Appendix

A. Theorems 1-3 and Proofs

A.1 Theorem 1. When a new task t learns in the TCL, if and only if each task of 1 to $t \leq T$ learning tasks occupies its own independent and mutually weight subspace, which is orthogonal to other weight subspaces of the learned tasks in the whole weight space S_w of the model, and the weight subspace spanned/crossed by the gradient of task t is contained in its weight subspace $\boldsymbol{W}_t$, the learning of the new task t will not interfere with the knowledge (weights $\boldsymbol{W}_{t-1}$) of the previously learned $t-1$ tasks, i.e., there is no CF.

Proof. **1) Necessity**. Given a DNN model, if $\boldsymbol{W}_t$ and $\boldsymbol{W}_{t-1}$ are the orthogonal weight subspaces (i.e., weight matrices) of the model after tasks t and $t-1$ learning respectively, and weight subspace spanned by $\nabla \mathbb{W}_t$ to be contained in $\boldsymbol{W}_t$ in task t learning, then Eq. (1) becomes Eq.(13) as follows:

$$\boldsymbol{W}_t^{i+1} = \boldsymbol{W}_t^i - \lambda \nabla \boldsymbol{W}_t^i, \lambda \nabla \boldsymbol{W}_t^i \in \boldsymbol{W}_t^i \tag{13}$$

where $\boldsymbol{W}_t^{i+1}$ and $\boldsymbol{W}_t^i$ are the $(i+1)$th and ith iteration optimization results of the weight subspace of task t learning respectively, and $\boldsymbol{W}_{t-1}$ remains the original value.

2) Sufficiency. In learning a new task t, if there is no interference in both $\mathbb{W}_t$ and $\mathbb{W}_{t-1}$ in Eq.(1), i.e., no CF issue, which must be $\mathbb{W}_t \cdot \mathbb{W}_{t-1} = 0$ and the projection of $\lambda \nabla \mathbb{W}_t$ on $\mathbb{W}_{t-1}$ is zero. ■

A.2 Theorem 2. The upper bound of the minimum number of learnable tasks of method OWM (Zeng et al., 2019) is $rank(P_{OWM})$, which is the rank of the orthogonal projection operator P_{OWM} formulated as follows:

$$rank(P_{OWM}) = r, r \leq rank(\mathbb{W}) \leq min(m, n) \tag{14}$$

where $\mathbb{W} \in \mathbb{R}^{m \times n}$ is the weight matrix of the model, and $min(m, n)$ is the minimum value of the weight matrix dimensions. See OWM (Zeng et al., 2019) for proof.

A.3 Theorem 3. The upper bound of the minimum number of learnable tasks of the proposed OWS-based TCL method is $\sum_{t=1}^{T} rank(\boldsymbol{B}_t) \gg min(m, n)$, where $rank(\boldsymbol{B}_t)$ is the rank of bases $\boldsymbol{B}_t = \{\boldsymbol{B}_t^l\}_{l=1}^{L}$ (see Eq. (18)) of the weight subspace of task t, and the other symbols are the same as before.

Proof. Based on the optimization theory (Saad, 2003; Shah et al., 1992), as $1 < rank(\boldsymbol{B}_t) \leq min(m, n)$ ($t \in [1, T]$) and $rank(\boldsymbol{B}_t)$ will increase with learning of more tasks, the upper bound of the minimum number of learnable tasks of the proposed OWS-based TCL method at least is

$$\sum_{t=1}^{T} rank(\boldsymbol{B}_t) > 1 + 2+, ..., +min(m, n) = T(2 + min(m, n))/2 \gg min(m, n), T \gg 2 \tag{15}$$

where T is the total member of learned tasks. ■

B. The Architecture and Pipeline of the Proposed SATCL

The proposed method SATCL aims to deal with both CF and KT well in its CL lifespan. To this end, it contains three components embedded in some novel techniques with a KB shown in Fig.3, i.e., Online Validation Dataset Generator, Similar/Dissimilar Tasks Detector, and Dissimilar/Similar Tasks Processor. Each component respectively performs the following functions.

Online Validation Dataset Generator. Using the input dataset $\mathbb{D}_t$ of task t, it generates validation dataset $\mathbb{D}'_t$ online corresponding to the dataset $\mathbb{D}_t$ for similarity/dissimilarity calculation between learning tasks.

Similar/Dissimilar Tasks Detector. By interacting with SATCL's KB, it can determine whether a new task is similar to some previously learned tasks or not, so as to prepare for different processing of similar/dissimilar tasks.

