ESSR: Evolving Sparse Sharing Representation for Multi-task Learning

Yayu Zhang, *Student Member, IEEE,* Yuhua Qian*, *Member, IEEE,* Guoshuai Ma, Xinyan Liang, Guoqing Liu, Qingfu Zhang, *Fellow, IEEE* and Ke Tang, *Fellow, IEEE*

***Abstract*—Multi-task learning uses knowledge transfer among tasks to improve the generalization performance of all tasks. For deep multi-task learning, knowledge transfer is often implemented via sharing all hidden features of tasks. A major shortcoming is that it can lead to negative knowledge transfer across tasks when task correlation is weak. To overcome it, this paper proposes an evolutionary method to learn sparse sharing representations adaptively. By embedding the neural network optimization into evolutionary multitasking, our proposed method finds an optimal combination of tasks and sharing features. It can identify negative correlation and redundant features and then remove them from the hidden feature set. Thus, an optimal sparse sharing subnetwork can be produced for each task. Experiment results show that the proposed method achieve better learning performance with a smaller inference model than other related methods.**

***Index Terms*—Multi-task learning, evolutionary multitasking optimization, knowledge transfer, sharing representation.**

I. Introduction

To date, scientists have developed many machine learning techniques and methods to deal with various tasks [1]–[4], such as clustering, classification, regression, and more. Although these learning methods have made great success in many areas, they are mostly driven by single tasks or datasets [5]–[8]. It limits the adaptability of machine learning models to different learning tasks. For example, the model of image recognition has unsatisfactory performance in dealing with speaker recognition task. It means that we need to design new learning algorithms for changing tasks, which is time-consuming and laborious. To overcome it, researchers have begun exploring more general learning paradigms, which can adaptively cope with diverse learning tasks. To this end, several machine learning methods have been proposed, including transfer learning [9], automatic machine learning [10], [11], meta-learning [12], multi-task learning [13] and so on.

Multi-task Learning (MTL) deals with multiple related tasks at the same time, and it aims to improve the performance of each task. MTL improves generalization performance by using the domain information contained in the training signals of related tasks as an inductive bias, so it is also considered as an inductive transfer mechanism [14]. As a general learning paradigm, many learning paradigms such as semi-supervised and unsupervised learning [15], reinforcement learning [16], multi-modal learning [17]–[19], active learning [20] can combine with multi-task learning to improve their performance.

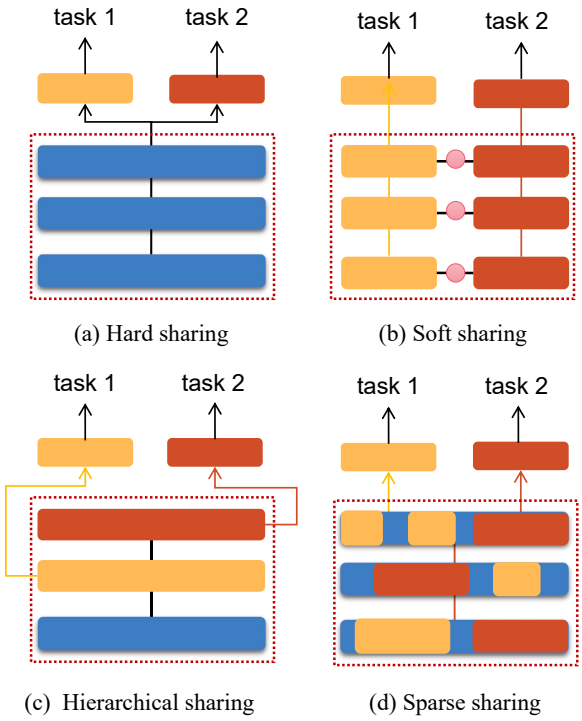


Fig. 1: The different sharing architecture of multi-task learning. The layers inside the red dotted box are task-sharing layers. The layers at the top of the model are task-specific layers. The blue block represents the tasks sharing parameters; The yellow and red blocks are the task-specific parameters; The solid pink circle represents the sharing mechanism of the model. (a) Given a learning model, tasks share all the underlying parameters. (b) Task 1 and Task 2 are given a separate model in advance, and the tasks share parameters at the same level through some mechanism. (c) Given a learning model, the sharing layers are divided into different tasks. (d) The parameters of sharing layers are divided into three categories: task 1, task 2, and the shared by task 1 and task 2.

*: Corresponding author.

Y. Zhang, Y. Qian, G. Ma, X. Liang and G. Liu are with the Institute of Big Data Science and Industry, Shanxi University, Taiyuan 030006, Shanxi, China; and also with the School of Computer and Information Technology, Shanxi University, Taiyuan, Shanxi 030006, China. Y. Qian also is with the Key Laboratory of Computational Intelligence and Chinese Information Processing of Ministry of Education, Taiyuan, 030006 Shanxi, China. G. Ma also is with School of Computer Science and Technology, North University of China, Taiyuan, Shanxi, 030051, China. (e-mail: zhang_yayu93@126.com and jinchengqyh@126.com).

Q. Zhang is with the Department of Computer Science, City University of Hong Kong, Hong Kong, and also with the Shenzhen Research Institute, City University of Hong Kong, Shenzhen 518057, China (e-mail: qingfu.zhang@cityu.edu.hk).

K. Tang is with Shenzhen Key Laboratory of Computational Intelligence, the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China (e-mail:tangk3@sustech.edu.cn)

Manuscript received **; revised **.

With the development of deep learning, deep multi-task learning (DMTL) has attracted much research attention. It has been applied in various areas such as computer vision [21], natural language processing [22], and speech recognition [23] with success. For neural networks, a commonly used multi-task learning approach is to design a parameterized model that shares a subset of parameters across different tasks [24]. As shown in Fig. 1(a), the model bottom layers are shared by all tasks, and the top layers are task specific [25]–[27]. Then a linearly weighted objective function can be optimized to determine the model parameters. In such a modeling approach, all tasks share the same latent features, and each sharing layer hidden units can be viewed as the common feature mapping or representation learned for the tasks [28]. Knowledge transfer across tasks is realized via shared parameters during learning. The sharing layers transform the input data from the original feature space to the learned feature representation.

However, in practice, the relevance among tasks can be very loose, or only a few tasks are correlated with each other. Thus the learned features may contain some negative correlation and redundant components [29]. Sharing all of them will transfer some destructive information across tasks and then lead to "negative knowledge transfer". This phenomenon, also known as "task interference" [30], [31], sometimes worsens the performance of multi-task learning. Therefore, the general modeling and learning method as shown Fig. 1(a) may not be always effective. To alleviate this issue, much effort has been made to study how to share features among tasks. Existing methods can be roughly classified into three categories: soft sharing, hierarchical sharing, and sparse sharing.

In soft sharing, each task is given a separate network in advance, and tasks realize the interaction of information through inter-access [32], [33] as shown Fig. 1(b). However, the growth of tasks is accompanied by the growth of the model's parameters, so such methods cannot be well extended to address many tasks learning problems. In hierarchical sharing, parameters are shared between corresponding layers of task-specific network [34], [35] as shown Fig. 1(c), which usually need prior knowledge to design the hierarchical sharing architecture. In recent years, the method of automatic identification task-sharing layers has been proposed, but the number of modules in each layer also needs to be given in advance [36]. The methods of sparse sharing, as shown in Fig. 1(d), divide the sharing layer's parameters according to tasks. Considering deep learning models often have millions or billions of parameters [37], [38], which causes a vast search space for finding the best combination of tasks and parameters. Existing sparse sharing methods improve learning efficiency and performance by dividing parameters randomly [30], [39]–[41]. However, the strategy of random division has the following limitation:

1) Random parameters division leads to the uncertainty of experimental results [40], [42].
2) The random dividing method based on dropout results in the inconsistencies between the training model and the inference model [30].
3) It cannot identify and reduce redundant information in the model [30], [40], [43].

As argued above, the existing methods of avoiding negative knowledge transfer by parameter assignment have disadvantages such as weak scalability, prior dependence, randomness, etc. In the paper, a multi-task information sharing method—evolving sparse sharing representation (ESSR) is proposed; it learns a low-dimensional and sparse sharing representation for each task adaptively and avoids the negative knowledge transfer. To ensure valuable knowledge transfer and improve the generalization performance of each task, ESSR seeks the combination of tasks and sharing features adaptively. It can also be viewed as a feature selection method, which removes the negative correlation and redundant features from all latent features [44], [45]. Specifically, for a base neural network, as shown in Fig. 1(a), the algorithm adaptively decouples a specific sparse sharing subnetwork for each task by evolution. Each subnetwork corresponds to a "main task", and other tasks as "auxiliary tasks" improve the performance of the main task by transferring knowledge on selected parameters. Inspired by evolutionary multitasking [46], the learning of task-specific subnetwork is modeled as a multitasking optimization problem, which is solved by a multifactorial evolutionary algorithm with a single population. Finally, the best subnetworks of multiple tasks are obtained simultaneously.

The main contributions of the paper are as follows:

- A knowledge sharing method ESSR is proposed to learn the sparse sharing representation of tasks by adaptively seeking the optimal combination of tasks and sharing features. The sharing representation of each task corresponds to a subnetwork which is obtained by decoupling a given base network.
- The learning of task representation (subnetworks) is formalized as an optimization problem, and an evolutionary multitasking optimization algorithm is introduced to find the best representation of all tasks simultaneously and adaptively.

The proposed method in this paper has the following advantages:

- ESSR reduces both the negative correlation and redundant features of tasks, which can avoid task interference caused by knowledge negative transfer in multi-task learning.
- ESSR gets a small subnetwork for each task adaptively by evolution, which reduces the uncertainty caused by random masking.

The remainder of this paper is as follows. Section II introduces the related work of this paper. Section III presents the proposed method ESSR. Section IV proves the effectiveness of the proposed method through experiments. Section V summarizes the work in this paper.

II. Related work

In this section, the definition of MTL is introduced first. Then the previous works that solve the task interference in DMTL are reviewed. Finally, a series of existing multitasking optimization studies are presented.

A. The Definition of Multi-Task Learning

***Definition** 1 (Multi-Task Learning):* Given a multi-task problem consisting of T learning tasks $\{\mathcal{T}_t\}_{t=1}^{T}$, where all the tasks or a subset of them are relevant. Multi-task learning aims to improve the learning performance of a model for $\mathcal{T}_t$ by using the knowledge contained in all or some of the T tasks [47].

Based on the definition of MTL, we let T tasks to be learned. A task $\mathcal{T}_t$ is accompanied by a training dataset $\mathcal{D}_t = \{(x_i^t, y_i^t)\}_{i=1}^{N_t}$, where N_t is the number of samples. For a given training dataset, MTL is learning a mapping from input space $\mathcal{X}^t$ to the target space $\mathcal{Y}^t$. A parameter hypothesis class for each task can be denoted as $\hat{y}_i^t = f^t(\varepsilon(x_i^t, \theta^{sh}), \theta^t) : \mathcal{X}^t \longrightarrow \mathcal{Y}^t$, where ε refers to the task-sharing layers of a neural network, θ^{sh} refers to the parameters of shared across tasks and θ^t is task-specific parameters. The task-specific loss function is $L_t(\theta^{sh}, \theta^t) = \frac{1}{N_t}\sum_i^{N_t} l(\hat{f}_i^t, y_i^t)$.

The naive method for solving multi-task learning is linear scalarization, which simply performs a linearly weighted sum of the losses for each individual task. The objective function is generally described in the following empirical risk minimization formulation:

$$\min_{\substack{\theta^{sh} \\ \theta^1,\dots\theta^T}} \sum_{t=1}^{T} w_t L_t(\theta^{sh}, \theta^t), \tag{1}$$

w_t is the weight of the t-th task. This method is often affected by the weight w and tends to a certain task as shown in Fig. 2(b) [48].

