Ethics and Reproducibility Statements

We have followed the ICLR Code of Ethics in our work.

For the reproducibility of the proposed STIL system, the source code can be downloaded from https://github.com/STILalg/STIL.

Appendix

CONTENTS

A THEOREMS 1-4 AND PROOFS

A.1 Theorem 1. The learning of the new task $t \leq T$ will not interfere with the knowledge (weights $\boldsymbol{W}_{t-1}$) of the previous learned $t-1$ tasks (i.e., there is no CF) *if and only if* 1) task t occupies its own independent weight subspace $\boldsymbol{W}_t$. that is orthogonal to all other weight subspaces of the previous tasks in the whole weight space $\boldsymbol{S}_w$ of the model, and 2) the modified weights spanned by the orthogonal gradient to the previous learned tasks are imposed in its weight subspace $\boldsymbol{W}_t$ of task t.

1) Sufficiency

Proof. Given a DNN model, if $\boldsymbol{W}_t$ and $\boldsymbol{W}_{t-1}$ (i.e., weight matrices of task t and $t-1$) are orthogonal weight subspaces of the model after learning tasks t and $t-1$ respectively, and if the weight subspace spanned by $\triangledown \mathbb{W}_t$ is $\boldsymbol{W}_t$ in learning task t, then Eq. (1) becomes Eq.(12) as follows:

$$\boldsymbol{W}_t^{i+1} = \boldsymbol{W}_t^i - \lambda \triangledown \boldsymbol{W}_t^i, \lambda \triangledown \boldsymbol{W}_t^i \in \boldsymbol{W}_t^i \tag{12}$$

where λ is the learning rate, and $\boldsymbol{W}_t^{i+1}$ and $\boldsymbol{W}_t^i$ are the $(i+1)$th and ith iteration optimization results of the weight subspace in learning task t respectively. Eq.(12) means that $\boldsymbol{W}_t$ has no interference with $\boldsymbol{W}_{t-1}$ of the previous learned task $t-1$ in the new task t learning. □

2) Necessity

Proof. In learning a new task t, if there is no interference in $\boldsymbol{W}_t$ and $\boldsymbol{W}_{t-1}$, i.e., no CF issue, which means $\boldsymbol{W}_t \cdot \boldsymbol{W}_{t-1} = 0$ and the projection of $\lambda \triangledown \boldsymbol{W}_t$ on $\boldsymbol{W}_{t-1}$ is zero, that is $\boldsymbol{W}_{t-1}^i - \lambda \triangledown \boldsymbol{W}_t^i = \boldsymbol{W}_{t-1}^i - 0$ (λ is the learning rate). Thus $\boldsymbol{W}_{t-1}^i$ of task $t-1$ would not be changed in the learning of task t. □

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:

$$rank(P_{OWM}) = r, r \leq rank(\mathbb{W}) \leq min(m, n) \tag{13}$$

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 OIWS-based TIL method is $\sum_{t=1}^{T} rank(\boldsymbol{B}_t) \gg min(m, n)$, where $rank(\boldsymbol{B}_t)$ is the rank of the bases $\boldsymbol{B}_t = \{\boldsymbol{B}_t^l\}_{l=1}^L$ (see Eq. (21)) of the weight subspace of task t).

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 the learning of more tasks, the upper bound of the minimum number of learnable tasks of the proposed OIWS-based TIL method is at least

$$\sum_{t=1}^{T} rank(\boldsymbol{B}_t) > 1 + 2 +, ..., + min(m, n) = T(2 + min(m, n))/2 \gg min(m, n), T \gg 2 \tag{14}$$

where T is the total member of the learned tasks. □

A.4 Theorem 4. The theoretical bounds for FWT and BWT of tasks i and t ($i < t$, $i \in [1, T-1]$, $t \in [2, T]$) in TIL are as follows.

$$\begin{cases} FWT : \epsilon_t(h) \leq \epsilon_i(h) + d(\mathcal{D'}_i, \mathcal{D'}_t) + \min\{\mathbb{E}_{\mathcal{D'}_{t-1}}\left[|l_i(\mathbf{x}) - l_t(\mathbf{x})|\right], \mathbb{E}_{\mathcal{D'}_t}\left[|l_i(\mathbf{x}) - l_t(\mathbf{x})|\right]\} \\ \\ BWT : \epsilon_i(h) \leq \epsilon_t(h) + d(\mathcal{D'}_i, \mathcal{D'}_t) + \min\{\mathbb{E}_{\mathcal{D'}_i}\left[|l_i(\mathbf{x}) - l_t(\mathbf{x})|\right], \mathbb{E}_{\mathcal{D'}_t}\left[|l_i(\mathbf{x}) - l_t(\mathbf{x})|\right]\} \end{cases}$$

