Task-aware Orthogonal Sparse Network for Exploring Shared Knowledge in Continual Learning

Yusong Hu [*1] **De Cheng†** [*1] **Dingwen Zhang** [2] **Nannan Wang†** [1] **Tongliang Liu** [3] **Xinbo Gao** [4]

Abstract

Continual learning (CL) aims to learn from sequentially arriving tasks without catastrophic forgetting (CF). By partitioning the network into two parts based on the Lottery Ticket Hypothesis—one for holding the knowledge of the old tasks while the other for learning the knowledge of the new task—the recent progress has achieved forget-free CL. Although addressing the CF issue well, such methods would encounter serious under-fitting in long-term CL, in which the learning process will continue for a long time and the number of new tasks involved will be much higher. To solve this problem, this paper partitions the network into three parts—with a new part for exploring the knowledge sharing between the old and new tasks. With the shared knowledge, this part of network can be learnt to simultaneously consolidate the old tasks and fit to the new task. To achieve this goal, we propose a task-aware **Orthogonal Sparse Network** (OSN), which contains shared knowledge induced network partition and sharpness-aware orthogonal sparse network learning. The former partitions the network to select shared parameters, while the latter guides the exploration of shared knowledge through shared parameters. Qualitative and quantitative analyses, show that the proposed OSN induces minimum to no interference with past tasks, $i.e.$, approximately no forgetting, while greatly improves the model plasticity and capacity, and finally achieves the state-of-the-art performances.

[*]Equal contribution [1]Xidian University, Xi'an, Shaanxi Province, China. [2]Northwestern Polytechnical University, Xi'an, Shaanxi Province, China. [3]University of Sydney. [4]Chongqing University of Post and Telecommunications, Chongqing, China.. Correspondence to: De Cheng <dcheng@xidian.edu.cn>, Nannan Wang <nnwang@xidian.edu.cn>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Continual learning (CL) aims to learn knowledge from sequential input data/tasks, to mimic human cognition to incrementally learn new concepts over his/her lifespan. Therefore, CL is also known as lifelong learning or incremental learning. However, this is a very challenging problem due to the *catastrophic forgetting* (CF) phenomenon, where the performance of the network on the old tasks will substantially decrease after training on the new task when facing a series of continuous data stream. To achieve long-term CL, the network should not only preserve old knowledge learned from old tasks to deal with CF, but also needs to obtain knowledge transfer from the old tasks to the new one, which is known as the stability-plasticity trade-off dilemma.

Recently, various CL methods have been proposed to deal with CF, which can be roughly divided into the following categories: *Regularization-based* methods (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018), *Rehearsal-based* methods (Aljundi et al., 2018; Ostapenko et al., 2019; Shin et al., 2017; Cheng et al., 2024a) and *Architecture-based* methods (Hung et al., 2019; Li et al., 2019; Yoon et al., 2018; Mallya & Lazebnik, 2018; Wang et al., 2022; Kang et al., 2022). For these *Architecture-based* methods, they are mainly based on parameter expansion or pruning. *Parameter expansion* methods (Hung et al., 2019; Li et al., 2019; Yoon et al., 2018) usually selectively reuse the parameters in the dense network to deal with forgetting and increase the size of the network to learn on new tasks, as shown in Fig. 1(b). *Parameter pruning* methods (Mallya & Lazebnik, 2018; Wang et al., 2022; Kang et al., 2022) assign a sub-network, which is pruned from the pre-allocated dense network, to each specific old task and only the unpruned parameters can be used for further training, as shown in Fig. 1(c). Although these methods have many merits, which are memory efficient without requiring to store extra old-task data and could achieve forget free in some situations, they still have the following limitations: 1) The *parameter expansion* methods have to keep increasing the network size to learn on new task, which may cause serious burden on training efficiency; 2) The pruned sub-network in the *parameter pruning* methods is monopolized by the old tasks leading to low model capacity for new tasks.

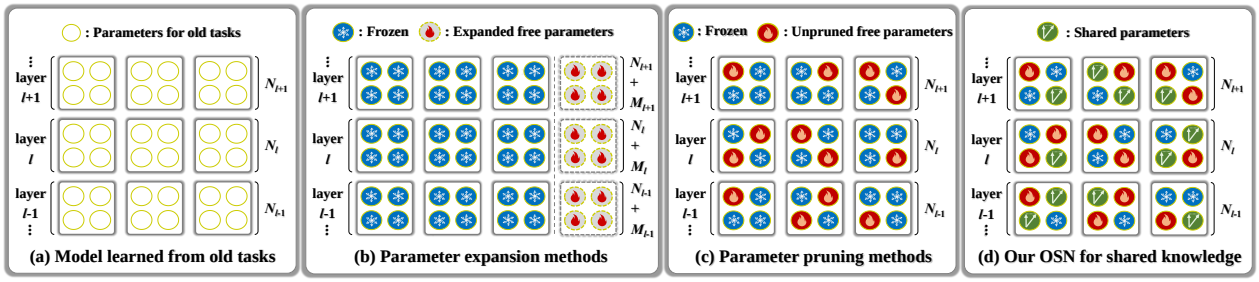

Figure 1. Comparison of our proposed OSN with other architecture-based methods. After learning from old tasks, a model is obtained as depicted in (a). In (b), parameter expansion methods increase the size of network ($i.e.$, the number of parameters increases from N_l to $N_l + M_l$ at l-th layer) to learn on the new tasks and frozen the original old parameters to deal with forgetting. As shown in (c), parameter pruning methods prune some of parameters to preserve old knowledge and only the unpruned free parameters can be used to learn from new tasks. In (d), our method OSN partitions the network into three parts, with a new part consisting shared parameters to explore shared knowledge.

Existing parameter pruning methods, such as (Kang et al., 2022), divide the network into two parts: the pruned sub-network and the remaining part. The former is frozen and can only be used to preserve old knowledge, while the latter is used solely for learning new knowledge. These limitations severely hinder the practical application for CL, especially for the long-term CL with more and more arriving tasks. Comprehensive analysis demonstrate the common reason behind these limitations is that, these methods usually neglect the shared knowledge between old and new tasks. As a result, it will greatly sacrifice the model plasticity and capacity, leading to negative effects in long-term CL. Therefore, to address this issue, we make an early effort to introduce the shared knowledge between the old and new tasks during model parameter pruning and updating in our method, which provide a new way to address the well-known stability-plasticity dilemma in CL.

Specifically, we propose a task-aware ***O**rthogonal **S**parse **N**etwork* (OSN) for exploring the shared knowledge in CL, through the shared model parameters in the previously pruned sub-network. The proposed OSN not only prunes a sparse sub-network for each task to deal with CF, but also searches for a set of shared parameters that are potentially important for knowledge transfer between old and new tasks in the sub-network, as shown in Fig. 1(d). After obtaining the shared parameters from the sub-network for the old tasks, the trainable model parameters for new task will be expanded including the task-shared and the unpruned model parameters. During training on the new task, the task-shared parameters are updated though an orthogonal projection to achieve no interference with the past tasks, while the traditional unconstrained gradient descent optimization are adopted on the unpruned model parameters. By this way, the shared knowledge can be learned through these task-shared parameters in the sub-network, thus both of the model stability and plasticity can be better achieved. Meanwhile, the model capacity can be greatly improved to benefit the practical application of CL, especially for the long-term scenario. Besides, we introduce a sharpness-aware orthogonal projection to replace existing approximate orthogonal projection, aiming to enhance knowledge sharing and improve the overall performance of the proposed OSN method.