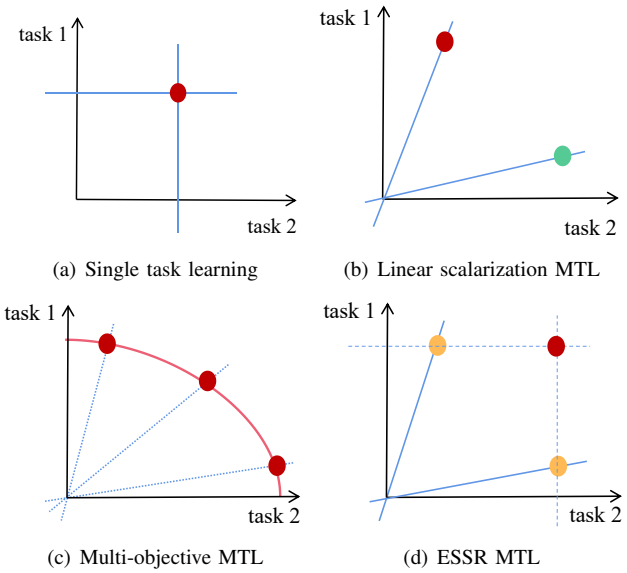
B. The Related Works of Alleviate the "Task Interference" of DMTL

After giving the definition and some symbols of MTL, we will review the research on solving the task interference in DMTL. These studies are roughly divided into the following three categories:

*1) **Adaptive Loss Weighting**:* Such methods aim at solving the task imbalance problem caused by the weight of loss function during learning.

Studies have shown that the performance of multi-task learner often depends on the relative weighting w_t of the task losses [49]. However, it is difficult to adjust these weights artificially. Therefore, a series of works have studied and discussed how to adjust the weight of task loss adaptively. Kendall et al. (2018) proposed a multi-task deep learning method in which the multiple losses are weighted by considering the homoscedastic uncertainty of each task. Chen et al. (2018) [50] proposed an adaptive loss weighting method—GradNorm, and it makes the weight w_i adaptively change with the training steps, i.e., $w_i = w_i(t)$. Inspired by GradNorm, Liu et al. (2019) [51] proposed a simple and effective adaptive weighting method—dynamic weight average (DWA).

*2) **Sharing Architecture Learning**:* These methods avoid the negative knowledge transfer in DMTL by parameter partition. It means that sub-network of task is decoupled from the given base network. The knowledge is transferred among the shared parameters.



(a) Single task learning (b) Linear scalarization MTL

(c) Multi-objective MTL (d) ESSR MTL

Fig. 2: Explanation of the solutions obtained by different methods. Take two tasks as examples, and the red dot or green dot in the figure is the final solution. (a) The solution of single-task learning. (b) The solution of linear scalarization multi-task learning. (c) The trade-off solution of multi-objective multi-task learning. (d) The solution of the ESSR method.

The previous studies can be divided into three categories: parameters-level, filters-level, and layers-level. Sun et al. (2019) [43] divided the parameters of the base neural network into different tasks. Bragman (2019) [52], Strahovski (2019) [40] and Pascal (2020) [30] divided filters into different tasks, and task-special subnetwork is obtained. Prellberg (2020) [36] and sun (2020) [41] proposed the partition method based on layer-level, in which the assignment of sharing layers and task-specific layers were given.

*3) **Multi-objective Multitasking**:* These methods aim at finding the Pareto optimal solutions with good trade-off among different tasks by casting multi-task learning as a multi-objective optimization (as shown in Fig. 2(c)).

Sener and Koltum (2018) [53] first focused on the conflict among tasks and regarded multi-task learning as a multi-objective optimization problem. Different from linear scalarization, the multi-objective optimization formulation of MTL uses a vector-valued loss:

$$\begin{aligned} &\min_{\substack{\theta^{sh} \\ \theta^1,\dots\theta^T}} L(\theta^{sh}, \theta^1, ..., \theta^T) \\ &= \min_{\substack{\theta^{sh} \\ \theta^1,\dots\theta^T}} (L_1(\theta^{sh}, \theta^1), ..., L_T(\theta^{sh}, \theta^T))^T. \end{aligned} \tag{2}$$

However, this method also finds one Pareto optimal solution across different tasks for a MTL problem instead of "Pareto optimality sets" with task preference.

Lin et al. (2018) [48] proposed Pareto multi-task learning (Pareto MTL) to find a set of well-distributed solutions that can represent different trade-offs across tasks. The main idea of Pareto MTL is to decompose a MTL problem into several constrained multi-objective subproblems with different trade-offs preferences across the tasks in the original MTL. The paper also shows that the Pareto MTL can be reformulated as a linear scalarization of tasks with adaptive loss weighting.

Subsequently, Mahapatra and Rajan (2020) [54] proposed a MOO optimization method — exact Pareto optimal (EPO) that can converge to the desired ray in loss space. Ma et al. (2020) [55] presented a novel and efficient method that generates locally continuous Pareto sets and Pareto fronts. It provides the possibility for continuous analysis of Pareto optimal solutions in machine learning problems. Li et al. (2021) [56] proposed a novel controllable Pareto multi-task learning framework, which enables the system to make real-time trade-off control across different tasks with a single model.

These studies focus on multi-task learning problems with task conflict, which can be considered as a particular aspect of multi-task learning research.

C. Evolutionary Multitasking Optimization

Evolutionary multitasking as a new paradigm in the field of optimization and evolutionary computation was proposed in 2016 [46]. Unlike multi-objective optimization, multi-tasking optimization aims to find the optimal solution for each task rather than trade-off solutions. Assuming there are T minimization problems, the objective function of evolutionary multitasking can be formulated as follows:

$$\begin{aligned}\{x_1, x_2, ..., x_T\} = argmin\,\{f_1(x), f_2(x), ..., f_T(x)\}\\ s.t. \quad x_i \in \Omega_t, i = 1, ..., T,\end{aligned} \tag{3}$$

where x_t and Ω_t are the feasible solution and feasible region of task t, respectively. It aims to optimize each task completely and concurrently with the help of implicit parallelism of population-based search. In [46], a multifactorial evolutionary algorithm (MFEA) is first proposed to solve this problem. Subsequently, researchers have primarily focused on addressing the issues related to task interference or more complex problems in learning algorithms [57], [58]. Multi-task evolutionary algorithms have been developed to mitigate the task interference during the optimization process, such as MFEA II [59], MO-MFO [60] , and others [61]–[66]. Furthermore, the evolutionary multitasking framework has been applied to the simultaneous optimization of multiple sparse reconstruction tasks [67] and feature subspaces generation [68].

In the paper, the MFEA algorithm is utilized to solve the sparse sharing representation of tasks. Therefore, several key definitions of MFEA are given. Specifically, assuming a population P with K individuals, several definitions associated with the individual $p_j, j \in \{1, 2, ..., K\}$ are shown as follows:

***Definition** 2 (Factorial Cost):* The factorial cost of individual p_j on task $\mathcal{T}_t$ is defined as $\Psi_t^j = f_t^j + \lambda\delta_t^j$, where λ is the penalty parameters, f_t^j and δ_t^j are the objective value and the total constraint violation, respectively. When the p_j is a feasible solution, it is represented only by the objective value f_t^j.

***Definition** 3 (Factorial Rank):* The *factorial rank* r_t^j of individual p_j on task $\mathcal{T}_t$ is an index that is obtained from the ascending order of the *factorial cost* Ψ_t.

***Definition** 4 (Skill Factor):* The *skill factor* τ_j of p_j is a index of task corresponding to the best *factorial rank*, which is denoted as $\tau_j = argmin_t \left\{r_t^j\right\}_{t=1}^{T}$.

***Definition** 5 (Scalar Fitness):* The *scalar fitness* φ_j of p_j is the inverse of τ_j; i,e. $\varphi_j = 1/min_{i\in\{1,...,T\}} r_i^j$.

III. Proposed Method

In this section, learning task sparse sharing representation is modeled as an optimization problem, and the ESSR method is proposed first. Then an evolutionary learning framework is adopted to solve the optimal combination of tasks and sharing features. Finally, we further explore the importance of features and task relationships in MTL.

A. Sparse Sharing Representation of MTL

To avoid negative knowledge transfer in MTL, a feasible approach is sharing some features among tasks. The process of selecting which features to share is a combinatorial optimization problem. It aims to maximize task's performance by identifying the ideal blend of tasks and shared features. In contrast to existing random combination methods, ESSR models the selection of sharing features as an optimization problem.

In a neural network with L sharing convolution layers, $l \in \{1, ..., L\}$ is the index of the sharing layers. Let $x^{l-1,k}$ be the input feature of k-th filter units of l-th layer, and x^l is the output feature of it. The convolution operation for task $\mathcal{T}_t, t \in \{1, ..., T\}$ can be described as:

$$x_t^{l,k} = f^{l,k}\left(\theta^{l,k}, x_t^{l-1,k}\right), \tag{4}$$

$f^{l,k}$ refers to a feature map of the k-th filter unit at the layer l, and $\theta^{l,k}$ is the parameters of filter. Given a neural network with K filters in the sharing layers, $\theta_\varepsilon = [\theta_1, \theta_2, ..., \theta_K]$ denotes the filters parameters of sharing layers and $\mathcal{F} = \{f_1, f_2, ..., f_K\}$ is the hidden sharing feature set of tasks. Depending on θ_ε and $\mathcal{F}$, the sharing layers transform the original inputs to the learned feature representation. However, when some of the information among tasks is weakly or even negatively correlated, feature set $\mathcal{F}$ will contain some negative correlation or redundant features. This results in negative knowledge transfer by sharing all the features during multi-task learning. Therefore, in this paper, we assume that all tasks share a small subset of features.

Let $M = [m_1, m_2, ..., m_K]$ is a mask, and m_k is a tensor filled with all 0 or all 1, and it has the same dimension with θ_k. Then, the following operations is given:

$$m_k \odot \theta_k = \begin{cases}\theta_k, & m_k = 1^{|\theta_k|}\\ 0^{|\theta_k|}, & m_k = 0^{|\theta_k|}\end{cases} \tag{5}$$

where $|\theta_k|$ denotes the dimension of θ_k, and $\odot$ denotes element-wise multiplication. If $m_k = 1^{|\theta_k|}$, the k-th filter is selected. Conversely, if $m_k = 0^{|\theta_k|}$, it means that the k-th filter is masked in neural network, the feature f_k is discarded. Moreover, the subspace spanned by the k-th feature will also be removed from $\mathcal{F}$. Therefore, the operation $M \odot \theta_\varepsilon$ is equivalent to select a sharing subset of features by masking the filters. After the sharing layers are masked, the task $\mathcal{T}_t$ gets a sparse sharing representation. Next, a learning method learns a mapping $\mathcal{F}$ from this sparse representation to a specific output, i.e. $y_t = \mathcal{F}\left(M \odot \theta_\varepsilon, x_t\right)$.

