, while EMT-NAS† stacks three times. EMT-NAS searched the network architecture's on C-10 and C-100 with a higher clas-

Table 3. Comparison on C-10 and C-100. $*$ indicates that the same network architecture is used on both C-10 and C-100, so their GPU days for searching are the same. † indicates normal cell stacking three times for better performance.

Model	Search Method	GPU Days	Params (M)	C-10 (%)	Params (M)	C-100 (%)
Wide ResNet [47]	manual	–	36.5	95.83	$*$	79.50
DenseNet [14]	manual	–	27.2	96.26	$*$	80.75
MobileNetV2 [35]	manual	–	2.1	94.56	$*$	77.09
PNAS [23]	SMBO	150	3.2	96.59±0.09	$*$	80.47
ENAS [28]	RL	0.45	4.6	97.11	$*$	80.57
NASNet-A [57]	RL	2000	3.3	**97.35**	–	–
Block-QNN-Faster [53]	RL	0.8	3.9	96.43	$*$	81.79
DARTS [25]	GD	1.5	3.3	97.00±0.14	$*$	79.48±0.31
SNAS [41]	GD	1.5	2.9	97.02	–	–
large-scale Evo [31]	EA	2750	5.4	94.6	40.4	77
Hier-EA [24]	EA	300	64	96.25±0.12	–	–
Amoebanet-A [30]	EA	3150	3.2	96.66±0.06	$*$	81.07
AE-CNN [39]	EA	27/36	2	95.7	5.4	79.15
SHEDA-CNN [20]	EA	0.58/0.97	10.88	96.36	18.64	78.84
Single-Tasking (Baseline)	EA	0.46	1.83	96.24±0.20	1.68	79.70±0.44
Single-Tasking† (Baseline)	EA	0.46	2.41	96.64±0.18	2.19	80.82±0.71
EMT-NAS (Ours)	EA	0.42	2.17	96.73±0.15	2.23	81.86±0.10
EMT-NAS† (Ours)	EA	0.42	2.91	97.04±0.04	2.97	**82.60±0.38**

sification accuracy than Single-Tasking (baseline), demonstrating that architecture knowledge transfer can effectively improve performance on C-10 and C-100. Compared to other NAS algorithms, our approach achieves better performance and more compact network architectures searched on C-10 and C-100 through architectural knowledge transfer, except for ENAS and NASNet-A on C-10. However, the models found by EMT-NAS have only two-thirds the number of parameters of models found by ENAS. On the other hand, NASNet-A consumes about 4,700 times more GPU days than EMT-NAS. These indicate that our approach can achieve a good performance-complexity trade-off. We also analysed two baselines for joint training (shared or unshared network architectures) in Supplementary Material D. The performance of the algorithms has degraded as data from different tasks are used to update the same supernet.

4.4. Transferability to ImageNet

As done in [25], we transferred the best network architectures found on C-10 and C-100 to ImageNet for training from scratch, to further validate the transferability of the architectures searched by EMT-NAS. The results comparing EMT-NAS with 15 models (or their variants) on ImageNet are given in Table 4. EMT-NAS-4-C-100 has a lower number of model parameters (2.49M) than all compared algorithms, while it has a higher top-1 accuracy (74.25%) except for MoblieNet-V3, MobileNetV2-1.0+CA, RepVGG-A1, AutoSpace, PC-DARTS and MUXNet-m. For better performance, we change the number of normal cells stacked from two to three to get EMT-NAS-C-100†. The top-1 accuracy of EMT-NAS-C-100† is 75.44%, which again surpasses the accuracy of the rest of the algorithms, and the number of model parameters is 3.25M. This demonstrates that EMT-NAS has good transferability between the network architectures searched on C-10 and C-100.

Table 4. Comparisons on ImageNet. † indicates normal cell stacking three times for better performance.