To summarize, our contributions in OSN are three-fold:

- By revisiting the relationship between the old and new tasks, we make an early effort to introduce the concept of shared knowledge to the CL task, providing a new way to address the well-known stability-plasticity dilemma.
- For exploring the shared knowledge in CL, we proposes two techniques: 1) the shared knowledge induced network partition to discover the network parameters that are suitable for capturing the shared knowledge for different tasks; 2) the sharpness-aware orthogonal sparse network learning strategy that learns the shared knowledge upon the discovered network parameters.
- Qualitative and quantitative analysis on five commonly used datasets show that our method can induce minimum to no interference with past tasks while greatly improves the model plasticity capacity to refresh the state-of-the-art performance in CL. Notable, when dealing with long-term CL case, our method can obtain more than 3.46%ACC performance improvement over the current SOTA methods.

2. Related Work

A series of continual learning methods have been proposed to deal with CF. *Regularization-based* methods (Kirkpatrick et al., 2017; Zenke et al., 2017; Aljundi et al., 2018) introduce a penalty term to regularize parameter updating on new

tasks. *Knowledge distillation based* methods (Li & Hoiem, 2016; Dhar et al., 2019; Lee et al., 2019) adopt a distillation loss to constrain the training on new tasks to prevent a significant distribution difference between the new and old models. *Rehearsal-based* methods usually select (Aljundi et al., 2019; Prabhu et al., 2020) or generate (Hu et al., 2018; Ostapenko et al., 2019; Shin et al., 2017) some representative data from old tasks to store them in a memory buffer, and both the data from the new task and the memory buffer are fed to the network during training. Some other methods integrate continual learning with other learning paradigms (Cheng et al., 2024b) or leverage specific model architectures (Cheng et al., 2023). *Orthogonal projection based* methods (Chaudhry et al., 2018b; Zeng et al., 2019; Wang et al., 2021; Kong et al., 2022; Lin et al., 2022a) adjust the parameter updating direction onto the orthogonal space of the old tasks to achieve better stability-plasticity trade-off. However, these methods are not very effective for dealing with CF due to the technical limitations, and the forgetting may be much severer in long-term CL.

Parameters allocation methods. *Architecture-based* methods could achieve lower CF and can be divided into two categories: 1) *parameter expansion*; 2) *parameter pruning*. Parameter expansion methods reuse the old model parameters to deal with CF and increase the size of the network to learn on new task (Hung et al., 2019; Li et al., 2019; Yoon et al., 2018). Parameter expansion methods suffer from low training efficiency and huge memory consumption, which makes it unable to perform well in long-term CL scenarios. Parameter pruning methods allocate a sparse sub-network pruned from the dense network for a specific old task, while only the unpruned parameters are used to learn on the new task(Serra et al., 2018; Mallya & Lazebnik, 2018; Wortsman et al., 2020; Kang et al., 2022; Wang et al., 2022). However, as the pruned sub-network is usually monopolized by the old task and solely used for preserving old knowledge, its potential contribution to the knowledge transfer has been ignored. Intuitively, the parameters which are important for the new task in the sub-network, should be applied to learn the shared knowledge between the old and new tasks. As the new tasks arrive continually, obtaining the shared knowledge become more crucial for achieving better knowledge transfer and network plasticity. Hence, we aim to select and apply those important shared parameters in the sub-network to learn on new task.

To solve the above challenges, we propose a novel parameter pruning method in CL, namely *Orthogonal Sparse Network* (OSN). In OSN, we not only allocate a pruned sparse sub-network to each specific old task, but also update the important shared parameters in the sub-network to obtain the shared knowledge between old and new tasks, to provide a new way to address the well-known plasticity-stability dilemma in CL.

3. Orthogonal Sparse Network in CL

In this section, we propose a novel parameter pruning methods in CL, namely ***O**rthogonal **S**parse **N**etwork* (OSN), which contains network partition and sharpness-aware orthogonal projection. In OSN, different from other parameter pruning methods, we partition the network into three parts with an extra part for exploring the shared knowledge between the old and new tasks. Then a sharpness-aware orthogonal projection is applied for learning the shared knowledge. Our proposed OSN is shown in Figure 1 and the detailed algorithm is in Algorithm 1.

3.1. Problem Definition

Given a continuous task stream $\{T_1, T_2, \cdots, T_N\}$ and all N tasks are sequentially input to an L-layer neural network $\mathcal{F}(\cdot, \theta^l)$, where $l \in \{0, 1, \cdots, L\}$ and θ^l are the model parameters. We denote the dataset of the t-th task as $\mathcal{D}_t = \{X_t, Y_t\}$, where X_t and Y_t are the input feature set and its corresponding label set, respectively. In continual learning setting, when training on the t-th task, only the dataset $\mathcal{D}_t$ is available and the network is parameterized as $\mathcal{F}(\mathcal{D}_t, \theta_t^l)$. During the inference phase, if the task identity t of each test sample is given, such a continual learning setting is called task-incremental learning (TIL) (Ven & Tolias, 2019), otherwise it becomes class-incremental learning (CIL) (Ven & Tolias, 2019). In this paper, we mainly focus on the TIL.

After training on the t-th task, the model should not only perform well on all old tasks $\{T_1, T_2, \cdots, T_{t-1}\}$, but also perform well on the new task $\{T_t\}$ during the model inference. This requires the continual learning model to have two important characteristics, namely *stability* and *plasticity*. The stability indicates the ability of the model to maintain old knowledge and the plasticity indicates the capacity of the model to learn from the new task.

3.2. Shared Knowledge Induced Network Partition

To deal with catastrophic forgetting, a series of parameter pruning methods (Mallya & Lazebnik, 2018; Wang et al., 2022) have been proposed for CL. One of the most representative methods is Winning Subnetworks (WSN) (Kang et al., 2022), which is inspired by the Lottery Ticket Hypothesis (Frankle & Carbin, 2019). WSN prunes the network into two parts: the pruned sub-network and the unconstrained sub-network, which are used to preserve old knowledge and learn new knowledge, respectively. Notably, the pruning of the sub-network is represented by a binary mask $\mathcal{M}$, which can be updated task-by-task through Huffman encoding (Huffman, 1952). Formally, after training on the task T_{t-1}, the pruning of the sub-network can be written as

follows,

$$\min_{\mathcal{M}^l_{t-1}\in\{0,1\}} \mathcal{L}(\theta^l_{t-1}\odot\mathcal{M}^l_{t-1},\mathcal{D}_{t-1})-\mathcal{L}(\theta^l_{t-1},\mathcal{D}_{t-1}), \tag{1}$$