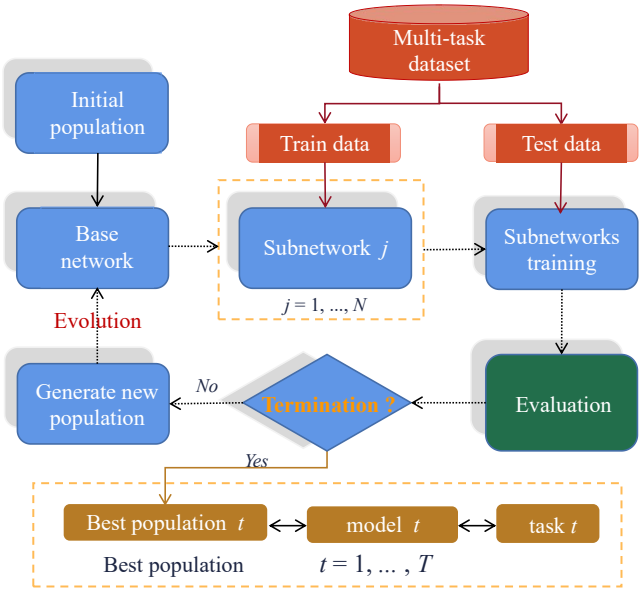


Fig. 3: The flowchart of the ESSR. The first step involves the initialization of the population P comprising N individuals, followed by decoding each individual into a corresponding mask. Secondly, each mask corresponds to a subnetwork for a given base network. And all the subnetworks are trained in parallel. Then each individual (subnetwork) is evaluated based on it scalar fitness (SF). If the termination condition is met, the algorithm stops running; Otherwise, the algorithm selects a new population according to SF and cycles into the next iteration. Finally, our framework learns a specific subnetwork for each task.

Because of the variations among tasks, it is challenging to identify a single mask that optimizes the performance of all tasks. ESSR learns the mask set based on task preferences. Drawing inspiration from evolutionary multitasking, we formalize the multitask learning problem as follows:

$$\begin{aligned}&\{(M,\theta)_1,(M,\theta)_2,...,(M,\theta)_T\}\\&=\underset{(M,\theta)}{\operatorname{argmin}}\left\{L_1\left(M\odot\theta_\varepsilon,\theta^1\right),L_2\left(M\odot\theta_\varepsilon,\theta^2\right),\right.\\&\left.\ldots,L_T\left(M\odot\theta_\varepsilon,\theta^T\right)\right\},M\in\{\mathbf{0},\mathbf{1}\}^K\end{aligned}\tag{6}$$

where K denotes the number of filters; $\mathbf{0}$ and $\mathbf{1}$ are the tensors filled with 0 and 1 respectively. $(M,\theta)_t$ is a 2-tuple consisting of the optimal mask and parameters of task $\mathcal{T}_t$. Eq. (6) is a multitasking optimization problem that aims to find the T optimal combination of mask and network parameters simultaneously. The subnetworks of tasks are obtained by optimizing Eq. (6).

Let $\Theta=\{M,\theta_\varepsilon,\theta^t\}$ be a parameters set. Eq. (6) can be equivalently written in the following general form:

$$\begin{aligned}&\{\Theta_1,\Theta_2,...,\Theta_T\}\\&=\underset{\Theta}{\operatorname{argmin}}\left\{L_1\left(\Theta;\mathcal{X}\right),L_2\left(\Theta;\mathcal{X}\right),...,L_T\left(\Theta;\mathcal{X}\right)\right\},\end{aligned}\tag{7}$$

$\mathcal{X}\in D=\{D_t\}_{t=1}^T$. Thus, ESSR can also be interpreted as determining the best possible solution for each task, as shown in Fig. 2(d).

B. Evolving Sparse Sharing Representation for MTL

In this section, an adaptable learning framework is proposed to solve the problem (6). There are two types parameters to be optimized: masks and neural network parameters. The task's masks are utilized to obtain subnetwork architectures, whereas all tasks jointly optimize neural network parameters. Additionally, evolutionary algorithms update the task masks based on the subnetwork's performance. Algorithm 1 represents the pseudocode of ESSR. It consists of training subnetworks in parallel and updating mask set, they are alternately optimized during learning. The algorithm is explained in detail as follows:

Algorithm 1: ESSR Algorithm Framework

Input: Multi-task dataset $\mathcal{D}=\{D_t\}_{t=1}^T$;
Base network Net;
Output: Task-specific subnetworks.
1 Initialize a population (mask set);
2 **for** g = 1 *to* G **do**
3 Part 1: Training subnetworks in parallel (**Algorithm 2**) ;
4 Part 2: Updating mask set (**Algorithm 3**);
5 **end**
6 **return** Task-specific subnetworks.

*1) **Subnetwork Training**:* When a mask is provided, the corresponding subnetwork is jointly optimized by all tasks. Algorithm 2 outlines the pseudocode of subnetworks training. During training, the task gradients are averaged to obtain the final subnetwork parameter gradient g_θ (lines 2-9 of Algorithm 2). The Adam algorithm then optimizes the parameters using g_θ. The task gradient is transmitted to parameter θ by back propagation of weighted loss λL_t. And the weight λ is assigned differently for each task, depending on whether it is the "main task" of the subnetwork. The "main task" has a larger λ than other tasks. Lines 4-6 of the Algorithm 3 show how the subnetwork is assigned its "main task". This mechanism guarantees that each subnetwork is uniquely linked with a "main task", while other tasks serve as "auxiliary tasks" that provide valuable information for "main task" during training. For the initial population without prior task information, all tasks receive the same loss function weights λ.

*2) **Updating Mask Set**:* Considering only the mask set, the optimization objective function is simplified as follows:

$$\begin{aligned}&\{M_1,M_2,...,M_T\}\\&=\underset{M}{\operatorname{argmin}}\left\{L_1\left(M\right),L_2\left(M\right),...,L_T\left(M\right)\right\}.\end{aligned}\tag{8}$$

This is a multitasking optimization problem. In this paper, the classic single population MFEA algorithm is introduced to solve the masks of tasks simultaneously. In the following, we give the detailed steps of the evolutionary algorithm:

Encoding and decoding: The individuals in the population are randomly initialized. Assuming the base network contains K filters in the sharing layers. Individual chromosomes are encoded and decoded as follows:

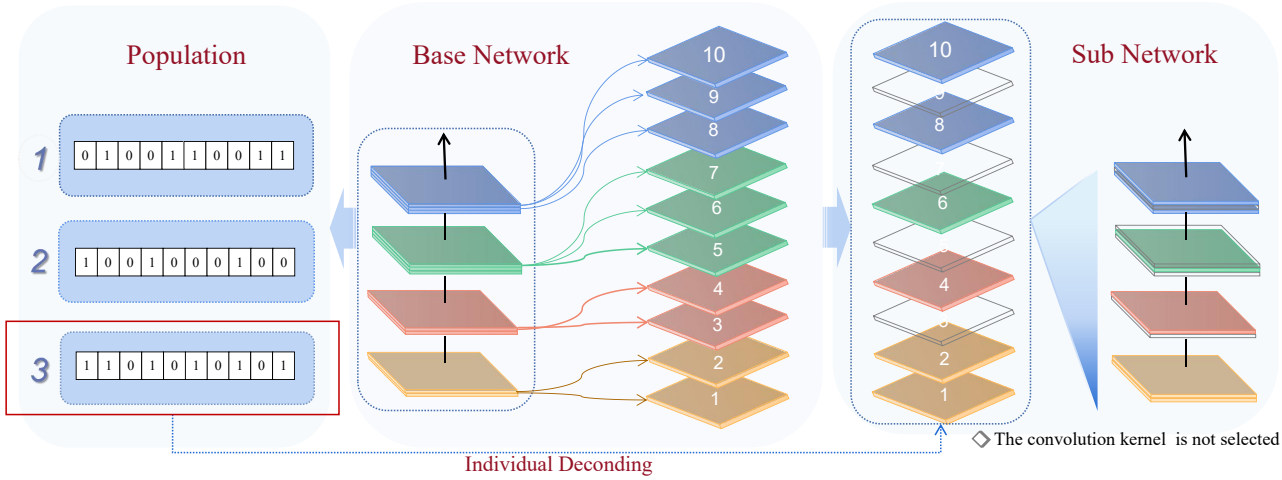


Fig. 4: Illustration of the process of filter selection. The figure shows a convolution neural network containing four sharing convolution layers. The rhomboid blocks with different colors in the network represent different layers, constituting ten filters in total, numbered sequentially. Individuals in the population encode based on the ten-element vectors using either 0 or 1. For example, individual 3 is considered, and the position 3, 5, 7, and 9 are zeroed, meaning that filters 3, 5, 7, and 9 will be discarded during learning. Finally, subnetwork generating from the 3rd individual is established.

Encoding: Individual chromosomes within population P are encoded as $K-$dimension vectors filled with 0 and 1; i.e.$\forall p \in P$, $p = (p_1, p_2, ..., p_K)$, and $p_k \in \{0, 1\}$.

Decoding: The mask M is generated after decoding the individual. If the $p_k = 1$, then the m_k is a tensor filled with 1. It means that the i-th filter in sharing layers is selected. Similarly, if $p_k = 0$, then m_k is a tensor filled with 0. At the same time, the i-th filter in sharing layers is masked. Fig. 4 shows the process of filter selection and subnetwork generation after giving a population.

Individual task division: In this paper, we represent the *factorial cost* of the individual using negative task accuracy. The *factorial cost* of individual p_j on task $\mathcal{T}_t$ is denoted as $\Psi_t^j = -acc_t^j$, where acc_t^j refers to the accuracy of individual j on task $\mathcal{T}_t$. Once the subnetworks are trained, calculating the *factorial cost* of each individual becomes feasible. The *factorial rank* and *skill factor* of individuals are obtained accordingly based on Definition 3 and Definition 4, respectively. Individuals are assigned to different tasks according to their *skill factor*. *In the initialization stage*, the *skill factor* τ_j of the individual is obtained by comparing its performance across different tasks. If the individual is optimal on the t-th task, then $\tau_j = t$. If an individual shows identical performance across several tasks, it will be randomly assigned to one of those tasks. *During the population evolution stage*, individual acquire the *skill factor* through cultural communication inherited from their parents as outlined in lines 4-12 of the Algorithm 3.

Through division, each individual (subnetwork) corresponds to a unique learning task that we refer to as the "main task". Thus, we can manually adjust the weight of loss function based on the individual's *skill factor* so that the learning algorithm is biased towards learning the "main task".

Population evolution and genetic transfer: The reproduction of individuals within a population is achieved through *crossover* and *mutation*. Crossover is the main mechanism for genetic transfer.

Algorithm 2: Subnetworks Training

Input: M: A population (mask set);
T: The number of tasks;
a: A number between 0.5 and 1;
$D = \{D_t\}_{t=1}^{T}$: Multi-task dataset;
Net: A base network;
Output: Ψ: Population Factorial Cost;

```
/* Subnetworks run in parallel.                    */
for i = 1 to |M| do
    /* Subnetwork training.                        */
    for t = 1 to T do
        if t = M_i.skill factor then
            g^t= ∇_θ a L_t(M_i ⊙ θ_ε; D_t^train) ; // M_i ⊙ θ_ε
              indicates masking the Net.
        else
            g^t= ∇_θ (1 − a) L_t(M_i ⊙ θ_ε; D_t^train)
        end
    end
    Gradient g_θ = (1/T) Σ_{i=1}^{T} g^t;
    Update θ using g_θ;
    Obtain subnetwork Net_i;
    /* Subnetwork testing.                         */
    for t= 1 to T do
        Calculate task accuracy acc_t^i;
    end
    Factorial Cost Ψ_i = {−acc_1^i, ..., −acc_T^i};
end
Return Ψ
```

Genetic transfer occurs on inter-tasks or intra-tasks by crossover operators during population evolution. However, the crossover of individuals between tasks increases the risk of negative knowledge transfer while realizing genetic transfer. The optimization algorithm must balance exploitation and exploration of the search space. So the random mating prob-

Algorithm 3: Population Evolution

Input: P_g: Current population;
Output: P_{g+1}: Next generation population.