Model	Top-1 (%)	Top-5 (%)	Params (M)	GPU Days	Search Method
ShuffleNet 2.0 [51]	73.7	–	5.4	–	manual
MoblieNet-V3 [13]	75.2	–	5.1	–	manual
MobileNeXt-0.75 [54]	72.6	–	2.5	–	manual
MobileNetV2-1.0+CA [12]	74.3	–	3.95	–	manual
RepVGG-A1 [5]	74.46	–	12.78	–	manual
PNAS [23]	74.2	91.9	5.1	225	SMBO
NASNet-A [57]	74.0	91.6	5.3	2000	RL
DPP-Net-Panacea [6]	74.2	91.79	4.8	8	RL
AutoSpace [55]	75.3	–	4.3	8.33	RL
DARTS [25]	73.3	91.3	4.7	4	GD
SNAS [41]	72.7	91.8	4.3	1.5	GD
PC-DARTS [43]	74.9	92.2	5.3	0.1	GD
HourNAS-C [45]	74.1	91.6	4.8	0.1	GD
Amoebanet-A [30]	74.5	92.0	5.1	3150	EA
MUXNet-m [26]	75.3	92.5	3.4	176	EA
Single-Tasking-C-10 (Baseline)	73.18	91.22	2.41	0.23	EA
Single-Tasking-C-100 (Baseline)	72.88	91.06	2.25	0.22	EA
Single-Tasking-C-10† (Baseline)	74.59	92.09	3.15	0.23	EA
Single-Tasking-C-100† (Baseline)	74.11	91.86	2.97	0.22	EA
EMT-NAS-C-10 (Ours)	73.83	91.63	2.38	0.42	EA
EMT-NAS-C-100 (Ours)	74.25	91.80	2.49		
EMT-NAS-C-10† (Ours)	75.11	92.39	3.13	0.42	EA
EMT-NAS-C-100† (Ours)	**75.44**	**92.55**	3.25		

4.5. Ablation Studies

To verify the effectiveness of the evolutionary multi-tasking NAS algorithm and the fitness re-evaluation method, four algorithms are compared, namely evolutionary single-Tasking NAS, which serves as the baseline (denoted as Single-tasking in Table 5), evolutionary single-tasking NAS with the fitness re-evaluation method, called Single-Tasking+Reeval, evolutionary multi-tasking NAS without fitness re-evaluation, called Multi-Tasking, and evolutionary multi-tasking NAS with the fitness re-evaluation method, denoted as Multi-Tasking+Reeval.

Evolutionary Multi-Tasking: By comparing the results of Single-tasking and Multi-Tasking (③ vs ① and ④ vs ②), evolutionary multi-tasking NAS can improve the performance on both C-10 and C-100, confirming the effectiveness of knowledge transfer when simultaneously searching for the optimal neural architectures on similar tasks.

Fitness Re-Evaluation: By comparing the results with and without fitness re-evaluation (④ vs ③ and ② vs ①), we can see that the proposed fitness re-evaluation method improves the performance on both C-10 and C-100. This confirms that the fitness re-evaluation method is effective in enhancing the performance by alleviating fitness fluctuations in the search based on sampled evaluations.

Search Efficiency: We take the total runtime of Single-Tasking, and count the percentage increase or decrease of the other three algorithms relative to it. We also calculate three parts of the runtime, including time for training model parameters on the training dataset, time for evaluating model performance on the validation dataset, and time for searching. First, we note that the time taken by the evolutionary multi-tasking is reduced by 50% compared to single-tasking(③ vs ① and ④ vs ②). Because the compu-

Table 5. Comparison of evolutionary single- and multi-tasking optimisation with/without fitness re-evaluation on CIFAR-10 and -100.