where $\mathcal{L}$ is the cross entropy loss function and the symbol $\odot$ represents the multiplication operation at the corresponding position. Through Eq. 1, we can obtain the most important top-$c(\%)$ weights from θ^l_{t-1} with a given sparsity ratio c. A value of 1 in $\mathcal{M}^l_{t-1}$ indicates the corresponding θ^l_{t-1} is pruned ($i.e.$, remained parameters), while a value of 0 indicates the opposite. In WSN, to preserve the old knowledge, the pruned sub-network is monopolized by the corresponding old task (Kang et al., 2022), which means the sub-network is frozen when training on the new task. Formally, when training on task T_t through the mask $\mathcal{M}^l_{t-1}$, the updating of θ^l_t can be written as follows (Kang et al., 2022),

$$\theta^l_t \leftarrow \theta^l_{t-1}-\eta\cdot\left(\frac{\partial\mathcal{L}}{\partial\theta}\odot(1-\mathcal{M}^l_{t-1})\right), \tag{2}$$

where η is the learning rate. Eq. 2 indicates that, when training on the task T_t, the gradient of the pruned sub-network parameter is set as zero while other unpruned parameters are updated unrestrictedly to learn on task T_t.

Through Eq. 1 and Eq. 2, WSN has achieved forget-free CL. However, when training on the new task, the pruned sub-network is only used for preserving old knowledge while its potential contribution to the new knowledge transfer is ignored. For those parameters in the sub-network, which are also important for learning knowledge on new task, monopolized only by the old tasks will lead to a poor network plasticity. We refer those parameters as *shared parameters* and aim to apply them to learn the shared knowledge between old and new tasks. Especially in long-term CL, since a large number of tasks continually arrive, such shared knowledge is not only effective for deal with forgetting but also more crucial for the new knowledge transfer. Hence, the key issue becomes how to obtain the important shared parameters and how to apply them to learn the shared knowledge.

Among existing parameter pruning based methods, the network parameters are pruned only based on their contribution in preserving old knowledge, as described in Eq. 1. The pruned sub-network can effectively retain old knowledge, but the importance of its parameters in acquiring new knowledge is neglected. The shared parameters should not only be effective for preserving old knowledge but also be important for the new knowledge transfer. Hence, we introduce a task-aware network partition strategy to obtain the shared parameters for exploring the shared knowledge. Different from existing parameter pruning methods, our proposed network partition strategy divides the network into three parts, which contains an extra new part for shared parameters. Specially, we compute the importance of the pruned sub-network parameters obtained from Eq. 1 for learning on new task. Based on the importance, we partition the sub-network into shared network parameters and the remaining pruned sub-network parameters. We apply a weight importance function CWI$(\cdot)$ (Wang et al., 2022) to quantitatively calculate the importance of each parameter in the pruned sub-network when training on the new task, $i.e.$,

$$CWI(\cdot)=\|\cdot\|_1+\|\frac{\partial\mathcal{L}(\theta^l_t,\mathcal{D}_t)}{\partial(\cdot)}\|_1, \tag{3}$$

where $\|\cdot\|_1$ is the $L1$ norm. When training on the new task T_t, we compute the importance of sub-network parameters, $i.e.$, CWI$(\theta^l_t\odot\mathcal{M}^l_{t-1})$, and select the most important top-$k(\%)$ ones. In Eq. 3, CWI$(\cdot)$ mainly relies on the training gradient to assess the importance of sub-network parameters for learning on new task. It is reliable because the gradient can directly and obviously demonstrate the impact of parameters' changes on the final training loss. In practice, the network partition strategy serves as a warm-up training before the main OSN algorithm. Similarly, we apply an extra binary mask to represent the network partition, denoted as m^l_t. Thus, the parameters which are selected by the network re-partition in OSN can be represented by the intersection of $\mathcal{M}^l_{t-1}$ and m^l_t, $i.e.$, $(\mathcal{M}^l_{t-1}\cap m^l_t)\odot\theta^l_t$.

3.3. Orthogonal Sparse Network learning

Next, we utilize these shared parameters $(\mathcal{M}^l_{t-1}\cap m^l_t)\odot\theta^l_t$ to acquire shared knowledge between the old and new tasks. Since the learnt knowledge is shared, we must ensure that the shared parameters can be not only learnt to simultaneously consolidate the old tasks but also fit to the new one. Therefore, we specifically project the updating gradients of the shared parameters on the new task, onto the orthogonal space of old tasks' feature space. This updating strategy can achieve better balance between model stability and plasticity, which can be derived from the following lemma.

Lemma 3.1. *Given a L-layer neural network $\mathcal{F}$, it has been trained on a continuous task stream $\{T_1,T_2,\cdots,T_{t-1}\}$. The features and weights of each layer for each task in $\mathcal{F}$ are X^l_{t-1} and θ^l_{t-1}, respectively. When a new task T_t is feed to $\mathcal{F}$, if the parameter updating $\Delta\theta^l_t$ is orthogonal to the old tasks feature construction $\bar{X}^l_{t-1}=[X^l_1,X^l_2,\cdots,X^l_{t-1}]$, $i.e.$,*

$$\theta^l_t=\theta^l_{t-1}+\Delta\theta^l_t,\quad \bar{X}^l_{t-1}\cdot\Delta\theta^l_t=0, \tag{4}$$

we have

$$\mathcal{F}(\mathcal{D}_p,\theta^l_p)=\mathcal{F}(\mathcal{D}_p,\theta^l_t),\quad p=0,1,\cdots,t-1. \tag{5}$$

The detailed proof can refer to (Wang et al., 2021). Lemma 3.1 suggests that, updating the network parameters in the orthogonal space of the input features from old tasks at each training step when training on a new task, allows the

network to preserve its training loss on all previous tasks and prevent forgetting. Such an orthogonal projection based training strategy allows the network parameters to fit the new task while simultaneously maintaining their performances on the old tasks, which means that it has the potential to explore shared knowledge. Therefore, we project the training gradients of the shared parameters $(\mathcal{M}_{t-1}^l \cap m_t^l) \odot \theta_t^l$ on the new task onto the orthogonal feature space of the old tasks to obtain shared knowledge between the new and old tasks. That is,

$$\theta_t^l \leftarrow \theta_{t-1}^l - \eta \cdot P_{t-1}^l \cdot \left(\frac{\partial \mathcal{L}}{\partial \theta} \odot (\mathcal{M}_{t-1}^l \cap m_t^l) \right), \quad (6)$$

where P_{t-1}^l is the orthogonal projection constructed by $\bar{X}_{t-1}^l$. Eq. 6 indicates that, the parameter updating of the pruned shared parameters, which is represented by $\mathcal{M}_{t-1}^l \cap m_t^l$, is confined in the orthogonal space spanned by P_{t-1}^l to obtain the shared knowledge. Moreover, we also assign different learning strategies to the other parts of the network parameters, $i.e.$, parameters in the pruned sub-network that are not selected by the network partition strategy, and parameters in the dense network that are are not pruned. The former, which can be written as $(\mathcal{M}_{t-1}^l - \mathcal{M}_{t-1}^l \cap m_t^l) \odot \theta_t^l$, is used to preserve old knowledge and it remains frozen when training on new task. The latter, which can be written as $(1 - \mathcal{M}_{t-1}^l) \odot \theta_t^l$, is used for promoting new knowledge and it serves as the free parameters for training on the new task. Formally, they can be written as follows,

$$\theta_t^l \leftarrow \theta_{t-1}^l - \eta \cdot \left(\frac{\partial \mathcal{L}}{\partial \theta} \odot (1 - \mathcal{M}_{t-1}^l) \right). \quad (7)$$

Eq. 6 and Eq. 7 indicate that our OSN divides the network parameters into three different parts and assigns different learning strategies to each part.