$R \leftarrow \emptyset$; $C \leftarrow \emptyset$;
for t = *1 to* $|P_g|/2$ **do**
  Two candidate parents p_i and p_j in P are selected randomly;
  if $\tau_i == \tau_j$ *or rmp<0.3* **then**
    Obtain offspring c_1 and c_2 by the crossover;
    The *Skill Factor* of offspring is assigned randomly according to the parent;
    Put c_1 and c_2 in C;
  else
    Obtain offspring c_1 and c_2 by the mutation;
    The *Skill Factor* of individuals is assigned according to their parents;
    Put c_1 and c_2 in C;
  end
end
Evaluate the individuals in offspring C according to **Algorithm 2**;
Put P_g and C into the matching pool R;
Calculate *Factorial Rank* of individuals in R;
Calculate *Scalar Fitness* of individuals in R;
Select N best *Scalar Fitness* individuals from R to form P_{g+1};
Return P_{g+1};

ability (*rmp*) is used to adjust this balance. If *rmp*<0.3, inter-task individuals reproduce through crossover operator; otherwise, the offspring will be produced through mutation. The details of population evolution are shown in Algorithm 3.

Evaluation and Selection: The performance of individual is evaluated based on the *scalar fitness* (refer to Definition 5). Individual with higher *scalar fitness* have greater adaptability to multi-task environments. And the "elite selection" operator is used to obtain a new population, that is, N individuals with optimal *scalar fitness* in the population are selected into the next generation.

C. Exploring Feature Importance of Tasks

ESSR aims to eliminate valueless components from all the hidden features so that the retained features can provide helpful knowledge for tasks. However, it is essential to consider the specific roles that these features play in the transfer of knowledge. Further discussion is necessary.

Assuming $\mathcal{T} = \{\mathcal{T}_t\}_{t=1}^{T}$ is task set and $\mathcal{F} = \{f_k\}_{k=1}^{K}$ is the all *hidden feature set* of tasks. $\mathcal{U}_t \subseteq \mathcal{F}$ denotes the *task sharing feature set* of task $\mathcal{T}_t$, which is learned by ESSR. According to the utilization of tasks, we divide features in $\mathcal{F}$ into three categories: common feature, redundant feature, and supporting feature.

***Definition** 6 (Common feature):* The common features in $\mathcal{F}$ are used for all tasks. The set of all common features is denoted as $C(\mathcal{F})$; i.e. for $\forall f_k \in C(\mathcal{F}) \subseteq \mathcal{F}$ satisfying $f_k \in \mathcal{U}_t, t = 1, ..., T$.

***Definition** 7 (Redundant feature):* The redundant features in $\mathcal{F}$ are not used by any tasks. All redundant features constitute the set $\mathcal{R}(\mathcal{F})$. For $\forall f_k \in \mathcal{R}(\mathcal{F}) \subseteq \mathcal{F}$, $f_k \notin \mathcal{U}_t, t = 1, ..., T$.

***Definition** 8 (Supporting feature):* The supporting features in $\mathcal{F}$ are used for some tasks. The set of all supporting features is denoted as $\mathcal{S}(\mathcal{F}) = \mathcal{F} - C(\mathcal{F}) - \mathcal{R}(\mathcal{F})$.

In addition, we also define the *negative correlation feature* of task and *task-special feature* as follows:

***Definition** 9 (Negative correlation feature):* The set of negative correlation features $\mathcal{N}(\mathcal{U}_t)$ of task $\mathcal{T}_t$ is composed of the features that do not belong to $\mathcal{U}_t \cup \mathcal{R}(\mathcal{F})$. i.e. $\mathcal{N}(\mathcal{U}_t) = \mathcal{F} - \mathcal{U}_t - \mathcal{R}(\mathcal{F})$.

***Definition** 10 (Task-special feature):* The set of task-special feature $\mathcal{TS}(\mathcal{F})$ is composed of the features that are used by only one task; i.e., $\mathcal{TS}(\mathcal{F}) = \cup_{t=1}^{T}\mathcal{U}_t - \cap_{t=1}^{T}\mathcal{U}_t$, $\mathcal{TS}(\mathcal{F}) \subseteq \mathcal{S}(\mathcal{F})$.

The above definitions facilitate us to analyze and understand the purpose of ESSR more clearly. The *common features* are the core components of all tasks and indispensable to $\mathcal{F}$. The knowledge in the *redundant features* is either contained in *task sharing features* or valueless for improving the performance of tasks. And *negative correlation features* of a task are the main factor that lead to negative knowledge transfer. The *task-special feature* corresponds to a task uniquely and provides knowledge to it. ESSR obtains the set of *task sharing feature* by identifying and reducing *redundant feature* $\mathcal{R}(\mathcal{F})$ and *negative correlation feature* $\mathcal{N}(\mathcal{U}_t)$ in $\mathcal{F}$. The set of *task feature*, learned by ESSR, is the essential of tasks, which suffices to dig all basic concepts occurring in hidden knowledge.

D. Discovering Task Correlation in MTL

In the paper, we assumes that there exists some correlation among tasks. Would it be possible to further investigate this correlation by analyzing the learned features?

In reference [14], a simple example shows that there are more shared units in the output layers when tasks are relevant. The outputs of unrelated tasks tend to use different hidden units. When tasks are more related, it indicates a higher degree of shared features between them. Therefore, we can measure the similarity and difference between tasks according to the set of *task sharing features* (the set of filters that are used by tasks).

For two finite sets A and B, the measure $H(A,B) = \frac{|A \cap B|}{|A \cup B|}, (A \cup B \neq \emptyset)$ is used to portray the similarities of them and $1 - H(A,B)$ is used to describe the differences of them [69], [70]. Therefore, we define the metrics of task similarity and difference.

***Definition** 11 (Similarity and Difference measure):* Let $\mathcal{U}_i$ and $\mathcal{U}_j$ are the set of task sharing features of tasks $\mathcal{T}_i$ and $\mathcal{T}_j$, respectively. Then the similarity and difference between tasks $\mathcal{T}_i$ and $\mathcal{T}_j$ are defined below:

$$S(\mathcal{T}_i, \mathcal{T}_j) = \frac{|\mathcal{U}_i \cap \mathcal{U}_j|}{|\mathcal{U}_i \cup \mathcal{U}_j|}, \tag{9}$$

and

$$D(\mathcal{T}_i, \mathcal{T}_j) = \frac{1}{K}\left(|\mathcal{U}_i \cup \mathcal{U}_j| - |\mathcal{U}_i \cap \mathcal{U}_j|\right). \tag{10}$$

For any two tasks, greater similarity and smaller differences indicate that they are more related. Therefore, the ESSR method has two purposes: it can explore the optimal sharing feature subset of the task; and it can also further explain the task relationship.

IV. Experimental Studies

A. Datasets

In this paper, experiments are carried out on three heterogeneous feature multi-task datasets: DKL-MNIST, Cifar100, and Omniglot.

- **DKL-mnist**: This dataset is constructed by 28×28 gray images which are from EMNIST and KMNIST. EMNIST dataset contains 26 Letters classes and 10 Digits classes. The KMNIST datasets (kuzushiji-mnist) of Japanese Cursive Script has 10 kuzushiji classes. So, DKL-mnist is regarded as multi-task dataset which contains three tasks. The DKL-mnist is divided into a training set and a testing set; the training set is composed of 1500 samples selected from digits, letters, and kuzushiji in equal amounts, and other $70,800$ samples as testing set.
- **Cifar100**: According to [36], the cifar100 consists of 20 coarse labels, and each label is regarded as a task. Each task contains 5 classes and 500×5 training samples (500 per class). So, the whole dataset includes 50,000 training data and $10,000$ testing data. All the samples in Cifar100 are 32×32 RGB images.
- **Omniglot** : As a standard MTL dataset, the handwritten characters of the Omniglot database are grouped into different tasks by the 50 different alphabets, and the number of classes in tasks is different. As a few-shot dataset containing only 20 samples per class, it is harder to learn. Since training and testing data are not predefined in omniglot, we randomly split off 20% as test examples from each alphabet according to reference [36]. All the examples are 105×105 gray images.

B. Experimental Settings

1) Model and parameters:

- For DKL-mnist, the base neural network consists of three convolutional layers and two dense layers. Each convolution layer has 32 filters whose size is 3×3. So, there are 96 filters on sharing layers. All the convolutions go through batch normalization, ReLU activation, and 2×2 max-pooling input to the next layer. The first dense layer has 128 units and is followed by a ReLU activation. The second dense layer is a task-specific layer. For this dataset, there are three task-specific layers, and the units of each layer are consistent with the number of task classes.
- For Cifar100, we use the ResNet18 architecture as the base network. The network configurations are consistent with [36], which remove the 3×3 max-pooling layer from the original ResNet18 and set the convolution stride parameters again. The sharing layers contain 4800 filters.
- For Omniglot, the base neural network consists of four convolution layers and a single dense layer. Each convolution layer has 53 filters which size 3×3. So, there are 212 filters on sharing layers. All the convolutions go through normalization, ReLU activation, and 2×2 max-pooling input to the next layer. And the number of units of the final dense layer (task-specific layer) and the classes of tasks are equal.

During evolution, the population size is set to 20. The maximum evolutionary generation of DKL-mnist and Cifar100 with 10 tasks is 20, and the other datasets are 10. During neural network training, the Adam algorithm with a learning rate of 10^{-3} is used to optimize the network parameters. The three datasets are trained for $5,000$, $20,000$, and $20,000$ times, respectively, and the test interval is set to 200. For effective learning, only $5,000$ iterations are carried out in mask learning for datasets Cifar100 and Omniglot, and the subnetwork goes through $15,000$ iterations after the best masks of tasks are learned. In addition, we introduce the multi-task warmup (MTW) during learning; that is, the base network is pre-trained 200 times before masking. During the model training, the weights of the loss function are set to 0.8 and 0.2 for "main task" and "auxiliary tasks", respectively.

2) Comparative methods: We compare ESSR method with seven related methods on three multi-task datasets. There are two baseline methods (STL and MTL) and five related methods for address task interference in MTL.

- **STL**: The single-task learning method (STL) is used as baseline.
- **MTL**: A standard multi-task learning approach in which tasks share all sharing layer parameters.
- **Prior-MTL**: The convolution layers are divided into the sharing layers and the task-specific layers artificially.
- **LWS** [36]: It learns the assignment between sharing layers and task-specific layers adaptively.
- **SE** [39]: It masks the filters of base network by introducing the squeeze-and-excitation module.
- **TR** [40]: Task Routing divides the filter output into tasks by randomly giving multiple binary masks.
- **MR** [30]: Maximum Roaming inspired by dropout that randomly varies the parameter partitioning.

SE, TR, MR, and ESSR divide task parameters by masking the sharing layers filter. Compared with other methods, ESSR obtains masks through optimization rather than random assignment.

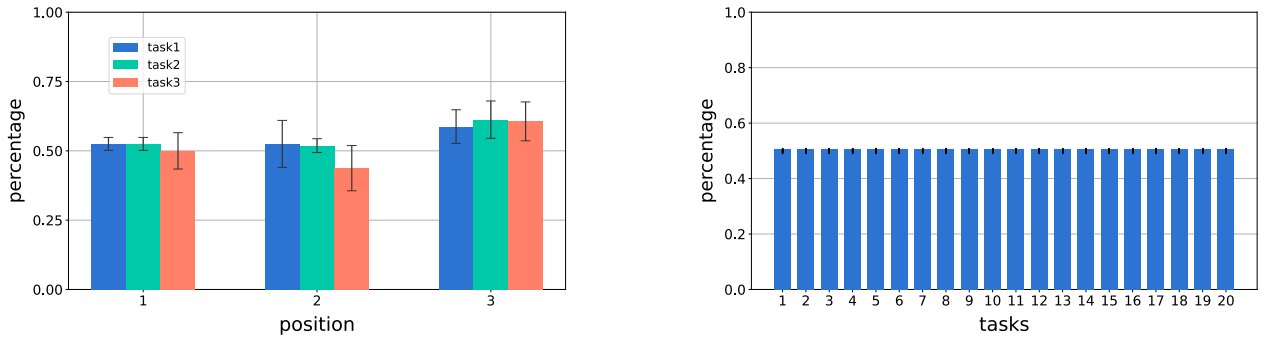
3) Performance metrics: Four indexes, accuracy, precision, recall, and F-score, are used to evaluate the performance of methods.