Index	Algorithm	Task 1			Task 2			Total Time (%) ↓	Training Time (%) ↓	Evaluation Time (%) ↓	Search Time (%) ↓	Average Params (M)	Average FLOPs (M)
		C-10 (%)	Params (M)	Transfer (%)	C-100 (%)	Params (M)	Transfer (%)						
①	Single-Tasking	96.23±0.20	1.83	96.07±0.25	79.70±0.44	1.68	78.95±0.60	+0.0	+0.0	+0.0	+0.0	1.83	334
②	Single-Tasking+Reeval	96.29±0.25	1.74	96.26±0.18	80.48±0.28	1.90	78.77±0.65	-5.1	**-4.1**	-15.1	-1.4	1.81	323
③	Multi-Tasking	96.49±0.16	2.02	–	80.68±0.43	2.03	–	-2.1	+3.8	**-37.4**	-49.4	2.06	382
④	Multi-Tasking+Reeval	**96.73±0.15**	2.17	–	**81.86±0.10**	2.23	–	**-8.7**	-4.0	-35.9	**-51.6**	1.97	355

tational complexity of the Evolutionary Multi-Tasking and Single-Tasking algorithms is almost the same, except for the time spent on training and evaluation. Second, ③ vs ① and ④ vs ②, the time spent on the evaluation is reduced by about 36%. The population size and the number of generations are the same for single and multi-tasking, i.e., the number of fitness evaluations is the same, so that the total time to optimise multiple tasks simultaneously will be less than the sum of the runtimes when the tasks are optimised one by one. Finally, ④ vs ③ and ② vs ① with a small percentage reduction in the training time. To investigate this further, we reproduce the sampled training process and chose two model metrics, namely the number of parameters and the number of floating point operations (FLOPs). Specifically, when each network architecture was trained once, two metrics were recorded until five runs of the algorithm have stopped, and then averaged and listed in the column *Average Params (M)* and *Average FLOPs (M)* of Table 5, respectively. The results show a small reduction in the average number of parameters and the average FLOPs for both ④ vs ③ and ② vs ①. It is assumed that the fitness re-evaluation method allows the algorithm to favour network architectures consisting of operations with a small number of parameters or a low computational complexity in the search process. This may explain the fact that introducing the fitness re-evaluation method can reduce the time spent by the model on the training set by a small amount. Overall, EMT-NAS consumes less runtime than its single-task counterparts.

Model Transferability: We transferred the network architectures that Single-tasking and Single-tasking+Reeval searched on C-10 and C-100, respectively, to C-100 and C-10 for retraining. The results are presented in column *Transfer (%)*, which show that through transfer learning, the performance of the network architectures on the new dataset is much lower than that of the network architectures obtained by evolutionary multi-tasking NAS (Multi-Tasking and Multi-Tasking+Reeval) search. Therefore, evolutionary multi-tasking NAS is more competitive than evolutionary single-tasking NAS with transfer learning.

4.6. Sensitivity Analysis

In EMT-NAS, RMP is a key parameter that controls the degree of knowledge transfer between tasks. Here, we set the random mating probability $RMP = 0, 0.1, 0.2, 0.3, 0.4, 0.5$ and note that when $RMP = 0$, two single tasks run in the same population without any knowledge transfer between the tasks. Table 6 shows that the validation and test accuracies increase as RMP increases in the beginning, and then decrease as RMP becomes larger, indicating that too much or too little knowledge transfer between tasks can degrade the performance of EMT-NAS. The best performance was achieved on the validation set and test set when $RMP = 0.3$.

Table 6. Comparison of EMT-NAS when random mating probability is set to 0, 0.1, 0.2, 0.3, 0.4, and 0.5.

RMP	Task 1	Validation ACC (%)	Test ACC (%)	Task 2	Validation ACC (%)	Test ACC (%)
0	C-10	88.36±0.69	96.28±0.09	C-100	61.48±1.65	80.31±0.40
0.1	C-10	87.36±1.60	96.48±0.18	C-100	58.85±4.10	80.46±0.71
0.2	C-10	88.32±1.02	96.52±0.18	C-100	59.67±0.47	81.11±0.54
0.3	C-10	**88.76±1.05**	**96.73±0.15**	C-100	**61.69±2.15**	**81.86±0.10**
0.4	C-10	88.09±1.25	96.49±0.18	C-100	61.27±3.09	81.26±0.27
0.5	C-10	88.52±1.16	96.52±0.17	C-100	61.42±2.57	81.15±0.31

We also analysed the influence of the population size, crossover and mutation probabilities, and the number of generations on the performance. The results are presented in Supplementary Material E.