Proof. Recall that $\epsilon_t(h) = \epsilon_t(h, l_t)$ and $\epsilon_i(h) = \epsilon_i(h, l_i)$. Let g_i and g_t be the density functions of ${\mathcal{D}'}_i$ and ${\mathcal{D}'}_t$ respectively. For the theoretical bound of FWT,

$$\begin{aligned}
\epsilon_t(h) &= \epsilon_t(h) + \epsilon_i(h) - \epsilon_i(h) + \epsilon_i(h, l_t) - \epsilon_i(h, l_t) \\
&\leq \epsilon_i(h) + |\epsilon_i(h, l_t) - \epsilon_i(h, l_i)| + |\epsilon_t(h, l_t) - \epsilon_i(h, l_t)| \\
&\leq \epsilon_i(h) + \mathbb{E}_{{\mathcal{D}'}_i}[|l_i(\mathbf{x}) - l_t(\mathbf{x})|] + |\epsilon_t(h, l_t) - \epsilon_i(h, l_t)| \\
&\leq \epsilon_i(h) + \mathbb{E}_{{\mathcal{D}'}_i}[|l_i(\mathbf{x}) - l_t(\mathbf{x})|] + \int |g_i(\mathbf{x}) - g_t(\mathbf{x})||h(\mathbf{x}) - l_t(\mathbf{x})| d\mathbf{x} \\
&\leq \epsilon_i(h) + \mathbb{E}_{{\mathcal{D}'}_i}[|l_i(\mathbf{x}) - l_t(\mathbf{x})|] + d_1({\mathcal{D}'}_i, {\mathcal{D}'}_t).
\end{aligned} \tag{15}$$

For the theoretical bound of BWT, in the first line of the above Eq. (15), we could instead choose to add and subtract $\epsilon_t(h, l_t)$ rather than $\epsilon_i(h, l_t)$, which would result in the same bound only with the expectation taken with respect to ${\mathcal{D}'}_t$ instead of ${\mathcal{D}'}_i$. Choosing the smaller of the two gives us the bound of BWT. □

B Network Architectures, hyperparameters, Training Details, and Datasets

Network Architectures and hyperparameters. To test the efficacy and scalability of our method, we use various DNN models/backbones on the 11 benchmark datasets. We use a 3-layer fully connected network (FCN) with two hidden layers of 100 units each for PMINIST, F-CelebA-1, F-CelebA-2, (EMNIST, F-EMNIST-1) and (CIFAR 100, F-CelebA-1) following (Lopez-Paz & Ranzato, 2017) and (LeCun et al., 1998). For experiments with CIFAR 100 we use a 5-layer AlexNet similar following (Serrà et al., 2018). For experiments with CIFAR-100 Sup, we use a 5-layer LeNet-5 (Yoon et al., 2020). For experiments with MiniImageNet, 5-Datasets, F-EMNIST-1 and F-EMNIST-2, similar to (Chaudhry et al., 2019), 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 (Mallya & Lazebnik, 2018). For PMNIST, F-CelebA-1 and F-CelebA-2, we evaluate and compare our STIL 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 STIL in the 'multi-head' setting, where each task has a separate head or classifier. The correspondence between the training dataset and its network structure, as well as the training hyperparameters used by each network structure, are shown in Table 4.

Datasets Details. Eleven benchmark image classification datasets are used in our experiments, which are divided into the following categories:

–Dissimilar tasks datasets. (1.1). PMNIST (Permuted MNIST, 10 tasks) (Lecun et al., 1998). It is a variant of MNIST dataset where each task is considered as a random permutation of the original MNIST pixels. We create 10 sequential tasks using different permutations where each task has 10 classes. (1.2). CIFAR-100 (10 tasks) (Krizhevsky & Hinton, 2009). It is constructed by randomly splitting 100 classes of CIFAR-100 (Krizhevsky & Hinton, 2009) into 10 tasks with 10 classes per task. (1.3). CIFAR 100 Sup (20 tasks) (Krizhevsky & Hinton, 2009): It is constructed by splitting 100 classes of CIFAR 100 into 20 tasks with 5 classes of the same attributes per task. (1.4). MiniImageNet (20 tasks) (Vinyals et al., 2016): It is constructed by splitting 100 classes of miniImageNet into 20 sequential tasks where each task has 5 classes. (1.5). 5-Datasets (5 tasks) (Ebrahimi et al., 2020): It includes CIFAR-10, MINIST, SVHN (Netzer et al., 2011), notMNIST (Bulatov, 2011) and Fashion MNIST (Xiao et al., 2017), where the classification of each dataset is considered as a task.

–Similar tasks datasets. (2.1) F-EMINIST-1 and (2.2) F-EMINIST-2 (10/35 tasks). They are similar task datasets from *federated learning*, which are constructed by randomly choosing 10/35 tasks from two publicly available federated learning datasets (Caldas et al., 2018). (2.3) F-CelebA-1 and (2.4) F-CelebA-2 (10/20 tasks). They are also similar task datasets from *federated learning*, which are constructed by randomly choosing 10/20 tasks from two publicly available federated learning datasets (Caldas et al., 2018). Each of the 10/20 tasks contains images of a celebrity labeled by whether he/she is smiling or not. Note that for the four datasets (2.1)-(2.4), the training and testing sets are already provided in (Caldas et al., 2018). We further split about 10% of the original training set and kept it for validation purposes.