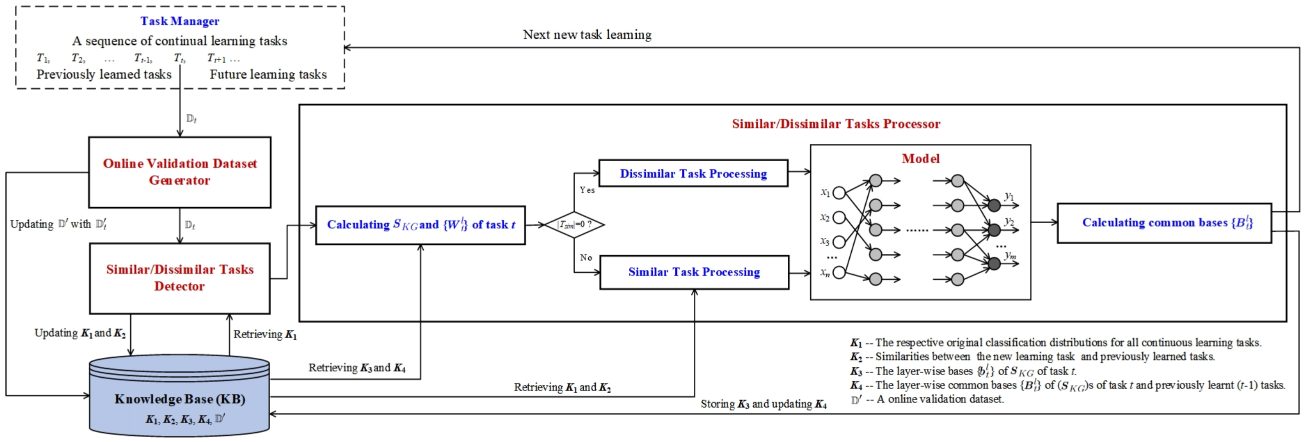
Task Manager
A sequence of continual learning tasks
T_1, T_2, … T_{t-1}, T_t, T_{t+1} …
Previously learned tasks
Future learning tasks
Next new task learning
$\mathbb{D}_t$
Online Validation Dataset Generator
Updating $\mathbb{D}'$ with $\mathbb{D}'_t$
$\mathbb{D}_t$
Similar/Dissimilar Tasks Detector
Updating K_1 and K_2
Retrieving K_1
Knowledge Base (KB)
K_1, K_2, K_3, K_4, $\mathbb{D}'$
Retrieving K_3 and K_4
Retrieving K_1 and K_2
Storing K_3 and updating K_4
Similar/Dissimilar Tasks Processor
Calculating S_{KG} and $\{W_t^l\}$ of task t
$|T_{sim}|=0$?
Yes
No
Dissimilar Task Processing
Similar Task Processing
Model
x_1
x_2
x_3
x_n
y_1
y_2
y_m
Calculating common bases $\{B_t^l\}$
K_1 -- The respective original classification distributions for all continuous learning tasks.
K_2 -- Similarities between the new learning task and previously learned tasks.
K_3 -- The layer-wise bases $\{b_t^l\}$ of S_{KG} of task t.
K_4 -- The layer-wise common bases $\{B_t^l\}$ of (S_{KG})s of task t and previously learnt (t-1) tasks.
$\mathbb{D}'$ -- A online validation dataset.

Figure 3: The architecture and pipeline of the proposed SATCL.

Similar/Dissimilar Tasks Processor. By the proposed OWS-based TCL strategy, with joint optimization of bi-objective loss $\mathcal{L}_{dis}$, it performs dissimilar tasks learning with the minimal or no CF. And by interacting with its KB, the proposed OWS-based TCL strategy and a novel contrastive loss $\mathcal{L}_{sim}$ with constrains, it performs similar tasks learning with adaptive and forward and backward KT.

Notably as the SATCL is a closed-loop system with automatically forward and backward information transfer, it can be continuously optimized or evolved to learn better and better as shown in Tables/Figs. 8-9.

C. The Network Structures, Superparameters, Training Details and Experimental Platform

To test the efficacy and scalability of our method, we use various DNN models/backbones on the five benchmark datasets. We use fully-connected network (FCN) with two hidden layers of 100 units each for PMINIST following Ref. (Lopez-Paz & Ranzato, 2017b). For experiments with CIFAR-100 we use a 5-layer AlexNet similar to Ref. (Serrà et al., 2018b). For experiments with CIFAR-100 Sup we use a 5-layer LeNet-5 (Yoon et al., 2020). For experiments with MiniImageNet and 5-Datasets, similar to Ref. (Chaudhry et al., 2019c), we use a reduced ResNet 18 architecture. No bias units and batch normalization parameters are learned for the first task and shared with all the other tasks following Ref. (Mallya & Lazebnik, 2018). For PMNIST, we evaluate and compare our SATCL in the 'single-head' setting (Hsu et al., 2018; Farquhar & Gal, 2018) where all tasks share the final classifier layer and inference is performed without task hint. For all other experiments, we evaluate our SATCL in the 'multi-head' setting, where each task has a separate classifier. The correspondence between the training dataset and its network structure, as well as the training superparameters used by each network structure, are shown in Table 4.