To obtain P_{t-1}^l, existing work (Wang et al., 2021; Saha & Roy, 2021) usually apply *Singular Value Decomposition* (SVD) technique on the feature space of old tasks to get approximate null space or orthogonal basis. These two methods are equivalent as they utilize the singular vectors corresponding to the singular values of the feature construction matrix $\bar{X}_{t-1}^l$. However, the orthogonal projection obtained by such matrix iterations or SVD technique is typically an approximate result, which can not strictly satisfy Lemma 3.1. It may result in the shared parameters in Eq. 6 being unable to effectively consolidate old knowledge, leading to negative effects on knowledge sharing.

Therefore, we propose a sharpness-aware orthogonal projection instead for better knowledge sharing. The sharpness-aware orthogonal projection is designed to seek for the flat minima in the loss function rather than the sharp one. Arriving at a minima from a flat loss surface is helpful for improving the model stability in CL (Yang et al., 2023).

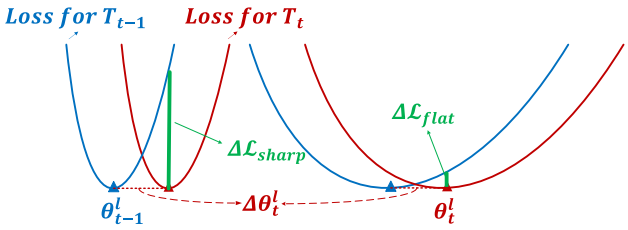

Figure 2. Illustration of sharpness-aware orthogonal projection. With the same $\Delta\theta_t^l$, the loss change $\Delta\mathcal{L}_{flat}$ in the flat loss surface is much smaller than $\Delta\mathcal{L}_{sharp}$ in the sharp loss surface.

Assuming that $-\eta \cdot P_{t-1}^l \cdot \left(\frac{\partial \mathcal{L}}{\partial \theta} \odot (\mathcal{M}_{t-1}^l \cap m_t^l) \right)$ in Eq. 6 is $\Delta\theta_t^l$ and we have $\theta_t^l = \theta_{t-1}^l + \Delta\theta_t^l$, where $\Delta\theta_t^l$ represents the change from the old knowledge θ_{t-1}^l to the new knowledge θ_t^l. As shown in Figure 2, with the same $\Delta\theta_t^l$, the loss change $\Delta\mathcal{L}_{flat}$ from task T_{t-1} (depicted in the blue curve) to task T_t (depicted in the red curve) on a flat loss function surface is much smaller than $\Delta\mathcal{L}_{sharp}$ on a sharp one. This indicates that, when updating on the new task T_t on a flat loss function surface, the obtained θ_t^l have a minimal impact on the loss for the old task T_{t-1} and can maintain performance. Therefore, applying the sharpness-aware orthogonal projection can better preserve old knowledge while not impeding the new knowledge, thereby improving the knowledge sharing. Hence, we propose to improve the loss function of OSN by seeking flat minima as follows,

$$\mathcal{L} \leftarrow \mathcal{L} + \max_{\|\delta\|_2 \leq \rho} \left(\mathcal{L}(\theta + \delta, \cdot) - \mathcal{L}(\theta, \cdot) \right), \quad (8)$$

where ρ is a given threshold and δ is a small perturbation. Besides, to improve the overall performance of the sharpness-aware orthogonal projection, we introduce a data perturbation to extend the training data distribution. We select any two sample pairs $\{x_{i,t}, y_{i,t}\}$ and $\{x_{j,t}, y_{j,t}\}$ from a mini batch of $\mathcal{D}_t$ at each training step to generate a perturbed one, $i.e.$,

$$\tilde{x}_t = \gamma x_{i,t} + (1-\gamma) x_{j,t}, \tilde{y}_t = \gamma y_{i,t} + (1-\gamma) y_{j,t}, \quad (9)$$

where $\gamma \in [0, 1]$ is a given weight hyper-parameter. Both the original and perturbed data are used to minimize the loss function in Eq. 8.

After seeking for the flat loss surface, the sharpness-aware orthogonal projection can be written as follows,

$$P_{t-1}^l = U_{2,t-1}^l (U_{2,t-1}^l)^\top, \quad (10)$$

where $U_{2,t-1}^l$ is the singular vector to the minimum singular value that can be obtained through SVD technique as follows,

$$\bar{X}_{t-1}^l = U_{t-1}^l \Sigma_{t-1}^l (U_{t-1}^l)^\top, \quad (11)$$

where Σ_{t-1}^l is the singular value matrix and U_{t-1}^l is the corresponding singular vector. $U_{2,t-1}^l$ can be achieved ac-

cording to the following criteria with a given threshold ϵ_{th}^l,

$$\|U_{2,t-1}^l \Sigma_{2,t-1}^l (U_{2,t-1}^l)^\top\|_F \leq (1-\epsilon_{th}^l)\|\bar{X}_{t-1}^l\|_F, \quad (12)$$

where $\|\cdot\|_F$ is the Frobenius norm.

Algorithm 1 OSN in Continual Learning

Input: Tasks series $\{T_1, T_2, \cdots, T_N\}$, Dataset $\mathcal{D}_t$ for T_t.
Output: A neural network parameterized as $\mathcal{F}(\cdot, \theta_N^l)$, a binary mask $\mathcal{M}_N^l$.

Initializing network parameters θ_0^l, orthogonal projection P_0^l, cross entropy loss function $\mathcal{L}$, learning rate η, pruning sparsity c, network partition percentage k;
for $T_t \in \{T_1, T_2, \cdots, T_N\}$ **do**
  for $l \in \{1, 2, \cdots, L\}$ **do**
    if $t = 1$ **then**
      Train θ_1^l on task T_1;
      Obtain $\mathcal{M}_1^l$ to prune top-c(%) weights by Eq. 1;
      Construct $\bar{X}_t^l$ and obtain P_t^l by Eq. 10;
    else
      Obtain top-k(%) shared weights by Eq. 3, represented by m_t^l;
      Apply Eq. 9 and Eq. 8 to get new $\mathcal{L}$;
      Train θ_t^l on task T_t by Eq. 6 and Eq. 7;
      Update $\mathcal{M}_1^l$ to prune top-c(%) weights by Eq. 1;
      Construct $\bar{X}_t^l$ and obtain P_t^l by Eq. 10;
    end if
  end for
end for

4. Experiments

4.1. Datasets and Evaluation Metrics

We employ a series of experiments on five commonly used datesets, including PMNIST (Deng, 2012), Split CIFAR-100 (Krizhevsky, 2009), CIFAR-100 Superclass (Yoon et al., 2018), 5-Datasets (Saha & Roy, 2021) and Split TinyImageNet (Krizhevsky et al., 2017). To further test the perfermance of OSN in long-term continual learning, we introduce another two datasets with a larger number(≥ 40) of tasks, $i.e.$, Split TinyImageNet and Split CIFAR-100-50. The detailed information and setups of these datasets are introduced in Appendix B.1.