C. Comparing with Related Methods

In this section, we compare the ESSR with seven related methods. The experimental results are summarized in Table I and Table IV, where $\#N$ is the number of tasks. The p refers to the utilization rate of filters. The gen is the generation of the population evolutionary. The results presented in the table represent the average performance of the algorithm across all tasks. All experiments have been conducted five

TABLE I: COMPARATIVE STUDY OF RELATED ALGORITHMS ON DKL-MNIST AND CIFAR100 DATASET

Datasets	$\#N$	Model	Accuracy	Precision	Recall	F-score	Avg. Rank
DKL-mnist	3	STL	0.817 ± 0.002	0.632 ± 0.002	0.651 ± 0.002	0.629 ± 0.002	-
		MTL	0.798 ± 0.010	0.614 ± 0.010	0.635 ± 0.010	0.610 ± 0.011	4
		Prior-MTL	0.813 ± 0.006	0.630 ± 0.007	0.651 ± 0.007	0.628 ± 0.007	3
		LWS	0.837 ± 0.003	0.657 ± 0.005	0.676 ± 0.004	0.655 ± 0.005	1
		SE	0.777 ± 0.009	0.597 ± 0.008	0.619 ± 0.009	0.592 ± 0.010	6
		TR(p = 0.9)	0.773 ± 0.015	0.595 ± 0.016	0.617 ± 0.022	0.590 ± 0.020	7
		MR(p = 0.9)	0.789 ± 0.026	0.607 ± 0.022	0.627 ± 0.021	0.602 ± 0.023	5
		ESSR(gen = 20)	**0.827 ± 0.002**	**0.644 ± 0.003**	**0.664 ± 0.002**	**0.641 ± 0.003**	2
Cifar100	10	STL	0.689 ± 0.005	0.677 ± 0.009	0.678 ± 0.009	0.647 ± 0.008	-
		MTL	0.683 ± 0.011	0.674 ± 0.012	0.677 ± 0.013	0.640 ± 0.014	6.25
		Prior-MTL	0.694 ± 0.005	0.683 ± 0.006	0.688 ± 0.006	0.653 ± 0.006	3.75
		LWS	0.684 ± 0.007	0.670 ± 0.005	0.673 ± 0.004	0.638 ± 0.004	6.75
		SE	0.707 ± 0.005	0.704 ± 0.006	0.699 ± 0.006	0.670 ± 0.005	1.75
		TR($p = 0.9$)	0.695 ± 0.007	0.691 ± 0.005	0.685 ± 0.005	0.657 ± 0.006	3.25
		MR($p = 0.9$)	0.689 ± 0.006	0.683 ± 0.006	0.679 ± 0.003	0.651 ± 0.005	4.75
		ESSR(gen = 20)	**0.710 ± 0.004**	**0.707 ± 0.004**	**0.699 ± 0.006**	**0.673 ± 0.005**	1
Cifar100	20	STL	0.672 ± 0.004	0.661 ± 0.005	0.662 ± 0.005	0.631 ± 0.005	-
		MTL	0.673 ± 0.005	0.662 ± 0.006	0.668 ± 0.006	0.637 ± 0.014	7
		Prior-MTL	0.697 ± 0.006	0.686 ± 0.006	0.687 ± 0.005	0.655 ± 0.006	3
		LWS	0.684 ± 0.008	0.671 ± 0.009	0.677 ± 0.008	0.642 ± 0.009	6
		SE	0.708 ± 0.006	0.705 ± 0.005	0.698 ± 0.004	0.670 ± 0.006	2
		TR(p = 0.9)	0.687 ± 0.002	0.684 ± 0.003	0.676 ± 0.003	0.649 ± 0.003	4.75
		MR(p = 0.9)	0.687 ± 0.005	0.685 ± 0.004	0.678 ± 0.004	0.650 ± 0.005	4
		ESSR(gen = 10)	**0.709 ± 0.005**	**0.709 ± 0.005**	**0.700 ± 0.006**	**0.674 ± 0.005**	1



(a) DKL-mnist (b) Cifar100

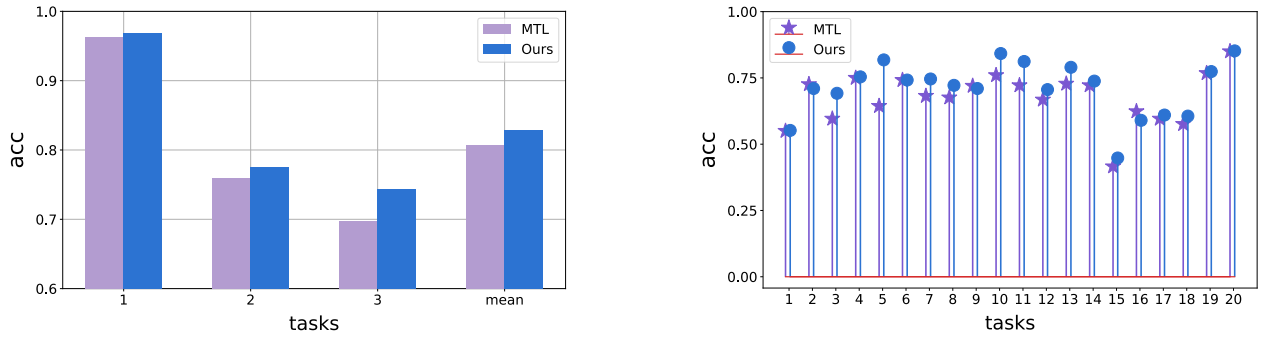
Fig. 5: The utilization ratio of the filter. The percentage of the filters used for each layer for each task on DKL-mnist (a) and Cifar100 (b). The figure shows the mean and standard deviation of the five experiments.

times and reported with their respective mean and standard deviation (presented as "mean ± std"). The optimal results are underlined, while the results obtained by ESSR are presented in bold. On Cifar100, two groups of experiments are conducted for 10 and 20 tasks, respectively, to verify the impact of task size on algorithm performance. Considering time consumption, the experiment is conducted only once on Omniglot. The utilization rate of filters of ESSR method on each task is presented in Fig. 5. Then we compare the performance of our algorithm on each task with MTL method (refer to Fig. 6). In addition, the performance of the final individual on each task is presented in Table III. From Table I - V and Fig. 5 - 6, some findings are obtained:

(1) ***The classical MTL method sometimes suffers from negative knowledge transfer during multi-task learning.*** As shown in Table I, the STL method work better than MTL on DKL-mnist. This implies that with the traditional MTL method, where all hidden features are shared during training, there is a negative transfer of knowledge among tasks. What is the reason behind this phenomenon? DKL-mnist with a few (500) training samples and numerous testing samples, the information contained in training samples is insufficient for each task. A learning task will rely on lots of training signals from other tasks to improve performance during learning. However, the correlation among tasks is weak and inconsistent for the DKL-mnist dataset. (This conclusion will be obtained from Figure: 7 in Section IV.D.). The traditional MTL method trains each task equally without considering the difference among tasks, which allows some negative correlation features to be transferred across tasks and leads to negative knowledge transfer phenomenon.

Compared with the MTL and STL, ESSR improves performance on all four metrics. The superior performance indicates that ESSR can leverage the sparse modeling to discover the discriminative features (*task sharing feature set*).

(2) ***ESSR focuses on improving the performance of each task.*** To demonstrate the superiority on individual tasks, we further compare the accuracy of ESSR with MTL on each task.



(a) DKL-mnist (b) Cifar100

Fig. 6: Comparing the accuracy of ESSR with MTL in each task on DKL-mnist(a) and Cifar100 with 20 tasks(b).

TABLE II: THE VALUE OF T-STATISTIC IN THE T-TEST

$H_0 : u_0 = u_j, H_1 : u_0 \neq u_j; \quad t_{0.05,8} = 2.3060.$

Datasets	Model	Acc	Pre	Rec	F-score
DKL-mnist	MTL	17.985	18.1735	17.985	17.1957
	prior-MTL	14	11.6264	11.2937	10.7959
	LWS	17.5412	14.1005	16.9706	15.1851
	SE	34.2997	34.791	30.8697	29.6833
	TR	22.5687	19.0372	13.4561	15.9492
	MR	9.2164	10.5392	11.0931	10.6342
cifar100-10	MTL	14.5893	16.5	9.718	14.0394
	prior-MTL	15.8037	21.0494	8.1989	16.1955
	LWS	20.3961	36.546	22.8035	34.5705
	SE	2.9632	2.6312	**0**	2.6833
	TR	11.767	15.8037	11.3369	12.9564
	MR	18.4182	21.0494	18.8562	19.6774
cifar100-20	MTL	32.1994	38.0595	23.8514	15.7411
	prior-MTL	9.7173	18.6249	10.5271	15.3858
	LWS	16.76	23.3432	14.5465	19.6574
	SE	**0.8098**	3.5777	**1.7541**	3.2391
	TR	25.8377	27.1163	22.6274	27.1163
	MR	19.6774	23.7055	19.2953	21.4663

* u_0 is the mean of ESSR; u_j is the mean of compared method.
* If the absolute value of the t-statistic (T-value) is greater than this critical value, the null hypothesis can be rejected. The null hypothesis is rejected, showing that the results are reasonably accurate and not by chance.

For DKL-mnist (refer to Fig. 6(a)), ESSR outperforms MTL on all three tasks. Significantly, the proposed method increases accuracy by 11% on task 3. For Cifar100 with 20 tasks (refer to Fig. 6(b)), the ESSR method improves the performance of most tasks. In particular, on task 5, the accuracy of ESSR method is improved by 27% compared with MTL. It indicates that the proposed method aims at improving the performance of each task, not some tasks.

(3) ***The performance of ESSR outperforms most of the related methods on three datasets.*** From the Table I and Table IV, the ESSR ranks first on Cifar 100 and Omniglot, and second on DKL-mnist, which means the ESSR has a good performance in the comparison of various competing methods. Especially on the Cifar100 with 20 tasks, ESSR achieves optimal performance after only 10 generation evolution. Although the performance of ESSR is slightly lower than LWS on DKL-mnist, the smaller inference model of tasks is obtained. Furthermore, the t-test in Table: II demonstrates significant statistical differences between the ESSR and compared method.

(4) ***Compared with random sparse sharing methods (TR, MR), the ESSR has fewer parameters.*** In the experiments, when the output rate of filters of pre-task is 90% ($p = 0.9$), TR and MR have the best performance. However, ESSR only uses about $40\% - 60\%$ filters in each sharing layer on DKL-mnist as shown in Fig. 5(a). And the selected filters for each task account for 50% of the overall on sharing layers, as shown in Figure. 5(b). This shows that ESSR with less filters performs better than TR and MR. Especially on Omniglot dataset with a small number of training samples and many tasks, ESSR has apparent advantages. The results in Table V also proves that ESSR has a smaller inference time(IT) and model size(MS).

In summary, ESSR can find a better match of tasks and features to alleviate negative knowledge transfer of multi-task learning. It works well with a larger sparsity of the model compared to other related methods.

D. Analyzing the Importance of Features and the Relationship of Tasks

We further analyzed the relationship between the filters obtained by ESSR in terms of their consistency with features, in order to explore the significance of various features in the MTL, as well as the interrelationships between tasks.