All of the experiments were conducted on the platform: Intel(R) Xeon(R) Gold 6230 CPU 2.10GHz, 251GB RAM, and GPU - GeForce RTX 2080 Ti with 12GB MC (graphics card Memory Capacity). And all the experimental results are standard deviation values over 5 different runs with the random DNN parameters for each run.

D. Algorithms

D.1 Algorithm 1 LWSimilarity. The calculating the layer-wise similarity between tasks i and t shown in Sec.4.1 Step 4.2 is Algorithm 1 LWSimilarity, which is as follows:

(1) Feeding the validation dataset $\mathbb{D}'$ into $model_{ori}$ and $model_{CL}$ sequentially to obtain their corresponding layer-wise **representations** $\{x'^{l}_{j}\}$ and $\{x^{l}_{j}\}$ of $model_{ori}$ and $model_{CL}$, respectively, where $j \in [1, t]$ and $l \in [1, L-1]$.

(2) Calculating the layer-wise similarity between tasks i and t by metric SDM (Eq. 3). Note that in this case, the distance in Eq. (3) corresponds to the distance between the **representations**.

(3) Outputting the results of the layer-wise similarities between tasks i and t ($i \in [1, t-1]$) to Similar Task Processing shown in Fig. 3. Note that the layer-wise similarity may select different tasks for different layers, which provides a more

Table 4: The Datasets, network architectures and superparameters.

Datasets	Backbone	Batch Size	Epochs	λ	Optimizer	n_s	ϵ_t^l
PMNIST	3-Layers FCN	10	5	0.01	SGD	300	0.95/ 0.99/ 0.99
CIFAR 100	AlexNet	64	100	0.01		125	See Note 1 of the table
CIFAR 100 Sup	LeNet-5	64	100	0.01		125	See Note 2 of the table
MiniImageNet	ResNet-18	64	200	0.10		100	0.985 in all layers
5-Datasets		64	100	0.10		100	0.965 in all layers

1. The initial values for all layers are set to 0.97. When learning the second task, it is set to 0.973, and when learning the third task, it is set to 0.976, that is, the threshold of each layer is increased by 0.003 in turn. When learning the 10th task, the threshold of each layer reaches 0.997.
2. The initial values for all layers are set to 0.98. When learning the second task, it is set to 0.981, and when learning the third task, it is set to 0.982, that is, the threshold is increased by 0.001 layer by layer. When learning the 20th task, the threshold of each layer reaches 0.999.

Algorithm 1 LWSimilarity

Input: The validation dataset $\mathbb{D}'$; the original model $model_{ori}$ and learning model $model_{CL}$ of SATCL;
Output: The set $\{\mathbb{T}_{sim}^l\}_{l=1}^{L-1}$ of similar tasks between task t and previously learned (t-1) tasks;

```
1:  for each task $j \in [1, t]$ do
2:      Feeding the validation dataset $\mathbb{D}'_j$ into $model_{ori}$ and $model_{CL}$, respectively, before learning task $t$;
3:      for each layer $l \in [1, L-1]$ do
4:          Calculating the layer-wise representations $\{x_j'^l\}$ and $\{x_j^l\}$ of $model_{ori}$ and $model_{CL}$, respectively;
5:      end for
6:  end for
7:  for each layer $l \in [1, L-1]$ do
8:      $\mathbb{T}_{sim}^l \leftarrow \emptyset$;
9:  end for
10: for each task $i \in [1, t-1]$ do
11:     for each layer $l \in [1, L-1]$ do
12:         Calculating the $dis'$ and $dis$ between tasks $i$ and $t$ by Wasserstein distance Eq. (4);
13:         // judging the similarity between tasks $i$ and $t$ by metric SDM shown in Eq. (3);
14:         if $dis < dis'$ then
15:             $\mathbb{T}_{sim}^l \leftarrow \mathbb{T}_{sim}^l \cup i$;
16:         end if
17:     end for
18: end for
19: return $\{\mathbb{T}_{sim}^l\}_{l=1}^{L-1}$;
```

fine-grained characterization of task similarity in terms of layer-level features of the model (see Table 5).