To compare with other CL methods, we use two common continual learning evaluation metrics, average accuracy (ACC) and backward transfer (BWT). ACC is the average test accuracy of all tasks after training, and BWT measures the forgetting degree on the old tasks after learning from the new one. Formally, ACC and BWT are defined as follows,

$$ACC = \frac{1}{N}\sum_{t=1}^{N} A_{N,t}, \quad BWT = \frac{1}{N-1}\sum_{t=1}^{N-1} A_{N,t} - A_{t,t}, \quad (13)$$

where $A_{N,t}$ is the test accuracy on the t-th task after training sequentially on all N tasks. Both ACC and BWT are the larger the better. A high ACC is more important than a high BWT under the same conditions, because ACC represents the overall performance of the network stability and plasticity, while BWT only reflects the forgetting degree, $i.e.$, the stability.

4.2. Implementation Details

For fair comparisons, we strictly follow the experimental settings in (Kang et al., 2022), including using the same backbone network on the corresponding dataset and so on. We conduct all experiments on PMNIST using a 2-layer MLP structure to extract features and an extra output unit as a classifier. We use a modified version of AlexNet followed by (Saha & Roy, 2021) for Split CIFAR-100 and a modified LeNet (Yoon et al., 2018) for CIFAR-100 Superclass. Similarly, a reduced ResNet-18 (Kang et al., 2022) is used on 5-Datasets. For Split TinyImageNet, we use the backbone followed by (Kang et al., 2022), which is composed by 4-layer convolution structure and 3 fully connected layers.

Followed by (Kang et al., 2022), we use Adam as an initial model optimizer with momentum 0.9. For PMNIST, the batch size, initial learning rate and epoch are 10, 1e-3 and 15, respectively. Each task is trained for 50 epochs on CIFAR-100, 80 epochs on 5-Datasets and 40 epochs on Split TinyImageNet. All the experiments are implemented on four NVIDIA 2080Ti GPUs with PyTorch.

4.3. Experimental Results

In Table 1, we compare our proposed OSN with other state-of-the-art CL methods, including EWC (Kirkpatrick et al., 2017), PackNet (Mallya & Lazebnik, 2018), SupSup (Wortsman et al., 2020), La-MaML (Joseph & Gu, 2021), FS-DGPM (Deng et al., 2021), GPM (Saha & Roy, 2021), TRGP (Lin et al., 2022b), WSN (Kang et al., 2022), Connector (Lin et al., 2022a), DualGPM (Liang & Li, 2023), API (Liang & Li, 2023) and DFGP (Yang et al., 2023).

For PMNIST, our OSN achieves 97.19% in ACC, which is 0.68% superior to the second best method WSN and 2.55% higher than DFGP. For Split CIFAR-100, the ACC of OSN is 7.78% higher than API and 1.68% higher than the second best method Connector. Our OSN achieves 63.81% in ACC on CIFAR-100 Superclass, which is 3.61% superior to API and 2.11% higher than the second best method SupSup in ACC. For 5-Datasets, OSN outperforms any other CL methods in Table 1. For Split TinyImageNet, OSN achieves 76.23% in ACC, which is 3.46% higher than the second best method WSN.

For those parameter pruning methods, such as PackNet, SupSup and WSN, they can achieve no forgetting, $i.e.$, the

Table 1. Comparisons of different continual learning methods on several real datasets. '-' indicates the method is not employed on the corresponding dataset. All the experiments results are obtained under three independent runs with the same experimental setups.

Method	PMNIST		Split CIFAR-100		CIFAR-100 Superclass		5-Datasets		Split TinyImageNet	
	ACC(%)[↑]	BWT(%)[↑]	ACC(%)[↑]	BWT(%)[↑]	ACC(%)[↑]	BWT(%)[↑]	ACC(%)[↑]	BWT(%)[↑]	ACC(%)[↑]	BWT(%)[↑]
EWC (Kirkpatrick et al., 2017)	92.01	-0.03	68.80	-2.02	-	-	88.64	-0.04	-	-
PackNet (Mallya & Lazebnik, 2018)	96.37	0.00	72.39	0.00	58.78	0.00	92.81	0.00	55.46	0.00
SupSup (Wortsman et al., 2020)	96.31	0.00	75.47	0.00	61.70	0.00	93.28	0.00	59.60	0.00
La-MaML (Joseph & Gu, 2021)	-	-	71.37	-5.39	54.44	-6.65	-	-	66.90	-9.13
FS-DGPM (Deng et al., 2021)	-	-	74.33	-2.71	58.81	-2.97	-	-	70.41	-2.11
GPM (Saha & Roy, 2021)	94.96	-0.02	72.48	-0.90	57.33	-0.37	90.87	-0.01	67.39	1.45
TRGP (Lin et al., 2022b)	96.34	-0.80	74.46	-0.90	58.25	-2.32	92.16	-0.12	68.32	1.71
WSN (Kang et al., 2022)	96.51	0.00	75.86	0.00	61.34	0.00	92.30	0.00	71.96	0.00
Connector (Lin et al., 2022a)	-	-	78.10	-0.30	56.20	-0.40	85.50	-2.90	-	-
DualGPM (Liang & Li, 2023)	-	-	71.33	-0.12	57.60	-1.00	88.70	-1.90	-	-
API (Liang & Li, 2023)	-	-	72.00	0.37	60.20	-0.20	91.10	-0.50	-	-
DFGP (Yang et al., 2023)	94.64	-0.01	74.59	0.00	-	-	92.09	-0.01	-	-
OSN (Ours)	**97.19** ± **0.01**	0.00 ± 0.01	**79.78** ± **0.06**	-0.01 ± 0.01	**63.81** ± **0.06**	-0.12 ±0.02	**93.32** ± **0.09**	0.01 ± 0.02	**75.42** ± **0.38**	-0.52 ± 0.04

Table 2. Comparisons of different pruning based continual learning methods on Split CIFAR-100 and Split TinyImageNet.

Method	Split CIFAR-100		Split TinyImageNet	
	ACC(%)[↑]	CAP(%)[↓]	ACC(%)[↑]	CAP(%)[↓]
PackNet (Mallya & Lazebnik, 2018)	72.39	96.38	55.46	188.67
SupSup (Wortsman et al., 2020)	75.47	129.00	59.60	214.52
WSN (Kang et al., 2022)	75.86	41.88	72.11	37.50
OSN(Ours)	**79.78**	**25.09**	**75.42**	**30.50**

BWT equals 0.00%. But their overall performance is limited by the insufficient model plasticity. Our proposed OSN achieves great improvements on the model plasticity, by updating the shared parameters in the sub-network and obtaining the shared knowledge between old and new tasks. As shown in Table 1, the overall performance of our OSN is much better than other parameter pruning methods. Moreover, in long-term CL dataset, our OSN can achieve more excellent performance compared to other parameter pruning methods. For example, on Split TinyImageNet which exists 40 tasks, our OSN can achieve 75.42% in ACC which is 3.46% higher than the second best method WSN. This indicates the superiority of OSN under long-term CL scenarios.