–Mixed tasks datasets. (3.1) (EMNIST, F-EMNIST-1) (20 tasks). It is a randomly mixed similar and dissimilar task sequences constructed from EMNIST (LeCun et al., 1998) and F-EMNIST-1. (3.2) (CIFAR-100, F-CelebA-1)(20 tasks). It is a randomly mixed similar and dissimilar task sequences constructed from CIFAR-100 (10 tasks) and F-EMNIST-1 (10 tasks).

The sample sizes of the training/validation/testing are as follows: (1.1) PMNIST 6000 / 300 / 700, (1.2) CIFAR100 5000/300/700, (1.3) CIFAR 100 Sup 5000 / 300 / 700, (1.4) MiniImageNet 5000 / 200 / 800, and (1.5) 5-Datasets, which. Ihas 5 tasks in total, and the samples of per task are 50000 / 10000 / 10000, 50000 / 10000 / 10000, 63257 / 10000 / 26032, 50000 / 10000 / 10000, and 10000/6854/1872 respectively.

Running Environment. 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 seeds.

Table 4: Datasets, network architectures and hyperparameters.

Datasets	**Backbone**	**Batch Size**	**Epochs**	λ	**Optimizer**	ϵ_t^l	θ
PMNIST	3-Layers FCN	10	5	0.01	SGD	0.95/ 0.99/ 0.99	0.75
CIFAR 100	AlexNet	64	100	0.01	SGD	0.97 in all layers	0.75
CIFAR 100 Sup	LeNet-5	64	100	0.01	SGD	0.98 in all layers	0.75
MiniImageNet	ResNet 18	64	200	0.10	SGD	0.985 in all layers	0.80
5-Datasets	ResNet 18	64	100	0.10	SGD	0.965 in all layers	0.95
F-EMNIST-1	ResNet 18	64	50	0.10	SGD	0.965 in all layers	0.80
F-EMNIST-2	ResNet 18	64	50	0.10	SGD	0.965 in all layers	0.80
F-CelebA-1	3-Layers FCN	64	50	0.01	SGD	0.95/ 0.99/ 0.99	0.75
F-CelebA-2	3-Layers FCN	64	50	0.01	SGD	0.95/ 0.99/ 0.99	0.70
(EMNIST, F-EMNIST-1)	3-Layers FCN	64	50	0.01	SGD	0.95/ 0.99/ 0.99	0.70
(CIFAR 100, F-CelebA-1)	3-Layers FCN	64	50	0.01	SGD	0.95/ 0.99/ 0.99	0.70

[1] ϵ_t^l is a feasible empirical threshold for the matrix approximation in Eq. (18), and θ is the distance threshold in Eq.(11) on each dataset.

C ALGORITHMS

C.1 Algorithm 1 LWSimilarity. It calculates the layer-wise similarity between tasks i and t shown in **Step 4**, Sec.4.3, which is as follows:

(1) Feeding the sampled repay data $\mathbb{D}'$ into $model_{ori}$ and $model_{CL}$ sequentially to obtain their corresponding layer-wise **representations** $\{x_j'^l\}$ and $\{x_j^l\}$ 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. 11). Note that in this case, the distance in Eq.(10) corresponds to the distance between the **representations** of tasks i and t.

(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 Figure 1(a). Note 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 (see Table 6).

C.2 Algorithm 2 basesComputing

Definition 1 (Layer-wise representation and representation matrix $\boldsymbol{R}_t^l$). Given a DNN model with L layers, for the input data $\boldsymbol{x}_{t,i}$ of task t, denote $\boldsymbol{x}_{t,i}^l$ as the input of layer l, namely the **representation** of $\boldsymbol{x}_{t,i}$ at layer l, and $\boldsymbol{x}_{t,i}^1 = \boldsymbol{x}_{t,i}$. The output $\boldsymbol{x}_{t,i}^{l+1}$ for layer l is computed by $\boldsymbol{x}_{t,i}^{l+1} = f(\boldsymbol{W}_t^l, \boldsymbol{x}_{t,i}^l)$, where $f(\cdot)$ is the mapping function of the network layer. When learning task t, we only have access to dataset $\mathbb{D}_t$. The **representation matrix** of layer l is defined as $\boldsymbol{R}_t^l = [\boldsymbol{x}_{t,1}^l, \boldsymbol{x}_{t,2}^l, ..., \boldsymbol{x}_{t,n_s}^l]$ for task t, which can be constructed by concatenating n_s (which is a number of sampled data from the training