D.2 Algorithm 2 basesComputing. Computing the layer-wise $\boldsymbol{B}_t^l$ bases of the **space $\boldsymbol{S^l}$ of significant representation** corresponding to the S_{KG} (see Def.4) of task t. In proposed SATCL, $\boldsymbol{B}_1^l$ and $\boldsymbol{B}_t^l$ ($t \in [2, T]$) are calculated differently as follows:

Calculating $\boldsymbol{B}_1^l$. After learning task 1, first, a layer-wise **representation matrix** $\{\boldsymbol{R}_t^l\}_{l=1}^L$ ($t = 1$, see Def. 1) is constructed using dataset $\mathbb{D}_1$ for each layer of the model with L layers. Second, the SVD on $\boldsymbol{R}_t^l = \boldsymbol{U}_t^l \boldsymbol{\Sigma}_t^l (\boldsymbol{V}_t^l)^\intercal$ ($\in \mathbb{R}^{m \times n}$) by Eq.(5) is calculated. Here, we employ the property of $\boldsymbol{U}_t^l$ and $\boldsymbol{V}_t^l$ being orthogonal to each other to get the two orthogonal submatrices of $\boldsymbol{R}_t^l$, which is corresponding to the original subspaces $\boldsymbol{S}_{KG}$ and $\boldsymbol{S}_{RG}$, respectively. Third, the k-rank approximation $(\boldsymbol{R}_t^l)_k$ of $\boldsymbol{R}_t^l$ by Eq. (6) is calculated with as little loss of information as possible. As the dimensions m and n of $\boldsymbol{R}_t^l$ may be very large, this step can avoid subsequently very complicated matrix operations. In this paper, let ϵ_t^l is a superparameter of the model shown in Table 4 in Appendix C, which is a feasible empirical value. Based on the $(\boldsymbol{R}_t^l)_k$, the **space $\boldsymbol{S^l}$ of significant representation** for task t at layer l can be derived by Eq. (7), i.e., $\boldsymbol{B}_t^l = \boldsymbol{b}_t^l = \{\boldsymbol{u}_{t,1}^l, \boldsymbol{u}_{t,2}^l, ..., \boldsymbol{u}_{t,j}^l, ..., \boldsymbol{u}_{t,k}^l\} (t = 1)$.

Calculating $\boldsymbol{B}_t^l, t \in [2, T]$. The calculations are performed as follows. First, the layer-wise representation matrix $\{\boldsymbol{R}_t^l\}_{l=1}^L$ is constructed using dataset $\mathbb{D}_t$ after learning task t only. Second, the orthogonal projection $\hat{\boldsymbol{R}}_t^l$ of $\boldsymbol{R}_t^l$ for task t is calculated

Algorithm 2 basesComputing

```
Input: The training data $\mathbb{D}_t$ of learning task $t$;
Output: The layer-wise bases $\boldsymbol{b}_t^l$ of task $t$ and common bases $\boldsymbol{B}_t^l$ between task $t$ and previously learned $t-1$ tasks;
 1: $\{\boldsymbol{b}_t^l\} \leftarrow \emptyset$, $\{\boldsymbol{B}_t^l\} \leftarrow \emptyset$;
 2: for each layer $l \in [1, L]$ do
 3:     Calculating its layer-wise representation matrix $\boldsymbol{R}_t^l$ after learning task $t$ (see Def.1);
 4:     if $t == 1$ then
 5:         Calculating the SVD result of $\boldsymbol{R}_t^l$ by Eq.(5);
 6:         Calculating the $k$-rank approximation $(\boldsymbol{R}_t^l)_k$ of $\boldsymbol{R}_t^l$ by Eq. (6);
 7:         Selecting the top-$k$ vectors of $\boldsymbol{U}_t^l$ from $(\boldsymbol{R}_t^l)_k$ to construct $\boldsymbol{b}_t^l$;
 8:         $\boldsymbol{B}_t^l \leftarrow \boldsymbol{b}_t^l$;
 9:     else
10:         Calculating the orthogonal projection $\hat{\boldsymbol{R}}_t^l$ of $\boldsymbol{R}_t^l$ by Eq.(16);
11:         Calculating the SVD result of $\hat{\boldsymbol{R}}_t^l$ by Eq.(5);
12:         The $k$ new orthogonal bases for the minimum value of $k$ is selected for satisfying Eq. (17);
13:         Selecting the top-$k$ vectors of $\hat{\boldsymbol{U}}_t^l$ from $(\hat{\boldsymbol{R}}_t^l)_k$ to construct $\boldsymbol{b}_t^l$;
14:         $\{\boldsymbol{b}_t^l\} \leftarrow \{\boldsymbol{b}_t^l\} \bigcup \boldsymbol{b}_t^l$;
15:         Calculating the common bases $\boldsymbol{B}_t^l$ of task $t$ and previously learnt $t-1$ tasks by Eq. (18)
16:         $\{\boldsymbol{B}_t^l\} \leftarrow \{\boldsymbol{B}_t^l\} \bigcup \boldsymbol{B}_t^l$;
17:     end if
18: end for
19: Storing $\{\boldsymbol{b}_t^l\}$ and $\{\boldsymbol{B}_t^l\}$ to the KB of SATCL;
20: return $\{\boldsymbol{B}_t^l\}$;
```