5. Conclusion

In this work, evolutionary multi-tasking is applied to NAS for the first time. Extensive experiments have demonstrated that for multiple classification tasks from different datasets (data modalities), architectural knowledge transfer and fitness re-evaluation method allow EMT-NAS to outperform each task obtained simultaneously over a single task, and that the runtime of EMT-NAS is less than the sum of the runtimes of solving multiple single tasks independently. Although this work achieves implicit architectural knowledge transfer, more precise, explicit knowledge transfer between tasks in evolutionary multi-tasking NAS is desirable and will be investigated in the future.

Acknowledgment

This work is supported by National Natural Science Foundation of China (62136003, 61988101), the Programme of Introducing Talents of Discipline to Universities (the 111 Project) under Grant B17017, Fundamental Research Funds for the Central Universities (222202317006) and Shanghai AI Lab. Y. Jin is supported by an Alexander von Humboldt Professorship for AI endowed by the German Federal Ministry of Education and Research.

References

[1] Shai Ben-David and Reba Schuller Borbely. A notion of task relatedness yielding provable multiple-task learning guarantees. *Machine learning*, 73(3):273–287, 2008. 2

[2] Yassine Benyahia, Kaicheng Yu, Kamil Bennani Smires, Martin Jaggi, Anthony C Davison, Mathieu Salzmann, and Claudiu Musat. Overcoming multi-model forgetting. In *International Conference on Machine Learning*, pages 594–603. PMLR, 2019. 3

[3] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 1, 2

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2

[5] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13733–13742, 2021. 7

[6] Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 517–531, 2018. 3, 7

[7] Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12387–12396, 2019. 2

[8] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. *Advances in neural information processing systems*, 28, 2015. 1, 6

[9] Abhishek Gupta, Yew-Soon Ong, and Liang Feng. Multifactorial evolution: Toward evolutionary multitasking. *IEEE Transactions on Evolutionary Computation*, 20(3):343–357, 2015. 2

[10] Chunlin He, Yong Zhang, Dunwei Gong, Xianfang Song, and Xiaoyan Sun. A multi-task bee colony band selection algorithm with variable-size clustering for hyperspectral images. *IEEE Transactions on Evolutionary Computation*, 2022. 3

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 1, 6

[12] Qibin Hou, Daquan Zhou, and Jiashi Feng. Coordinate attention for efficient mobile network design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13713–13722, 2021. 7

[13] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019. 7

[14] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 1, 7

[15] Minbin Huang, Zhijian Huang, Changlin Li, Xin Chen, Hang Xu, Zhenguo Li, and Xiaodan Liang. Arch-graph: Acyclic architecture relation predictor for task-transferable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11881–11891, 2022. 3

[16] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1946–1956, 2019. 1, 6

[17] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018. 2

[18] Apoorv Khattar, Srinidhi Hegde, and Ramya Hebbalaguppe. Cross-domain multi-task learning for object detection and saliency estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3639–3648, 2021. 2

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 5

[20] Jian-Yu Li, Zhi-Hui Zhan, Jin Xu, Sam Kwong, and Jun Zhang. Surrogate-assisted hybrid-model estimation of distribution algorithm for mixed-variable hyperparameters optimization in convolutional neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. 3, 7

[21] Peng Liao, Chaoli Sun, Guochen Zhang, and Yaochu Jin. Multi-surrogate multi-tasking optimization of expensive Problems. *Knowledge-Based Systems*, 205:106262, 2020. 3

[22] Bingyan Liu, Yifeng Cai, Yao Guo, and Xiangqun Chen. Transtailor: Pruning the pre-trained model for improved transfer learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 8627–8634, 2021. 3

[23] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018. 1, 7

[24] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. In *International Conference on Learning Representations*, 2018. 3, 7

[25] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable Architecture Search. In *International Conference on Learning Representations*, 2018. 3, 7