Algorithm 1 LWSimilarity

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

for each task $j \in [1, t]$ **do**
  Feeding the validation dataset $\mathbb{D}'_j$ into $model_{ori}$ and $model_{CL}$, respectively, before learning task t;
  for each layer $l \in [1, L-1]$ **do**
    Calculating the layer-wise **representations** $\{x_j'^{l}\}$ and $\{x_j^{l}\}$ of $model_{ori}$ and $model_{CL}$, respectively;
  end for
end for
for each layer $l \in [1, L-1]$ **do**
  $\mathbb{T}_{sim}^{l} \leftarrow \emptyset$;
end for
for each task $i \in [1, t-1]$ **do**
  for each layer $l \in [1, L-1]$ **do**
    Calculating the dis' and dis between tasks i and t by Wasserstein distance (Panaretos & Zemel, 2019) and Eq. (10);
    // judging the similarity between tasks i and t by metric SDM shown in Eq. (11);
    if $dis < dis'$ and $|dis - dis'| < \theta$ **then**
      $\mathbb{T}_{sim}^{l} \leftarrow \mathbb{T}_{sim}^{l} \cup i$;
    end if
  end for
end for
return $\{\mathbb{T}_{sim}^{l}\}_{l=1}^{L-1}$;

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

$\{\boldsymbol{b}_t^l\} \leftarrow \emptyset, \{\boldsymbol{B}_t^l\} \leftarrow \emptyset$;
for each layer $l \in [1, L]$ **do**
  Calculating its layer-wise representation matrix $\boldsymbol{R}_t^l$ after learning task t (see Def.1);
  if $t == 1$ **then**
    Calculating the SVD result of $\boldsymbol{R}_t^l$ by Eq.(17);
    Calculating the k-rank approximation $(\boldsymbol{R}_t^l)_k$ of $\boldsymbol{R}_t^l$ by Eq. (18);
    Selecting the top-k vectors of $\boldsymbol{U}_t^l$ from $(\boldsymbol{R}_t^l)_k$ to construct $\boldsymbol{b}_t^l$;
    $\boldsymbol{B}_t^l \leftarrow \boldsymbol{b}_t^l$;
  else
    Calculating the orthogonal projection $\hat{\boldsymbol{R}}_t^l$ of $\boldsymbol{R}_t^l$ by Eq.(19);
    Calculating the SVD result of $\hat{\boldsymbol{R}}_t^l$ by Eq.(17);
    The k new orthogonal bases for the minimum value of k is selected for satisfying Eq. (20);
    Selecting the top-k vectors of $\hat{\boldsymbol{U}}_t^l$ from $(\hat{\boldsymbol{R}}_t^l)_k$ to construct $\boldsymbol{b}_t^l$;
    $\{\boldsymbol{b}_t^l\} \leftarrow \{\boldsymbol{b}_t^l\} \bigcup \boldsymbol{b}_t^l$;
    Calculating the common bases $\boldsymbol{B}_t^l$ of task t and previously learnt $t-1$ tasks by Eq. (21)
    $\{\boldsymbol{B}_t^l\} \leftarrow \{\boldsymbol{B}_t^l\} \bigcup \boldsymbol{B}_t^l$;
  end if
end for
Storing $\{\boldsymbol{b}_t^l\}$ and $\{\boldsymbol{B}_t^l\}$ to the KB of STIL;
return $\{\boldsymbol{B}_t^l\}$;

dataset) representations along the column obtained from the forward pass of n_s random samples from the current training dataset through the model after learning task t.

Definition 2 ($\boldsymbol{S}_{KG}$ and $\boldsymbol{S}_{RG}$). Given a gradient space $\boldsymbol{S}_g$ for a DNN model, we divide $\boldsymbol{S}_g$ into two orthogonal subspaces: Core/Key Gradient Space (denoted by $\boldsymbol{S}_{KG}$) and Residual Gradient Space

(denoted by $\boldsymbol{S}_{RG}$). $\boldsymbol{S}_{KG}$ and $\boldsymbol{S}_{RG}$ have the properties: (1) they are orthogonal and complementary subspaces to each other, and (2) let $\boldsymbol{S}_{KG}$ be the gradient subspace spanned by important gradients of previously learned $t-1$ tasks. So, for a new task t learning, its gradient steps along $\boldsymbol{S}_{KG}$ would induce a high level of interference on the learned tasks, whereas gradient steps along $\boldsymbol{S}_{RG}$ have minimum CF on the learned tasks.

The calculation process of the Algorithm 2 basesComputing is as follows.

Computing the layer-wise $\boldsymbol{B}_t^l$ bases of the **space $\boldsymbol{S^l}$ of significant representation** corresponding to the S_{KG} of task t. In proposed STIL, $\boldsymbol{B}_1^l$ 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.(17) 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. (18) 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 be a hyperparameter 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. (16), i.e., $\boldsymbol{B}_t^l = \boldsymbol{b}_t^l = \{\boldsymbol{u}_{t,1}^l, \boldsymbol{u}_{t,2}^l, ..., \boldsymbol{u}_{t,j}^l, ..., \boldsymbol{u}_{t,k}^l\} (t=1)$.

$$\boldsymbol{S^l} = span(\boldsymbol{b}_t^l = \{\boldsymbol{u}_{t,1}^l, \boldsymbol{u}_{t,2}^l, ..., \boldsymbol{u}_{t,j}^l, ..., \boldsymbol{u}_{t,k}^l\}),\ \boldsymbol{u}_{t,j}^l \in \boldsymbol{U}_t^l \in (\boldsymbol{R}_t^l)_k, j \in [1,k] \tag{16}$$

Calculating $\boldsymbol{B}_t^l (t \in [2,T])$. For clarity, we first introduce following definitions.