as follows:

$$\hat{\boldsymbol{R}}_t^l = \boldsymbol{R}_t^l - \boldsymbol{R}_t^l \boldsymbol{B}_{t-1}^l (\boldsymbol{B}_{t-1}^l)^\intercal = \boldsymbol{R}_t^l - \boldsymbol{R}_{t,Proj}^l \tag{16}$$

Third, the SVD on $\hat{\boldsymbol{R}}_t^l$ is calculated by Eq. (5) and the k new orthogonal bases for the minimum value of k are selected for satisfying the following inequality:

$$||\boldsymbol{R}_{t,Proj}^l||_F^2 + ||(\hat{\boldsymbol{R}}_t^l)_k||_F^2 \geq \epsilon_t^l ||\boldsymbol{R}_t^l||_F^2 \tag{17}$$

where the ϵ_t^l is a superparameter of the model shown in Table 4.

Forth, based on the calculated $(\hat{\boldsymbol{R}}_t^l)_k$ by Eq. (17), the layer-wise bases $\boldsymbol{b}_t^l = \{\hat{\boldsymbol{u}}_{t,1}^l, \hat{\boldsymbol{u}}_{t,2}^l, ..., \hat{\boldsymbol{u}}_{t,k}^l\}$ $(l \in [1, L])$ of task t is obtained and stored to KB. As the common bases $\boldsymbol{B}_t^l$ of both $\boldsymbol{B}_{t-1}^l$ and $\boldsymbol{b}_t^l$ must be updated to ensure the newly added bases are unique and orthogonal to the existing bases $\boldsymbol{B}_{t-1}^l$, the new bases $\boldsymbol{b}_t^l$ is added to $\boldsymbol{B}_{t-1}^l$ as follows for next iteration:

$$\boldsymbol{B}_t^l = \{\boldsymbol{B}_{t-1}^l, \boldsymbol{b}_t^l\}, l \in [1, L] \tag{18}$$

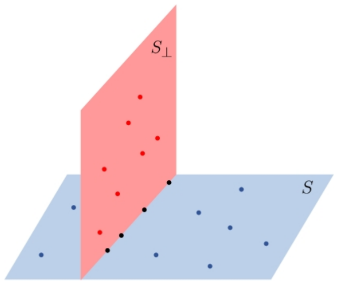

Figure 4: Schematic diagram of two orthogonal spaces with some data overlap in their orthogonal section.

$$\begin{array}{c|ccccc}
 & 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 1.00 & & & & \\
2 & 0.90 & 1.00 & & & \\
3 & 0.53 & 0.47 & 1.00 & & \\
4 & 0.00 & 0.00 & 0.00 & 1.00 & \\
5 & 0.52 & 0.00 & 0.48 & 0.00 & 1.00
\end{array}$$

(a)

$$\begin{array}{c|cccccccccc}
 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\
\hline
1 & 1.00 & & & & & & & & & \\
2 & 0.52 & 1.00 & & & & & & & & \\
3 & 0.47 & 0.53 & 1.00 & & & & & & & \\
4 & 0.00 & 0.00 & 0.00 & 1.00 & & & & & & \\
5 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & & & & & \\
6 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & & & & \\
7 & 0.00 & 0.49 & 0.51 & 0.00 & 0.00 & 0.00 & 1.00 & & & \\
8 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & & \\
9 & 0.00 & 0.00 & 0.48 & 0.00 & 0.00 & 0.00 & 0.52 & 0.00 & 1.00 & \\
10 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00
\end{array}$$

(b)

Figure 5: The task-wise tasks' similarity matrix. (a) on 5-Datasets (5 tasks) and (b) on CIFAR-100 (10 tasks).