[26] Zhichao Lu, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Muxconv: Information multiplexing in convolutional neural networks. In *Proceedings of the IEEE/CVF Conference*

on Computer Vision and Pattern Recognition (CVPR), pages 12044–12053, 2020. 3, 7

[27] Alan Tan Wei Min, Yew-Soon Ong, Abhishek Gupta, and Chi-Keong Goh. Multiproblem surrogates: Transfer evolutionary multiobjective optimization of computationally expensive problems. *IEEE Transactions on Evolutionary Computation*, 23(1):15–28, 2017. 3

[28] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018. 3, 7

[29] Kangjia Qiao, Kunjie Yu, Boyang Qu, Jing Liang, Hui Song, and Caitong Yue. An evolutionary multitasking optimization framework for constrained multiobjective pptimization problems. *IEEE Transactions on Evolutionary Computation*, 26(2):263–277, 2022. 3

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4780–4789, 2019. 3, 7

[31] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017. 3, 7

[32] Zhongzheng Ren and Yong Jae Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 762–771, 2018. 2

[33] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 2

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[35] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4510–4520, 2018. 1, 7

[36] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *Advances in neural information processing systems*, 31, 2018. 2

[37] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 2

[38] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020. 2

[39] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. Completely automated cnn architecture design based on blocks. *IEEE Transactions on Neural Networks and Learning Systems*, 31(4):1242–1254, 2019. 3, 7

[40] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021. 2

[41] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. Snas: Stochastic neural architecture search. In *International Conference on Learning Representations*, 2018. 3, 7

[42] Jin Xu, Xu Tan, Kaitao Song, Renqian Luo, Yichong Leng, Tao Qin, Tie-Yan Liu, and Jian Li. Analyzing and mitigating interference in neural architecture search. In *International Conference on Machine Learning*, pages 24646–24662. PMLR, 2022. 3

[43] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. Pc-darts: Partial channel connections for memory-efficient architecture search. In *International Conference on Learning Representations*, 2019. 3, 7

[44] Jiancheng Yang, Rui Shi, and Bingbing Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195. IEEE, 2021. 2, 5, 6

[45] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Jianyuan Guo, Wei Zhang, Chao Xu, Chunjing Xu, Dacheng Tao, and Chang Xu. Hournas: Extremely fast neural architecture search through an hourglass lens. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10896–10906, 2021. 3, 7

[46] Feiyang Ye, Baijiong Lin, Zhixiong Yue, Pengxin Guo, Qiao Xiao, and Yu Zhang. Multi-objective meta learning. *Advances in Neural Information Processing Systems*, 34:21338–21351, 2021. 2

[47] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016. 7

[48] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018. 2

[49] Fangfang Zhang, Yi Mei, Su Nguyen, Mengjie Zhang, and Kay Chen Tan. Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, 25(4):651–665, 2021. 3

[50] Haoyu Zhang, Yaochu Jin, Ran Cheng, and Kuangrong Hao. Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance. *IEEE Transactions on Evolutionary Computation*, 25(2):371–385, 2020. 3, 4, 6

[51] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018. 1, 7

[52] Yu Zhang and Qiang Yang. An overview of multi-task learning. *National Science Review*, 5(1):30–43, 2018. 1, 2

[53] Zhao Zhong, Zichen Yang, Boyang Deng, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Blockqnn: Efficient

block-wise neural network architecture generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(7):2314–2328, 2020. 3, 7

[54] Daquan Zhou, Qibin Hou, Yunpeng Chen, Jiashi Feng, and Shuicheng Yan. Rethinking bottleneck structure for efficient mobile network design. In *European Conference on Computer Vision*, pages 680–697. Springer, 2020. 7

[55] Daquan Zhou, Xiaojie Jin, Xiaochen Lian, Linjie Yang, Yujing Xue, Qibin Hou, and Jiashi Feng. Autospace: Neural architecture search with less human interference. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 337–346, 2021. 7

[56] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc Le. Automl for large scale image classification and object detection. *Google AI Blog*, 2:2017, 2017. 1, 6

[57] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, 2018. 3, 5, 7