Definition 3 (The SVD of $\boldsymbol{R}_t^l$). Based on Singular Value Decomposition (SVD) (Deisenroth et al., 2020), the SVD of $\boldsymbol{R}_t^l$ for task t is defined as:

$$\boldsymbol{R}_t^l = \boldsymbol{U}_t^l \boldsymbol{\Sigma}_t^l (\boldsymbol{V}_t^l)^\intercal, \boldsymbol{R}_t^l \in \mathbb{R}^{m\times n}, l \in [1,L] \tag{17}$$

where $\boldsymbol{U}_t^l$ and $\boldsymbol{V}_t^l$ are left and right singular value matrices which are orthogonal to each other, and Σ_t^l contains the singular values along its main diagonal. If the rank of matrix $\boldsymbol{R}_t^l$ is r ($r \leq min(m,n)$), $\boldsymbol{R}_t^l$ can be expressed as $\boldsymbol{R}_t^l = \sum_{i=1}^r \boldsymbol{\sigma}_i \boldsymbol{u}_i \boldsymbol{v}_i^\intercal$, where $\boldsymbol{u}_i \in \boldsymbol{U}_t^l$ and $\boldsymbol{v}_i \in \boldsymbol{V}_t^l$ are left and right singular vectors and $\boldsymbol{\sigma}_i \in diag(\boldsymbol{\Sigma}_t^l)$ are singular values.

Definition 4 (The k-rank approximation of $\boldsymbol{R}_t^l$ for task t.) With the matrix approximation (Hadsell et al., 2006), the k-rank approximation of $\boldsymbol{R}_t^l$ is defined as:

$$||(\boldsymbol{R}_t^l)_k||_F^2 \geq \epsilon_t^l ||\boldsymbol{R}_t^l||_F^2, l \in [1,L] \tag{18}$$

where $||\cdot||_F$ is the Frobenius norm of the matrix and $\epsilon_t^l \in (0,1]$ is a feasible empirical threshold.

The calculations of $\boldsymbol{B}_t^l$ ($t \in [2,T]$) 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 as follows:

$$\hat{\boldsymbol{R}}_t^l = \boldsymbol{R}_t^l - \boldsymbol{R}_t^l \boldsymbol{B}_{t-1}^l (\boldsymbol{B}_{t-1}^l)^\intercal = \boldsymbol{R}_t^l - \boldsymbol{R}_{t,Proj}^l \tag{19}$$

Third, the SVD on $\hat{\boldsymbol{R}}_t^l$ is calculated by Eq. (17) and the k new orthogonal bases for the minimum value of k are selected for satisfying the following inequality:

$$||\boldsymbol{R}_{t,Proj}^l||_F^2 + ||(\hat{\boldsymbol{R}}_t^l)_k||_F^2 \geq \epsilon_t^l ||\boldsymbol{R}_t^l||_F^2 \tag{20}$$

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. (20), 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 of STIL. 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:

$$\boldsymbol{B}_t^l = \{\boldsymbol{B}_{t-1}^l, \boldsymbol{b}_t^l\}, l \in [1,L] \tag{21}$$

C.3 Algorithm 3 STIL. See Figure 1(a) for the calculation procedure of Algorithm 3 STIL, and the STIL source code is available at https://github.com/STILalg/STIL.

Algorithm 3 STIL

Input: The training dataset $\mathbb{D} = \{\mathbb{D}_t\}_{t=1}^{T}$; the original model $model_{ori}$ and learning model $model_{CL}$ of STIL; **Output:** The learned model $model_{CL}$ after learning all tasks;

for each task $t \in [1, T]$ **do**
  Generating the validation dataset $\mathbb{D}'_t$ of $\mathbb{D}_t$ online by **Online Validation Dataset Generator** of STIL;
  if $t == 1$ **then**
    Training $model_{CL}$ on $\mathbb{D}_t$ by using the cross-entropy loss function without imposing any constraint on weight updates of the model;
  else
    Calculating the layer-wise OIWS: $\boldsymbol{W}_t^l$ 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 $model_{CL}$
    $\{\mathbb{T}_{sim}^l\}_{l=1}^{L-1} \leftarrow$ 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 STIL;
    if $\{\mathbb{T}_{sim}^l\}_{l=1}^{L-1} == \emptyset$ **then**
      Setting cross entropy $\mathcal{L}_{dis}$ calculated as the loss function of $model_{CL}$;
    else
      Setting $\mathcal{L}_{sim}$ calculated by Eq. (9) as the loss function of $model_{CL}$ for preparing knowledge transfer (KT);
    end if
    Training $model_{CL}$ on $\mathbb{D}_t$ with **Similar/Dissimilar Tasks Processor** of STIL, where the learning takes the gradients of task t in the directions orthogonal to S_{KG} of the previous t-1 tasks by Eq. (6) to avoid CF, and performs weight updates of the model by Eq. (7);
  end if
  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;
end for
return $model_{CL}$;

Table 5: The efficiency and memory comparisons of STIL and state-of-the-art KT baselines.