Algorithm 3 SATCL

```
Input: The training dataset $\mathbb{D} = \{\mathbb{D}_t\}_{t=1}^{T}$; the original model $model_{ori}$ and learning model $model_{CL}$ of SATCL;
Output: The learned model $model_{CL}$ after learning all tasks;
 1: for each task $t \in [1, T]$ do
 2:     Generating the validation dataset $\mathbb{D}'_t$ of $\mathbb{D}_t$ online by Online Validation Dataset Generator of SATCL;
 3:     if $t == 1$ then
 4:         Training $model_{CL}$ on $\mathbb{D}_t$ by using the cross-entropy loss function without imposing any constraint on weight
            updates of the model;
 5:     else
 6:         Calculating the layer-wise OWS: $\boldsymbol{W}_t^l$ of task $t$ by $\boldsymbol{W}_t^l = \boldsymbol{W}_{t-1}^l - \boldsymbol{W}_{t-1}^l \boldsymbol{B}_{t-1}^l (\boldsymbol{B}_{t-1}^l)^\intercal$ in $model_{CL}$
 7:         $\{\mathbb{T}_{sim}^l\}_{l=1}^{L-1} \leftarrow$ Calculating the layer-wise similar/dissimilar tasks between task $t$ and previously learnt $t-1$ tasks
            by Algorithm 1 LWSimilarity embedded in Similar/Dissimilar Tasks Detector of SATCL;
 8:         if $\{\mathbb{T}_{sim}^l\}_{l=1}^{L-1} == \varnothing$ then
 9:             Setting $\mathcal{L}_{dis}$ calculated by Eq. (10) as the loss function of $model_{CL}$;
10:         else
11:             Setting $\mathcal{L}_{sim}$ calculated by Eq. (11) as the loss function of $model_{CL}$ for preparing knowledge transfer (KT);
12:         end if
13:         Training $model_{CL}$ on $\mathbb{D}_t$ with Similar/Dissimilar Tasks Processor of SATCL, where the learning takes the
            gradients of task $t$ in the directions orthogonal to $S_{KG}$ of the previous $t$-1 tasks by Eq. (8) to avoid CF, and
            performs weight updates of the model by Eq. (9);
14:     end if
15:     Calculating the layer-wise common bases $\{\boldsymbol{B}_t^l\}$ between task $t$ and previously learned $t$-1 tasks by Algorithm 2
        basesComputing for next new task learning;
16: end for
17: return $model_{CL}$;
```

D.3 Algorithm 3 SATCL. See Fig.3 for the calculation procedure of Algorithm 3 SATCL, and SATCL source code is available at https://github.com/satclalg/SATCL.

E. Additional Experimental Results

E.1 ACC and BWT Performances of the Proposed SATCL of Task-wise vs. Layer-wise on Five Datasets

Table 5: ACC and BWT performances of the proposed SATCL of Task-wise vs. Layer-wise on five datasets.

Datasets	PMNIST		CIFAR 100		CIFAR 100 Sup		MiniImageNet		5-Datasets	
	10 Tasks		10 Tasks		20 Tasks		20 Tasks		5 Tasks	
Methods	ACC(%)	BWT	ACC(%)	BWT	ACC(%)	BWT	ACC(%)	BWT	ACC(%)	BWT
Task-wise SATCL	94.79 $\pm$ 0.04	-0.03 $\pm$ 0.01	74.27 $\pm$ 0.14	-0.01 $\pm$ 0.01	58.93 $\pm$ 0.02	-0.01 $\pm$ 0.01	61.52 $\pm$ 0.36	-0.02 $\pm$ 0.01	92.01 $\pm$ 0.07	-0.02 $\pm$ 0.01
Layer-wise SATCL	**95.96** $\pm$ 0.03	**-0.02 $\pm$ 0.01**	**75.57** $\pm$ 0.13	**-0.00** $\pm$ 0.00	**60.35** $\pm$ 0.01	**-0.01** $\pm$ 0.00	**62.64** $\pm$ 0.31	**-0.01**$\pm$ 0.01	**93.14** $\pm$ 0.06	**-0.01** $\pm$ 0.01
Gain of Layer-wise	1.17 $\pm$ 0.01	0.01 $\pm$ 0.00	1.30 $\pm$ 0.01	0.01 $\pm$ 0.01	1.42 $\pm$ 0.01	0.00 $\pm$ 0.01	1.12 $\pm$ 0.05	0.01 $\pm$ 0.00	1.13 $\pm$ 0.01	0.01 $\pm$ 0.00

As the tasks in different datasets may have similarities shown in Fig. 5, we should carry out forward KT to assist task t learning if a new task t is similar to some of the previously learnt t-1 tasks. Moreover, we have known that the layer-wise similarity may select different tasks for different layers, which provides a more fine-grained characterization of task similarity in terms of layer-level features of the model. To show the efficacy of Layer-wise tasks similarity comparison, we conducted the experiments for ACC and BWT performances with the standard deviation values over 5 different runs of the proposed SATCL of Task-wise vs. Layer-wise on five datasets. As shown in Table 5, the Layer-wise SATCL (i.e., SATCL) can achieve an average ACC/BWT gain of 1.23%/0.01 over the Task-wise SATCL on all five datasets, which further confirms that the similarity comparison of Layer-wise tasks is superior to that of Task-wise tasks in terms of improving ACC/BWT performances.