For DKL-mnist, we calculate the percentage of intersection, union, and symmetric difference of the filter sets of three tasks and pairwise tasks, respectively. The ratio of intersection and union is also given. The experimental results are presented in Fig. 7. By analyzing the filters used for three tasks, the following observations can be obtained:

- The features shared by the three tasks are less than 20% of the total features. It implies that less than 20% features belong to the set of *common features*.
- The union of task filters accounts for about 90% of the total. It implies that 10% of features learned from the base networks are invalid or redundant. In other words, the base model of DKL-mnist may be over-parameterized.
- The other 70% of features belong to *supporting feature*, and among them *task-special features* accounts for 50% of overall.
- The ratio of intersection and union of three tasks is 0.2. It indicates that the similarity of tasks is 0.2.

TABLE III: ACCURACY OF FINAL INDIVIDUALS ON EACH TASK

Datasets	POP	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	**main task**
DKL-mnist	pop_1	**0.969**	0.749	0.720								1
	pop_2	0.962	**0.774**	0.699								2
	pop_3	0.957	0.743	**0.746**								3
cifar100-10	pop_1	**0.560**	0.682	0.642	0.712	0.700	0.700	0.662	0.622	0.624	0.760	1
	pop_2	0.516	**0.692**	0.648	0.720	0.692	0.684	0.676	0.626	0.612	0.722	2
	pop_3	0.534	0.674	**0.654**	0.722	0.752	0.664	0.676	0.642	0.676	0.744	3
	pop_4	0.550	0.670	0.608	**0.766**	0.702	0.716	0.682	0.652	0.638	0.740	4
	pop_5	0.546	0.662	0.608	0.722	**0.778**	0.726	0.690	0.654	0.626	0.764	5
	pop_6	0.518	0.656	0.624	0.732	0.722	**0.772**	0.684	0.614	0.630	0.748	6
	pop_7	0.508	0.676	0.596	0.686	0.737	0.714	**0.712**	0.644	0.630	0.738	7
	pop_8	0.550	0.672	0.606	0.720	0.722	0.688	0.694	**0.702**	0.622	0.748	8
	pop_9	0.538	0.644	0.590	0.678	0.750	0.656	0.674	0.642	**0.688**	0.740	9
	pop_{10}	0.508	0.680	0.592	0.696	0.726	0.682	0.704	0.628	0.682	**0.814**	10

TABLE IV: COMPARATIVE STUDY OF RELATED ALGORITHMS ON OMNIGLOT DATASET

Datasets	**Model**	**Accuracy**	**Precision**	**Recall**	**F-score**	**Avg. Rank**
Omniglot	STL	0.764	0.538	0.625	0.567	-
	MTL	0.788	0.571	0.659	0.599	4
	Prior	0.679	0.471	0.564	0.498	5
	LWS	0.789	0.591	0.667	0.616	2.25
	SE	0.276	0.162	0.211	0.169	7
	TR	0.517	0.337	0.427	0.361	6
	MR	0.792	0.581	0.667	0.608	2.5
	ESSR	**0.833**	**0.635**	**0.712**	**0.661**	1

TABLE V: COMPARASION OF INFERENCE TIMES AND NETWORK SIZE OF RELATED METHODS ON DKL-MNIST DATASET

Model	ESSR	MTL	Prior-MTL	LWS	SE	TR	MR
IT(s)	32.71	33.92	32.52	34.20	59.88	43.73	42.98
MS(KB)	246.8	246.9	612.4	695.0	261.4	249.7	249.7

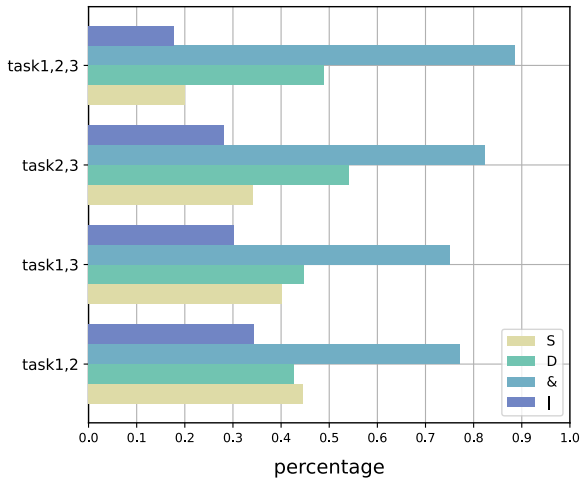


Fig. 7: Analyzing the learned features on DKL-mnist. Where $'|'$ is the percentage of set intersection, $'\&'$ is the percentage of a union of the set, $'D'$ is the percentage of difference between the set union and intersection, and it can be used to measure the difference between tasks; $'S'$ is the ratio of intersection and union of sets and it can be used to measure the similarity of tasks.

Overall, the knowledge obtained by the task is mainly from *task-special features*, and only a few *common features* provide knowledge for all tasks. It indicates that there are fewer similarities and more differences among tasks.

By analyzing the filters used in pairwise tasks, we find that the intersection of features between any two tasks accounts for about 30% of the total features. The union of features between any two tasks accounts for about $75\% - 82\%$ of the total features. The similarities and differences between tasks can be calculated according to Definition 11.

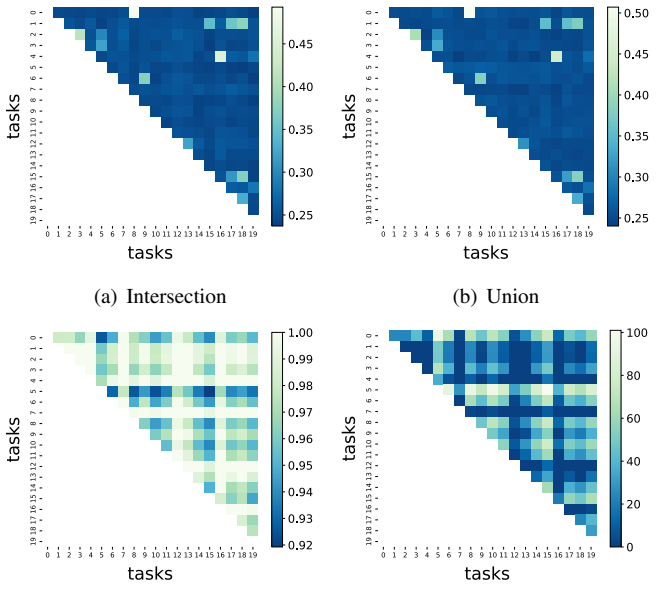
- The difference between task 1 with task 2 and task 3 are about 0.42 and 0.45, respectively. The difference between task 2 with task 3 is about 0.54;
- The similarities of task 1 with task 2 and task 3 are about 0.45 and 0.4, respectively. In addition, the similarity of task 2 with task 3 is about 0.35;

In general, the algorithm identifies that task 1 and task 2 have the strongest correlation. In contrast, the correlation between task 2 and task 3 is weaker.

For Cifar100, all filters are used during multi-task learning. In addition, there is no filter contained in all tasks. Therefore, the feature universal set of Cifar100 does not contain *common feature* and *redundant feature*. So, heat maps are used to show the relationship between tasks. Fig. 8 (a) and (b) show that the proportion of intersection and union between any two tasks accounts for about $25\% - 30\%$ of all filters. Fig. 8 (c) and (d) show that task 4, task 7, task 12, and task 16 have strong similarities and small differences with all tasks.

E. Parameter Sensitivity Analysis

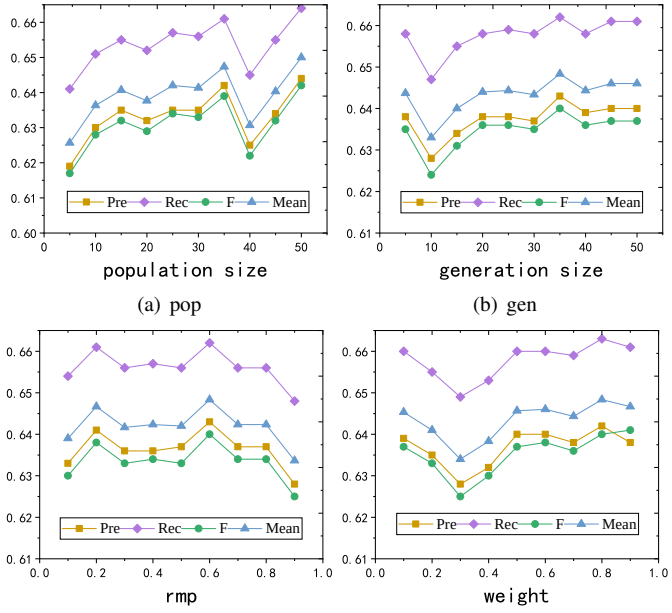
In this section, we verify the influence of the hyperparameters on the performance of multi-task learning. Considering the time consumption, the experiment is only carried out on a small DKL-mnist dataset. Each individual iterates 2000 times during training. Other parameters are consistent with the above experiments. The results are shown in Figure 9. The values of metrics in the figure are the average of all task performances. Mean is the average of three metrics. Results show that algorithm with a larger population size or generation size performs well. The outliers in the figure may be caused by the randomness of the algorithm. The sensitivity analysis of rmp shows that when the parameters are 0.2 and 0.6, all metrics achieve better performance. It means that a good balance between exploitation and exploration is obtained with $rmp = 0.2$ or 0.6 in this case. Figure 9(c) shows the variation of the algorithm performance with the loss function weight of the main task. Overall, the algorithm tends to choose a larger weight value. When the weight is 0.8, the algorithm has the best performance.



(a) Intersection (b) Union

(c) Similarity (d) Difference

Fig. 8: Illustration of the filters co-occurrence relation between tasks, for Cifar100 with 20 tasks. (a) and (b) show the proportion of the intersection and union of the task filter set to the total number, respectively. (c) shows the similarities between tasks. (d). shows the differences between tasks.



(a) pop (b) gen

(c) rmp (d) weight

Fig. 9: Illustration of influence of hyperparameters on algorithm performance for DKL-mnist dataset. (a) Influence of population size on algorithm performance; (b) Influence of maximum evolutionary generation on algorithm performance; (c) Influence of rmp on algorithm performance; (d) The influence of the weights of the loss function on algorithm performance.

TABLE VI: COMPARATIVE STUDY OF SEVERAL EVOLUTIONARY ALGORITHMS ON DKL-MNIST DATASET

Model	Accuracy	Precision	Recall	F-score
-GA	0.817 ± 0.003	0.631 ± 0.003	0.651 ± 0.003	0.628 ± 0.003
-MFDE	0.821 ± 0.003	0.637 ± 0.004	0.657 ± 0.004	0.635 ± 0.004
-MFPSO	0.821 ± 0.006	0.638 ± 0.007	0.659 ± 0.007	0.635 ± 0.007
-MFEA	0.827 ± 0.002	0.644 ± 0.003	0.664 ± 0.002	0.641 ± 0.003

F. Applicability of Other Multifactorial Evolutionary Algorithms to ESSR

To demonstrate the applicability of ESSR, two multifactorial evolutionary optimization algorithms[1] MFPSO and MFDE are also embedded into our multi-task learning framework. In addition, the single-task genetic algorithm — GA is also compared with the multi-task modeling method to verify the advantages of multitasking algorithms. The objective function of ESSR-GA is shown below:

$$(M,\theta) = \underset{M,\theta}{\operatorname{argmin}} \sum_{t=1}^{T} L_t(M \odot \theta^{sh}, \theta^t). \tag{11}$$

This is a single-objective optimization problem; the unique solution of (M,θ) is obtained by minimized Eq.(11).