Datasets	OWM		GPM		CAT		TRGP		CUBER		STIL(**Ours**)	
	T(S)	M(G)	T(S)	M(G)	T(S)	M(G)	T(S)	M(G)	T(S)	M(G)	T(S)	M(G)
PMNIST	18.15	5.12	**10.17**	1.29	15.32	3.92	13.22	1.36	16.68	1.52	10.67	**0.80**
CIFAR100	24.85	15.63	2.51	1.59	3.28	4.22	2.82	1.91	2.91	2.11	**2.02**	**1.12**
CIFAR100 Sup	–	–	1.59	1.53	2.41	4.22	1.62	1.61	1.87	1.92	**1.31**	**1.24**
MiniImageNet	–	–	3.52	1.91	3.54	4.41	4.72	2.97	5.53	3.53	**2.43**	**1.86**
5-Datasets	–	–	14.74	8.49	12.54	4.22	15.73	11.58	26.64	13.11	**7.10**	**6.89**
F-EMNIST-1	–	–	**1.08**	**1.06**	2.16	3.62	1.37	1.93	1.56	2.14	1.16	1.17
F-EMNIST-2	–	–	3.06	**1.25**	19.46	3.11	3.76	2.25	4.09	2.56	**1.38**	1.39
F-CelebA-1	–	–	**0.14**	**0.82**	0.18	3.25	0.20	1.40	0.37	1.63	0.17	0.87
F-CelebA-2	–	–	**0.17**	**0.87**	0.74	3.78	0.48	1.53	0.67	1.71	0.19	0.95
(EMNIST, F-EMNIST-1)	–	–	**1.26**	1.73	8.68	4.24	2.16	2.71	3.35	3.26	1.64	**1.67**
(CIFAR 100, CelebA-1)	–	–	**1.30**	**0.82**	9.26	3.17	2.35	2.03	3.69	2.71	1.78	0.89
Average	21.56	10.38	3.59	1.94	7.05	3.83	4.40	2.84	6.12	3.29	**2.70**	**1.71**

T(S)–Time (Second); M(G)–Memory (GB);
"–" indicates that the source codes are not provided by the baselines leading to no experimental results.

D Additional Experimental Results

D.1 Efficiency and Memory Performance Comparisons

To verify the efficiency of the proposed STIL in time and memory required for the model training, we conducted the efficiency comparison experiments in terms of the time spent per epoch, and the amount of memory used by the baselines. Note that the few methods can tackle both CF and KT, e.g.,

CAT, TRGP and CUBER, and all the OG-based methods, OWM, GPM, TRGP and CUBER, follow the same basic processing framework. The experimental results are shown in Table 5.

It is worth noting that although the proposed STIL method added the OIWS and KT processing mechanisms for its end compared with GPM, its time and space performances are better than GPM on some datasets, which is due to the optimizations by STIL made in its implementation. GPM loads in all its stored information during its computation, while our STIL loads only the information needed for computation on demand. In addition, during model training, STIL added the processing of early stop of model convergence, that is, once the convergence of model training/learning is determined, the training is immediately ended, unlike GPM, which always needs to be iteratively completed.

D.2 ACC and BWT Performances of STIL of Task-wise vs. Layer-wise on Five Datasets

As the tasks in different datasets may have similarities, we should carry out forward KT to assist task t learning if a new task t is similar to some of the previously learned t-1 tasks. Moreover, we know that 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 task similarity comparison, we conducted the experiments for ACC and BWT performances with the standard deviation values over 5 different runs of the proposed STIL of Task-wise vs. Layer-wise on five datasets. As shown in Table 6, the Layer-wise STIL (i.e., STIL) can achieve an average ACC/BWT gain of 1.23%/0.01 over the Task-wise STIL 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.