E.2 The Accuracy Evolution of Task t of the Proposed SATCL and Baselines on a Variety of Datasets.

To show the BWT and/or knowledge transfer (KT) performance of the proposed SATCL and the-state-of-art baselines, we randomly selected a task on different datasets to carry out its accuracy (denoted by Acc (%), the average Acc result of 5 different runs) evolution experiments of the task. The experimental results shown in Tables 6-9 and Figs. 6-9 demonstrate the following:

(1) Compared with the baselines, the proposed SATCL has stronger resistance to forgetting than that of the baselines while maintaining high accuracy, which is due to the proposed OWS-based TCL strategy of SATCL (see Tables/Figs. 6-9).

(2) More interestingly, benefited from the KT with its $\mathcal{L}_{sim}$ shown in Eq. (11), the proposed SATCL has the trend of increasing accuracy instead of decreasing accuracy in terms of a task t as the number of learning tasks increases (see Tables/Figs. 8-9). It is worth noting that this phenomenon also appears in baselines CAT and TRGP with KT ability (see Table 9/Fig. 9), which further proves the necessity and importance of KT in CL as noted in (Ke et al., 2020).

In a word, the above experimental results not only show the effectiveness of the proposed SATCL in resisting forgetting and performing KT, but also verify the superiority of SATCL over the baselines in ACC, BWT and KT performances.

Table 6: Accuracy evolution of task 5 on PMNIST.

Dataset	**PMNIST**					
Learnt Tasks	5	6	7	8	9	10
Method	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)
LwF	95.8	94.1	92.2	88.8	85.1	83.7
GPM	97.0	96.4	95.8	94.8	94.1	92.5
SupSup	91.7	91.7	91.7	91.7	91.7	91.7
CAT	93.3	93.3	93.3	93.3	93.3	93.3
TRGP	**97.2**	**97.1**	**97.1**	**97.1**	**96.8**	**96.9**
SATCL(Ours)	96.8	96.7	96.5	96.3	96.1	96.0

Table 7: Accuracy evolution of task 1 on 5-Datasets.

Dataset	**5-Datasets**				
Learnt Tasks	1	2	3	4	5
Method	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)
LwF	76.4	72.2	70.9	70.3	68.9
GPM	76.1	75.7	72.5	72.6	72.4
SupSup	72.3	71.3	71.3	71.3	71.3
CAT	59.8	58.7	57.2	57.2	57.2
TRGP	75.8	75.8	75.9	75.8	75.8
SATCL(Ours)	**82.2**	**81.5**	**79.8**	**79.7**	**79.5**

Table 8: Accuracy evolution of task 1 on dataset CIFAR 100.

Dataset	**CIFAR 100**									
Learnt Tasks	1	2	3	4	5	6	7	8	9	10
Method	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)
LwF	75.3	71.8	71.4	69.7	68.4	67.6	64.1	63.4	59.3	59.4
GPM	77.6	77.2	77.5	75.4	75.3	75.7	76.1	76.3	75.8	75.7
SupSup	61.1	61.1	61.1	61.1	61.1	61.1	61.1	61.1	61.1	61.1
CAT	60.2	60.1	60.2	60.1	60.1	60.3	60.1	60.2	60.1	60.1
TRGP	75.8	75.1	75.1	74.7	75.5	75.6	75.9	75.5	75.4	75.3
SATCL(Ours)	**78.2**	**76.7**	**77.6**	**78.4**	**78.2**	**80.1**	**77.4**	**78.1**	**79.3**	**79.2**

Table 9: Accuracy evolution of task 6 on dataset CIFAR 100 Sup.

Dataset	**CIFAR 100 Sup**														
Learnt Tasks	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Method	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)	Acc(%)
LwF	59.8	59.6	60.2	57.2	57.2	55.8	56.5	56.2	53.8	54.8	55.8	55.2	55.2	53.6	52.6
GPM	62.2	61.2	61.4	62.6	61.2	59.8	60.2	59.2	60.3	60.4	58.4	59.6	58.8	58.4	58.6
SupSup	53.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8	50.8
CAT	51.2	47.5	47.5	47.5	47.5	47.5	47.5	47.5	47.5	50.2	50.2	50.2	50.2	50.2	50.2
TRGP	62.6	62.1	62.4	**62.2**	61.6	**61.6**	60.2	61.2	60.4	60.2	60.4	**61.2**	61.8	61.6	61.2
SATCL(Ours)	**62.8**	**63.2**	**62.6**	61.6	**61.8**	**61.6**	**62.6**	**62.7**	**61.6**	**62.4**	**61.6**	**61.2**	**62.3**	**62.6**	**62.4**