4.4. Ablation Study and Analysis

Network capacity improving in CL and long-term CL. We compare the network capacity of our OSN with other parameter pruning methods. Followed by (Kang et al., 2022), we introduce a metric CAP to quantify the network capacity in each parameter pruning method. Formally, $CAP = (1 - S) + \frac{(1-\alpha)N}{32}$, where α is average mask compression rate and N is the number of the tasks. S is the percentage of non-fixed parameters. Therefore, CAP represents the sum of the percentage of non-fixed parameters and the compression efficiency of binary encoding. The smaller the CAP is, the better the network capacity will be. We employ some experiments on Split CIFAR-100 and Split TinyImageNet to demonstrate the improvement of our proposed OSN in CAP. As shown in Table 2, for Split CIFAR-100, our OSN achieve 25.09% in CAP, which is better than 96.38% of PackNet, 129.00% of SupSup and 99.13% of WSN. For Split TinyImageNet, the CAP of OSN is superior to all other methods' in Table 2, $e.g.$, 11.15% higher than the second best method WSN. The results in Table 2 strongly demonstrate that our OSN can greatly improve the model capacity compared to other existing parameter pruning methods.

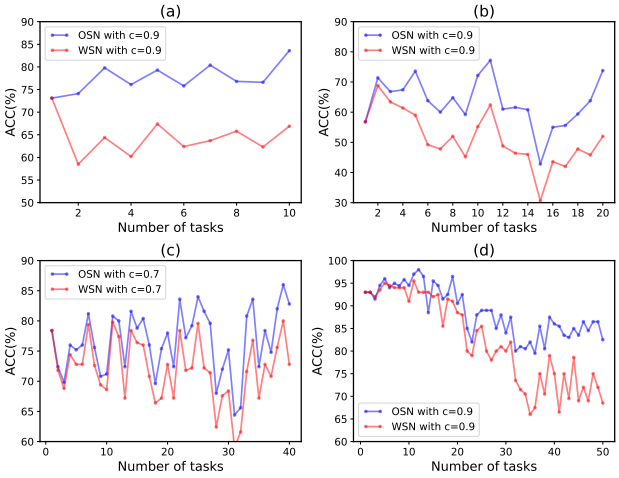

Figure 3. The diagonal ACC(%) of OSN and WSN with the same sparsity ratio c on CL datasets or long-term CL settings. (a) : Split CIFAR-100; (b): CIFAR-100 Superclass; (c): Split TinyImageNet; (d): Split CIFAR-100-50.

Model plasticity improving in CL and long-term CL. We then employ some experiments to assess the contribution of the shared knowledge learned through the orthogonal projection to the model plasticity. Usually, the model plasticity can be measured by IM (Chaudhry et al., 2018a) or FWT (Lopez-Paz & Ranzato, 2017). Both of the two measurements calculate the difference between $A_{t,t}$ and A_t^*, where A_t^* is the testing accuracy of the jointly trained model on task T_t. Given that in the same backbone, A_t^* can be considered as a constant, we use the testing accuracy on each new task to measure the model plasticity, $i.e.$, the diagonal ACC $A_{t,t}$. In Figure 3, c represents the ratio of pruned parameters in the sub-network. The larger c is, the more parameters in the sub-network are remained. As shown in Figure 3 (a) and (b), with the same sparsity ratio c, our OSN

Table 3. Ablation studies of each component in OSN on Split CIFAR-100 and CIFAR-100 Superclass. 'OP' indicates orthogonal projection and 'Sharpness-aware OP' indicates sharpness-aware orthogonal projection

Module				Split CIFAR-100			CIFAR-100 Superclass		
Pruning	Network Partitioning	OP	Sharpnesss-aware OP	ACC(%)[↑]	BWT(%)[↑]	CAP(%)[↓]	ACC(%)[↑]	BWT(%)[↑]	CAP(%)[↓]
✓				75.86	**0.00**	99.13	61.34	**0.00**	80.93
✓		✓		77.12	-0.28	**6.88**	62.13	-0.54	**13.75**
✓	✓	✓		78.23	-0.02	25.09	63.21	-0.21	38.75
✓	✓		✓	**79.78**	-0.01	25.09	**63.81**	-0.12	38.75

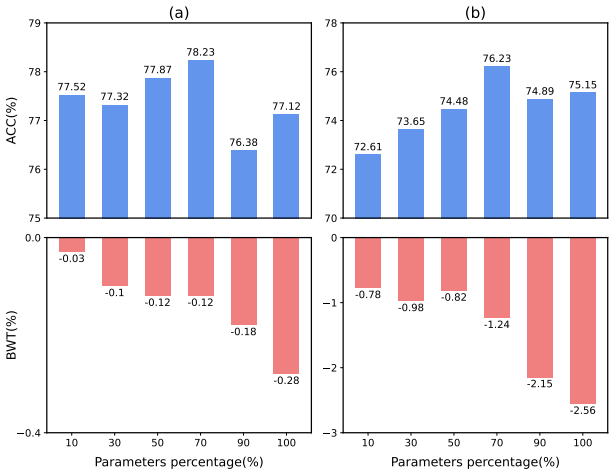

Figure 4. Comparisons of ACC(%) and BWT(%) with different parameters percentage selected by the network partition strategy on dataset (a): Split CIFAR-100 and (b): Split TinyImageNet.

outperforms WSN in diagonal ACC on both Split CIFAR-100 and CIFAR-100 Superclass, which proves the crucial role of shared knowledge in enhancing model plasticity. Besides, we further conduct some experiments on long-term CL settings to illustrate the superiority of our proposed OSN in long-term CL. As shown in Figure 3 (c) and (d), with the same sparsity ratio c, OSN outperforms WSN on both Split TinyImageNet and Split-CIFAR-100-50, and the strengths of OSN become more evident as the number of tasks increases, which indicates shared knowledge in OSN plays a very important role in long-term CL.

OSN vs. WSN with different sparsity ratio. In Figure 5, we compare the ACC of OSN and WSN with different sparsity ratio c on both Split CIFAR-100 and Split TinyImageNet. As c increases, the performance of WSN drops significantly, while OSN continues to perform well. This phenomenon is even more severe on long-term datasets Split TinyImageNet. Since OSN acquires shared knowledge, the impact of network sparsity on its performance stability is minimal even in long-term CL.