Table 6: ACC and BWT performances of STIL 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 STIL	95.98 $\pm$ 0.04	-0.03 $\pm$ 0.01	73.98 $\pm$ 0.14	-0.01 $\pm$ 0.01	59.36 $\pm$ 0.02	-0.01 $\pm$ 0.01	63.99 $\pm$ 0.36	-0.02 $\pm$ 0.01	92.33 $\pm$ 0.07	-0.02 $\pm$ 0.01
Layer-wise STIL	**97.15** $\pm$ 0.03	**-0.02** $\pm$ **0.01**	**75.28** $\pm$ 0.13	**0.21** $\pm$ 0.00	**60.78** $\pm$ 0.01	**-0.01** $\pm$ 0.00	**65.11** $\pm$ 0.31	**-0.01** $\pm$ 0.01	**93.46** $\pm$ 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

D.3 The Accuracy Evolution of Task t of STIL and Baselines on a Variety of Datasets

To show the BWT and/or knowledge transfer (KT) performance of the proposed STIL and 5 strong TIL baselines of various categories, 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 are shown in Tables 7-10 and Figures 2-5 demonstrate the following:

(1) Compared with the baselines, the proposed STIL has stronger resistance to forgetting than that of the baselines while maintaining high accuracy, which is due to the proposed OIWS-based TCL strategy of STIL (see Tables 7-10 and Figures 2-5).

(2) More interestingly, benefited from the KT with its $\mathcal{L}_{sim}$ shown in Eq. (8), the proposed STIL 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 7-10 or Figures 4 - 5). It is worth noting that this phenomenon also appears in baselines CAT and TRGP with KT ability (see Table 10/Figure 5), 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 STIL in resisting forgetting and performing KT, but also verify the superiority of STIL over the baselines in ACC, BWT, and KT performances.

Table 7: ACC 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**
STIL(Ours)	96.8	96.7	96.5	96.3	96.1	96.0

Table 8: ACC 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
STIL(Ours)	**82.2**	**81.5**	**79.8**	**79.7**	**79.5**

Table 9: ACC 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
STIL(Ours)	**78.2**	**76.7**	**77.6**	**78.4**	**78.2**	**80.1**	**77.4**	**78.1**	**79.3**	**79.2**

Table 10: ACC 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
STIL(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**

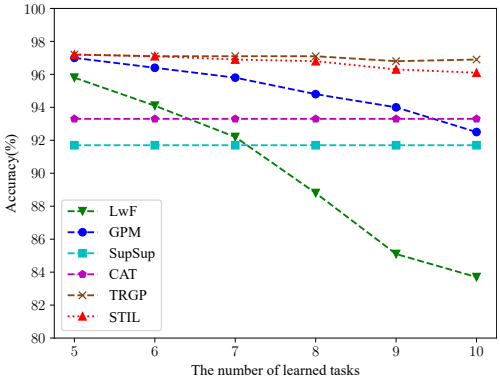

Figure 2: The diagram of the data in Table 7.

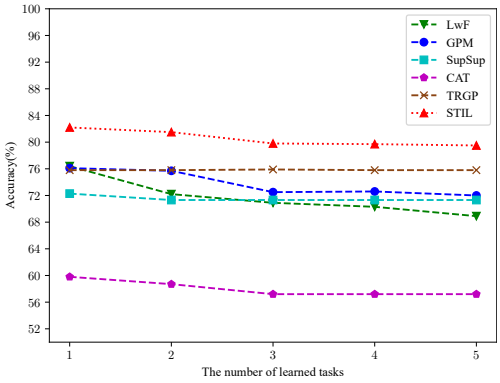

Figure 3: The diagram of the data in Table 8.

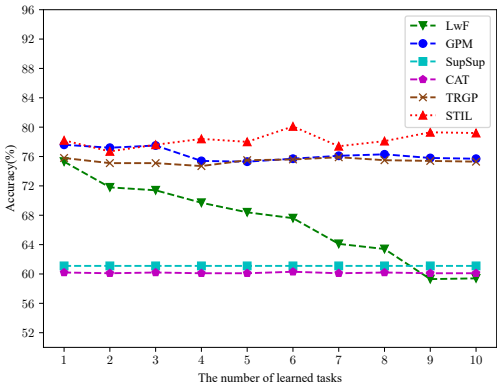

Figure 4: The diagram of the data in Table 9.

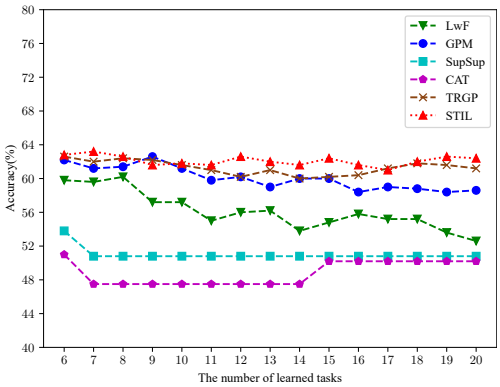

Figure 5: The diagram of the data in Table 10.

D.4 The KT Performance of STIL and Baselines on Mixed Tasks Datasets

Table 11: The KT performances of our STIL and 5 strong baselines with/without KT mechanisms over 5 different runs on two mixed tasks datasets.