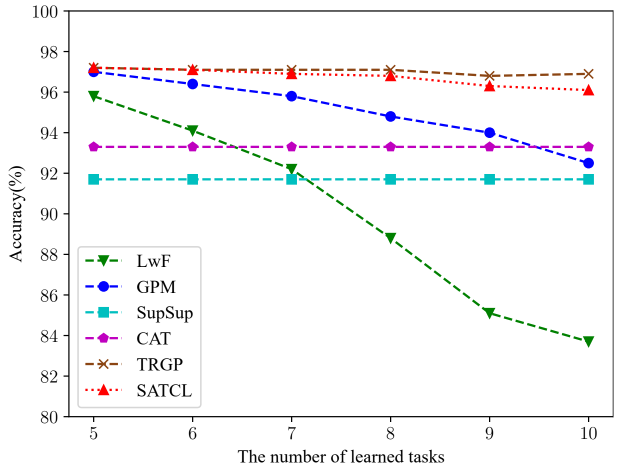

Figure 6: Schematic diagram of the data in Table 6.

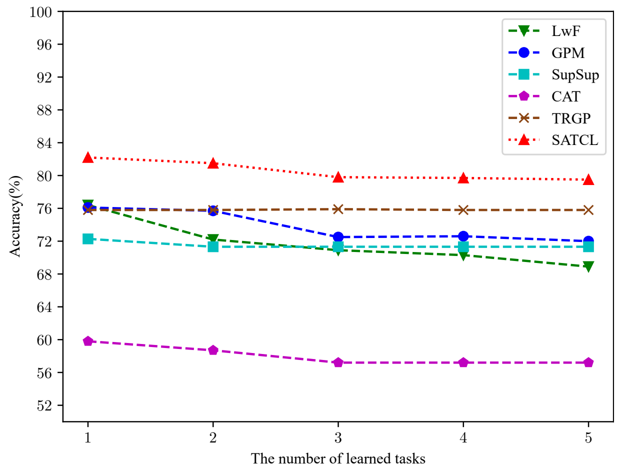

Figure 7: Schematic diagram of the data in Table 7.

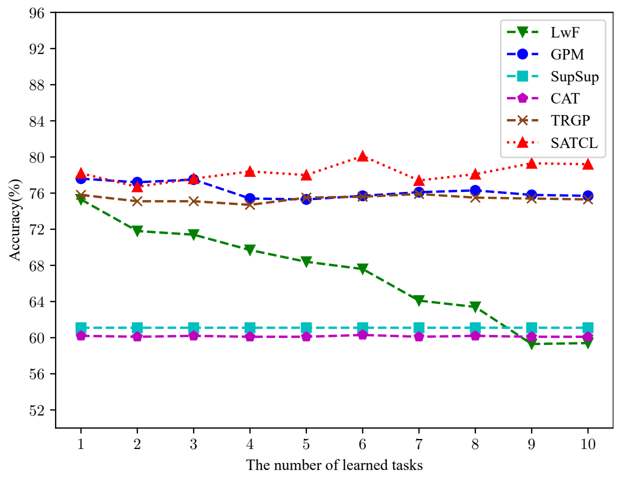

Figure 8: Schematic diagram of the data in Table 8.

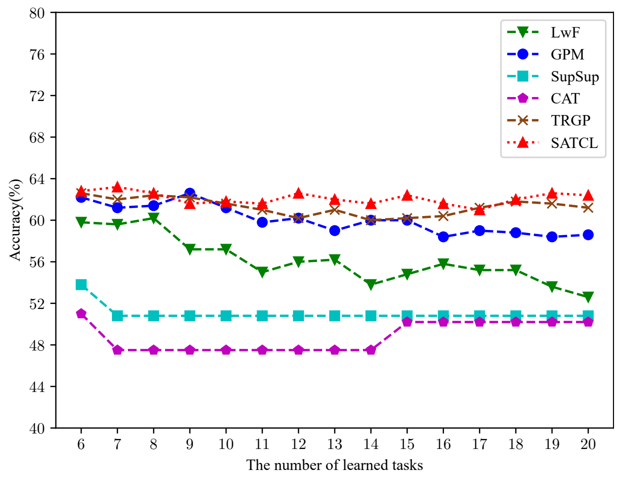

Figure 9: Schematic diagram of the data in Table 9.