Parameters sensitivity in the network partition. We change the k in Algorithm 1 to change the number of selected shared parameters. As shown in Figure 4, when k is large, $i.e.$, a large number of shared parameters are selected, the network may suffer from a little forgetting, $i.e.$,

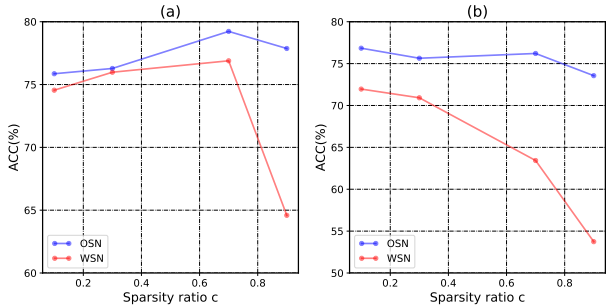

Figure 5. Comparisons between OSN and WSN with different sparsity ratio c on dataset (a): Split CIFAR-100 and (b): Split TinyImageNet.

BWT is small. This is because, compared to the frozen model to preserve old knowledge($i.e.$, $k = 0$), the shared knowledge learned through the approximate orthogonal projection can cause little forgetting. When k become small, the forgetting also decreases. In fact, by varying the number of selected shared parameters, our OSN can achieve a stability-plasticity trade-off, and appropriate number of parameters selected by the network re-partitioning is positive to improve the overall performance.

Ablation studies of each component. Table 3 shows the ablation studies results on Split CIFAR-100 and CIFAR-100 Superclass. As shown in Table 3, every component has a significant contribution to improving the overall performance of OSN. In OSN, pruning while ignoring the shared knowledge can only achieve 75.86% and 61.34% of the ACC on Split CIFAR-100 and CIFAR-100 Superclass. Applying the network re-partitioning and obtaining the shared knowledge through the orthogonal projection (OP) help to improve the performance of OSN, $i.e$, 78.23% and 63.21%. Moreover, a sharpness-aware OP also contributes to improving the overall performances of OSN.

5. Conclusion

In this paper, we propose a task-aware orthogonal sparse network (OSN) to search for the shared knowledge between old and new tasks in CL and further in long-term CL. Different from other parameter pruning methods, we divide the network parameters into three parts, with a new part called shared parameters for exploring the shared knowl-

edge between old and new tasks. The shared parameters are selected by the network partition strategy and updated through a sharpness-aware orthogonal projection to obtain shared knowledge. Qualitative and quantitative analysis show that our OSN can achieve superior performances to existing state-of-the-art methods in both CL and long-term CL settings. In the future, we will try to extend our work to the class incremental learning settings.

Acknowledgements

This work was supported in part by the National Science and Technology Major Project (NO.2022ZD0119004), in part by NSFC under Grant NO.62176198 and U22A2096, in part by the Key R&D Program of Shaanxi Province under Grant 2024GX-YBXM-135, in part by the Key Laboratory of Big Data Intelligent Computing under Grant BDIC-2023-A-004.

Impact Statement

This paper is presented to advance the field of continual learning, contributing to the progress in machine learning. While we acknowledge that continual learning techniques may have broad societal impacts, including but not limited to enhancing the adaptability of artificial intelligence systems to dynamic environments, protecting data privacy, and more, we believe that our work is unlikely to have direct negative societal impacts.

References

Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M., and Tuytelaars, T. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 139–154, 2018.

Aljundi, R., Lin, M., Goujaud, B., and Bengio, Y. Gradient based sample selection for online continual learning. *Neural Information Processing Systems,Neural Information Processing Systems*, Jan 2019.

Bulatov, Y. Notmnist dataset. *Google (Books/OCR), Tech. Rep.[Online]. Available: http://yaroslavvb. blogspot. it/2011/09/notmnist-dataset. html*, 2, 2011.

Chaudhry, A., Dokania, P. K., Ajanthan, T., and Torr, P. H. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 532–547, 2018a.

Chaudhry, A., Ranzato, M., Rohrbach, M., and Elhoseiny, M. Efficient lifelong learning with a-gem. *International Conference on Learning Representations,International Conference on Learning Representations*, Sep 2018b.

Cheng, D., Wang, G., Wang, B., Zhang, Q., Han, J., and Zhang, D. Hybrid routing transformer for zero-shot learning. *Pattern Recognition*, 137:109270, 2023.

Cheng, D., Ji, Y., Gong, D., Li, Y., Wang, N., Han, J., and Zhang, D. Continual all-in-one adverse weather removal with knowledge replay on a unified network structure. *IEEE Transactions on Multimedia*, 2024a.

Cheng, D., Li, Y., Zhang, D., Wang, N., Sun, J., and Gao, X. Progressive negative enhancing contrastive learning for image dehazing and beyond. *IEEE Transactions on Multimedia*, 2024b.

Deng, D., Chen, G., Hao, J., Wang, Q., and Heng, P.-A. Flattening sharpness for dynamic gradient projection memory benefits continual learning. *Advances in Neural Information Processing Systems*, 34:18710–18721, 2021.

Deng, L. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. doi: 10.1109/MSP.2012.2211477.

Dhar, P., Singh, R. V., Peng, K.-C., Wu, Z., and Chellappa, R. Learning without memorizing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5138–5146, 2019.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *International Conference on Learning Representations,International Conference on Learning Representations*, Mar 2019.

Hu, W., Lin, Z., Liu, B., Tao, C., Tao, Z., Ma, J., Zhao, D., and Yan, R. Overcoming catastrophic forgetting for continual learning via model adaptation. In *International conference on learning representations*, 2018.

Huffman, D. A. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952. doi: 10.1109/JRPROC.1952.273898.

Hung, C.-Y., Tu, C.-H., Wu, C.-E., Chen, C.-H., Chan, Y.-M., and Chen, C.-S. Compacting, picking and growing for unforgetting continual learning. *Neural Information Processing Systems,Neural Information Processing Systems*, Jan 2019.

Joseph, J. and Gu, A. Reproducibility report: La-maml: Look-ahead meta learning for continual learning. *CoRR*, abs/2102.05824, 2021. URL https://arxiv.org/abs/2102.05824.

Kang, H., Mina, R. J. L., Madjid, S. R. H., Yoon, J., Hasegawa-Johnson, M. A., Hwang, S. J., and Yoo, C. D. Forget-free continual learning with winning subnetworks. In *International Conference on Machine Learning*,

2022. URL https://api.semanticscholar.org/CorpusID:250340593.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Kong, Y., Liu, L., Wang, Z., and Tao, D. Balancing stability and plasticity through advanced null space in continual learning. In *European Conference on Computer Vision*, pp. 219–236. Springer, 2022.

Krizhevsky, A. Learning multiple layers of features from tiny images. Jan 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, pp. 84–90, May 2017. doi: 10.1145/3065386. URL http://dx.doi.org/10.1145/3065386.

Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.

Lee, K., Lee, K., Shin, J., and Lee, H. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 312–321, 2019.

Li, X., Zhou, Y., Wu, T., Socher, R., and Xiong, C. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning*, pp. 3925–3934. PMLR, 2019.

Li, Z. and Hoiem, D. *Learning without Forgetting*, pp. 614–629. Jan 2016. doi: 10.1007/978-3-319-46493-0_37. URL http://dx.doi.org/10.1007/978-3-319-46493-0_37.