In this paper, the learning tasks mask is a discrete optimization problem, so the solutions generated by MFPSO and MFDE must be a binary string containing only 0 or 1. Therefore, for MFPSO and MFDE, the encoded individual needs to be transformed into a binary string before decoding it. For MFPSO, without changing the velocity update formula, the position formula is redefined as follows:

$$x_{ij} = \begin{cases} 0, & rand \geq Sigmoid(v_{ij}) \\ 1, & otherwise \end{cases}, \tag{12}$$

and

$$Sigmoid(v_{ij}) = \frac{1}{1+e^{-v_{ij}}}.$$

where v_{ij} and x_{ij} refer to the velocity and position of particle i on j-th component, respectively; $rand$ represents a random number in the range of 0 and 1. For MFDE, the intermediate individuals p generated by differential and mutation operators are transformed as follows:

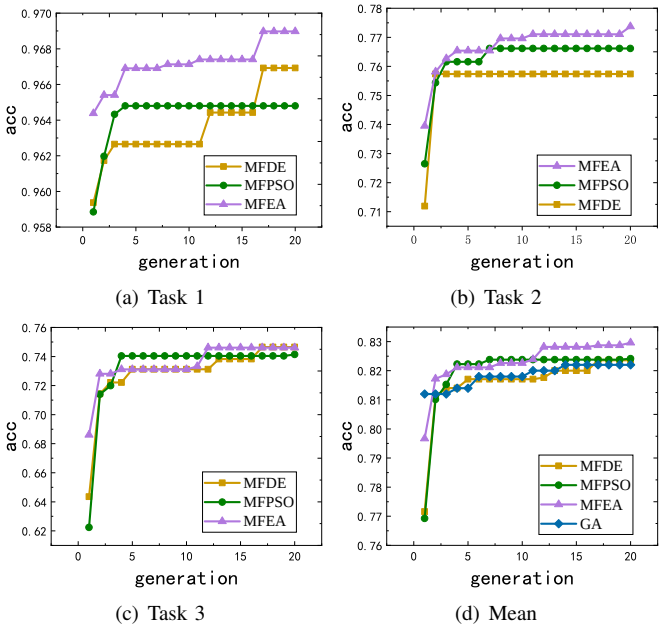
$$p_{ij} = \begin{cases} 0, & rand \geq Sigmoid(p_{ij}) \\ 1, & otherwise \end{cases} \tag{13}$$

where p_{ij} refers to the j-th element of i-th intermediate individual.

The comparative experiments of DKL-mnist are carried out. In all experiments, a population of 20 individuals evolved over 20 generations. The performance and evolution behaviors of these algorithms are shown in Table VI and Fig. 10, respectively. The results in Table VI are the "mean ± std" of five times experiments on all tasks.

As shown in Table VI, the developed multifactorial evolutionary algorithms work well. The performance of multi-task modeling methods (MFDE, MFPSO, and MFEA) are better than single-task modeling (GA), and MFEA has the best performance. The performance of MFDE is similar to that of MFPSO, but MFDE has less deviation and higher stability. Then, we analyze the evolution behavior of the algorithm accuracy and the average task accuracy across three tasks, and the experimental results are presented in Fig. 10. And the following findings are given: (1) MFEA shows strong learning ability in all tasks, while MFDE and MFPSO tend to fall into local minimum too early. MFPSO finds the best accuracy on

[1]The code is available at http://www.bdsc.site/websites/MTO/index.html.



(a) Task 1 (b) Task 2

(c) Task 3 (d) Mean

Fig. 10: Illustration of the evolution behavior of different evolutionary algorithms for DKL-mnist dataset. (a) The evolutionary behavior of Task 1; (b) The evolutionary behavior of Task 2; (c) The evolutionary behavior of Task 3; (d) Mean evolutionary behavior.

3-rd and 4-th generation of task 1 and task 3, respectively. For MFDE, the best accuracy obtained in task 2 has only evolved for two generations. (2) GA outperforms MF- methods in the initial phase, but the accuracy growth is not significant during the 20 generations evolution. In contrast, multi-task learning methods with poor initial performance exhibit strong search ability later. Consequently, ESSR based on multi-task modeling is reasonable compared to single-task, and MFEA is more applicable to the ESSR learning framework among multi-task modeling approaches.

V. Conclusion

To avoid task interference caused by negative knowledge transfer in MTL, ESSR is proposed in this paper. It aims to evolve a sparse sharing representation for each task. ESSR can find the combination of tasks and sharing features adaptively by multitasking modeling, and learning a sparse sharing subnetwork for each task. The multitasking optimization problem is solved by a single population multifactorial evolutionary algorithm (MFEA). The experiments demonstrate that ESSR can mitigate the phenomenon of knowledge negative transfer in MTL and obtain a small inference model.

However, this paper only investigates the performance of ESSR in heterogeneous features MTL. The applicability of this method in homogeneous features MTL, especially for computer vision tasks with big data and network architecture, needs further study. Due to differences in data features, it is necessary to explore new iterative and gradient updating approaches to ensure the performance of the algorithm. We also need to design more appropriate evolution and evaluation approaches to reduce the time consumption caused by the data scale. In addition, the proposed methods will be extended to Pareto multi-task learning in the future.

Acknowledgment

This work was supported by National Key Research and Development Program of China (No. 2021ZD0112400), National Natural Science Foundation of China (No. 62136005, No. 61976129), The Science and Technology Major Project of Shanxi under Grant (No.202201020101006), Hong Kong General Research Fund under Grant CityU-9043352, The Key Basic Research Foundation of Shenzhen under Grant JCYJ20220818100005011, Young Scientists Fund of the Natural Science Foundation of Shanxi (No. 20210302124549, No. 202203021222183), Scientific and Technological Innovation Programs of Higher Education Institutions in Shanxi (2021L286, 2020CG007), The Natural Science Foundation of Shanxi Province, China (No.20210302123455), The Key R&D Program of Shanxi Province, China (No. 202202020101004), The Open Project Program of the Key Laboratory of Embedded System and Service Computing of Ministry of Education (Tongji University), (ESSCKF 2021-04).

References

[1] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[2] H. Cheng, Y. Qian, Y. Guo, K. Zheng, and Q. Zhang, "Neighborhood information-based method for multivariate association mining," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[3] F. Li, Y. Qian, J. Wang, C. Dang, and L. Jing, "Clustering ensemble based on sample's stability," *Artificial Intelligence*, vol. 273, pp. 37–55, 2019.

[4] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4388–4403, 2021.

[5] J. Wang, Y. Qian, F. Li, J. Liang, and Q. Zhang, "Generalization performance of pure accuracy and its application in selective ensemble learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[6] Y. Chen, Y. Guo, Q. Chen, M. Li, W. Zeng, Y. Wang, and M. Tan, "Contrastive neural architecture search with neural architecture comparators," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9502–9511.

[7] Y. Guo, Y. Zheng, M. Tan, Q. Chen, Z. Li, J. Chen, P. Zhao, and J. Huang, "Towards accurate and compact architectures via neural architecture transformer," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 10, pp. 6501–6516, 2021.

[8] J. Liu, B. Zhuang, Z. Zhuang, Y. Guo, J. Huang, J. Zhu, and M. Tan, "Discrimination-aware network pruning for deep model compression," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 8, pp. 4035–4051, 2021.

[9] W. Dai, O. Jin, G.-R. Xue, Q. Yang, and Y. Yu, "Eigentransfer: a unified framework for transfer learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 193–200.

[10] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 394–407, 2019.

[11] H. X. Choong, Y.-S. Ong, A. Gupta, and R. Lim, "Jack and masters of all trades: One-pass learning of a set of model sets from foundation models," *arXiv preprint arXiv:2205.00671*, 2022.

[12] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[13] F. Heuer, S. Mantowsky, S. Bukhari, and G. Schneider, "Multitask-centernet (MCN): Efficient and diverse multitask learning using an anchor free approach," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 997–1005.

[14] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[15] M. Bernardini, L. Romeo, E. Frontoni, and M.-R. Amini, "A semi-supervised multi-task learning approach for predicting short-term kidney disease evolution," *IEEE Journal of Biomedical and Health Informatics*, vol. 25, no. 10, pp. 3983–3994, 2021.

[16] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine, "Reset-free reinforcement learning via multi-task learning: Learning dexterous manipulation behaviors without human intervention," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6664–6671.

[17] H. Zhang, S. Qian, Q. Fang, and C. Xu, "Multi-modal meta multi-task learning for social media rumor detection," *IEEE Transactions on Multimedia*, vol. 24, pp. 1449–1459, 2021.

[18] X. Liang, Y. Qian, Q. Guo, H. Cheng, and J. Liang, "AF: An association-based fusion method for multi-modal classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 9236–9254, 2022.

[19] X. Liang, Q. Guo, Y. Qian, W. Ding, and Q. Zhang, "Evolutionary deep fusion method and its application in chemical structure recognition," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 883–893, 2021.

[20] B. Zhou, X. Cai, Y. Zhang, W. Guo, and X. Yuan, "MTAAL: Multi-task adversarial active learning for medical named entity recognition and normalization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 16, 2021, pp. 14 586–14 593.

[21] A. Li, L. Jiao, H. Zhu, L. Li, and F. Liu, "Multitask semantic boundary awareness network for remote sensing image segmentation," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021.

[22] S. Chen, Y. Zhang, and Q. Yang, "Multi-task learning in natural language processing: An overview," *arXiv preprint arXiv:2109.09138*, 2021.

[23] S. Raghavan and K. Shubham, "Hybrid unsupervised and supervised multitask learning for speech recognition in low resource languages," in *Proc. Workshop on Machine Learning in Speech and Language Processing*, 2021.

[24] Y. Wang, Z. Zhao, B. Dai, C. Fifty, D. Lin, L. Hong, and E. H. Chi, "Small towers make big differences," *arXiv preprint arXiv:2008.05808*, 2020.

[25] H. Bilen and A. Vedaldi, "Integrated perception with recurrent multi-task neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[26] R. Ranjan, V. M. Patel, and R. Chellappa, "Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 1, pp. 121–135, 2017.

[27] X. Liu, P. He, W. Chen, and J. Gao, "Multi-task deep neural networks for natural language understanding," *arXiv preprint arXiv:1901.11504*, 2019.

[28] Y. Zhang and Q. Yang, "An overview of multi-task learning," *National Science Review*, vol. 5, no. 1, pp. 30–43, 2018.

[29] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[30] L. Pascal, P. Michiardi, X. Bost, B. Huet, and M. Zuluaga, "Maximum roaming multi-task learning," in *35th AAAI Conference on Artificial Intelligence*, vol. 35, no. 10, 2021, pp. 9331–9341.

[31] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning," in *ICML*, 2011.

[32] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert, "Cross-stitch networks for multi-task learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3994–4003.

[33] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille, "NDDR-CNN: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3205–3214.

[34] A. Søgaard and Y. Goldberg, "Deep multi-task learning with low level tasks supervised at lower layers," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 231–235.

[35] K. Hashimoto, C. Xiong, Y. Tsuruoka, and R. Socher, "A Joint Many-Task Model: Growing a neural network for multiple nlp tasks," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 2017, pp. 1923–1933.

[36] J. Prellberg and O. Kramer, "Learned weight sharing for deep multi-task learning by natural evolution strategy and stochastic gradient descent," in *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020, pp. 1–8.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[39] K. Maninis, I. Radosavovic, and I. Kokkinos, "Attentive single-tasking of multiple tasks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1851–1860.

[40] G. Strezoski, N. v. Noord, and M. Worring, "Many task learning with task routing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1375–1384.

[41] X. Sun, R. Panda, R. Feris, and K. Saenko, "Adashare: Learning what to share for efficient deep multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[42] C. Rosenbaum, T. Klinger, and M. Riemer, "Routing networks: Adaptive selection of non-linear functions for multi-task learning," in *International Conference on Learning Representations*, 2018.

[43] T. Sun, Y. Shao, X. Li, P. Liu, H. Yan, X. Qiu, and X. Huang, "Learning sparse sharing architectures for multiple tasks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, 2020, pp. 8936–8943.