Dataset	(EMNIST, F-EMNIST-1) (20 Tasks)			(CIFAR 100, F-CelebA-1) (20 Tasks)			Average		
Methods	ACC(%)	FWT(%)	BWT(%)	ACC(%)	FWT(%)	BWT(%)	ACC(%)	FWT(%)	BWT(%)
ONE	**77.44**	None	None	**64.50**	None	None	**70.97**	None	None
GPM	73.69 $\pm$ 0.32	-3.65	**0.38**	64.28 $\pm$ 0.28	0.13	-0.37	68.99	-1.89	**0.01**
HAT	70.70 $\pm$ 0.18	-6.26	0.00	56.82 $\pm$ 0.13	-7.68	**0.00**	63.76	-6.77	0.00
CAT	74.61 $\pm$ 0.19	-0.45	-2.19	61.94 $\pm$ 0.16	-2.56	**0.00**	68.28	-1.51	-1.10
TRGP	75.53 $\pm$ 0.28	0.12	-1.36	61.92 $\pm$ 0.21	-1.17	-1.55	68.73	-0.48	-1.46
CUBER	**77.23** $\pm$ 0.28	**0.53**	-0.73	**64.85** $\pm$ 0.31	**1.09**	-0.73	**71.04**	**0.81**	-0.73
STIL(Ours)	**78.01** $\pm$ 0.23	0.16	**0.43**	**65.46** $\pm$ 0.14	**1.23**	**0.33**	**71.74**	0.70	**0.38**

[1] ONE – building a model for each task independently using a separate neural network, which has no knowledge transfer and no forgetting involved (denoted as None). The backbone 3-Layer FCN is used in all baselines on the two mixed tasks datasets. The blue results mean the best prior results.

The experimental results shown in Table 11 demonstrates that the performance of STIL's ACC, FWT and BWT are superior to those of five strong baselines on two mixed data sets, which fully verifies the effectiveness of STIL's forgetting elimination and KT mechanism.

D.5 The Number of Learnable Tasks of STIL and OG-based OWM

Figure 6 shows the number of learnable tasks of STIL and OG-based OWM, respectively.

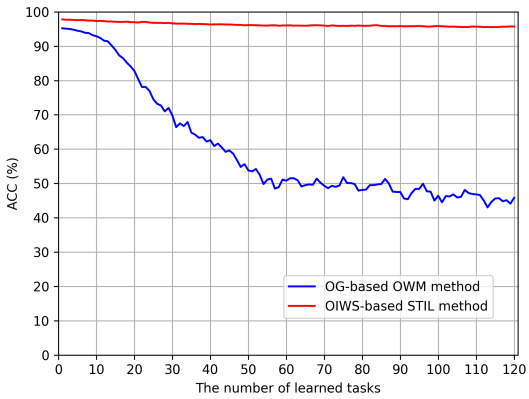

Figure 6: Accuracy changes as the number of learned tasks increasing for OWM and STIL.

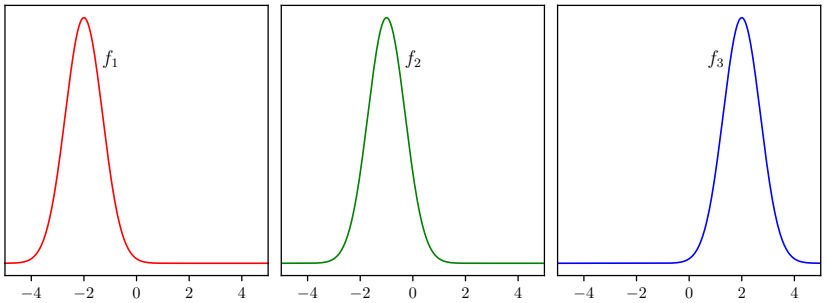

Figure 7: The schematic diagram of the difference between Euclidean distance and Wasserstein distance. The figure shows three distributions f_1(red), f_2(green) and f_3(blue). Each pair has the same distance in the Euclidean space. But in the Wasserstein space, f_1 and f_2 are closer as the shapes/geometries of f_1 and f_2 are more similar overall.

ONE ACC Performance on 5-Datasets						
T1	**T2**	**T3**	**T4**	**T5**	**ACC (%)**	
79.86	99.43	99.41	94.83	94.37	**93.58**	
STIL ACC Performance on 5-Datasets						
79.10						
79.10	99.30					
79.10	99.30	99.30				
79.10	99.30	99.30	94.57		**ACC (%)**	**BWT (%)**
79.10	99.30	99.30	94.52	95.10	93.46	-0.01

ONE ACC Performance on 5-Datasets						
T1	**T2**	**T3**	**T4**	**T5**	**ACC (%)**	
79.86	99.43	99.41	94.83	94.37	**93.58**	
CUBER ACC Performance on 5-Datasets						
79.10						
79.10	99.30					
79.10	99.30	98.12				
79.10	99.30	98.12	92.15		**ACC (%)**	**BWT (%)**
79.10	99.30	98.12	92.60	90.64	91.95	0.03

Figure 8: The ACC performance of the proposed STIL and CUBER from Table 1 on 5-Datasets.