Liang, Y.-S. and Li, W.-J. Adaptive plasticity improvement for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7816–7825, 2023.

Lin, G., Chu, H., and Lai, H. Towards better plasticity-stability trade-off in incremental learning: A simple linear connector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 89–98, June 2022a.

Lin, S., Yang, L., Fan, D., and Zhang, J. TRGP: trust region gradient projection for continual learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022b. URL https://openreview.net/forum?id=iEvAf8i6JjO.

Lopez-Paz, D. and Ranzato, M. Gradient episodic memory for continual learning. *Neural Information Processing Systems,Neural Information Processing Systems*, Jun 2017.

Mallya, A. and Lazebnik, S. Packnet: Adding multiple tasks to a single network by iterative pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. doi: 10.1109/cvpr.2018.00810. URL http://dx.doi.org/10.1109/cvpr.2018.00810.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Reading digits in natural images with unsupervised feature learning. Jan 2011.

Ostapenko, O., Puscas, M., Klein, T., Jahnichen, P., and Nabi, M. Learning to remember: A synaptic plasticity driven framework for continual learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. doi: 10.1109/cvpr.2019.01158. URL http://dx.doi.org/10.1109/cvpr.2019.01158.

Prabhu, A., Torr, P. H. S., and Dokania, P. K. *GDumb: A Simple Approach that Questions Our Progress in Continual Learning*, pp. 524–540. Jan 2020. doi: 10.1007/978-3-030-58536-5_31. URL http://dx.doi.org/10.1007/978-3-030-58536-5_31.

Saha, G. and Roy, K. Gradient projection memory for continual learning. *International Conference on Learning Representations,International Conference on Learning Representations*, May 2021.

Serra, J., Suris, D., Miron, M., and Karatzoglou, A. Overcoming catastrophic forgetting with hard attention to the task. In *International conference on machine learning*, pp. 4548–4557. PMLR, 2018.

Shin, H., Lee, J. K., Kim, J., and Kim, J. Continual learning with deep generative replay. *Advances in neural information processing systems*, 30, 2017.

Ven, G. and Tolias, A. Three scenarios for continual learning. *arXiv: Learning,arXiv: Learning*, Apr 2019.

Wang, S., Li, X., Sun, J., and Xu, Z. Training networks in null space of feature covariance for continual learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2021. doi: 10.1109/cvpr46437.2021.00025. URL http://dx.doi.org/10.1109/cvpr46437.2021.00025.

Wang, Z., Zhan, Z., Gong, Y., Yuan, G., Niu, W., Jian, T., Ren, B., Ioannidis, S., Wang, Y., and Dy, J. Sparcl: Sparse continual learning on the edge. *Advances in Neural Information Processing Systems*, 35:20366–20380, 2022.

Wortsman, M., Ramanujan, V., Liu, R., Kembhavi, A., Rastegari, M., Yosinski, J., and Farhadi, A. Supermasks in superposition. *Neural Information Processing Systems,Neural Information Processing Systems*, Jun 2020.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv: Learning,arXiv: Learning*, Aug 2017.

Yang, E., Shen, L., Wang, Z., Liu, S., Guo, G., and Wang, X. Data augmented flatness-aware gradient projection for continual learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5630–5639, 2023.

Yoon, J., Yang, E., Lee, J., and Hwang, S. J. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

Zeng, G., Chen, Y., Cui, B., and Yu, S. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, pp. 364–372, Aug 2019. doi: 10.1038/s42256-019-0080-x. URL http://dx.doi.org/10.1038/s42256-019-0080-x.

Zenke, F., Poole, B., and Ganguli, S. Continual learning through synaptic intelligence. In *International conference on machine learning*, pp. 3987–3995. PMLR, 2017.

A. Binary Mask Encoding in OSN

In our OSN, a binary mask is applied to represent the pruning of the sub-network from the dense network, $i.e.$, $\mathcal{M}_{t-1}^{l}$ in Eq. 2. With the increasing number of pruned parameters in the sub-network, the binary encoding should be efficiently updated with an effective compression algorithm. Followed by (Kang et al., 2022), we use *Huffman Encoding* (Huffman, 1952) to update the binary mask. The encoding efficiency is often considered as an important factor that affecting the model capacity. Specially, Huffman encoding can achieve an approximately 78% lossless compression, which can help significantly improve model capacity.

B. Experimental Details

In this section, we provide more experimental details in our proposed OSN algorithm. For fair comparisons, We strictly follow the experimental settings employed in (Kang et al., 2022).

B.1. Datasets

The experiments in our OSN contains several datasets: **PMNSIT (Permuted MNIST)**. PMNIST is constructed by different random permutations to the original MNIST (Deng, 2012). PMNIST contains ten different tasks. **Split CIFAR-100**: We Split CIFAR-100 (Krizhevsky, 2009) into 10 different tasks and each task contains 10 disjoint different classes in CIFAR-100. **CIFAR-100 Superclass** is constructed by splitting CIFAR-100 into 20 different classes. Each task contains 5 different classes which are semantically related. **5-Datasets** (Saha & Roy, 2021) contains 5 different tasks and each task is constructed by a separate dataset, including CIFAR-10 (Krizhevsky, 2009), MNIST (Deng, 2012), SVHN (Netzer et al., 2011), FashionMNIST (Xiao et al., 2017) and notMNIST (Bulatov, 2011).

To evaluate the performance of our OSN in long-term CL, we further introduce some datasets with larger number of tasks for long-term CL experimental setups. **Split TinyImageNet** (Krizhevsky et al., 2017) is constructed by splitting TinyImageNet into 40 different tasks and each task contains 5 different tasks. **Split CIFAR-100-50** is a dataset constructed for long-term CL, which contains 50 different tasks and each task contains 2 different disjoint classes.

B.2. Training Details

For all experiments on PMNIST, we use a 2-layer MLP and each hidden layer contains 100 neurons. We set the epoch as 15 and the initial learning rate as 0.001. We use Adam as the initial optimizer. The batch size is set as 10. For all experiments on Split CIFAR-100 and Split CIFAR-100-50, we use a modified version of AlexNet followed by (Saha & Roy, 2021). We set the epoch as 50 and the initial learning rate as 0.001. We use Adam as the initial optimizer. The batch size is set as 64. On CIFAR-100 Superclass, we use a modified LeNet as the backbone followed by (Saha & Roy, 2021). The epoch and the initial learning rate are set as 120 and 0.001, respectively. We use Adam as the initial optimizer and the batch size is set as 64. For the experiments on 5-Datasets, we use a reduced ResNet-18 (Saha & Roy, 2021) as the backbone network. We set the epoch as 80 and the initial learning rate as 0.001. We use Adam as the initial optimizer and set the batch size as 64. We use a backbone followed by (Saha & Roy, 2021) on Split TinyImageNet. We set the epoch as 40 and the initial learning rate as 0.001. We use Adam as the initial optimizer and set the batch size as 64. All the experiments are implemented on four NVIDIA 2080Ti GPUs with PyTorch.