[44] R. K. Ando, T. Zhang, and P. Bartlett, "A framework for learning predictive structures from multiple tasks and unlabeled data." *Journal of Machine Learning Research*, vol. 6, no. 11, 2005.

[45] J. Chen, J. Zhou, and J. Ye, "Integrating low-rank and group-sparse structures for robust multi-task learning," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 42–50.

[46] A. Gupta, Y. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.

[47] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[48] X. Lin, H. Zhen, Z. Li, Q. Zhang, and S. Kwong, "Pareto multi-task learning," *Advances in neural information processing systems*, vol. 32, pp. 12 060–12 070, 2019.

[49] A. Kendall, Y. Gal, and R. 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*, 2018, pp. 7482–7491.

[50] Z. Chen, V. Badrinarayanan, C. Lee, and A. Rabinovich, "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks," in *International Conference on Machine Learning*. PMLR, 2018, pp. 794–803.

[51] S. Liu, E. Johns, and A. J. Davison, "End-to-end multi-task learning with attention," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1871–1880.

[52] F. J. Bragman, R. Tanno, S. Ourselin, D. C. Alexander, and J. Cardoso, "Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1385–1394.

[53] O. Sener and V. Koltun, "Multi-task learning as multi-objective optimization," *arXiv preprint arXiv:1810.04650*, 2018.

[54] D. Mahapatra and V. Rajan, "Multi-task learning with user preferences: Gradient descent with controlled ascent in pareto optimization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6597–6607.

[55] P. Ma, T. Du, and W. Matusik, "Efficient continuous pareto exploration in multi-task learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 6522–6531.

[56] X. Lin, Z. Yang, Q. Zhang, and S. Kwong, "Controllable pareto multi-task learning," *arXiv preprint arXiv:2010.06313*, 2020.

[57] T. Wei, S. Wang, J. Zhong, D. Liu, and J. Zhang, "A review on evolutionary multi-task optimization: Trends and challenges," *IEEE Transactions on Evolutionary Computation*, 2021.

[58] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multitasking for evolutionary optimization of expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 44–58, 2017.

[59] K. K. Bali, Y. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: Mfea-ii," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 69–83, 2019.

[60] A. Gupta, Y. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE transactions on cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2016.

[61] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y. Ong, K. Tan, and A. K. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE transactions on cybernetics*, vol. 49, no. 9, pp. 3457–3470, 2018.
[62] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 858–869, 2019.
[63] Z. Liang, H. Dong, C. Liu, W. Liang, and Z. Zhu, "Evolutionary multitasking for multiobjective optimization with subspace alignment and adaptive differential evolution," *IEEE Transactions on Cybernetics*, 2020.
[64] J. Lin, H. L. Liu, B. Xue, M. Zhang, and F. Gu, "Multiobjective multitasking optimization based on incremental learning," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 824–838, 2019.
[65] Z. Tang, M. Gong, Y. Wu, W. Liu, and Y. Xie, "Regularized evolutionary multitask optimization: Learning to intertask transfer in aligned subspace," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 262–276, 2020.
[66] L. Zhou, L. Feng, K. C. Tan, J. Zhong, Z. Zhu, K. Liu, and C. Chen, "Toward adaptive knowledge transfer in multifactorial evolutionary computation," *IEEE transactions on cybernetics*, vol. 51, no. 5, pp. 2563–2576, 2020.
[67] H. Li, Y. Ong, M. Gong, and Z. Wang, "Evolutionary multitasking sparse reconstruction: Framework and case study," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 733–747, 2018.
[68] B. Zhang, A. K. Qin, and T. Sellis, "Evolutionary feature subspaces generation for ensemble classification," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 577–584.
[69] Y. Yao, "Information granulation and rough set approximation," *International Journal of Intelligent Systems*, vol. 16, no. 1, pp. 87–104, 2001.
[70] Y. Qian, H. Cheng, J. Wang, J. Liang, W. Pedrycz, and C. Dang, "Grouping granular structures in human granulation intelligence," *Information Sciences*, vol. 382, pp. 150–169, 2017.

Yayu Zhang (Student Member, IEEE) received the BSc degree at the Department of Mathematics from Changzhi University, China, in 2015. Currently, she is a Ph.D. candidate at the Institute of Big Data Science and Industry, Shanxi University. Her research interests include evolutionary computation, machine learning and their applications.

Yuhua Qian (Member, IEEE) received the MS and Ph.D. degrees in computers with applications from Shanxi University, Taiyuan, China, in 2005 and 2011, respectively. He is currently a Professor with the Key Laboratory of Computational Intelligence and Chinese Information Processing, Ministry of Education, Shanxi University. He is best known for multigranulation rough sets in learning from categorical data and granular computing. He is involved in research on machine learning, pattern recognition, feature selection, granular computing, and artificial intelligence. He has authored over 100 articles on these topics in international journals. He served on the Editorial Board of the International Journal of Knowledge-Based Organizations and Artificial Intelligence Research. He has served as the Program Chair or Special Issue Chair of the Conference on Rough Sets and Knowledge Technology, the Joint Rough Set Symposium, and the International Conference on Intelligent Computing, etc., and also PC Members of many machine learning, and data mining conferences.

Guoshuai Ma (Student Member, IEEE) Guoshuai Ma is a Ph.D. candidate at school of Computer and Information Technology, Shanxi University. Before this, he got bachelor's degree in management from the school of economics and management, northeast electric power university, in 2015. His research interest includes data mining and complex network.

Xinyan Liang received the BS degree at the School of Computer and Information technology from Shanxi University, China, in 2014. And he received Ph.D. degree in computer science technology from Shanxi University, in 2022. He was a visiting scholar at the University of Hong Kong, Hong Kong, China, in 2018. His main research interests include multi-modal/view machine learning, information fusion, the evolutionary intelligence, granular computing, and their applications. He has published several journal papers in his research fields, including *IEEE Transactions on Pattern Analysis and Machine Intelligence, IEEE Transactions on Evolutionary Computation*, etc.

Guoqing Liu (Student Member, IEEE) received the BSc degree at the school of Mathematical sciences from Shanxi University, China, in 2016. Currently, she is a Ph.D. candidate at the Institute of Big Data Science and Industry, Shanxi University. Her research interests include reinforcement learning, machine learning.

Qingfu Zhang (Fellow, IEEE) (M'01-SM'06-F'17) received the BSc degree in mathematics from Shanxi University, China in 1984, the MSc degree in applied mathematics, and the Ph.D. degree in information engineering from Xidian University, China, in 1991 and 1994, respectively. He is a Chair Professor of Computational Intelligence at the Department of Computer Science, City University of Hong Kong. His main research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications. Dr. Zhang is an Associate Editor of the IEEE Transactions on Evolutionary Computation and the IEEE Transactions on Cybernetics. He is a Web of Science highly cited researcher in Computer Science since 2016.

Ke Tang (Fellow, IEEE) (M'07-SM'13) received the B.Eng. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2002 and the Ph.D. degree from Nanyang Technological University, Singapore, in 2007. From 2007 to 2017, he was with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, China, first as an Associate Professor from 2007 to 2011 and later as a Professor from 2011 to 2017. He is currently a Professor with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China. He has over 6000 Google Scholar citation with an H-index of 36. He has published over 60 journal papers and over 80 conference papers. His current research interests include evolutionary computation, machine learning, and their applications. Dr. Tang was a recipient of the Royal Society Newton Advanced Fellowship in 2015 and the 2018 IEEE Computational Intelligence Society Outstanding Early Career Award. He is an Associate Editor of the IEEE Transactions on Evolutionary Computation and served as a member of Editorial Boards for a few other journals.

LIST OF FIGURES

1 The different sharing architecture of multi-task learning. The layers inside the red dotted box are task-sharing layers. The layers at the top of the model are task-specific layers. The blue block represents the tasks sharing parameters; The yellow and red blocks are the task-specific parameters; The solid pink circle represents the sharing mechanism of the model. (a) Given a learning model, tasks share all the underlying parameters. (b) Task 1 and Task 2 are given a separate model in advance, and the tasks share parameters at the same level through some mechanism. (c) Given a learning model, the sharing layers are divided into different tasks. (d) The parameters of sharing layers are divided into three categories: task 1, task 2, and the shared by task 1 and task 2. 1

2 Explanation of the solutions obtained by different methods. Take two tasks as examples, and the red dot or green dot in the figure is the final solution. (a) The solution of single-task learning. (b) The solution of linear scalarization multi-task learning. (c) The trade-off solution of multi-objective multi-task learning. (d) The solution of the ESSR method. 3

3 The flowchart of the ESSR. The first step involves the initialization of the population P comprising N individuals, followed by decoding each individual into a corresponding mask. Secondly, each mask corresponds to a subnetwork for a given base network. And all the subnetworks are trained in parallel. Then each individual (subnetwork) is evaluated based on it scalar fitness (SF). If the termination condition is met, the algorithm stops running; Otherwise, the algorithm selects a new population according to SF and cycles into the next iteration. Finally, our framework learns a specific subnetwork for each task. 5

4 Illustration of the process of filter selection. The figure shows a convolution neural network containing four sharing convolution layers. The rhomboid blocks with different colors in the network represent different layers, constituting ten filters in total, numbered sequentially. Individuals in the population encode based on the ten-element vectors using either 0 or 1. For example, individual 3 is considered, and the position 3, 5, 7, and 9 are zeroed, meaning that filters 3, 5, 7, and 9 will be discarded during learning. Finally, subnetwork generating from the 3rd individual is established. 6

5 The utilization ratio of the filter. The percentage of the filters used for each layer for each task on DKL-mnist (a) and Cifar100 (b). The figure shows the mean and standard deviation of the five experiments. 9

6 Comparing the accuracy of ESSR with MTL in each task on DKL-mnist(a) and Cifar100 with 20 tasks(b). 10

7 Analyzing the learned features on DKL-mnist. Where $'|'$ is the percentage of set intersection, $'\&'$ is the percentage of a union of the set, $'D'$ is the percentage of difference between the set union and intersection, and it can be used to measure the difference between tasks; $'S'$ is the ratio of intersection and union of sets and it can be used to measure the similarity of tasks. 11

8 Illustration of the filters co-occurrence relation between tasks, for Cifar100 with 20 tasks. (a) and (b) show the proportion of the intersection and union of the task filter set to the total number, respectively. (c) shows the similarities between tasks. (d). shows the differences between tasks. . 12

9 Illustration of influence of hyperparameters on algorithm performance for DKL-mnist dataset. (a) Influence of population size on algorithm performance; (b) Influence of maximum evolutionary generation on algorithm performance; (c) Influence of rmp on algorithm performance; (d) The influence of the weights of the loss function on algorithm performance. 12

10 Illustration of the evolution behavior of different evolutionary algorithms for DKL-mnist dataset. (a) The evolutionary behavior of Task 1; (b) The evolutionary behavior of Task 2; (c) The evolutionary behavior of Task 3; (d) Mean evolutionary behavior. 13

List of Tables

I COMPARATIVE STUDY OF RELATED ALGORITHMS ON DKL-MNIST AND CIFAR100 DATASET 9
II THE VALUE OF T-STATISTIC IN THE T-TEST 10
III ACCURACY OF FINAL INDIVIDUALS ON EACH TASK 11
IV COMPARATIVE STUDY OF RELATED ALGORITHMS ON OMNIGLOT DATASET 11
V COMPARASION OF INFERENCE TIMES AND NETWORK SIZE OF RELATED METHODS ON DKL-MNIST DATASET 11
VI COMPARATIVE STUDY OF SEVERAL EVOLUTIONARY ALGORITHMS ON DKL-MNIST DATASET 12